E. Ü. FEN BİLİMLERİ ENSTİTÜSÜ

EGE ÜNİVERSİTESİ

YÜKSEK LİSANS TEZİ

JAVACARD YAZILIMLARININ OTOMATİK ÜRETİLMESİ İÇİN BİR YÖNTEM

Miray TOSUN

Tez Danışmanı : Doç. Dr. Geylani Kardaş

Uluslararası Bilgisayar Anabilim Dalı

Sunuş Tarihi : 07.10.2015

Bornova-İZMİR

2015

EGE ÜNİVERSİTESİ FEN BİLİMLERİ ENSTİTÜSÜ

(YÜKSEK LİSANS TEZİ)

JAVACARD YAZILIMLARININ OTOMATİK ÜRETİLMESİ İÇİN BİR YÖNTEM

Miray TOSUN

Tez Danışmanı : Doç. Dr. Geylani Kardaş

Uluslararası Bilgisayar Anabilim Dalı

Sunuş Tarihi : 07.10.2015

Bornova-İZMİR

2015

Miray TOSUN tarafından YÜKSEK LİSANS tezi olarak sunulan “JavaCard Yazılımlarının Otomatik Üretilmesi İçin Bir Yöntem” başlıklı bu çalışma EÜ Lisansüstü Eğitim ve Öğretim Yönetmeliği ile EÜ Fen Bilimleri Enstitüsü Eğitim ve Öğretim Yönergesi’nin ilgili hükümleri uyarınca tarafımızdan değerlendirilerek savunmaya değer bulunmuş ve 07.10.2015 tarihinde yapılan tez savunma sınavında aday oybirliği/~~oyçokluğu~~ ile başarılı bulunmuştur.

Jüri Üyeleri:		**İmza**
Jüri Başkanı	**: Doç. Dr. Geylani KARDAŞ**	**.....**
Raportör Üye	**: Doç. Dr. Orhan DAĞDEVİREN**	**.....**
Üye	**: Doç. Dr. Hürevren KILIÇ**	**.....**

EGE ÜNİVERSİTESİ FEN BİLİMLERİ ENSTİTÜSÜ

ETİK KURALLARA UYGUNLUK BEYANI

EÜ Lisansüstü Eğitim ve Öğretim Yönetmeliğinin ilgili hükümleri uyarınca Yüksek Lisans Tezi / ~~Doktora Tezi~~ olarak sunduğum "Java Card Yazılımlarının Otomatik Üretilmesi İçin Bir Yöntem" başlıklı bu tezin kendi çalışmam olduğunu, sunduğum tüm sonuç, doküman, bilgi ve belgeleri bizzat ve bu tez çalışması kapsamında elde ettiğimi, bu tez çalışmasıyla elde edilmeyen bütün bilgi ve yorumlara atıf yaptığımı ve bunları kaynaklar listesinde usulüne uygun olarak verdiğimi, tez çalışması ve yazımı sırasında patent ve telif haklarını ihlal edici bir davranışımın olmadığını, bu tezin herhangi bir bölümünü bu üniversite veya diğer bir üniversitede başka bir tez çalışması içinde sunmadığımı, bu tezin planlanmasından yazımına kadar bütün safhalarda bilimsel etik kurallarına uygun olarak davrandığımı ve aksinin ortaya çıkması durumunda her türlü yasal sonucu kabul edeceğimi beyan ederim.

7 /10 /2015

İmzası

Adı-Soyadı

Miray TOSUN

ÖZET

JAVACARD YAZILIMLARININ OTOMATİK ÜRETİLMESİ İÇİN BİR YÖNTEM

TOSUN, Miray

Yüksek Lisans Tezi, Uluslararası Bilgisayar Anabilim Dalı
Tez Danışmanı: Doç. Dr. Geylani Kardaş
Ekim 2015, 42 sayfa

Java Card platformu ve beraberinde gelen uygulama programlama arayüzü akıllı kart uygulamalarının geniş kullanıma sahip Java programlama dili ile nesne yönelimli olarak geliştirilmesine imkan vermektedir. Ancak işlemci ve bellek yönünden oldukça kısıtlı olan bu kartlar için sadece Java dilinin bir alt kümesi kullanılabilmektedir ve geliştiriciler bilgisayar ve akıllı kart arasındaki paket alışverişi için alt seviye bir protokolün detay ve kısıtları ile uğraşmak durumundadırlar. Bu tez çalışması kapsamında Java Card uygulamalarının geliştirilmesinde görülen bu zorlukları gidermek amacıyla kart üzeri uygulamaların model güdümlü geliştirilmesine imkan veren DSL4JavaCard isimli bir alana özgü dil tüm bileşenleri ile birlikte geliştirilmiştir. Dilin soyut sözdiziminin dayandığı üstmodel ve modelleme aşamasında bir takım kısıtların kontrol edilmesini sağlayan somut sözdizimi tanımlandıktan sonra DSL4JavaCard modellerinden Java Card uygulama kodları otomatik olarak elde edilebilmektedir. Ayrıca önerilen dilin durum çalışması üzerinden kullanımının örneklendirilmesi ve değerlendirmesi yine bu tez içerisinde yer almaktadır. Değerlendirmeler sonucunda DSL4JavaCard'ın ilgili literatürde yer alan önceki çalışmalardan farklı olarak özellikle Java Card güvenlik bileşenlerinin de model güdümlü geliştirilebilmesine imkan verdiği ve işletimsel semantiğinin temel Java Card bileşenlerinin tamamının, kullanıcı tanımlı ve iş uygulamasına özel öğelerin ise önemli bir kısmının otomatik üretilmesini sağladığı görülmüştür.

Anahtar sözcükler: Alana özgü dil, Model güdümlü geliştirme, Akıllı kart, Java Card, DSL4JavaCard

ABSTRACT

A METHODOLOGY FOR THE AUTOMATIC GENERATION OF JAVACARD SOFTWARE

TOSUN, Miray

MSc. in International Computer
Supervisor: Assoc. Prof. Dr. Geylani Kardaş
October 2015, 42 pages

Java Card platform enables the object-oriented design and implementation of smart card applications. However, only a subset of Java programming language can be used due to limited processing and storage capabilities of these devices. Further, developers are forced to deal with the constraints and details of a low-level communication protocol for the packet transmission between smart cards and host computers. In order to eliminate those difficulties of Java Card software development and support the model-driven development of on-card applications, a domain-specific language, called DSL4JavaCard has been developed in this thesis study. The definition of both a metamodel for the language's abstract syntax and a concrete syntax for checking some contraints during smart card software modelling in the thesis paves the way of automatic generation of Java Card application codes from DSL4JavaCard instance models. In addition, case study-based exemplification of the language usage and evaluation are also provided in the thesis. Achieved results show that DSL4JavaCard differentiates from the previous related work specifically by enabling the model-driven development of Java Card security components. Moreover, it is experienced that operational semantics of the language succeeds in automatic generation of all fundamental Java Card components required in a smart card application while providing the automatic generation of most of the user-defined components and code blocks specific to the bussiness domain of the smart card application.

Keywords: Domain-specific language, Model-driven development, Smart card, Java Card, DSL4JavaCard

TEŞEKKÜR

Öncelikle bu tez konusu üzerinde bana çalışma imkanı sunan ve danışmanım olan Doç. Dr. Geylani Kardaş'a, deneyimleri, bilgileri ve önerileriyle araştırma ve geliştirmeye yönlendirmesi, sağladığı kaynaklar ile destek olmasından dolayı ve ayrıca bu uzun süreçte bana sabrından dolayı teşekkürü bir borç bilirim.

Tezim boyunca yanımda olan canım aileme, her ihtiyacım olduğunda yanımda olan Emre'ye ve bitirebilmem için tüm izinleriyle yanımda olan Obase ailesine teşekkürü bir borç bilirim. Ayrıca bana destek olan ve her öğlen kahvemi yapan iş arkadaşlarıma teşekkür ederim.

İÇİNDEKİLER

Sayfa

İÇİNDEKİLER (devam)

ŞEKİLLER DİZİNİ

Şekil Sayfa

ŞEKİLLER DİZİNİ (devam)

ÇİZELGELER DİZİNİ

Çizelge Sayfa

KISALTMALAR DİZİNİ

Kısaltmalar

APDU	Uygulama protokolü veri birimi
API	Uygulama programlama arayüzü
CIM	Hesaplama bağımsız model
DSL	Alana özdü dil
EEPROM	Elektronik Silinebilir Programlanabilir Salt Okunur Bellek
JCRE	Java kart çalıştırma ortamı
JCVM	Java kart sanal makinesi
MDA	Model güdümlü mimari
MDD	Model güdümlü geliştirme
OCL	Nesne kısıtlama dili
OMG	Nesne yönetim grubu
PIM	Platform bağımsız model
PIN	Kişisel kimlik numarası
PSM	Platforma özgü model
ROM	Salt okunur bellek
SIM	Abone kimlik modülü
UML	Birleşik modelleme dili

KISALTMALAR DİZİNİ (devam)

Kısaltmalar

XML	Genişletilebilir işaretleme dili
XMI	XML üst veri değişimi

1. GİRİŞ

Kendine özel işlemcisi ve belleği olan ve verilerin üzerinde güvenli bir şekilde işlenmesini ve saklanmasını sağlayan akıllı kartlar günümüzde başta mobil iletişim ve bankacılık alanı olmak üzere çeşitli sektörlerde yaygın olarak kullanılmaktadır. Donanım özellikleri masaüstü bir bilgisayara kıyasla oldukça kısıtlı kalan bu kartlar üzerinde çalıştırılacak yazılımların geliştirilmesi de yine masaüstü uygulamalara göre oldukça zahmetli ve zaman alıcıdır. Yazılım geliştiricilerin alt seviye bir akıllı kart iletişim protokolüne (ISO/IEC 7816) (ISO/IEC Standart,1995) hakim olması, sınırlı bellek kaynakları kullanmak amacıyla sadece çok ilkel veri yapılarını oluşturması, onaltılı (ing. "hexadecimal") yapıdaki veri paketlerini hazırlaması ve yine bu paketlerin onaltılı yapıda işlenmesini yazılımsal olarak sağlaması gerekmektedir.

İlk olarak Sun Microsystems tarafından geliştirilen ancak şu an Oracle tarafından geliştirilmesi ve dağıtımı sürdürülen Java Card yazılım çerçevesi ve platformu (Oracle Co., 2015) yazılım geliştiricilere Java programlama dilini kullanarak akıllı kartlar için ISO/IEC 7816 standartına dayalı ve nesne tabanlı olarak yazılım geliştirme imkanını sunmakta ve yukarıda değinilen akıllı kart yazılım geliştirme zorluklarını azaltmaya önemli ölçüde yardımcı olmaktadır. Ancak akıllı kartların bellek kısıtları nedeniyle Java Card çerçevesi Java programlama dilinin çok küçük bir alt kümesini kapsamaktadır ve temel veri tipleri olarak sadece *boolean, short* ve *byte* kullanılabilmektedir. Tamsayılar, karakterler ve *String* yapıları Java Card içerisinde kullanılamamaktadır. Bunların dışında yine çok boyutlu diziler, dinamik sınıf yükleme, çöp toplayıcı, nesne serileştirme ve klonlama Java Card'da desteklenmemektedir. Üstelik onaltılı formattaki veri paketlerinin oluşturulması ve işlenmesi zorluğu devam etmektedir. Alana özgü dillerin (ing. "domain-specific language") (DSL)(van Deursen. et al., 2000; Mernik et al., 2005; Fowler, 2011) geliştiricilere sağladığı soyutlama seviyesi, ifade gücü ve kullanım kolaylığı Java Card uygulamalarının geliştirilmesi sırasında karşılaşılan zorlukları gidermeye yardımcı olabilir. Test ve hata ayıklama imkanlarının çok kısıtlı olduğu bu sistemler için özellikle

modelleme seviyesinde kontroller akıllı kart yazılımı geliştirmeyi kolaylaştırabilir. Ayrıca DSL'lerin bir çoğunun sunduğu otomatik kod üretimi özelliği de kart uygulamalarının daha hızlı ve etkin geliştirilmesini sağlayabilir. DSL'lerin tüm bu avantajları göz önüne alınarak bu tez çalışmasında Java Card uygulamalarının model güdümlü geliştirilmesini ve yazılım kodlarının otomatik üretilmesini sağlayan DSL4JavaCard isimli bir DSL geliştirilmiştir.

DSL4JavaCard'ın sözdizimi Java Card çerçevesini tam olarak destekleyen bir üstmodele bağlı olarak geliştirilmiştir. İçerdiği bakış açıları akıllı kart temel yapıları haricinde güvenlik ve doğrulamaya ait varlıkları ve ilişkileri de desteklemektedir. Bu yönüyle literatürdeki bir çok benzer çalışmadan (Coglio, A., 2003; Gomes et al., 2007; Moebius et al., 2009; Sarıtaş ve Kardaş, 2011; Saritas and Kardas, 2014) farklılaşmaktadır. Somut sözdizim bir takım akıllı kart modeli kontrollerinin ve doğrulama kurallarının işletilmesiyle desteklenmektedir. Bunlara ek olarak DSL4JavaCard'ın işletimsel semantiği modellenen yazılımlara ait Java Card kodlarının otomatik olarak elde edilmesini sağlamaktadır. Tüm bu dil bileşenleri ve bunların entegrasyonu yukarıda değinilen ve tezin bir sonraki bölümünde anlatılacak olan literatürdeki diğer çalışmalar göz önüne alındığında DSL4JavaCard'a akıllı kart yazılımı geliştirme alanı için önerilen ilk DSL olma özelliğini kazandırmaktadır (Tosun ve Kardaş, 2015).

Tezin geriye kalan kısmında ilk olarak literatürdeki ilgili çalışmalar anlatılmıştır. Üçüncü bölümde DSL4JavaCard üstmodelinin çıkarılması ve kod dönüşümleri için kullanılan alt yapı anlatılmaktadır. Dördüncü bölümde DSL4JavaCard'ın soyut sözdizimi yer almaktadır. Beşinci bölümde DSL4JavaCard somut sözdizimi ve içeriğine değinilmektedir. Altıncı bölümde otomatik kod üretiminin sağlandığı işletimsel semantik anlatılmıştır. Yedinci bölüm iki farklı durum çalışması üzerinden DSL4JavaCard'ın değerlendirmesini içermektedir. Elde edilen sonuç ve ileriye yönelik çalışmalar son bölüm olan sekizinci bölümde yer almaktadır.

2. İLGİLİ ÇALIŞMALAR

Java Card platformu için model güdümlü yaklaşımın ve/veya Java Card yazılımlarının otomatik olarak üretilmesinin göz önüne alındığı ilk çalışmanın (Coglio, 2003) olduğu söylenebilir. Bu çalışmada Java Card uygulamalarının (ing. "Applet") üretilebilmesi ve denetlenebilmesi için hazırlanan bağımsız bir sertifikasyon mekanizması önerilmiştir. Önerilen yöntemin resmi bir tanımlaması ise (Coglio and Green, 2008)'de verilmiştir. DSL4JavaCard model kontrolünü ön tanımlı bir üstmodel aracılığı ile gerçekleştirdiğinden bu çalışmalardan ayrışmaktadır. Bonnet ve ark. kart üzeri yazılımların model güdümlü olarak kişiselleştirilmesini (Bonnet et al., 2004a) ve bu çerçevenin akıllı kart konfigürasyonuna adaptasyonunu (Bonnet et al., 2004b) önermektedir. Bu tezde ortaya konan yaklaşım ise akıllı kart konfigurasyonu ve kart donanımı üretiminden ziyade akıllı kart yazılımlarının model güdümlü geliştirilmesini desteklemektedir. Öte yandan Moebius ve ark. (2009a) güvenli akıllı kart uygulamalarını UML temelli sistem modelleri üzerinden geliştirmek için model güdümlü bir yöntem önermiştir. Bu yöntemin Java Card'lar için bir uygulaması ise (Moebius et al., 2009b)'de tanıtılmıştır. Ancak model dönüşümlerinin ve kod üretiminin nasıl uygulandığı bu çalışmalarda belli değildir. Ayrıca bu çalışmalar kart uygulamalarının otomatik geliştirilmesinden çok güvenli geliştirilmesi üzerinde durmaktadır.

B Metoduna dayalı olarak Java Card uygulamalarının geliştirilmesini öneren Tatibouet ve ark. (2003) Java Card platformuna özel iletişim ve kod ayrıntılarını birleştirmek için kodun elle değiştirilmesi gerektiğinden bahsetmiştir. Gomes ve ark. (2007) ise (Tatibouet et al., 2003)'te anlatılan yöntemi otomatik hale getirmeyi amaçlamaktadır ancak çalışma sadece fikirlerden ibarettir ve yazarlarının da belirttiği gibi otomatik kod üretimi için bu yöntemin bir araç desteğine ihtiyacı vardır.

DSL4JavaCard'in öncül çalışmaları olarak kabul edilebilecek olan (Sarıtaş, 2011; Sarıtaş ve Kardaş, 2011)'de Java Card yazılımlarının model güdümlü geliştirilmesi için bir üstmodel ve bu üstmodele dayalı kod dönüşümleri tanıtılmıştır. Ancak bu çalışmalarda sadece temel kart bileşenlerinin model

güdümlü geliştirilmesi söz konusudur. Java Card'ların istisna ve güvenlik bileşenlerinin geliştirilmesi göz önüne alınmamıştır.

Nikseresht ve Ziarati (2011), akıllı kart uygulamalarının model güdümlü mimariye göre platform bağımsız ve platforma özel modellenmesini göz önüne alsa da önerilen yöntem uygulandığında platform bağımlı diye adlandırılan modellerin aslında sadece platform bağımsız modellere ek alanlar katmaktan ibaret olduğu ve model güdümlü mimarinin önerdiği soyutlama seviyelerini sağlayamadığı görülmektedir. Ayrıca bu çalışma sadece dosya tabanlı akıllı kart uygulamalarının geliştirilmesini kapsamakta; Java Card gibi nesne yönelimli akıllı kart uygulamarının geliştirilmesini desteklememektedir.

Akıllı kart yazılımlarının model güdümlü geliştirilmesi ve otomatik elde edilmesi kapsamında en güncel çalışmanın (Saritas and Kardas, 2014)'te gerçekleştirildiği görülmektedir. Bu çalışmada model güdümlü mimariye dayalı olarak kart üzeri yazılımların platform bağımsız modellenmesi ve tanımlanan modelden modele ve modelden metne dönüşümler sonucunda bu modellerin aralarında Java Card'ın da bulunduğu çeşitli akıllı kart işletim ortamlarında çalışacak şekilde kodlarının üretilmesi sağlanmıştır. Genel bir model güdümlü mimari sunan bu çalışmadaki Java Card üstmodeli de DSL4JavaCard'in içerdiği bakış açılarına sahip değildir ve sadece temel akıllı kart bileşenlerini desteklemektedir. PIN oluşturma haricindeki akıllı kart güvenlik ve şifreleme fonksiyonları (Saritas and Kardas, 2014)'te kapsam dışındadır.

Bu bölümde değinilen çalışmalar ile tezde geliştirilen DSL4JavaCard'ın özet karşılaştırması Tablo 2.1'de yer almaktadır.

Tablo 2.1. DSL4JavaCard ve diğer çalışmaların karşılaştırılması

ÇALIŞMA	ÇALIŞMANIN ÖZELLİKLERİ	DSL4JAVACARD İLE KARŞILAŞTIRMA
Coglio, 2003 Coglio and Green, 2008	Java Card uygulamalarının üretilebilmesi ve denetlenebilmesi için hazırlanan bağımsız bir sertifikasyon mekanizması ile kontrolün yazılımın sonrasında yapılması hedeflenmiştir.	DSL4JavaCard ile model kontrolü ön tanımlı bir üstmodel aracılığı ile gerçekleştirilmektedir.
Bonnet et al., 2004a Bonnet et al., 2004b	Kart üzeri yazılımların model güdümlü olarak kişiselleştirilmesi ve akıllı kart konfigürasyonuna adaptasyonu sağlanmıştır.	DSL4JavaCard akıllı kart konfigurasyonu ve kart donanımı üretiminden ziyade akıllı kart yazılımlarının model güdümlü geliştirilmesini desteklemektedir.
Moebius et al., 2009a Moebius et al., 2009b	UML temelli sistem modelleri üzerinden kart uygulamalarının güvenli bir model güdümlü yöntemle geliştirilmesi amaçlanmaktadır.	DSL4JavaCard UML'e bağlı kalmaksızın daha geniş bir üstmodele bağlı olarak kart üzeri uygulamaların geliştirilmesini sağlayan bir soyut sözdizime sahiptir.
Tatibouet et al., 2003 Gomes, et al., 2007	Java Card için formal modellemeye dayalı bir kod üretim yöntemi önerilmiştir. Ancak uygulamayı destekleyen bir araç bulunmamaktır.	Önerilen kod üretimini gerçekleştirmeyi sağlayan bir araç desteği bulunmaktadır.
Sarıtaş, 2011 Sarıtaş ve Kardaş, 2011 Saritas and Kardas, 2014	Java Card temel kart bileşenlerinin model güdümlü geliştirilmesi sağlanmaktadır.	DSL4JavaCard sözdizimi temel bileşenlere ek olarak kart üzeri istisna ve güvenlik bileşenlerini de barındırmaktadır.
Nikseresht and Ziarati, 2011	Platform bağımsız kart modellerine ek alanlar katarak platform bağımlı kart modellerinin oluşturulması öngörülmektedir. Java Card platfomu yerine sadece dosya tabanlı kart uygulamalarının geliştirilmesi amaçlanmıştır.	DSL4JavaCard dosya tabanlı kart ortamları yerine Java Card'a dayalı bir biçimde kart uygulamalarının nesne yönelimli geliştirilmesini desteklemektedir.

3. ALTYAPI

Bu bölümde tezde kullanılan ve DSL4JavaCard'in altyapısını oluşturan çeşitli donanım/yazılım teknolojileri ve yazılım geliştirme yöntemleri anlatılmaktadır. İlk olarak akıllı kart teknolojilerinden ve Java Card'lardan bahsedilmekte sonrasında ise model güdümlü yazılım geliştirme ve alana özgü dil anlatılmaktadır. En son olarak ise model güdümlü yazılım geliştirme için sunulan Eclipse platformu ve Xpand ortamı tanıtılmaktadır.

3.1 Akıllı Kart Teknolojisi

Akıllı kartlar, diğer bir adı ile entegre devre kartı (ing. "integrated circuit") ya da çip kartı (ing. "chip card"), günümüzde sağlık, ulaşım, finans ve telekominikasyon gibi bir çok sektörde güvenilir olması sebebiyle tercih edilmekte ve yaygın olarak kullanılmaktadır (Chen, 2000). Güvenli oluşları, kullanım ve taşımadaki kolaylıkları sebebiyle her geçen gün çeşitli sektörlerde kullanımları daha da yaygın hale gelmektedir.

Bir işlemcisi ve bellek birimi bulunduran akıllı kartlar, okuyucu ile eşleştiği zaman farklı uygulamalar için gerekli işlem gücüne sahip olabilen plastik kartlardır (Chen, 2000). Güvenli ve kontrollü erişim sayesinde, yetkili kişiler tarafından bilgilerin görülebildiği ve işlem yapılabildiği ticari akıllı kartlar, veri taşıma ve veri işleme özelliklerini bünyesinde barındırır.

Akıllı kartların ya da tezde bahsi geçen Java Card'ların tüm dış görünüşlerinin yanı sıra içeriğinde bulunan temas noktaları, veri alışverişi için kullanılan komut kümeleri ve uygulamaları belirlemek için kullanılan kimlik sistemleri ve veri elemanları ISO-7816 tarafından belirlenmiş ve standart hale getirilmiştir (ISO/IEC 7816 Standards, 1995).

Akıllı kartların genel yapısı incelendiğinde, bilgisayar içerisinde bulunan temel parçaları bünyesinde barındırdığı görülmektedir. Ancak daha düşük hız ve daha düşük kapasitede olduğu da bilinmektedir. Akıllı kartlar genel olarak Abone kimlik modülü (ing. "Subscriber Identity Module") SIM kart veya kredi kartları

büyüklüğünde olup birçok çeşide sahiptir. Bellekli, temaslı ve temazsız kartlar, akıllı kart çeşitlerinden birkaçına örnektir. En küçük 8-bit işlemci ve 5 Mhz ile başlayan gömülü bir işlemciye sahip ve aynı zamanda içerisinde Elektronik Silinebilir Programlanabilir Salt Okunur Bellek (ing. "Electrically Erasable Programmable Read-Only Memory") (EEPROM) ve Salt Okunur Bellek (ing. "Read-Only Memory") (ROM) belleğe sahip, ucuz ve küçük sistemlerdir. Bu sebeple yaygın ancak yine bu sebeple yazılımsal geliştirilmesi güç bir sanayi alanıdır.

Eski tür akıllı kartların genel özellikleri tek kart ve tek işlev olarak söylenebilir. Çoğunluğunda C programlama dili ya da Assembly dili kullanılmaktadır. Yazılımları ROM üzerinde ve güncellenmesi mümkün olmayan türdedir. Ancak zaman geçtikçe bu standart kartlar yerini akıllı, güncellenebilir ve birden çok işlevi içinde bulunduran kartlara bırakmıştır. Bunlara en iyi örnek ise Java Card'lardır.

3.1.1 Java Card Teknolojisi

Java Card (Chen, 2000), Java programlama dili ile yazılabilen, güvenli bir yapıya sahiptir. Farklı akıllı kart teknolojilerinde kullanılabilmektedir. Java Card teknolojilerinin en önemli özelliği Java dilinin bir alt kümesi olan özel bir kütüphane ile yazılıyor oluşudur. Bununla birlikte güvenli, şifreli ve verilerin kontrollü bir şekilde işlenmesi ile birden çok uygulamayı içerisinde barındırıyor oluşu, Java Card'ı günümüzde yaygın olarak görme sebepleri arasında sıralanabilir.

Java Card'ın yukarıda listelenen özellikleri aynı zamanda Java Card'ı bir çok alternatif akıllı kart ortamı ve akıllı kart yazılımı geliştirme diline göre tercih edilebilinir kılmaktadır. Çıktığı dönemde akıllı kart yazılım geliştirmeye getirdiği nesne yönelimi bakışaçısı bu paradigmaya aşina yazılımcıların daha kolay akıllı kart yazılımı geliştirmesine imkan vermiştir. Ayrıca tek kart üzerinde birden fazla uygulamanın yer alabilmesi, kart üzeri uygulamaların iletişimi, vb. özellikler de Java Card'ı muadillerine göre bir adım öne geçirmiştir.

Java Card'larda, uygulama içerisinde daha sonradan güncelleme yapabilme imkanı bulunmaktadır. Uygulamalar kullanılan bir uygulama programlama arayüzü (ing. "Application Programming Inteface") (API) sayesinde işletim sisteminden ayrılmıştır ve böylece uygulamalar işletim sisteminden bağımsız olarak geliştirilebilir. Java Card sistemleri içerisindeki senkronizasyon güvenliği, işletim sistemi ve uygulamaları birbirinden ayıran bir güvenlik duvarı (ing. "firewall") yapısı tarafından sağlanmaktadır. Böylelikle yapı kullanım alanını da genişletmekte, günümüzde kimlik ve finansal bilgiler gibi güvenliğin üst düzey olduğu alanlarda kullanıldığı gözlemlenmektedir.

Java Card'lardaki uygulamalar diğer masaüstü Java programlarından farklı şekilde çalışırlar. Java kütüphanesinin sınırlı bir alt yapısı ile çalıştığı için, içerisinde bazı tanımlamalar eksik bırakılmıştır. Java Card sadece ("byte", "short", "boolean") veri tipleri ile bazı durumlar için özel kullanılan ("int") veri tipini desteklemekte bunun dışında kalan veri tiplerini ise kısıtlı alan sebebi ile desteklememektedir. Tablo 3.1 içerisinde desteklenen ve desteklenmeyen özellikler listelenmiştir

Tablo 3.1. Java Card'da Desteklenen ve Desteklenmeyen Özellikler

DESTEKLENENLER	**DESTEKLENMEYENLER**
Boolean, byte, short veri tipleri	Float, char, string, long, double
Tek boyutlu diziler	Çok boyutlu diziler
Java Paketleri, sınıf, arayüz ve istisna sınıfları	Dinamik sınıf yükleme
Nesne tabanlı java programlama özellikleri ("OOP")	Çöp toplama sistemi ("Garbage collection")
Sanal metotlar	Nesnelerin serileştirilmesi
Aşırı yükleme ("Overloading")	Nesne klonlama
Dinamik nesne oluşturma	Çoklu işlem parçacıkları
Faaliyet alanları	
Yükleme kuralları	
Int veri tipi (seçmeli olarak)	

Java Card teknolojisi 3 temel parçadan oluşmaktadır: Java Card Sanal Makinası (ing. "Java Card Virtual Machine") (JCVM), Java Card Çalışma Ortamı (ing. "Java Card Runtime Environment") (JCRE) ve Java Card Uygulama Çatısı

(ing. “Java Card Application Programming Interface”) (Java Card API). JCVM kart-içi (ing. “On-Card”) ve kart-dışı (ing. “Off-Card”) olmak üzere iki farklı bölümden oluşmaktadır ve Java Card uygulamaları (ing. “Applet”) yazıldıktan sonra dönüştürücü (ing. “Converter”) dönüşümü sağladıktan sonra kart üzerine uygulama yüklenir. JCRE ise, akıllı kartlar için özel tasarlanmış işletim sistemi gibi davranmaktadır. Son olarak Java Card API, Java kütüphanesinin alt elemanları ile tasarlanmış olan Java Card uygulamalarının standart hale gelmesi için hazırlanmış sınıf ve paketleri (ör. (“javacard.framework”), (“javacard.security”), ve (“java.lang”)) içermektedir (Oracle Co., 1999).

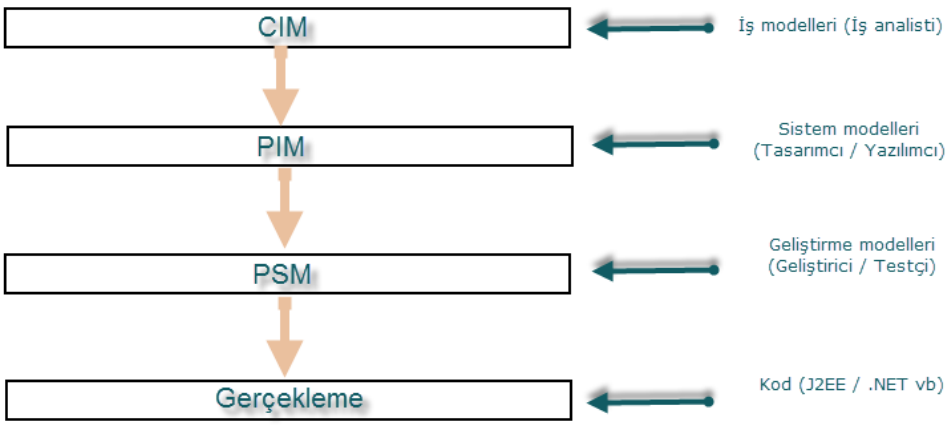
3.2 Model Güdümlü Yazılım Geliştirme ve Alana Özgü Dil

Modelleme, bir uygulamanın ya da bir sistemin geliştirilmeden önce karmaşık ve fazla detaylardan arındırılarak daha anlaşılır şekilde ifade edilmesini sağlamaktadır. Bu noktadan hareketle Model Güdümlü Yazılım Geliştirme (ing. “Model Driven Development”) (MDD) (Selic, 2003), genellikle yazılımın anlaşılabilir ve detaysız bir halinin modellenerek ve platformdan bağımsız şekilde tasarlanarak oluşturulması ve sonrasında bu modelin koda dönüştürülmesini kapsayan bir yazılım geliştirme yöntemidir. Temel amaç ana ihtiyaçları ve sistemin birbiri ile olan bağlantılarını daha iyi anlayabilmek ve koda dönüşüm sürecini hızlandırabilmektir.

MDD’nin temel kuralları ve metodolojileri Nesne Yönetim Grubu (ing. “Object Management Group) (OMG) tarafından Model Güdümlü Mimari (ing. “Model Driven Architecture”) (MDA) altında belirlenmiştir (OMG, 2003). MDA yazılım geliştirme süreçleri ile nesne yönelimli programlama teknikleri üzerinde kurulan modelleme yöntemleri sayesinde kod ile sistem arasında bir soyutlama sağlamıştır.

Bir sistem MDA ile modellenirken üç ana soyutlama seviyesi kullanılmaktadır. Programlama bağımsız model (ing. “Computational Independent Model”) (CIM), platform bağımsız model (ing. “Platform Independent Model”) (PIM) ve platforma özgü model (ing. “Platform Specific Model”) (PSM). Şekil-3.2’de bu seviyeler gösterilmiştir. Soyutlama seviyelerinin temel amacı, platforma özgü modellemeden uzaklıştırıp, platform bağımsız ve programlama bağımsız model oluşturarak daha soyut bakmayı sağlamak ve sonrasında platforma özgü modelleme ile kod üretimini sağlamaktır. CIM ve PIM seviyesinde sistem her

ortamda ve platformda çalışabilecek şekilde geliştirilir; bu da geliştiricilerin çok yönlü yazılım geliştirmesine olanak sağlamaktadır.



Şekil 3.2. Soyutlama Seviyeleri

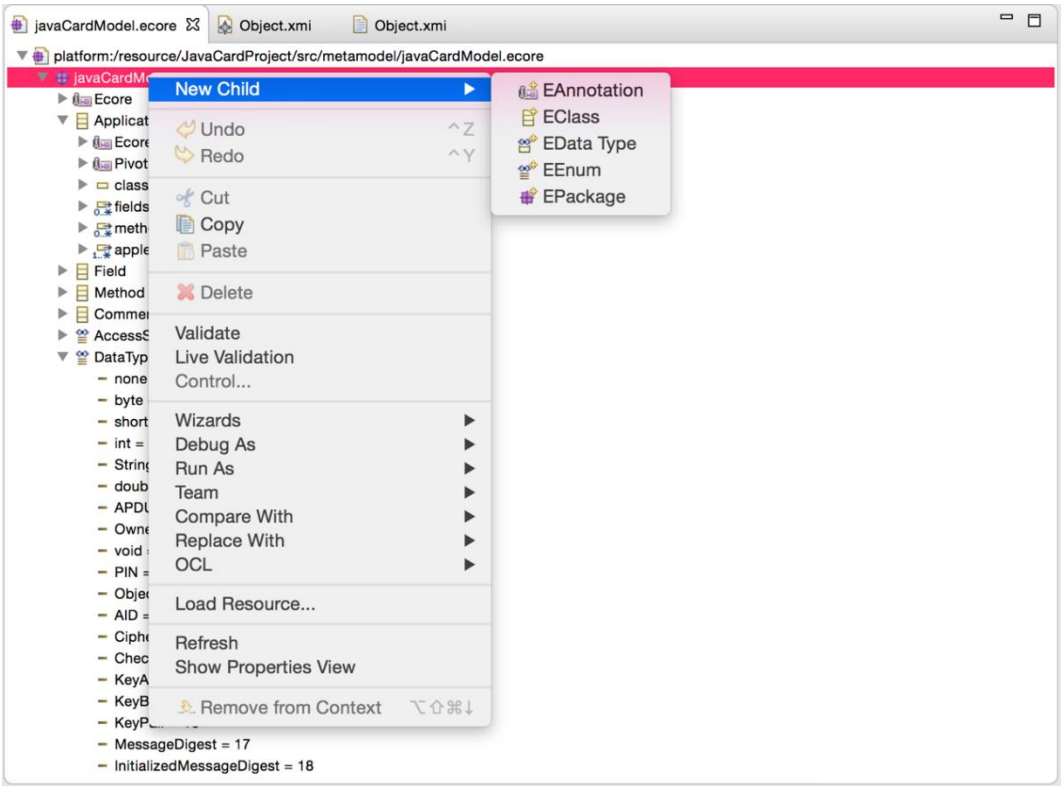
Öte yandan alana özgü dil (ing. "Domain Specific Language") (DSL) belli bir uygulama alanı için içeriğinde sadece o alana özgü özellik ve kavramları barındırararak o alan için uygulama geliştirmeyi kolaylaştıran yazılım geliştirme dilleridir (van Deursen et al., 2000). DSL'ler geliştiricilere sağladığı soyutlama seviyesi, ifade gücü ve kullanım kolaylığı sayesinde günümüzde yaygın olarak kullanılmaktadır. Genellikle bir DSL alan kavramları ve ilişkilerini modelleyen bir üstmodele dayalı bir soyut sözdizim, bu soyut sözdizimin yazılım geliştiriciler tarafından kullanılmasını sağlayan metinsel ya da görsel bir somut sözdizim ve bu sözdizime anlam sağlayan işletimsel ya da belirtimsel bir semantiğin bileşimi şeklinde geliştirilmektedir (Mernik et al., 2005). Bu tezde de Java Card uygulamalarının daha rahat ve etkili geliştirilmesini sağlayan DSL4JavaCard isimli DSL yukarıda belirtilen sözdizim ve semantik bileşenlere sahip olacak şekilde üretilmiştir.

3.3 Eclipse ve Xpand

Model güdümlü yazılım geliştirme sırasında ilk akla gelen bir üst modelin (ing. "meta-model") oluşturulmasıdır. Üst model sayesinde problemlere genel bir bakış açısı ile bakılması sağlanır ve bu seviyede geliştiricilerin yapabilecekleri hataları önlemek amacı ile model içerisine uygun validasyonlar eklenir. Tez

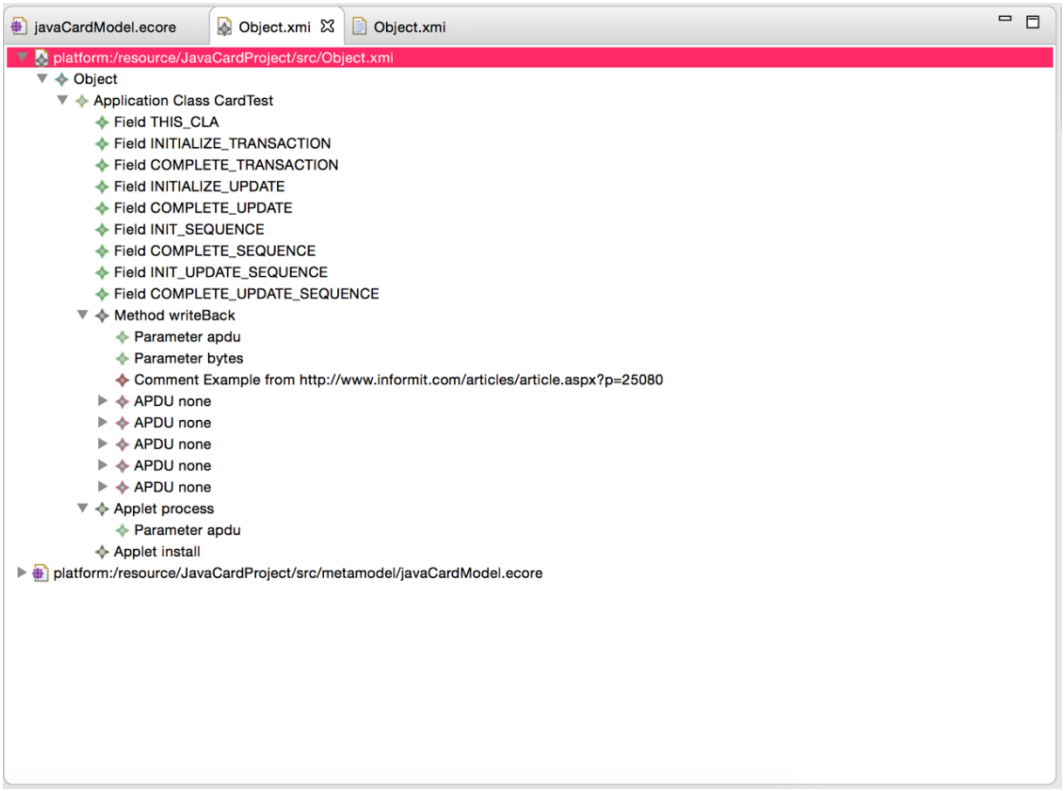
kapsamında DSL4JavaCard'in üst modelinin oluşturulması için EMF ("Eclipse Modelling Framework") kullanılmıştır (EMF Tutorial, 2015).

EMF, Eclipse modelleme çerçevesi olup uygulama geliştirmek için modele dayalı kod üretimini sağlayan üretim tesisidir. EMF'in çekirdeğini Ecore modeller oluşturur. Ecore model, çekirdek model (ing. "core meta-model") olarak geçmektedir. Şekil 3.3'te Ecore model örneği gösterilmiştir.



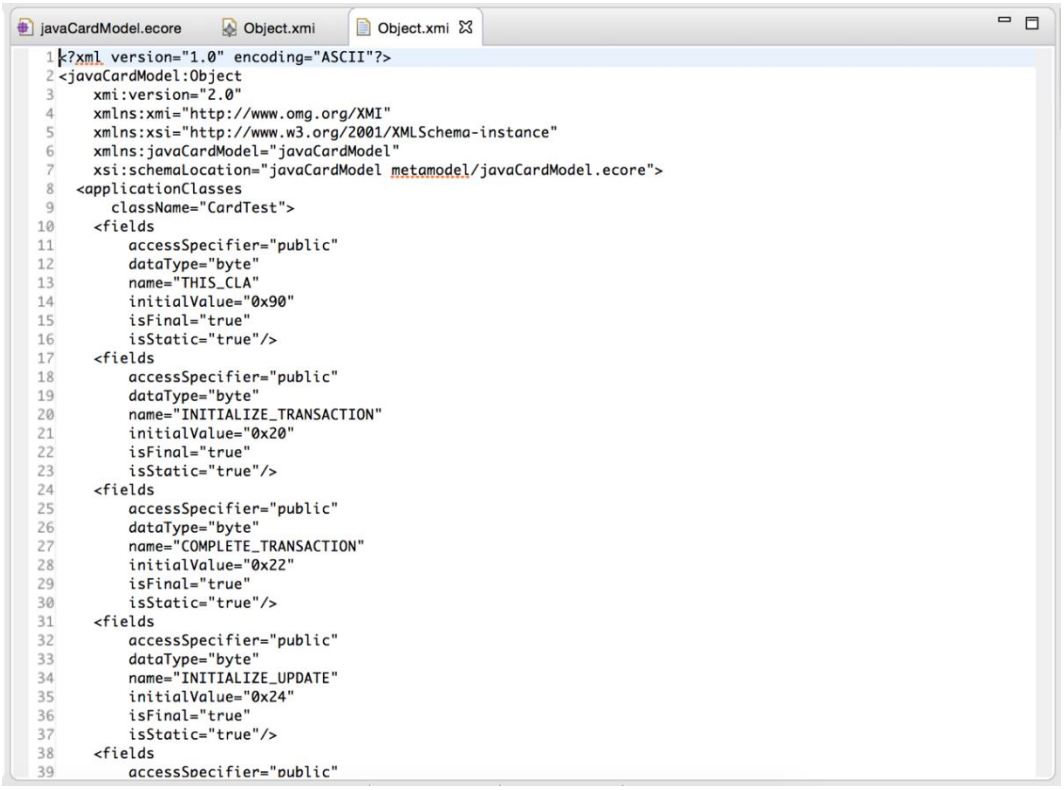
Şekil 3.3. Ecore uzantılı model örneği

Oluşturulan Ecore uzantılı üst modeller aynı zamanda XMI ("XML metadata interchange") formatında da saklanır. Bu format koda dönüşüm esnasında kullanılmak üzere oluşturulur. XMI, OMG'ye ait bir standart dil olup meta bilgi alışverişi için XML aracılığı ile oluşturulur (OMG, 2002). Eclipse üzerinde XMI yapısı Şekil 3.4 ve Şekil 3.5 üzerinde gösterilmiştir.



Şekil 3.4. XMI'ın görsel öğeler ile gösterimi

Yukarıdaki şekilde görüleceği üzere XMI kullanıcının oluşturduğu bir projede şekiller vasıtasıyla üretilebilir. Böylelikle kullanıcı sürükle-bırak ya da kopyala-yapıştır yolu ile kısa sürede istediği örnek modeli oluşturabilir. Bu oluşan modelin yazıya dökülmüş hali ise Şekil 3.5'te verilmiştir. XMI'ın yazıya dökülmüş şekli ile yine aynı şekilde bağlantıları ve birbirleriyle olan ilişkileri mevcuttur. Şekil 3.4'te *CardTest* adındaki ApplicationClass içerisine birden çok *Field* eklendiği gözlemlenir. Yine aynı şekilde bu görüntü Şekil 3.5'te de aynı şekilde görülür. *CardTest* adındaki *ApplicationClass* etiketi (ing. "tag") içerisinde birden çok *fields* etiketi ve mevcut özellikleri gözlemlenmektedir.



Şekil 3.5. XMI'ın metinsel gösterimi

Xpand, EMF tabanlı modellerin kod dönüşümleri için kullanılan ve aslında DSL'lerin EMF tabanlı olarak geliştirilmesini sağlayan bir şablon dilidir (Eclipse Xpand, 2009). Örneğin XMI ile serileştirilmiş bir Ecore modeli Xpand ile yazılan dönüşüm kuralları sonrasında Java kodlarına dönüştürülebilir. Şekil 3.6'da Xpand dili ile yazılmış dönüşüm kodlarına bir örnek gösterilmiştir. Kodlar hakkında daha ayrıntılı bilgi altıncı bölümde anlatılmıştır.

```
    «EXPAND Impl FOREACH fields»
     public static void install(byte[] bArray, short bOffset, byte bLength) {
     //install method automatically generated
       new «className»(bArray,bOffset,bLength);
   }
    protected «className»(byte[] bArray, short bOffset, byte bLength){
    //constructor method automatically generated
    register();
    }
    «EXPAND Meth FOREACH methods»
    «FOREACH applets AS app»
   «IF app.method.toString() != "install".toString()»
   «IF app.method.toString() == "process".toString()»
   public void «app.method»(«FOREACH app.parameters AS par SEPARATOR ','»«par.dataType» «par.name»«ENDFOREACH»){
   byte[] buffer = apdu.getBuffer();
   switch(buffer[ISO7816.OFFSET_INS])
   {//in the process method all functions are called
      «FOREACH methods AS m ITERATOR iter»
       case «iter.counter1» : «m.methodName»(«FOREACH m.parameters AS mpar SEPARATOR ','»«mpar.name»«ENDFOREACH»);
       return;
      «ENDFOREACH»
      default: ISOException.throwIt (ISO7816.SW_INS_NOT_SUPPORTED);
   }}«ELSEIF app.method.toString() == "select".toString()»
   public boolean «app.method»(){//  Perform any applet-specific session initialization.
   return true;}
   «ELSEIF app.method.toString() == "deselect".toString()»
   public void «app.method»(){
   // Perform appropriate cleanup.
   }«ELSE»«ENDIF»«ENDIF»«ENDFOREACH»
}«ENDFILE»
```

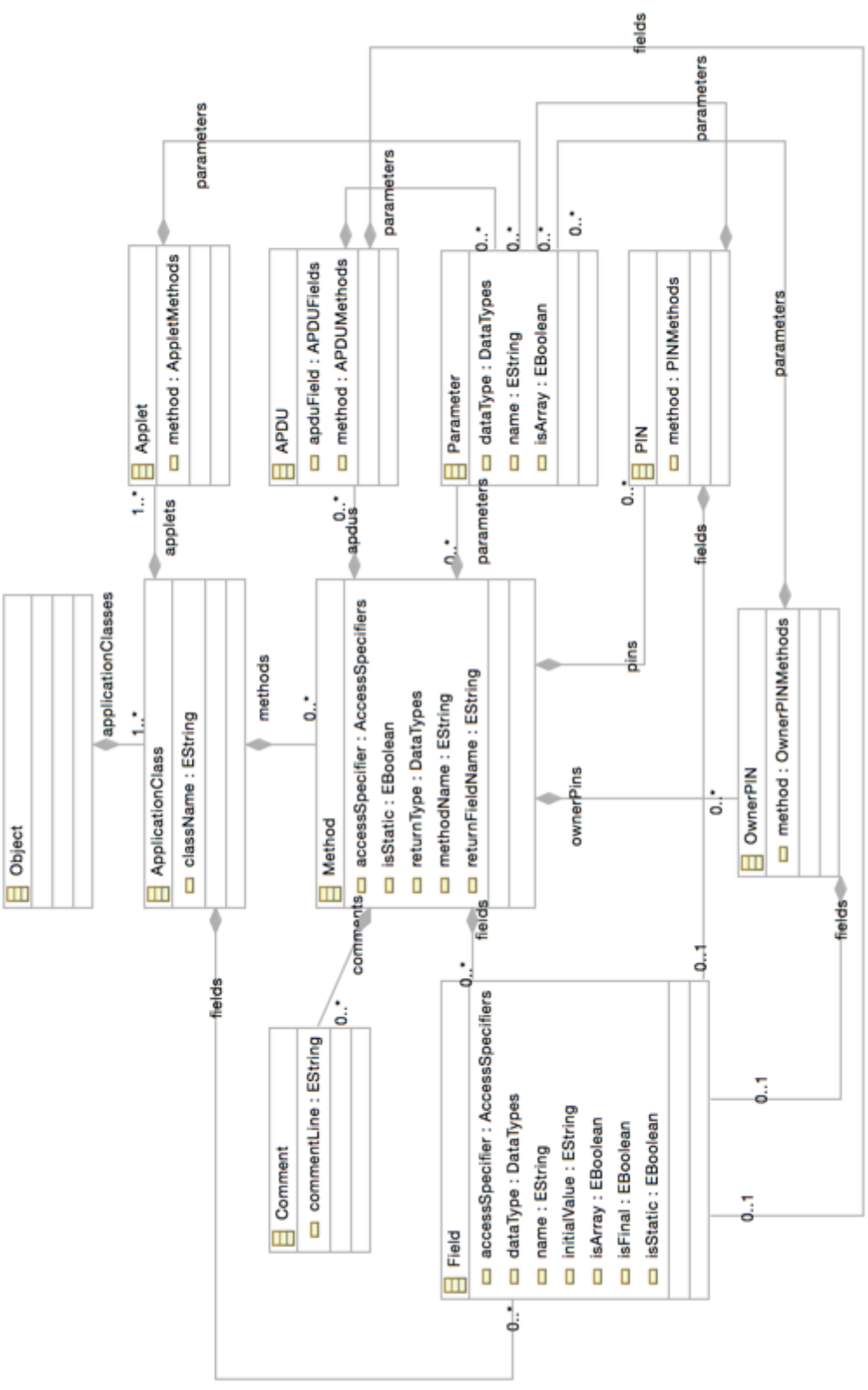
Şekil 3.6. Xpand dili ile kod yazım örneği

4. SOYUT SÖZDİZİMİ

DSL4JavaCard'ın soyut sözdizimi bu tezde geliştirilen bir üstmodele dayanmakta olup, Eclipse Ecore (Steinberg, D., et al., 2008) tabanlı olarak oluşturulmuştur. Bu üstmodel Java Card platformunun kart üzeri (ing."on-card") bileşenlerini ve bunların birbirleri ile ilişkilerini kapsamaktadır. İçeriğinde çok sayıda üstvarlık ve ilişkilerinden oluşan üstmodel 3 adet bakış açısına (ing. "viewpoint") sahiptir ve bu bakış açıları ilerleyen alt bölümlerde tanıtılmaktadır. Buna göre, Java Card Çekirdek bakış açısı (ing "core viewpoint"), Java Card'ın en temel üstvarlıklarını içinde barındırır ve bir sonraki bölümde anlatılmıştır. Onu izleyen bölümlerde İstisna (ing. "Exception viewpoint") ve Güvenlik (ing. "security viewpoint") bakış açısı ile Java Card temel üstvarlıklarına eklenmiş yeni özellikler ele alınmıştır. Böylelikle bu yeni özelliklerden İstisna bakış açısı temel Java Card üstvarlığının hata yakalama özelliğini içinde barındırırken Güvenlik bakış açısı ise Java Card'ın olmazsa olmaz güvenlik model elemanlarını içinde barındırır.

4.1 Çekirdek Bakış Açısı

Çekirdek bakış açısında (ing. "core viewpoint") herhangi bir Java Card uygulamasının geliştirilmesi sırasında ihtiyaç duyulan temel bileşenleri ve bunların ilişkileri yer almaktadır. Şekil 4.1'de DSL4JavaCard üstmodelinin bu bakış açısına ait Ecore diyagramı verilmiştir.



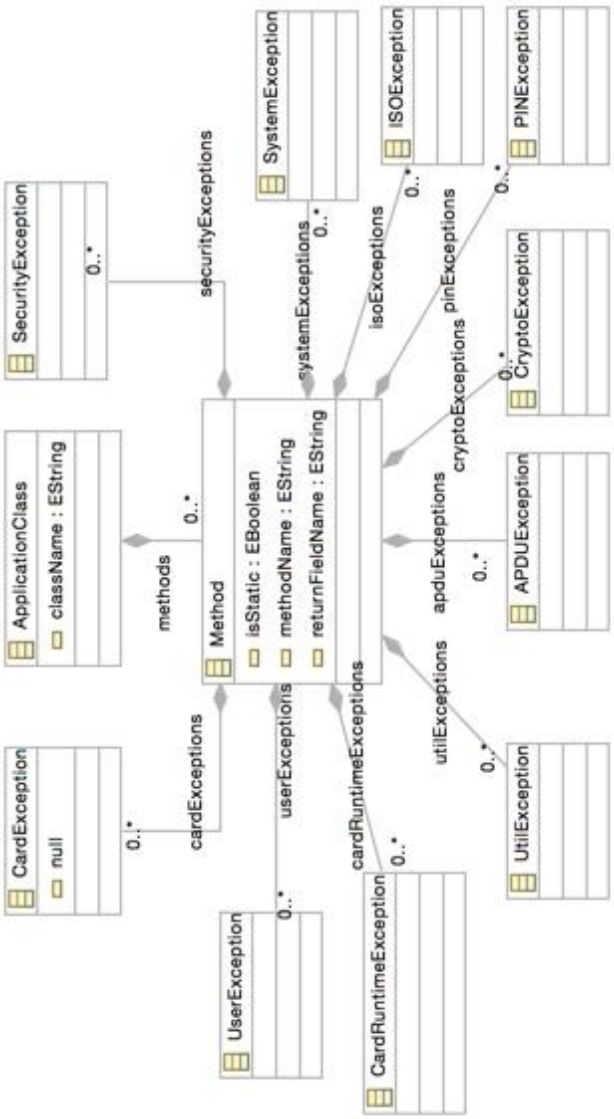
Şekil 4.1. DSL4JavaCard çekirdek bakış açısı üstmodeli

Java uygulaması denilince ilk akla gelen üstmodelin anahtar öğesi *Object* elemanıdır. *Object* elemanı Java Card uygulaması geliştirilirken olmazsa olmaz

ilk model elemanını temsil eder. *Object* elemanından sonra sırası ile oluşturulması gereken diğer model elemanı ise *ApplicationClass*'dır. *ApplicationClass* Java Card uygulamasına özel *Method, Field* ve *Applet* bileşenlerini barındırmaktadır. Bir Java Card uygulamasında temel unsurlar o uygulamaya ait sınıfın oluşturulması ve gerekli ise sabitlerin belirlenmesi ve metotlara ayrıştırılması ile oluşur. Üstmodel içerisinde *ApplicationClass* bunu temsil etmektedir. *Comment, Parameter* ve *Field* üstvarlıkları *Method* üstvarlığına bağlı ve *Method* içerisinde kullanılan bileşenlerdir. Buna göre *Comment* üstvarlığının örnekleri metot içerisinde kullanıcıya not yazabileceği bir alan sağlamaktadır. Bununla birlikte *Parameter* üstvarlığı metotların girdilerini belirtebilmek amacı ile üstmodelde yer almaktadır. Ayrıca metot içerisinde tanımlanmak istenen değişkenler *Field* bileşenleri ile oluşturulmaktadır. Şekil 4.1'de görüleceği üzere *Object, ApplicationClass* ve *Applet* haricinde tüm model elemanlarının ortak noktası *Method* üstmodel elemanıdır. *Method* içerisinde birçok elemanın kullanılmasına izin vererek, somut sözdizimi sırasında kullanıcının olası hatalarını önlemeye ve yanlış kullanımları engellemeye yardımcı olur. Üstmodelin diğer anahtar öğesi ise her bir Java Card uygulamasını temsil eden *Applet* model elemanıdır. *Object* üstmodelinin içinde barındırdığı her bir *Applet*, okuyucu tarafından gelen isteklere uygun cevapları geri döndürür. Uygulama Protokolü Veri Birimi ("Application Protocol Data Unit") (*APDU*) ise ISO / IEC 7816 ile standartları belirlenmiş, akıllı kart ile okuyucu arasındaki iletişimi sağlayan paket yapısını üstmodelde temsil etmektedir. Okuyucu tarafından bir APDU komutu gönderilir ve akıllı kart tarafından bu komuta cevap olarak yeni bir APDU paketi geri gönderilir. *PIN* ve bunu uygulayan *OwnerPIN* üstvarlıkları yine temel Java Card programlarında bulunan ve kart sahibinin kimlik doğrulamasını sağlayan model elemanlarıdır.

4.2 İstisna Bakış Açısı

İstisna bakış açısı (ing. "exception viewpoint") Java Card API içerisinde tanımlanmış ve yaygın olarak kullanılan istisna sınıflarının ve ilişkilerinin DSL4JavaCard bünyesinde modellenmesini sağlamaktadır. Şekil 4.2'de bu bakış açısına ait üstvarlıkları ve ilişkileri içeren kısmi DSL4JavaCard üstmodeli verilmiştir.



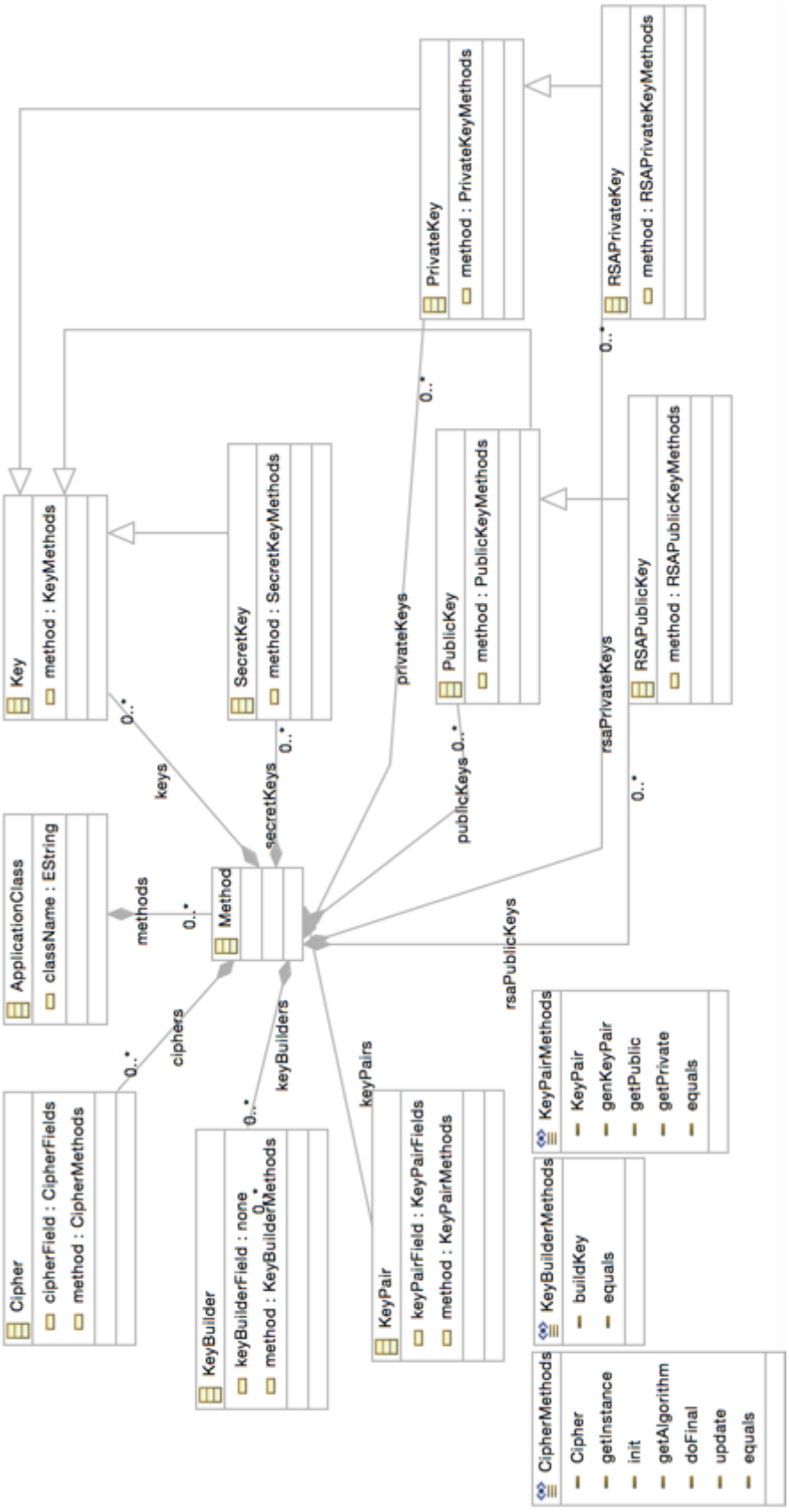
Şekil 4.2. DSL4JavaCard istisna bakış açısı üstmodeli

Exception bileşenleri üstmodel içerisinde *Method* üstvarlığı bünyesinde tanımlanabildiği gibi aynı zamanda metot oluşumu sırasında metodun istisna fırlatmasını da sağlayacak şekilde geliştiriciye sunulmuştur. Bu yapı ile kod oluşturma sırasında farklı alternatifler ile modelde gerçeğe yakınlık sağlanması da amaçlanmıştır. En çok kullanılan istisna çeşitleri *APDUException*, *CardException*, *PINException* ve *ISOException*'dır. *APDUException*, APDU alışverişi sırasında karşılaşılan istisnalar için modelde yer almaktadır. *CardException* ise akıllı kart ile iletişimdeki sorunları modellemektedir. *PINException*, *OwnerPIN* üstmodel elemanının kimlik doğrulama işlemleri sırasında oluşan istisnaların yakalanmasını sağlamaktadır. *ISOException*, yine

APDU alışverişi sırasında karşılaşılan istisnalar için kullanılmaktadır ve ISO 7816-4 cevap durumunu istisna kodu olarak içermektedir. *CryptoException* ise güvenlik bileşenlerinde şifreleme işlemleri sırasında çıkabilecek olan istisna durumları için modelde yer almaktadır. *SystemException*, Applet üstmodeli ile birlikte okuyucu tarafından gelen isteklere cevap verirken olabilecek istisna durumlarında kullanılmaktadır. *UserException* ise kullanıcı kaynaklı hata durumlarında çıkabilecek olan istisna durumları için eklenmiş bir model elemanıdır. Son olarak *SecurityException*, başka bir uygulama tarafından Java Card uygulamasına ait bir nesneye yasa dışı erişim yapılmak istendiği zaman oluşan istisnaların yakalanmasını sağlamaktadır.

4.3 Güvenlik Bakış Açısı

Temel akıllı kart modelinin oluşturulmasına ek olarak DSL4JavaCard sözdizimi kart üzeri şifreleme ve güvenlik fonksiyonlarının sağlanması amacıyla kullanılabilecek üstvarlıkları ve bunların ilişkilerini de içermektedir. Söz konusu güvenlik bakış açısına (ing. “security viewpoint”) ait üstvarlıkları ve ilişkileri içeren kısmi üstmodel Şekil 4.3’te verilmiştir.

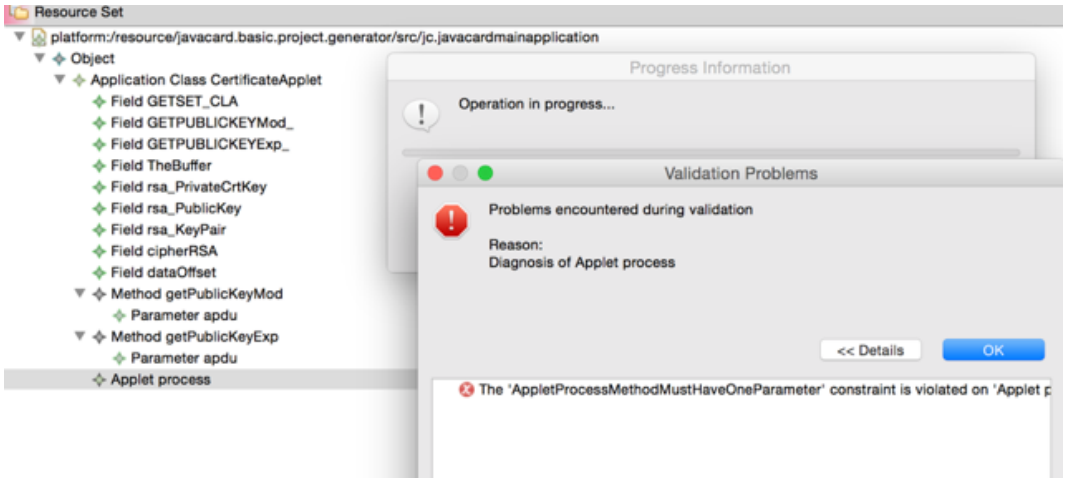


Şekil 4.3. DSL4JavaCard istisna bakış açısı üstmodeli

Java Card güvenlik bileşenleri, kart yazılımına ait metotlar içerisinde tanımlanacak şekilde üstmodelde yer almaktadır. Bu işlem yine *ApplicationClass* altında oluşturulan *method* model elemanı içerisinde tanımlayacağımız güvenlik elemanları ile oluşturulur. Örneğin *KeyPair* model elemanı anahtar çiftlerini oluşturmak için bir kapı görevi üstlenmiştir. İçerisindeki *getPublic* ve *getPrivate* metotları *PublicKey* ve *PrivateKey* döndürmektedir. *Cipher,* şifreleme (ing. "encryption") ve şifre çözme (ing. "decryption") gibi temel şifreleme işlemleri için kullanılan model elemanıdır. Birçok şifremeleme algoritmasını içerisinde bulundurur. *KeyBuilder* ise model içerisinde anahtarların oluşmasını sağlar. Şekil 4.3'te görülen anahtar model elemanlarının (*Key, SecretKey, PublicKey, PrivateKey, RSAPublicKey, RSAPrivateKey*) oluşturulması *KeyBuilder*'ın *buildKey* servisinin çağrımı üzerinden sağlanmaktadır.

5. SOMUT SÖZDİZİM

4. bölüm ve alt bölümlerde anlatılan DSL4JavaCard üstmodeline dayalı olarak akıllı kart yazılım modellerinin oluşturulabilmesi amacıyla geliştiricilere bir somut sözdizim sunulmuştur. Geliştirme aşamasında kodların doğru oluşabilmesi yani modelleme sırasında Java Card model kısıtlarının kontrol edilmesi ve dilin statik semantiğinin sağlanması için Eclipse EMF modelleri ile çalışabilen XPand'e (XPAND, 2015) dayalı bir modelleme ortamı geliştiricilere sunulmuştur. Geliştiricilerin hata yapmaması için kısıtların kontrolleri Nesne Kısıt Dili ("Object Constraint Language") (OCL) ile geliştirilmiştir. Örneğin Şekil 5.1'de geliştirilen Java Card modelinin DSL4JavaCard kurallarına göre doğrulanması süreci sırasında oluşan bir hatanın (çağrılan metodun tanımlanması için eksik parametre girişi) geliştiriciye gösterildiği ekran görüntüsü verilmiştir.



Şekil-5.1. Oluşturulan bir JavaCard modelinin DSL4JavaCard kurallarına göre doğrulanması

Sonrasında doğru kod üretiminin gerçekleştirilmesi amacıyla modelleme sırasında (bir başka deyişle DSL4JavaCard somut sözdizimi kullanıldığı sırada) mevcut statik semantik kontrollerin sağlanması amacıyla yazılmış olan kurallardan bir örnek Şekil 5.2'de verilmiştir. Örnekte bir Java Card geliştiricisi ApplicationClass örneğine bir isim vermediğinde ilk sıradaki kurala göre geliştiriciye hata verilecektir. Bir sonrakinde ise ApplicationClass içerisinde Applet model elemanına bağlı olan *process* ve *install* metotlarının çağırılması

gerekliliği kontrol edilmektedir. Çünkü her Java Card uygulamasında bu çağrımlara ihtiyaç vardır.

```
class ApplicationClass
{
    attribute className : String[?];
    property fields : Field[*] { ordered composes };
    property methods : Method[*] { ordered composes };
    property applets : Applet[+] { ordered composes };
    invariant ApplicationNameSizeMustGreaterThenZero('ApplicatioClass Must Have Name To Identify The Class'):
    className.size()>0;
    invariant WarningAplicationClassFieldsCanUseWithAccessSpecifier('Warning : You can use fields with Access Specifier'):
    self.fields->forAll(f | f.accessSpecifier <> AccessSpecifiers::none);
    invariant ApplicationClassMustHaveAppletProcessMethods('All Java Card Application Uses Applet process Method'):
    self.applets->size()>0 and self.applets->forAll(e | e.method <> AppletMethods::process);
    invariant ApplicationClassMustHaveAppletProcessMethods('All Java Card Application Uses Applet install Method'):
    self.applets->size()>0 and self.applets->forAll(e | e.method <> AppletMethods::install);
}
```

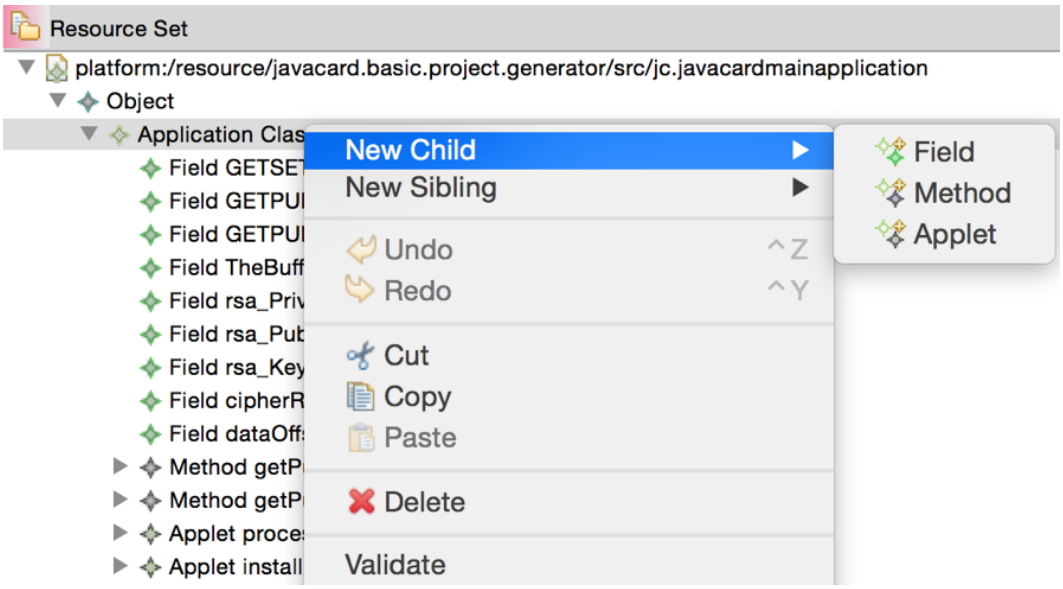
Şekil-5.2. Bir statik semantik kontrolü örneği

Başka bir örnek ise Şekil 5.3'te verilmiştir. Bu örnek içerisinde üstmodele ait *OwnerPIN* içerisinde *check* metodu kullanılırsa beraberinde gerekli olan parametre ve dönüş tipi olan boolean'ı hata mesajı olarak gönderir. Kullanıcı gerekli olan parametre ve dönüş tipi boolean olan bir field tanımladıktan sonra bu hata mesajı bir daha alınmayacaktır. Bir sonrakinde *getTriesRemaining* metodu kullanılması durumunda dönüş değeri için *byte* değeri istemektedir. 3. OCL kuralında ise *update* metodu için gerekli olan 3 parametre ihtiyacını kullanıcıya söylemektedir. Son iki örnekte de yukarıdaki anlatımlara benzer kurallar mevcut olmakla birlikte parametre ve dönüş tiplerini geliştiriciye hatırlatarak olası hataları önlenmesi amaçlanmıştır.

```
class OwnerPIN
{
    property fields : Field[?] { composes };
    attribute method : OwnerPINMethods[?];
    property parameters : Parameter[*] { ordered composes };
    property instanceVariable : InstanceVariable[?] { composes };
    invariant
    OwnerPINCheckMethodsMustHaveThreeParameterAndItHasBooleanReturnTypeSoAddOneBooleanField('check method must need three p
    self.method=OwnerPINMethods::check implies
    self.parameters->size()=3 and self.fields->size()=1;
    invariant
    OwnerPINgetTriesRemainingMethodHasNeedByteFiedl('getTriesRemaining Method has need one byte field for retrun value'):
    self.method=OwnerPINMethods::getTriesRemaining implies
    self.fields->size()=1 and self.fields->forAll(e | e.dataType <> DataTypes::byte);
    invariant
    OwnerPINUpdateMethodsMustHaveThreeParameter('update must have three Byte Array, short and byte parameters'):
    self.method=OwnerPINMethods::update implies
    self.parameters->size()=3;
    invariant OwnerPINSetValidatedFlagMethodsMustHaveOneParameter('setValidatedFlag must have boolean parameter'):
    self.method=OwnerPINMethods::setValidatedFlag implies
    self.parameters->size()=1;
    invariant
    OwnerPINisValidatedMethodHasNeedByteFiedl('isValidated Method has need one boolean field for retrun value'):
    self.method=OwnerPINMethods::isValidated implies
    self.fields->size()=1 and self.fields->forAll(e | e.dataType <> DataTypes::boolean);
}
enum OwnerPINMethods { serializable }
```

Şekil-5.3. Statik semantik kontrol örneği 2

Şekil 5.4'te ise geliştiricilere sunulan görsel modelleme ortamına ait bir ekran görüntüsü yer almaktadır. DSL4JavaCard'ın sözdizimine uygun bir şekilde bu modelleme ortamında kart yazılımına ait model oluşturulabilir. Modellemenin büyük bir kısmı model elemanlarına sağ tıklamak ve gelen menülerden o elemana ait içerebileceği yeni model elemanlarını oluşturmak ve/veya özellik değerlerini belirlemekten ibarettir. Böylelikle geliştiricinin özellikle kapsülleme veya DSL4JavaCard üstmodelindeki varlık ilişkilerine uymayan model kurgularına izin verilmemektedir. Yine modelleme ortamının bir diğer avantajı ise geliştiriciler kes-kopyala ya da oluşturulanı sürükle bırak yöntemi ile daha önceden oluşturulmuş olan model bileşenlerini istedikleri gibi taşıyabilirler. Ama taşıma, kopyalama ya da yapıştırma sırasında yine DSL4JavaCard model doğrulama kuralları otomatik olarak kontrol edilmektedir. Şekil 5.4'te de bunu gösteren bir oluşum bulunmaktadır. Geliştirici *ApplicationClass* içerisine ekleyebileceği model elemanlarını görür ve istediğini seçerek ekleme yapabilir. Ancak bu 3 farklı model elemanı dışında herhangi bir eklenti yapamamaktadır. Aynı şekilde sürükle-bırak ya da kes-yapıştır yöntemi de işe yaramaz. Burada ilgili kurallar devreye girer ve geliştiricinin hata yapması önlenir.



Şekil-5.4. DSL4JavaCard modelleme ortamına ait bir ekran görüntüsü

6. İŞLETİMSEL SEMANTİK

DSL4JavaCard sözdizimi ile oluşturulmuş modellerin Java Card platformunda işletilebilmesi için modellerin akıllı kartlar içerisine yüklenecek Java dosyalarına dönüştürülmesi gerekmektedir. Java dosyasına dönüşümlerin olabilmesi için Xpand dili (XPAND, 2009) kullanılarak modelden metne dönüşüm kuralları hazırlanmıştır. EMF tabanlı modellerden kod üretimini sağlayan Xpand dili ve Ecore'a dayalı DSL4JavaCard modeline dönüşüm kuralları eklenerek Java kodları oluşabilmektedir. Hazırlanan dönüşüm kurallarına bir örnek ise Şekil 6.1'de verilmiştir. DSL4JavaCard modeli üzerinde bu kurallar işletildiğinde her bir model elemanına karşılık uygun Java kodları ilgili *sınıf* (ing. "class"), *metot* (ing. "method") ve *alanları* (ing. "field") içerecek şekilde çıktı dosyasına yazılmış olur. Şekil 6.1'de gösterilen örnek kodlarda mavi renkli kısımlar kurala göre dosyaya yazılması gereken Java kodlarını gösterirken geriye kalan kodlar ise Xpand sözdimizi ile yazılmış dönüşüm kurallarını belirtmektedir.

Şekil 6.1'deki kurallar incelendiğinde Java Card Applet örneğinin girdi modeli üzerinde gezinme sonucunda bulunmasının ve modelde kurgulanmış ilişkilere göre de özellik ve metotlarının belirlenerek bunlara uygun Java Card kodlarının çıktı Java dosyasına yazılmasının sağlandığı görülmektedir. Dönüşüm kuralları şablon kod oluşturmanın ötesinde bir takım Java Card metodunun gövdesinin neredeyse tamamını üretebilmektedir. Örneğin mavi satırda Java Card uygulamalarında olmazsa olmaz *install* metodu otomatik oluşacak şekilde hazırlanmıştır. Bir altında ise *yapıcı metot* olan (ing. "constructor") ve ismi geliştiriciden gelmesi beklenen *ApplicationClass* ismine göre oluşturulacak şekilde hazırlanmıştır. Bir diğer örnek ise gelen APDU paketindeki komuta göre çalışması gereken metoda uygun diğer parametreler ile yönlendirmeyi sağlayan akıllı kart *process* metoduna ait *switch-case* bloğunun tamamı modelden koda dönüşümler sonucunda üretilebilmektedir.

```
«FILE className + ".java"»
public class «className» extends Applet {
«FOREACH methods AS m ITERATOR metIt»
     «FOREACH m.keyPairs AS k ITERATOR keyIt»
         «IF keyIt.counter1 == 1»
             «LET "KeyPair _" + m.methodName + k.method + ";" AS keyPairVar»
                «keyPairVar»«ENDLET»«ENDIF»«ENDFOREACH»«ENDFOREACH»
  «EXPAND Impl FOREACH fields»
   public static void install(byte[] bArray, short bOffset, byte bLength) {
   //install method automatically generated
     new «className»(bArray,bOffset,bLength);
 }
   protected «className»(byte[] bArray, short bOffset, byte bLength){
   //constructor method automatically generated
   register();
   }
   «EXPAND Meth FOREACH methods»
   «FOREACH applets AS app»
 «IF app.method.toString() != "install".toString()»
 «IF app.method.toString() == "process".toString()»
 public void «app.method»(«FOREACH app.parameters AS par SEPARATOR ','»«par.dataType» «par.name»«ENDFOREACH»){
 byte[] buffer = apdu.getBuffer();
 switch(buffer[ISO7816.OFFSET_INS])
 {//in the process method all functions are called
    «FOREACH methods AS m ITERATOR iter»
     case «iter.counter1» : «m.methodName»(«FOREACH m.parameters AS mpar SEPARATOR ','»«mpar.name»«ENDFOREACH»);
     return;
    «ENDFOREACH»
    default: ISOException.throwIt (ISO7816.SW_INS_NOT_SUPPORTED);
 }}«ELSEIF app.method.toString() == "select".toString()»
 public boolean «app.method»(){//  Perform any applet-specific session initialization.
 return true;}
 «ELSEIF app.method.toString() == "deselect".toString()»
 public void «app.method»(){
 // Perform appropriate cleanup.
 }«ELSE»«ENDIF»«ENDIF»«ENDFOREACH»
}«ENDFILE»
```

Şekil-6.1. DSL4JavaCard modellerinden JavaCard dosyalarını otomatik oluşturan dönüşüm kurallarından bir örnek

Bir diğer örnek ise Şekil 6.2'de verilmiştir. Örnekte mavi ile belirtilen alanların az olması geliştiricinin eklemiş olduğu isim ve özelliklere göre dizayn edileceğini anlatır. Bu safhadan sonra oluşan tüm kodlar geliştiricinin seçimlerine ve kuralların sıralamasına göre gelişmektedir. Öncelikle *JavaCardModel* içerisinde *Method*'lar aranmaktadır. *Method* bulunduktan sonra *AccessSpecifier* ile metotlara ait *Public, Private, Protected* gibi erişim belirteçleri geliştiricinin seçimine göre eklenir. Sonrasında metoda ait dönüş tipi yine geliştiricinin seçimine göre eklenecektir. Seçim sonrasında ise dönüş tipi için özel bir ad beklenir böylelikle metot sonunda "*return xxx*" (xxx örnek olarak eklenmiştir ve xxx bir değişken adını temsil etmektedir) olarak eklenmesi sağlanacaktır. Sonrasında ise metoda ait parametreler ve tipleri eklenir. Süslü parantezden bir önceki yapı ise hata durumunda yakalaması için metotlara entegre edilen istisna tipleridir. Eğer geliştirici istisna özelliğini metot içerisinde kullanır ise *"throws ExceptionType"* şeklinde oluşması sağlanır.

```
«DEFINE Meth FOR javaCardModel::Method»
«accessSpecifier» «IF isStatic»static«ELSE»«ENDIF»
«IF returnType.toString()!="none".toString()»«returnType»«ELSE»«ENDIF»
«methodName»(«FOREACH parameters AS mpar SEPARATOR ','»
«IF mpar.isArray»«mpar.dataType»[] «mpar.name»«ELSE»«mpar.dataType» «mpar.name»«ENDIF»«ENDFOREACH»)
«IF !exceptions.isEmpty»throws «FOREACH exceptions AS mex SEPARATOR ','»«mex.exceptionType»«ENDFOREACH»«ELSE»«ENDIF»
{
    «EXPAND field FOREACH fields»
    «EXPAND comment FOREACH comments»
    «EXPAND keyp(methodName) FOREACH keyPairs»
    «EXPAND cipher(methodName) FOREACH ciphers»
    «EXPAND apdu(methodName) FOREACH apdus»
}
«ENDDEFINE»
```

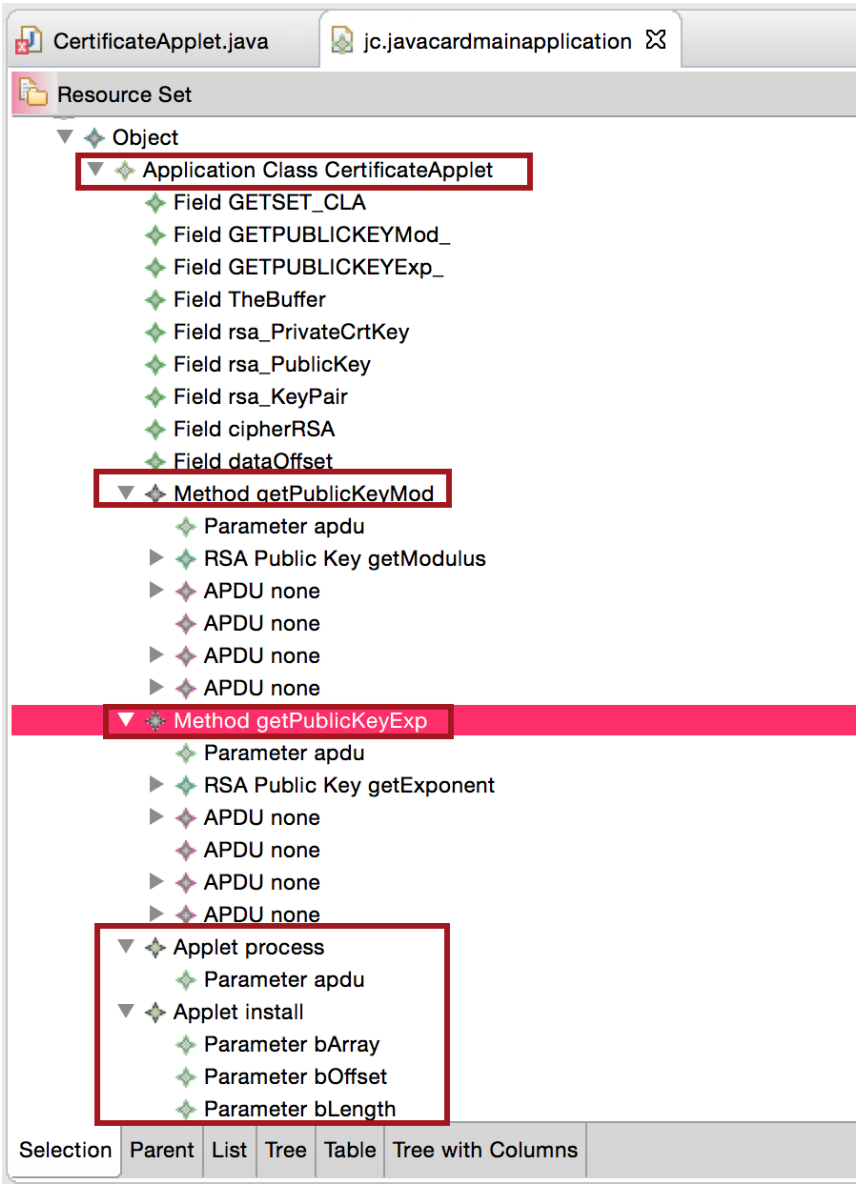
Şekil-6.2. Java metotlarını oluşturan DSL4JavaCard dönüşüm kuralları

7. DURUM ÇALIŞMASI

Bu bölümde Java Card uygulamalarının DSL4JavaCard kullanılarak nasıl modelleneceği ve modelden koda dönüşümlerin nasıl gerçekleştiği iki uygulama üzerinden örneklenmiştir ve başarımları değerlendirilmiştir. Değerlendirmenin gerçekçi olması bakımından seçilen örneklerde herkes tarafından tam sürüm kodlarına erişilebilinir Java Card uygulamaları temel alınmıştır ve DSL4JavaCard'a dayalı yazılım geliştirme çıktısı ile bu tam sürüm kodlar kıyaslanmaktadır.

7.1 Sertifika Uygulaması ("CertificateApplet")

DSL4JavaCard'ın Java Card uygulamalarını geliştirmede kullanılmasını değerlendirmek amacıyla yürütülen ilk durum çalışması bir sertifika uygulamasıdır. Çalışma uygulamalarda şifreleme amacıyla kullanılabilecek bir anahtar ikilisini akıllı kart içerisinde oluşturan ve bu anahtar çiftinden genel anahtar yapısında olanını (ing. "public key") akıllı kart ile ana bilgisayar (ing. "host") arasındaki bağlantı üzerinden güvenli bir şekilde istenildiğinde ana bilgisayara yollayabilen bir Java Card uygulamasının DSL4JavaCard kullanılarak geliştirilmesini kapsamaktadır. *CertificateApplet* adlı bu uygulamaya ait yazılım modelinin DSL4JavaCard sözdizimi kullanılarak oluşturulması Şekil 7.1'de gösterilmiştir. Oluşturulan modelden dilin işletimsel semantiği kullanılarak otomatik olarak üretilen Java Card kodlarından bir parçası ise Şekil 7.2'de gösterilmektedir. Uygulama için gerekli Java Card şifreleme ve güvenlik bileşenlerinin DSL4JavaCard sözdizimi ile modellenebilidiği görülmektedir. Yine model içerisinde *Applet* bileşenine ait *process* ve *install* metotlarının yanı sıra geliştiricinin kendi oluşturabildiği *getPublicKeyExp*, vb. metotlar da yer almaktadır. Oluşturulan kodlar içerisinde de *PublicKey* ve ona ait metot çağırımlarının üretilebildiği; bunun yanı sıra, *APDU* kullanımına ait metotların da modele bağlı olarak üretildiği gözlenmektedir. Geliştirilen model üzerinde gezinme ve otomatik kod üretimi Intel Core i5 2,6 GHz, 8GB RAM ve 256 GB SSD sabit diske sahip bir masaüstü bilgisayarda 1 sn.'den daha kısa sürede tamamlanmıştır.



Şekil-7.1. *CertificateApplet* uygulamasına ait DSL4JavaCard modeli

```
import javacard.framework.*;
import javacard.security.*;
import javacardx.crypto.*;
public class CertificateApplet extends Applet {

    RSAPublicKey _getPublicKeyModRSAPublicKey;//You may delete or rename to use this variable in your methods
    APDU _getPublicKeyModAPDU;//You may delete or rename to use this variable in your methods

    RSAPublicKey _getPublicKeyExpRSAPublicKey;//You may delete or rename to use this variable in your methods
    APDU _getPublicKeyExpAPDU;//You may delete or rename to use this variable in your methods

    static final byte GETSET_CLA = (byte) 0x85;
    static final byte GETPUBLICKEYMod_ = (byte) 0x20;
    static final byte GETPUBLICKEYExp_ = (byte) 0x30;
    byte TheBuffer[];
    RSAPrivateCrtKey rsa_PrivateCrtKey;
    RSAPublicKey rsa_PublicKey;
    KeyPair rsa_KeyPair;
    Cipher cipherRSA;
    final short dataOffset = (short) ISO7816.OFFSET_CDATA;
    public static void install(byte[] bArray, short bOffset, byte bLength) {
        //install method automatically generated
        new CertificateApplet(bArray, bOffset, bLength);
    }
    protected CertificateApplet(byte[] bArray, short bOffset, byte bLength) {
        //constructor method automatically generated
        register();
    }

    private void getPublicKeyMod(APDU apdu) {

        byte buffer[] = _getPublicKeyModAPDU.getBuffer();

        _getPublicKeyModAPDU.setOutgoing();

        _getPublicKeyModAPDU.setOutgoingLength((short) 128);

        _getPublicKeyModAPDU.sendBytesLong(buffer, ISO7816.OFFSET_CDATA,
                (short) 128);

        _getPublicKeyModRSAPublicKey.getModulus(buffer, ISO7816.OFFSET_CDATA);

    }
    private void getPublicKeyExp(APDU apdu) {

        byte buffer[] = _getPublicKeyExpAPDU.getBuffer();

        _getPublicKeyExpAPDU.setOutgoing();

        _getPublicKeyExpAPDU.setOutgoingLength((short) 4);

        _getPublicKeyExpAPDU.sendBytesLong(buffer,
                (short) ISO7816.OFFSET_CDATA, (short) 4);

        _getPublicKeyExpRSAPublicKey.getExponent(buffer, ISO7816.OFFSET_CDATA);

    }

    public void process(APDU apdu) {
        byte[] buffer = apdu.getBuffer();
        switch (buffer[ISO7816.OFFSET_INS]) {//in the process method all functions are called

            case 1 :
                getPublicKeyMod(apdu);
                return;

            case 2 :
                getPublicKeyExp(apdu);
                return;

            default :
                ISOException.throwIt(ISO7816.SW_INS_NOT_SUPPORTED);
        }
```

Şekil-7.2. *CertificateApplet* uygulamasına ait modelden otomatik üretilen JavaCard kodları

DSL4JavaCard'ın söz konusu uygulamanın geliştirilmesindeki üretkenliğini değerlendirmek amacıyla durum çalışması sonucunda sadece otomatik kod dönüşümü ile elde edilen ve üzerinde henüz herhangi bir ekleme, çıkarma, vb. değişiklik yapılmamış Java Card kodları ile aynı uygulamanın (CertificateApplet) (https://community.oracle.com/thread/2195878) adresinde yer alan ve tam sürüm kaynak koduna herkesin erişebildiği Java Card kodlarının bir karşılaştırması gerçekleştirilmiştir. Belirtilen adresteki kodlar, ilgili uygulama için gerekli tüm kodlardır ve doğrudan derlenip akıllı kart üzerinde çalıştırılabilecek özelliktedirler. Kod satır sayısı ("Line of Code) (LOC) bazında karşılaştırıldığında

DSL4JavaCard kullanılarak otomatik üretilen kodların tam sürüm uygulamanın %77'sini oluşturduğu; bir başka deyişle tüm programın %77'sinin sadece DSL4JavaCard modellemesi ile elde edilebildiği görülmüştür. LOC ölçümünün yanı sıra üretilen kodun etkinliğini gözlemlemek amacıyla otomatik kod üretimi sonrasında akıllı kart uygulamasının hangi kısımlarının tam olarak üretilebildiği veya üretilemediği incelenmiştir. İnceleme sonucu Tablo 7.1'de verilmiştir. Buna göre temel Java Card bileşenlerinin tamamının, kullanıcı tanımlı ve iş uygulamasına özel öğelerin ise önemli bir kısmının DSL4JavaCard kullanılarak otomatik bir şekilde üretilebildiği görülmektedir.

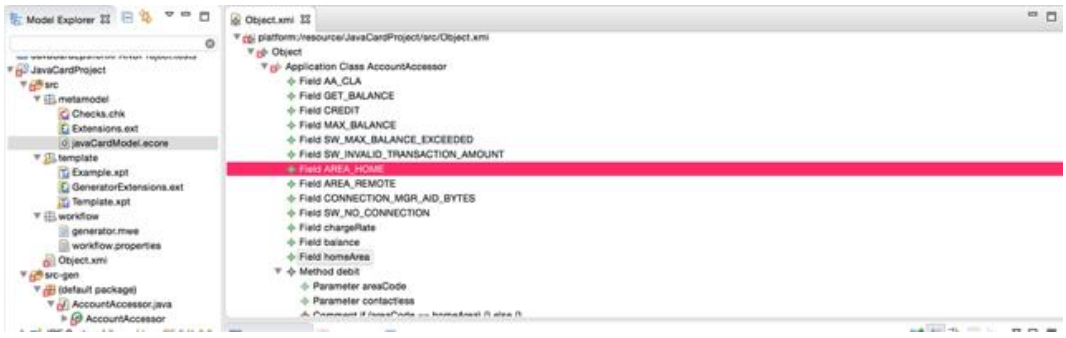
Tablo 7.1. DSL4JavaCard kullanılarak Java Card kod bileşenlerinin üretilmesine ait değerlendirme sonuçları

JAVA CARD KOD BİLEŞENLERİ	OTOMATİK KOD ÜRETİMİNİN DEĞERLENDİRİLMESİ
İşleme (ing. "process"), yükleme (ing. "install") gibi temel Java Card applet metotları	Tamamı üretilebiliyor.
Kullanıcı tanımlı ve uygulamaya özel metotlar	Metot tanımlarının tamamı üretilebiliyor. Ancak kod üretimi sonrasında gövdelerinde önemli ölçüde eklentiye ihtiyaç duyuluyor.
Sabit veri tipleri	Tamamı üretilebiliyor.
Sınıf özellikleri ve metot değişkenleri	Tamamı üretilebiliyor.
İstisna nesneleri	Tamamı üretilebiliyor. Seçenekler ile istisna nesneleri farklı alternatiflerle oluşturulabiliyor. 20 farklı istisna sınıfını destekliyor.
Anahtarlama, şifreleme ve yetkilendirme gibi güvenlik bileşenleri	Tüm güvenlik metotları ve özellik tanımları otomatik üretilebiliyor. Ancak metot içerisinde kod üretimi sonrasında eklentilere ihtiyaç duyuluyor.
Açıklama alanları	Tamamı üretilebiliyor.

7.2 Hesap Uygulaması ("AccountAccessor")

DSL4JavaCard ile yapılan bir diğer uygulama ise *AccountAccessor* uygulamasıdır. Uygulamada ağ hizmeti sağlayan bir şirketin hesap bilgilerini izleyen ve hesaba ait yeterli kredi miktarının olup olmadığını kontrol eden akıllı kart yapısı mevcuttur. Bunun için öncelikle hesap bilgilerini ayarlayan ve sonrasında kredi hesabı ile kalan bakiyeyi bulan metotlara ihtiyaç duymaktadır.

Şekil 7.3'de DSL4JavaCard sözdizimi kullanılarak oluşturulan ilgili uygulamaya ait yazılım modeli ve Şekil 7.4'te otomatik kod dönüşümü sonrası bu modelden elde edilen söz konusu uygulamanın Java Card üzerinde çalıştırılabilir Java kodlarının bir kısmı gösterilmiştir. Yine kod üretimi performansı zamansal olarak değerlendirildiğinde model üzerinde gezinme ve otomatik kod üretiminin 1 sn.'den daha kısa sürede tamamlandığı tespit edilmiştir.



Şekil-7.3. *AccountAccessor* uygulamasına ait DSL4JavaCard modeli

```
public class AccountAccessor extends Applet {

        APDU _creditAPDU;//You may delete or rename to use this variable in your methods
        static final byte AA_CLA = (byte) 0x80;
        static final byte GET_BALANCE = (byte) 0x10;
        static final byte CREDIT = (byte) 0x20;
        static final short MAX_BALANCE = (short) 0x7FFF;
        static final short SW_MAX_BALANCE_EXCEEDED = (short) 0x6A54;
        static final short SW_INVALID_TRANSACTION_AMOUNT = (short) 0x6A55;
        static final byte AREA_HOME = (byte) 0;
        static final byte AREA_REMOTE = (byte) 1;
        static final byte CONNECTION_MGR_AID_BYTES[] = {(byte) 0xA0, 0, 0, 0,
                (byte) 0x62, 0x03, 0x01, 0x0C, 0x0B, 0x2};
        static final short SW_NO_CONNECTION = (short) 0x6905;
        private short chargeRate[];
        private short balance;
        private short homeArea;
        public static void install(byte[] bArray, short bOffset, byte bLength) {
                //install method automatically generated
                new AccountAccessor(bArray, bOffset, bLength);
        }
        protected AccountAccessor(byte[] bArray, short bOffset, byte bLength) {
                //constructor method automatically generated
                register();
        }
        private boolean debit(short areaCode, boolean contactless) {
                short amtToDebit;
                amtToDebit = chargeRate[AREA_HOME];
                amtToDebit = chargeRate[AREA_REMOTE];
                balance = (short) (balance - amtToDebit);
                //if (areaCode == homeArea) {} else {}
                //if (balance >= amtToDebit) {}

                return false;
        }
        private void credit(APDU apdu) {

                byte buffer[] = _creditAPDU.getBuffer();//this method need byte Array Field
                short creditAmount = (short) ((short) (buffer[ISO7816.OFFSET_CDATA] << (short) 8) | (buffer[ISO7816.OFFSET_CDATA + 1]));
                balance = (short) (balance + creditAmount);
                byte byteRead = (byte) _creditAPDU.setIncomingAndReceive();//this method need short Field
                byte numBytes = (byte) (buffer[ISO7816.OFFSET_CLA]); // You can change this usage
                ISOException.throwIt(ISO7816.SW_WRONG_LENGTH);
        }
```

Şekil-7.4. *AccountAccessor* uygulamasına ait DSL4JavaCard modelinden oluşturulan kodların bir kısmı

DSL4JavaCard'ın söz konusu uygulamanın geliştirilmesindeki üretkenliğini tekrar değerlendirmek amacıyla durum çalışması sonucunda sadece otomatik kod dönüşümü ile elde edilen ve üzerinde henüz herhangi bir ekleme, çıkarma, vb.

değişiklik yapılmamış Java Card kodları ile aynı uygulamanın (*AccountAccessor*) (https://kenai.com/projects/javacard/sources/subversion/content/samples/classic/ClassicChannels/src/com/sun/jchowto/channels/AccountAccessor.java?rev=25) adresinde yer alan ve tam sürüm kaynak koduna herkesin erişebildiği Java Card kodlarının bir karşılaştırması gerçekleştirilmiştir. Yine LOC bazında karşılaştırıldığında DSL4JavaCard kullanılarak otomatik üretilen kodların tam sürüm uygulamanın %61'ini oluşturduğu; bir başka deyişle tüm programın %61'inin sadece DSL4JavaCard modellemesi ile elde edilebildiği görülmüştür. Bir önceki uygulamadaki otomatik kod üretimi başarımına göre bu oranın düşüklüğünün sebebi bu uygulama içerisinde temel Java ifadelerinin ve matematiksel ifadelerin ("if/else") oldukça fazla kullanılmasıdır. DSL4JavaCard ile modelleme yapılırken bu tarz yapılar "*Comment Line*" olarak eklenebilir ancak doğrudan otomatik üretilen kod içerisine yazmada kullanılabilecek üstmodel varlıkları DSL4JavaCard üstmodelinde yoktur. Yine LOC ölçümünün yanı sıra üretilen kodun etkinliğini gözlemlemek amacıyla otomatik kod üretimi sonrasında akıllı kart uygulamasının hangi kısımlarının tam olarak üretilebildiği veya üretilemediği incelenmiştir. İnceleme sonucu Tablo 7.2'de verilmiştir.

Tablo 7.2. DSL4JavaCard kullanılarak Java Card kod bileşenlerinin üretilmesine ait değerlendirme sonuçları

JAVA CARD KOD BİLEŞENLERİ	**OTOMATİK KOD ÜRETİMİNİN DEĞERLENDİRİLMESİ**
Temel Java Card applet metotları	Tamamı üretilebiliyor.
Kullanıcı tanımlı ve uygulamaya özel metotlar	Metot tanımlarının tamamı üretilebiliyor. Ancak kod üretimi sonrasında gövdelerinde önemli ölçüde eklentiye ihtiyaç duyuluyor.
Sabit veri tipleri, sınıf özellikleri ve metot değişkenleri	Tamamı üretilebiliyor.
İstisna nesneleri	Tamamı üretilebiliyor. Seçenekler ile istisna nesneleri farklı alternatiflerle oluşturulabiliyor. Metotlar ile fırlatılabilecek istisnalarda eklenebiliyor.
Temel Java Card Güvenlik unsurları	Tüm güvenlik metotları ve özellik tanımları otomatik üretilebiliyor. Ancak metot içerisinde kod üretimi sonrasında eklentilere ihtiyaç duyuluyor.
Kontrol yapıları, mantıksal ve matematiksel yapılar	Açıklama satırı olarak eklenebiliyor. Ancak model içerisinde if-else try-catch yapısı bulunmuyor, mantıksal ve matematiksel yazımlara yer verilmemiştir. Değişiklik ihtiyacı duyuluyor.

8. SONUÇ VE İLERİYE YÖNELİK ÇALIŞMALAR

Günümüz gelişen yazılım dünyasında, DSL kullanımı ve model güdümlü yazılım geliştirme giderek yaygınlaşmakta; böylelikle alt seviye hataları, dil bağlılıkları ve zaman kaybı minimum seviyeye indirilmeye çalışılmaktadır. Java Card gibi yazılım geliştirilmesi ortam kısıtları nedeniyle nispeten zor alanlarda ise model güdümlü yaklaşımların tezin ilgili çalışmalar altbölümünde de anlatıldığı gibi çok olduğu ve geliştiricilerin yoğun bir şekilde bu yönde uğraştığı görülmektedir. Bu tez çalışması kapsamında da Java Card platformunda daha hızlı, daha etkin ve model güdümlü bir şekilde geliştirmeye yönelik bileşenleri içeren ve bütünleşik bir DSL olan DSL4JavaCard üretilmiştir. Geliştiriciler bu dilin ön tanımlı bir üstmodele dayalı somut sözdizimini kullanarak akıllı kart yazılımlarını modelleme aracılığıyla ve akıllı kart ortamının kısıtlarına uygun bir şekilde hazırlayabilmekte; hazırladıkları modellerin otomatik modelden metne dönüşümleri sonrasında Java Card platformunda işletilebilir yazılım kodlarını elde edebilmektedirler. Bunun için yazılım geliştiricilerin çoğunlukla sadece akıllı kart alan bilgisine sahip olması yeterlidir.

Dilin soyut sözdiziminin üstmodel temelli olması ve bu üstmodel içerisinde kart üzeri uygulamalara ait bileşenlerin ve ilişkilerin tanımlanmış olması UML'e dayalı analiz ve tasarım bilgisine sahip yazılım geliştiricilerin de önerilen dile adaptasyonunu kolaylaştırmaktadır. Şöyle ki; çoğu durumda DSL4JavaCard kullanılarak geliştirilen yazılım modellerinde kart üzeri Java Card dil bileşeni örnekleri ve ilişkilerinin kurgusu UML sınıf diyagramlarının ve iletişim diyagramlarının hazırlanması ile benzerlik göstermektedir. Ek olarak DSL4JavaCard ile geliştirilen bu modeller dokümantasyonun dışında işletilebilir özelliktedir ve akıllı kart yazılımları tanımlı DSL4JavaCard semantiğine göre bu modellerden otomatik olarak elde edilebilmektedir. Bu sayede sadece çalışan kodlar oluşturulmamış Java Card uygulamaları için hata ayıklama ve test edilme süreçlerinde de nispeten kolaylık sağlanmıştır. DSL4JavaCard işletimsel semantiğinin Java Card ortamı için doğru olduğu varsayıldığında kart uygulamaları için DSL4JavaCard kullanılarak otomatik üretilen kodların herhangi bir hata barındırmayacağı açıktır.

Gerçekleştirilen durum çalışmaları otomatik üretilen kodların örnek olarak alınan tam sürüm kodların önemli bir miktarının sadece modelleme ile elde edilmesine imkan verdiğini ve yine üretilen kodun niteliksel olarak da sistem ihtiyaçlarını önemli ölçüde karşıladığını göstermiştir.

Bu tezin devamı olacak çalışmalardan ilki, değişik boyutlarda karmaşıklığa sahip kart yazılımlarının geliştirilmesinde DSL4JavaCard'in kullanımının daha yapısal bir şekilde değerlendirilmesi ve yeni değerlendirme sonuçlarına göre dilde iyileştirmelerin sağlanması olabilir. Java diline ait ifadelerin, aritmetik ve mantık yapılarının da modelleme ortamına taşınarak %100'e yakın bir yazılımın geliştirilmesi bir başka çalışma konusudur. Bir diğer çalışma ise akıllı kart dışında kalan ("off-card") ancak akıllı kart uygulaması için gerekli ana bilgisayar yazılımlarının hazırlanması için ek bakış açılarının ve dil yapılarının DSL4JavaCard'a eklenmesi olabilir.

KAYNAKLAR DİZİNİ

Bonnet, s., Potonniee, O., Marvie, R., Geib, J-M, 2004a, "A Model-Driven Approach for Smart Card Configuration." Lecture Notes in Computer Science, 3286:416-435

Bonnet, S., Marvie R., Geib J-M., 2004b, "Putting Concern-Oriented Modeling into Practice." In: 2nd Nordic Woarkshop on UML, Modeling, Methods and Tools, Turku, Finland

Chen, Z., 2000, "Java Card Technology for Smart Cards : Arcitecture and Programmer's Guide", Addison-Wesley

Coglio, A., 2003, "Code generation for high-assurance Java Card applets." In: 3rd NSA Conference on High Confidence Software and Systems. Pp. 85-93

Coglio, A., Green, CA, 2008, "Constructive Approach to Correctness, Exemplified by a Generator for Certified Java Card Applets." Letcure Notes in Computer Science, 4171:57-63

Eclipse Xpand, "a staticly-typed template language" https://eclipse.org/modeling/m2t/?%20project=xpand (Erişim tarihi: Haziran 2015).

Eclipse EMF, "Eclipse Modeling Framework" http://www.eclipse.org/modeling/emf/ (Erişim tarihi: 2015).

Fowler, M., 2011, "Domain-specific Languages". Addison-Wesley Professional

Gomes, B.E.G., Moreira, A.M., Deharbe D., 2007, "Developing Java Card Applications with B." Electronic Notes in Theoretical Computer Science, 184: 81-96

ISO/IEC 7816 Standards, 1995, ISO/IEC Standards family for Identification cards – Integrated circuit cards. http://www.iso.org/iso/home/store/catalogue_tc/catalogue_tc_browse.htm?commid=45144 (Erişim tarihi : Haziran 2015)

Mernik, M., Heering, J., Sloane, A., 2005, "When and how to develop domain-specific languages." ACM Computing Surveys, 37(4):316-344

KAYNAKLAR DİZİNİ (devam)

Moebius, N., Stenzel, K., Grandy, H., Reif, W., 2009b, "Model-Driven Code Generation for Secure Smart Card Applications." In: 20th Australian Software Engineering Conference, pp. 44-53

Moebius, N., Stenzel, K., Grandy, H., Reif, W., 2009a, "SecureMDD: A Model-Driven Development Method for Secure Smart Card Applications." In: 4th International Conference on Availability, Reliability and Security, pp. 841-846

Nikseresht, A., Ziarati, K., 2011, "MDA Based Framework for the Development of Smart Card Based Application". In: 2011 International MultiConference of Engineers and Computer Scientist, pp. 1-6 .

OMG, "XML Metadata Interchange (XMI) Specification" http://www.omg.org/spec/XMI/ (Erişim tarihi: 2015).

OMG, 2003, Object Management Group Model driven architecture. ⟨http://www.omg.org/mda/⟩ (Erişim tarihi: Ağustos 2015).

Oracle, 1999, "Java Card API" https://docs.oracle.com/javacard/3.0.5/api/index.html (Erişim tarihi: 2015).

Oracle Co. Java Card Technology, Java Card Technology http://www.oracle.com/technetwork/java/embedded/javacard/ (Erişim tarihi : Haziran 2015)

Sarıtaş, H. B., 2011, "Akıllı Kart Yazılımlarının Model Güdümlü Geliştirilmesi", Yüksek Lisans Tezi, Ege Üniversitesi Fen Bilimleri Enstitüsü, 92 sayfa

Sarıtaş, H. B., Kardaş, G., 2011, "Java card yazılımlarının model güdümlü geliştirilmesi." Türkiye Bilişim Vakfı Bilgisayar Bilimleri ve Mühendisliği Dergisi, 4:19-28

Saritas, H. B., Kardas, G., 2014, "A model driven architecture fort he development of smart card software." Computer Languages, Systems & Structures, 40(2): 53-72

Selic, B., 2003, "The pragmatics of model-driven development", IEEE Software, vol. 20, no. 5, pp. 19-25.

KAYNAKLAR DİZİNİ (devam)

Steinberg, D., Budinsky, F., Merks, E., Paternostro, M., 2008, EMF: eclipse modeling framwork. Pearson Education

Tatibouet, B., Requet, A., Voisinet, J.C., Hammad, A., 2003, "Java Card code generation from B specifications." Lecture Notes in Computer Science 2885: 306-318

Tosun, M., Kardaş, G., 2015, "DSL4JavaCard: Java Card Platformu için bir Alana Özgü Dil", 9. Ulusal Yazılım Mühendisliği Sempozyumu (UYMS 2015), 9-11 Eylül 2015, İzmir, Türkiye, CEUR Workshop Proceedings (Kabul edildi)

van Deursen, A., Klint, P., Visser, J., 2000, **"**Domain-specific languages: an annotated bibliography." ACM SIGPLAN Notices, 35 (6): 26-36

ÖZGEÇMİŞ

Miray TOSUN
Zümrüt Evler Mah. Nish Adalar Sitesi B :16 D:9 Maltepe/İSTANBUL
e-mail: mmirayy.tosun@gmail.com

Eğitim Durumu

Yüksek Lisans	: 2010- , EGE Üniversitesi, Uluslararası Bilgisayar Enstitüsü
Lisans	: 2004-2009, İzmir Ekonomi Üniversitesi, Yazılım Mühendisliği
Lise	: 1997-2004, Karamürsel Anadolu Lisesi

Yabancı Dil

Türkçe	: anadil
İngilizce	: iyi derecede
İspanyolca	: orta derecede
Almanca	: başlangıç seviyesi

İş Deneyimi

- Kıdemli Yazılım Test Uzmanı, Obase (2013- Halen)
- Yazılım Test Uzmanı, KG Teknolojikleri, (2012-2013)
- Yazılım Test Uzmanı, Ericsson, (2012-2012)
- Yazılım Mühendisi, Unit, (2010-2012)
- Yazılım Mühendisi, Motif BİIgi ve İletişim Sis.Tic.AŞ, (2010-2010)
- Stajyer, Korozo Ambalaj San. Ve Tic. A.Ş (2007)
- Stajyer, HAVELSAN A.Ş (2006)

Programlama Dilleri Bilgisi

- Java, C/C++, C#

EKLER

Ek 1 İngilizce Türkçe Sözlük

Ek 2 DSL4JavaCard Ecore Kodları

Ek 3 DSL4JavaCard Xpand Kodları

EK-1

İngilizce - Türkçe Sözlük

Applet	Akıllı kart uygulaması
Application class	Uygulama sınıfı
Application programming interface	Uygulama programlama arayüzü
Application protocol data unit	Uygulama protokoli veri birimi
Chip card	Çipli Kart
Converter	Çevirici
Comment	Yorum
Computational independent model	Hesaplamalı bağımsız model
Decryption	Şifre çözme
Domain Specific Languge	Platforma özgü dil
Domain specific model	Platforma özgü model
Electrically erasable programmable read only memory	Elektrikle silinebilir programlanabilir salt okunur bellek
Continuity Index	Devamlılık Endeksi
Exception	İstisna
Encryption	Şifreleme
Firewall	Güvenlik duvarı
Hexadecimal	Onaltılık
Integrated circuit	Entegre devre
Install	Yükleme
Java card virtual machine	Java Kart sanal makinesi
Java card runtime environment	Java kart çalışma ortamı
Model driven development	Model güdümlü geliştirme
Model driven architecture	Model güdümlü mimari
Meta model	Meta modeli
Object	Nesne
Object constraint language	Nesne kısıtlama dili
Object management group	Nesne yönetim grubu
Off card	Kart dışı
On card	Kart üzeri
Platform independent model	Paltform bağımsız model
Personal identification number	Kimlik numarası

Parameter	Parametre
Process	Süreç
Read only memory	Salt okunur bellek
Subscriber identity module	Abone kimlik modülü
Unified modeling language	Birleşik model dili
XML metadata interchange	XML üst veril değişimi
Viewpoint	Bakış açısı

EK-2

DSL4JavaCard Ecore Kodları

```
package javaCardModel : javaCardModel = 'javaCardModel'
{
class ApplicationClass
{
attribute className : String[?];
property fields : Field[*] { ordered composes };
property methods : Method[*] { ordered composes };
property applets : Applet[+] { ordered composes };
invariant ApplicationNameSizeMustGreaterThenZero:
className.size()>0;
invariant AplicationClassFieldsMustHaveAccessSpecifier:
self.fields->forAll(f | f.accessSpecifier <> AccessSpecifiers::none);
invariant ApplicationClassNeedsOneProcessANDOneInstallMethos:
self.applets->size()>0;
}
class Field
{
attribute accessSpecifier : AccessSpecifiers[?];
attribute dataType : DataTypes[?];
attribute name : String[?];
attribute initialValue : String[?];
attribute isArray : Boolean[?];
attribute isFinal : Boolean[?];
attribute isStatic : Boolean[?];
invariant FieldMustHaveName:
self.name.size()>0;
invariant IfAccessSpecifierExistFieldTypeNotBeNone:
self.accessSpecifier <> AccessSpecifiers::none implies
self.dataType <> DataTypes::none;
}
class Method
{
attribute accessSpecifier : AccessSpecifiers[?];
attribute isStatic : Boolean[?];
attribute returnType : DataTypes[?];
attribute methodName : String[?];
attribute returnFieldName : String[?];
property parameters : Parameter[*] { ordered composes };
property comments : Comment[*] { ordered composes };
property fields : Field[*] { ordered composes };
property exceptions : MethodException[0..3] { ordered composes };
property aesKeys : AESKey[*] { ordered composes };
property aids : AID[*] { ordered composes };
property apdus : APDU[*] { ordered composes };
property apduExceptions : APDUException[*] { ordered composes };
property arithmaticExceptions : ArithmeticException[*] { ordered composes };
property arrayIndexOutOfBoundsExceptions : ArrayIndexOutOfBoundsException[*]
{ ordered composes };
property arrayStoreExceptions : ArrayStoreException[*] { ordered composes };
property cardExceptions : CardException[*] { ordered composes };
property checksums : Checksum[*] { ordered composes };
property ciphers : Cipher[*] { ordered composes };
property classCastExceptions : ClassCastException[*] { ordered composes };
property cryptoExceptions : CryptoException[*] { ordered composes };
property desKeys : DESKey[*] { ordered composes };
property dsaKeys : DSAKey[*] { ordered composes };
property dsaPrivateKeys : DSAPrivateKey[*] { ordered composes };
property dsaPublicKeys : DSAPublicKey[*] { ordered composes };
property ecKeys : ECKey[*] { ordered composes };
property ecPrivateKeys : ECPrivateKey[*] { ordered composes };
property ecPublicKeys : ECPublicKey[*] { ordered composes };
property hmacKeys : HMACKey[*] { ordered composes };
property indexOutOfBoundsExceptions : IndexOutOfBoundsException[*] { ordered composes };
property initializeMessageDigests : InitializedMessageDigest[*] { ordered composes };
```

```
property iso7816s : ISO7816[*] { ordered composes };
property isoExceptions : ISOException[*] { ordered composes };
property keys : Key[*] { ordered composes };
property keyAgreements : KeyAgreement[*] { ordered composes };
property keyBuilders : KeyBuilder[*] { ordered composes };
property keyEncryotions : KeyEncryption[*] { ordered composes };
property keyPairs : KeyPair[*] { ordered composes };
property koreanSEEDKeys : KoreanSEEDKey[*] { ordered composes };
property messageDigests : MessageDigest[*] { ordered composes };
property negativeArraySizeExceptions : NegativeArraySizeException[*]
{ ordered composes };
property nullPointerExceptions : NullPointerException[*] { ordered composes };
property ownerPins : OwnerPIN[*] { ordered composes };
property pins : PIN[*] { ordered composes };
property pinExceptions : PINException[*] { ordered composes };
property privateKeys : PrivateKey[*] { ordered composes };
property publicKeys : PublicKey[*] { ordered composes };
property randomData : RandomData[*] { ordered composes };
property rsaPrivateCrtKeys : RSAPrivateCrtKey[*] { ordered composes };
property rsaPrivateKeys : RSAPrivateKey[*] { ordered composes };
property rsaPublicKeys : RSAPublicKey[*] { ordered composes };
property secretKeys : SecretKey[*] { ordered composes };
property securityExceptions : SecurityException[*] { ordered composes };
property serviceExceptions : ServiceException[*] { ordered composes };
property signatures : Signature[*] { ordered composes };
property signatureMessageRecoveries : SignatureMessageRecovery[*] { ordered composes };
property systemExceptions : SystemException[*] { ordered composes };
property transactionExceptions : TransactionException[*] { ordered composes };
property userExceptions : UserException[*] { ordered composes };
property utils : Util[*] { ordered composes };
property utilExceptions : UtilException[*] { ordered composes };
invariant MethodMustHaveName:
self.methodName.size()>0;
invariant MethodMustHaveAccessSpecifier:
self.accessSpecifier <> AccessSpecifiers::none;
invariant MethodMustHaveReturnType:
self.returnType <> DataTypes::none;
invariant IfMethodDataTypeIsNotVoidThenReturnFieldNameMustBeZero:
self.returnType = DataTypes::void implies
self.returnFieldName=null;
invariant IfMethodHasPatametersThenParameterMustHaveDataType:
self.parameters->forAll(f | f.dataType <> DataTypes::none);
invariant IfMethodsFieldMustNotUseAccessSpecifiers:
self.fields->forAll(x | x.accessSpecifier <> AccessSpecifiers::none);
invariant IfMethodsHasExceptionSoExceptionMustNotBeNone:
self.exceptions->forAll(e | e.exceptionType <> ExceptionTypes::none);
invariant IfMethodsHasExceptionSoExceptionMustNotBeException:
self.exceptions->forAll(e | e.exceptionType <> ExceptionTypes::Exception);
}
class Comment
{
attribute commentLine : String[?];
}
enum AccessSpecifiers { serializable }
{
literal none;
literal public = 1;
literal private = 2;
literal protected = 3;
}
enum DataTypes { serializable }
{
literal none;
literal byte = 1;
literal short = 2;
literal int = 3;
literal _'String' = 4;
literal double = 5;
literal APDU = 6;
literal OwnerPIN = 7;
literal void = 8;
literal PIN = 9;
```

```
literal Object = 10;
literal AID = 11;
literal Cipher = 12;
literal Checksum = 13;
literal KeyAgreement = 14;
literal KeyBuilder = 15;
literal KeyPair = 16;
literal MessageDigest = 17;
literal InitializedMessageDigest = 18;
literal RandomData = 19;
literal Signature = 20;
literal DSAKey = 21;
literal DSAPrivateKey = 22;
literal DSAPublicKey = 23;
literal ECKey = 24;
literal ECPrivateKey = 25;
literal ECPublicKey = 26;
literal Key = 27;
literal PrivateKey = 28;
literal RSAPrivateKey = 29;
literal RSAPrivateCrtKey = 30;
literal PublicKey = 31;
literal RSAPublicKey = 32;
literal SecretKey = 33;
literal AESKey = 34;
literal DESKey = 35;
literal HMACKey = 36;
literal KoreanSEEDKey = 37;
literal SignatureMessageRecovery = 38;
literal ISO7816 = 38;
literal KeyEncryption = 39;
literal Util = 40;
literal UtilException = 41;
literal boolean = 42;
literal Applet = 43;
}
class Applet
{
attribute method : AppletMethods[?];
property parameters : Parameter[*] { ordered composes };
invariant AppletProcessMethodMustHaveOneParameter:
self.method=AppletMethods::process implies
self.parameters->size()=1 and self.parameters->forAll(f | f.dataType = DataTypes::APDU);
invariant AppletGetShareableInterfaceObjectMethodMustHaveTwoParameters:
self.method=AppletMethods::getShareableInterfaceObject implies
self.parameters->size()=2;
invariant AppletdeselectMethodDontNeedAnyThingItwillBeGeneratedAutomatically:
self.method=AppletMethods::deselect implies
self.parameters->size()=0;
}
enum AppletMethods { serializable }
{
literal Applet = 1;
literal getShareableInterfaceObject = 2;
literal register = 3;
literal deselect = 4;
literal select = 5;
literal process = 6;
literal install = 7;
literal selectingApplet = 8;
}
class APDU
{
property fields : Field[?] { composes };
property parameters : Parameter[*] { ordered composes };
attribute apduField : APDUFields[?];
attribute method : APDUMethods[?];
property instanceVariable : InstanceVariable[?] { composes };
invariant APDUgetBufferMethodsNeedByteArrayField:
self.method=APDUMethods::getBuffer implies
self.fields->size()=1 and self.fields->forAll(e | e.dataType <>
DataTypes::byte) and self.fields->forAll(e | e.isArray=true);
```

```
invariant APDUgetCLAChannelisSTATICAndMethodsNeedByteField:
self.method=APDUMethods::getCLAChannel implies
self.fields->size()=1 and self.fields->forAll(e | e.dataType <> DataTypes::byte);
invariant APDUgetCurrentAPDUisSTATICAndMethodsNeedAPDUField:
self.method=APDUMethods::getCurrentAPDU implies
self.fields->size()=1 and self.fields->forAll(e | e.dataType <> DataTypes::APDU);
invariant APDUgetCurrentAPDUBufferisSTATICAndMethodsNeedByteArrayField:
self.method=APDUMethods::getCurrentAPDUBuffer implies
self.fields->size()=1 and self.fields->forAll(e | e.dataType <>
DataTypes::byte) and self.fields->forAll(e | e.isArray=true);
invariant APDUgetCurrentStateMethodsNeedByteField:
self.method=APDUMethods::getCurrentState implies
self.fields->size()=1 and self.fields->forAll(e | e.dataType <> DataTypes::byte);
invariant APDUgetInBlockSizeisSTATICAndMethodsNeedShortField:
self.method=APDUMethods::getInBlockSize implies
self.fields->size()=1 and self.fields->forAll(e | e.dataType <> DataTypes::short);
invariant APDUgetIncomingLengthMethodsNeedShortField:
self.method=APDUMethods::getIncomingLength implies
self.fields->size()=1 and self.fields->forAll(e | e.dataType <> DataTypes::short);
invariant APDUgetNADMethodsNeedByteField:
self.method=APDUMethods::getNAD implies
self.fields->size()=1 and self.fields->forAll(e | e.dataType <> DataTypes::byte);
invariant APDUgetOffsetCdataMethodsNeedshortField:
self.method=APDUMethods::getOffsetCdata implies
self.fields->size()=1 and self.fields->forAll(e | e.dataType <> DataTypes::short);
invariant APDUgetOutBlockSizeisSTATICAndMethodsNeedShortField:
self.method=APDUMethods::getOutBlockSize implies
self.fields->size()=1 and self.fields->forAll(e | e.dataType <> DataTypes::short);
invariant APDUgetProtocolisSTATICAndMethodsNeedByteField:
self.method=APDUMethods::getProtocol implies
self.fields->size()=1 and self.fields->forAll(e | e.dataType <> DataTypes::byte);
invariant APDUisCommandChainingCLAMethodsNeedBooleanField:
self.method=APDUMethods::isCommandChainingCLA implies
self.fields->size()=1 and self.fields->forAll(e | e.dataType <> DataTypes::boolean);
invariant APDUisISOInterindustryCLAMethodsNeedBooleanField:
self.method=APDUMethods::isISOInterindustryCLA implies
self.fields->size()=1 and self.fields->forAll(e | e.dataType <> DataTypes::boolean);
invariant APDUisSecureMessagingCLAMethodsNeedBooleanField:
self.method=APDUMethods::isSecureMessagingCLA implies
self.fields->size()=1 and self.fields->forAll(e | e.dataType <> DataTypes::boolean);
invariant APDUisValidCLAMethodsNeedBooleanField:
self.method=APDUMethods::isValidCLA implies
self.fields->size()=1 and self.fields->forAll(e | e.dataType <> DataTypes::boolean);
invariant APDUReceiveBytesMethodsMustHaveOneParameterAndShortField:
self.method=APDUMethods::receiveBytes implies
self.parameters->size()=1 and self.fields->size()=1 and self.fields->forAll(e |
e.dataType <> DataTypes::short);
invariant APDUsendBytesMethodsMustHave2ParameterButDontNeedField:
self.method=APDUMethods::sendBytes implies
self.parameters->size()=2 and self.fields->size()=0;
invariant APDUsendBytesLongMethodsMustHave3ParameterButDontNeedField:
self.method=APDUMethods::sendBytesLong implies
self.parameters->size()=3 and self.fields->size()=0;
invariant APDUsetOutgoingAndSendMethodsMustHave2ParameterButDontNeedField:
self.method=APDUMethods::setOutgoingAndSend implies
self.parameters->size()=2 and self.fields->size()=0;
invariant APDUsetIncomingAndReceiveMethodsNeedShortField:
self.method=APDUMethods::setIncomingAndReceive implies
self.fields->size()=1 and self.fields->forAll(e | e.dataType <> DataTypes::short);
invariant APDUsetOutgoingMethodsNeedShortField:
self.method=APDUMethods::setOutgoing implies
self.fields->size()=1 and self.fields->forAll(e | e.dataType <> DataTypes::short);
invariant APDUsetOutgoingAndSendMethodsDontNeedAnyParametersAndField:
self.method=APDUMethods::setOutgoingAndSend implies
self.fields->size()=0 and self.parameters->size()=0;
invariant APDUsetOutgoingLengthMethodsMustHave1ParameterButDontNeedAnyField:
self.method=APDUMethods::setOutgoingLength implies
self.parameters->size()=1 and self.fields->size()=0;
invariant APDUsetOutgoingNoChainingMethodsNeedShortField:
self.method=APDUMethods::setOutgoingNoChaining implies
self.fields->size()=1 and self.fields->forAll(e | e.dataType <> DataTypes::short);
invariant APDUwaitExtensionisSTATICAndMethodDontNeedAnyParameterOrField:
```

```
self.method=APDUMethods::waitExtension implies
self.fields->size()=0 and self.parameters->size()=0;
invariant
WARNING_IfYouChooseAnyOfAPDUFieldsYouWontUseAPDUMethodsORIfYouDontChooseAnyOfAPDUFieldsY
ouCantUseAPDUMethods:
self.apduField<>APDUFields::none;
}
enum APDUFields { serializable }
{
literal none;
literal STATE_ERROR_NO_T0_GETRESPONSE = 1;
literal PROTOCOL_MEDIA_CONTACTLESS_TYPE_B = 2;
literal STATE_FULL_OUTGOING = 3;
literal PROTOCOL_MEDIA_USB = 4;
literal STATE_ERROR_NO_T0_REISSUE = 5;
literal PROTOCOL_TYPE_MASK = 6;
literal PROTOCOL_T0 = 7;
literal PROTOCOL_MEDIA_DEFAULT = 8;
literal STATE_OUTGOING_LENGTH_KNOWN = 9;
literal STATE_INITIAL = 10;
literal STATE_PARTIAL_INCOMING = 11;
literal STATE_OUTGOING = 12;
literal STATE_ERROR_T1_IFD_ABORT = 13;
literal PROTOCOL_T1 = 14;
literal STATE_FULL_INCOMING = 15;
literal PROTOCOL_MEDIA_CONTACTLESS_TYPE_A = 16;
literal STATE_ERROR_IO = 17;
literal PROTOCOL_MEDIA_MASK = 18;
literal STATE_PARTIAL_OUTGOING = 19;
}
enum APDUMethods { serializable }
{
literal getInBlockSize = 1;
literal getCLAChannel = 2;
literal waitExtension = 3;
literal getCurrentAPDUBuffer = 4;
literal setIncomingAndReceive = 5;
literal getOutBlockSize = 6;
literal setOutgoingLength = 7;
literal setOutgoingAndSend = 8;
literal getBuffer = 9;
literal setOutgoing = 10;
literal receiveBytes = 11;
literal sendBytes = 12;
literal setOutgoingNoChaining = 13;
literal getNAD = 14;
literal getCurrentAPDU = 15;
literal getProtocol = 16;
literal getCurrentState = 17;
literal sendBytesLong = 18;
literal getIncomingLength = 19;
literal getOffsetCdata = 20;
literal isCommandChainingCLA = 21;
literal isISOInterindustryCLA = 22;
literal isSecureMessagingCLA = 23;
literal isValidCLA = 24;
}
abstract class PIN { interface }
{
property fields : Field[?] { composes };
attribute method : PINMethods[?];
property parameters : Parameter[*] { ordered composes };
property instanceVariable : InstanceVariable[?] { composes };
invariant PINCheckMethodsMustHaveThreeParameter:
self.method=PINMethods::check implies
self.parameters->size()=3;
}
enum PINMethods { serializable }
{
literal getTriesRemaining = 1;
literal check = 2;
literal isValidated = 3;
```

```
literal reset = 4;
}
class OwnerPIN
{
property fields : Field[?] { composes };
attribute method : OwnerPINMethods[?];
property parameters : Parameter[*] { ordered composes };
property instanceVariable : InstanceVariable[?] { composes };
invariant
OwnerPINCheckMethodsMustHaveThreeParameterAndItHasBooleanReturnTypeSoAddOneBooleanField(
'check method must need three parameter and one boolean field for retrun value'):
self.method=OwnerPINMethods::check implies
self.parameters->size()=3 and self.fields->size()=1;
invariant
OwnerPINgetTriesRemainingMethodHasNeedByteFiedl('getTriesRemaining Method has need one
byte field for retrun value'):
self.method=OwnerPINMethods::getTriesRemaining implies
self.fields->size()=1 and self.fields->forAll(e | e.dataType <> DataTypes::byte);
invariant
OwnerPINUpdateMethodsMustHaveThreeParameter('update must have three Byte Array, short
and byte parameters'):
self.method=OwnerPINMethods::update implies
self.parameters->size()=3;
invariant OwnerPINSetValidatedFlagMethodsMustHaveOneParameter('setValidatedFlag must
have boolean parameter'):
self.method=OwnerPINMethods::setValidatedFlag implies
self.parameters->size()=1;
invariant
OwnerPINisValidatedMethodHasNeedByteFiedl('isValidated Method has need one boolean field
for retrun value'):
self.method=OwnerPINMethods::isValidated implies
self.fields->size()=1 and self.fields->forAll(e | e.dataType <> DataTypes::boolean);
}
enum OwnerPINMethods { serializable }
{
literal OwnerPIN = 1;
literal getValidatedFlag = 2;
literal setValidatedFlag = 3;
literal getTriesRemaining = 4;
literal check = 5;
literal isValidated = 6;
literal reset = 7;
literal update = 8;
literal resetAndUnblock = 9;
}
enum ExceptionUsageTypes { serializable }
{
literal none;
literal tryCatch = 1;
literal ifElse = 2;
literal ThrowNew = 3;
}
class Parameter
{
attribute dataType : DataTypes[?];
attribute name : String[?];
attribute isArray : Boolean[?];
invariant ParameterMustHaveName('Parameters must have name'):
self.name.size()>0;
}
class Object
{
property applicationClasses : ApplicationClass[+] { ordered composes };
invariant ObjectMustHaveAtLeastOneApplicationClass:
self.applicationClasses->size()>0;
}
enum ExceptionTypes { serializable }
{
literal none;
literal Exception = 1;
literal CardException = 2;
literal UserException = 3;
```

```
literal RuntimeException = 4;
literal CardRuntimeException = 5;
literal APDUException = 6;
literal CryptoException = 7;
literal PINException = 8;
literal ISOException = 9;
literal ServiceException = 10;
literal SystemException = 11;
literal TransactionException = 12;
literal ArithmeticException = 13;
literal ArrayStoreException = 14;
literal ClassCastException = 15;
literal IndexOutOfBoundsException = 16;
literal ArrayIndexOutOfBoundsException = 17;
literal NegativeArraySizeException = 18;
literal NullPointerException = 19;
literal SecurityException = 20;
}
class MethodException
{
attribute exceptionType : ExceptionTypes[?];
}
abstract class Throwable { interface }
{
property fields : Field[?] { composes };
attribute method : ThrowableMethods[?];
property parameters : Parameter[*] { ordered };
property instanceVariable : InstanceVariable[?] { composes };
}
enum ThrowableMethods { serializable }
{
literal Throwable;
}
abstract class Exception { interface }
{
property fields : Field[?] { composes };
attribute method : ExceptionMethods[?];
property parameters : Parameter[*] { ordered composes };
attribute exceptionUsageType : ExceptionUsageTypes[?];
property instanceVariable : InstanceVariable[?] { composes };
invariant IfExceptionUseTryCatchJustAddOneField:
self.exceptionUsageType = ExceptionUsageTypes::tryCatch implies
self.fields->size()=1;
invariant IfExceptionUseThrowNewJustAddOneField:
self.exceptionUsageType = ExceptionUsageTypes::ThrowNew implies
self.fields->size()=1;
invariant ExceptionMethodsDoNotNeedParameter:
self.method = ExceptionMethods::Exception implies
self.parameters->size()=0;
}
enum ExceptionMethods { serializable }
{
literal Exception;
}
class CardException
{
property fields : Field[?] { composes };
attribute method : CardExceptionMethods[?];
property parameters : Parameter[*] { ordered composes };
attribute exceptionUsageType : ExceptionUsageTypes[?];
property instanceVariable : InstanceVariable[?] { composes };
invariant CardExceptionSetReasonMethodsMustHaveOneParameterButDontNeedAnyField:
self.method=CardExceptionMethods::setReason implies
self.parameters->size()=1 and self.fields->size()=0;
invariant CardExceptionThrowItisSTATICAndMethodsMustHaveOneParameterDontNeedField:
self.method=CardExceptionMethods::throwIt implies
self.parameters->size()=1 and self.fields->size()=0;
invariant IfCardExceptionUseTryCatchJustAddOneField:
self.exceptionUsageType = ExceptionUsageTypes::tryCatch implies
self.fields->size()=1;
invariant IfCardExceptionUseThrowNewJustAddOneField:
self.exceptionUsageType = ExceptionUsageTypes::ThrowNew implies
```

```
self.fields->size()=1;
invariant CardExceptiongetReasonMethodsNeedShortFieldButDontNeedAnyParameters:
self.method=CardExceptionMethods::getReason implies
self.parameters->size()=0 and self.fields->forAll(e | e.dataType <> DataTypes::short);
}
enum CardExceptionMethods { serializable }
{
literal CardException;
literal getReason = 1;
literal setReason = 2;
literal throwIt = 3;
}
class UserException
{
property fields : Field[?] { composes };
attribute method : UserExceptionMethods[?];
property parameters : Parameter[*] { ordered composes };
attribute exceptionUsageType : ExceptionUsageTypes[?];
property instanceVariable : InstanceVariable[?] { composes };
invariant UserExceptionMethodsMustHaveOneParameter:
self.method = UserExceptionMethods::UserException implies
self.parameters ->size()=1;
invariant UserExceptionThrowItMethodsMustHaveOneParameter:
self.method = UserExceptionMethods::throwIt implies
self.parameters ->size()=1;
invariant IfUserExceptionUseTryCatchJustAddOneField:
self.exceptionUsageType = ExceptionUsageTypes::tryCatch implies
self.fields->size()=1;
invariant IfUserExceptionUseThrowNewJustAddOneField:
self.exceptionUsageType = ExceptionUsageTypes::ThrowNew implies
self.fields->size()=1;
}
enum UserExceptionMethods { serializable }
{
literal UserException;
literal throwIt = 1;
}
abstract class RuntimeException { interface }
{
property fields : Field[?] { composes };
attribute method : RuntimeExceptionMethods[?];
property parameters : Parameter[*] { ordered composes };
attribute exceptionUsageType : ExceptionUsageTypes[?];
property instanceVariable : InstanceVariable[?] { composes };
invariant IfRuntimeExceptionUseTryCatchJustAddOneField:
self.exceptionUsageType = ExceptionUsageTypes::tryCatch implies
self.fields->size()=1;
invariant IfRuntimeExceptionUseThrowNewJustAddOneField:
self.exceptionUsageType = ExceptionUsageTypes::ThrowNew implies
self.fields->size()=1;
invariant RuntimeExceptionMethodsDoNotNeedAnyParameter:
self.parameters->size()=0;
invariant RuntimeExceptionEqualsMethodsNeedOneParameter:
self.parameters->size()=1;
}
enum RuntimeExceptionMethods { serializable }
{
literal RuntimeException;
literal equals = 1;
}
class CardRuntimeException
{
property fields : Field[?] { composes };
attribute method : CardExceptionMethods[?];
property parameters : Parameter[*] { ordered composes };
attribute exceptionUsageType : ExceptionUsageTypes[?];
property instanceVariable : InstanceVariable[?] { composes };
invariant CardRuntimeExceptionMethodsMustHaveOneParameter:
self.method = CardRuntimeExceptionMethods::CardRuntimeException implies
self.parameters ->size()=1;
invariant CardRuntimeExceptionSetReasonMethodsMustHaveOneParameterButDontNeedFields:
self.method = CardRuntimeExceptionMethods::setReason implies
```

```
self.parameters ->size()=1 and self.fields->size()=0;
invariant CardRuntimeExceptiongetReasonMethodDontNeedAnyParameterButNeedShortField:
self.method = CardRuntimeExceptionMethods::getReason implies
self.parameters ->size()=0 and self.fields->forAll(x|x.dataType = DataTypes::short);
invariant CardRuntimeExceptionThrowITisSTATICAndMethodsMustHaveOneParameter:
self.method = CardRuntimeExceptionMethods::throwIt implies
self.parameters ->size()=1;
invariant CardRuntimeExceptionEqualsMethodsMustHaveOnParameterAndBooleanField:
self.method = CardRuntimeExceptionMethods::equals implies
self.parameters ->size()=1 and self.fields->forAll(x|x.dataType = DataTypes::boolean);
invariant IfCardRuntimeExceptionUseTryCatchJustAddOneField:
self.exceptionUsageType = ExceptionUsageTypes::tryCatch implies
self.fields->size()=1;
invariant IfCardRunTimeExceptionUseThrowNewJustAddOneField:
self.exceptionUsageType = ExceptionUsageTypes::ThrowNew implies
self.fields->size()=1;
}
enum CardRuntimeExceptionMethods { serializable }
{
literal CardRuntimeException;
literal getReason = 1;
literal setReason = 2;
literal throwIt = 3;
literal equals = 4;
}
class UtilException
{
property fields : Field[?] { composes };
attribute utilExceptionField : UtilExceptionFields[?];
attribute method : UtilExceptionMethods[?];
attribute exceptionUsageType : ExceptionUsageTypes[?];
property parameters : Parameter[*] { ordered composes };
property instanceVariable : InstanceVariable[?] { composes };
invariant UtilExceptionMethodsMustHaveOneParameter:
self.method = UtilExceptionMethods::UtilException implies
self.parameters ->size()=1;
invariant UtilExceptionSetReasonMethodsMustHaveOneParameterOrsystemExceptionField:
self.method = UtilExceptionMethods::setReason implies
self.parameters ->size()=1 or self.utilExceptionField <> UtilExceptionFields::none;
invariant UtilExceptionThrowITMethodsMustHaveOneParameterOrsystemExceptionField:
self.method = UtilExceptionMethods::throwIt implies
self.parameters ->size()=1 or self.utilExceptionField <> UtilExceptionFields::none;
invariant UtilExceptionEqualsMethodsMustHaveOneObjectTypeParameter:
self.method = UtilExceptionMethods::equals implies
self.parameters ->size()=1 and self.parameters->forAll(x|x.dataType =
DataTypes::Object);
invariant UtilExceptionUseTryCatchJustAddOneField:
self.exceptionUsageType = ExceptionUsageTypes::tryCatch implies
self.fields->size()=1;
invariant UtilExceptionUseThrowNewJustAddOneField:
self.exceptionUsageType = ExceptionUsageTypes::ThrowNew implies
self.fields->size()=1;
invariant IfUtilExceptionUseIfElseJustAdd1ParameterOrsystemExceptionField:
self.exceptionUsageType = ExceptionUsageTypes::ifElse implies
self.parameters ->size()=1 or self.utilExceptionField <> UtilExceptionFields::none;
}
enum UtilExceptionFields { serializable }
{
literal none;
literal ILLEGAL_VALUE = 1;
literal TYPE_MISMATCHED = 2;
}
enum UtilExceptionMethods { serializable }
{
literal UtilException;
literal throwIt = 1;
literal getReason = 2;
literal setReason = 3;
literal equals = 4;
}
class APDUException
{
```

```
property fields : Field[?] { composes };
attribute method : APDUExceptionMethods[?];
attribute apduExceptionField : APDUExceptionFields[?];
property parameters : Parameter[*] { ordered composes };
attribute exceptionUsageType : ExceptionUsageTypes[?];
property instanceVariable : InstanceVariable[?] { composes };
invariant APDUExceptionMethodsMustHaveOneParameter:
self.method = APDUExceptionMethods::APDUException implies
self.parameters ->size()=1;
invariant APDUExceptionThrowItisSTATICAndMethodsMustHaveOneParameterOR1APDUExceptionFiel
dsButDontNeedAnyField:
self.method = APDUExceptionMethods::throwIt implies
(self.parameters ->size()=1 or self.apduExceptionField <>
APDUExceptionFields::none) and self.fields->size()=0;
invariant APDUExceptionSetReasonMethodsMustHaveOneParameterButDontNeedAnyField:
self.method = APDUExceptionMethods::setReason implies
self.parameters ->size()=1 and self.fields->size()=0;
invariant APDUExceptiongetReasonMethodsMustHaveOneParameterAndShortField:
self.method = APDUExceptionMethods::getReason implies
self.parameters ->size()=1 and self.fields->forAll(x|x.dataType = DataTypes::short);
invariant IfAPDUExceptionUseTryCatchJustAddOneField:
self.exceptionUsageType = ExceptionUsageTypes::tryCatch implies
self.fields ->size()=1;
invariant IfAPDUExceptionUseThrowNewJustAddOneField:
self.exceptionUsageType = ExceptionUsageTypes::ThrowNew implies
self.fields ->size()=1;
invariant IfAPDUExceptionUseIfElseJustAdd1ParameterOrsystemExceptionField:
self.exceptionUsageType = ExceptionUsageTypes::ifElse implies
self.parameters ->size()=1 or self.apduExceptionField <> APDUExceptionFields::none;
}
enum APDUExceptionFields { serializable }
{
literal none;
literal BAD_LENGTH = 1;
literal BUFFER_BOUNDS = 2;
literal ILLEGAL_USE = 3;
literal IO_ERROR = 4;
literal NO_T0_GETRESPONSE = 5;
literal NO_T0_REISSUE = 6;
literal T1_IFD_ABORT = 7;
}
enum APDUExceptionMethods { serializable }
{
literal none;
literal APDUException = 1;
literal throwIt = 2;
literal getReason = 3;
literal setReason = 4;
}
class CryptoException
{
property fields : Field[?] { composes };
attribute cryptoExceptionField : CryptoExceptionFields[?];
attribute method : CryptoExceptionMethods[?];
attribute exceptionUsageType : ExceptionUsageTypes[?];
property parameters : Parameter[*] { ordered composes };
property instanceVariable : InstanceVariable[?] { composes };
invariant CryptoExceptionThrowItisSTATICAndMethodsMustHaveOneParameterOR1CryptoException
Field:
self.method = CryptoExceptionMethods::throwIt implies
self.parameters ->size()=1 or self.cryptoExceptionField <> CryptoExceptionFields::none;
invariant CryptoExceptionSetReasonMethodsMustHaveOneParameterOR1CryptoExceptionFieldButD
ontNeedField:
self.method = CryptoExceptionMethods::setReason implies
(self.parameters ->size()=1 or self.cryptoExceptionField <>
CryptoExceptionFields::none) and self.fields->size()=0;
invariant IfCryptoExceptionUseTryCatchJustAddOneField:
self.exceptionUsageType = ExceptionUsageTypes::tryCatch implies
self.fields ->size()=1;
invariant IfCryptoExceptionUseThrowNewJustAddOneField:
self.exceptionUsageType = ExceptionUsageTypes::ThrowNew implies
self.fields ->size()=1;
```

```
invariant IfCryptoExceptionUseIfElseJustAdd1ParameterOrsystemExceptionField:
self.exceptionUsageType = ExceptionUsageTypes::ifElse implies
self.parameters ->size()=1 or self.cryptoExceptionField <> CryptoExceptionFields::none;
invariant CryptoExceptiongetReasonMethodsNeedShortFieldButDontNeedAnyParameters:
self.method = CryptoExceptionMethods::getReason implies
self.parameters ->size()=0 and self.fields->forAll(x|x.dataType = DataTypes::short);
}
enum CryptoExceptionFields { serializable }
{
literal none;
literal UNINITIALIZED_KEY = 1;
literal NO_SUCH_ALGORITHM = 2;
literal INVALID_INIT = 3;
literal ILLEGAL_VALUE = 4;
literal ILLEGAL_USE = 5;
}
enum CryptoExceptionMethods { serializable }
{
literal none;
literal CryptoException = 1;
literal throwIt = 2;
literal getReason = 3;
literal setReason = 4;
}
class PINException
{
property fields : Field[?] { composes };
attribute pinExceptionField : PINExceptionFields[?];
attribute method : PINExceptionMethods[?];
attribute exceptionUsageType : ExceptionUsageTypes[?];
property parameters : Parameter[*] { ordered composes };
property instanceVariable : InstanceVariable[?] { composes };
invariant PINExceptionThrowItMethodsMustHave1ParameterOrsystemExceptionField:
self.method = PINExceptionMethods::throwIt implies
self.parameters ->size()=1 or self.pinExceptionField <> PINExceptionFields::none;
invariant PINExceptionSetReasonMethodsMustHaveOneParameterOrsystemExceptionField:
self.method = PINExceptionMethods::setReason implies
self.parameters ->size()=1 or self.pinExceptionField <> PINExceptionFields::none;
invariant IfPINExceptionUseTryCatchJustAddOneField:
self.exceptionUsageType = ExceptionUsageTypes::tryCatch implies
self.fields ->size()=1;
invariant IfPINExceptionUseThrowNewJustAddOneField:
self.exceptionUsageType = ExceptionUsageTypes::ThrowNew implies
self.fields ->size()=1;
invariant PINExceptionEqualsMethodsMustHaveOneObjectTypeParameter:
self.method = PINExceptionMethods::equals implies
self.parameters ->size()=1 and self.parameters->forAll(x|x.dataType =
DataTypes::Object);
invariant IfPINExceptionUseIfElseJustAdd1ParameterOrsystemExceptionField:
self.exceptionUsageType = ExceptionUsageTypes::ifElse implies
self.parameters ->size()=1 or self.pinExceptionField <> PINExceptionFields::none;
}
enum PINExceptionFields { serializable }
{
literal none;
literal ILLEGAL_VALUE = 1;
}
enum PINExceptionMethods { serializable }
{
literal PINException;
literal throwIt = 1;
literal getReason = 2;
literal setReason = 3;
literal equals = 4;
}
class ISOException
{
property fields : Field[?] { composes };
attribute method : ISOExceptionMethods[?];
attribute exceptionUsageType : ExceptionUsageTypes[?];
property parameters : Parameter[*] { ordered composes };
property instanceVariable : InstanceVariable[?] { composes };
```

```
invariant ISOExceptionThrowItisSTATICAndMethodsMustHaveOneParameter:
self.method = ISOExceptionMethods::throwIt implies
self.parameters ->size()=1;
invariant ISOExceptionMethodsMustHaveOneParameter:
self.method = ISOExceptionMethods::ISOException implies
self.parameters ->size()=1;
invariant ISOExceptionSetReasonMethodsMustHaveOneParameter:
self.method = ISOExceptionMethods::setReason implies
self.parameters ->size()=1;
invariant IfISOExceptionUseTryCatchJustAddOneField:
self.exceptionUsageType = ExceptionUsageTypes::tryCatch implies
self.fields ->size()=1;
invariant IfISOExceptionUseThrowNewJustAddOneField:
self.exceptionUsageType = ExceptionUsageTypes::ThrowNew implies
self.fields ->size()=1;
invariant ISOExceptionEqualsMethodsMustHaveOneObjectTypeParameter:
self.method = ISOExceptionMethods::equals implies
self.parameters ->size()=1 and self.fields->forAll(x|x.dataType = DataTypes::Object);
invariant ISOExceptiongetReasonMethodsDontNeedAnyParameterButNeedShortField:
self.method = ISOExceptionMethods::getReason implies
self.parameters ->size()=0 and self.fields->forAll(x|x.dataType = DataTypes::short);
invariant ISOExceptionsetReasonMethodsNeed1ParameterButDontNeedAnyField:
self.method = ISOExceptionMethods::setReason implies
self.parameters ->size()=1 and self.fields->size()=0;
}
enum ISOExceptionMethods { serializable }
{
literal ISOException;
literal throwIt = 1;
literal getReason = 2;
literal setReason = 3;
literal equals = 4;
}
class ServiceException
{
property fields : Field[?] { composes };
attribute serviceExceptionField : ServiceExceptionFields[?];
attribute method : ServiceExceptionMethods[?];
attribute exceptionUsageType : ExceptionUsageTypes[?];
property parameters : Parameter[*] { ordered composes };
property instanceVariable : InstanceVariable[?] { composes };
invariant ServiceExceptionThrowItMethodsMustHaveOneParameter:
self.method = ServiceExceptionMethods::throwIt implies
self.parameters ->size()=1;
invariant ServiceExceptionSetReasonMethodsMustHaveOneParameter:
self.method = ServiceExceptionMethods::setReason implies
self.parameters ->size()=1;
invariant IfServiceExceptionUseTryCatchJustAddOneField:
self.exceptionUsageType = ExceptionUsageTypes::tryCatch implies
self.fields ->size()=1;
invariant IfServiceExceptionUseThrowNewJustAddOneField:
self.exceptionUsageType = ExceptionUsageTypes::ThrowNew implies
self.fields ->size()=1;
invariant ServiceExceptionEqualsMethodsMustHaveOneObjectTypeParameter:
self.method = ServiceExceptionMethods::equals implies
self.parameters ->size()=1 and self.parameters->forAll(x|x.dataType =
DataTypes::Object);
}
enum ServiceExceptionFields { serializable }
{
literal none;
literal CANNOT_ACCESS_IN_COMMAND = 1;
literal COMMAND_DATA_TOO_LONG = 2;
literal CANNOT_ACCESS_OUT_COMMAND = 3;
literal DISPATCH_TABLE_FULL = 4;
literal ILLEGAL_PARAM = 5;
literal REMOTE_OBJECT_NOT_EXPORTED = 6;
literal COMMAND_IS_FINISHED = 7;
}
enum ServiceExceptionMethods { serializable }
{
literal ServiceException;
```

```
literal throwIt = 1;
literal getReason = 2;
literal setReason = 3;
literal equals = 4;
}
class SystemException
{
property fields : Field[?] { composes };
attribute systemExceptionField : SystemExceptionFields[?];
attribute method : SystemExceptionMethods[?];
attribute exceptionUsageType : ExceptionUsageTypes[?];
property parameters : Parameter[*] { ordered composes };
property instanceVariable : InstanceVariable[?] { composes };
invariant SystemExceptionThrowItMethodsMustHaveOneParameterORsystemExceptionField:
self.method = SystemExceptionMethods::throwIt implies
self.parameters ->size()=1 or self.systemExceptionField ->size()=1;
invariant SystemExceptionSetReasonMethodsMustHaveOneParameter:
self.method = SystemExceptionMethods::setReason implies
self.parameters ->size()=1;
invariant IfSystemExceptionUseTryCatchJustAddOneField:
self.exceptionUsageType = ExceptionUsageTypes::tryCatch implies
self.fields ->size()=1;
invariant IfSystemExceptionUseThrowNewJustAddOneField:
self.exceptionUsageType = ExceptionUsageTypes::ThrowNew implies
self.fields ->size()=1;
invariant SystemExceptionEqualsMethodsMustHaveOneObjectTypeParameter:
self.method = SystemExceptionMethods::equals implies
self.parameters ->size()=1 and self.parameters->forAll(x|x.dataType =
DataTypes::Object);
invariant IfSystemExceptionUseIfElseJustAdd1ParameterOrsystemExceptionField:
self.method = ExceptionUsageTypes::ifElse implies
self.parameters ->size()=1 or self.systemExceptionField ->size()=1;
}
enum SystemExceptionFields { serializable }
{
literal none;
literal ILLEGAL_TRANSIENT = 1;
literal ILLEGAL_USE = 2;
literal NO_RESOURCE = 3;
literal NO_TRANSIENT_SPACE = 4;
literal ILLEGAL_AID = 5;
literal ILLEGAL_VALUE = 6;
}
enum SystemExceptionMethods { serializable }
{
literal SystemException;
literal throwIt = 1;
literal getReason = 2;
literal setReason = 3;
literal equals = 4;
}
class TransactionException
{
property fields : Field[?] { composes };
attribute transactionExceptionField : TransactionExceptionFields[?];
attribute method : TransactionExceptionMethods[?];
attribute exceptionUsageType : ExceptionUsageTypes[?];
property parameters : Parameter[*] { ordered composes };
property instanceVariable : InstanceVariable[?] { composes };
invariant TransactionExceptionThrowItMethodsMustHaveOneParameterOR1transactionExceptionF
ield:
self.method = TransactionExceptionMethods::throwIt implies
self.parameters -
>size()=1 or self.transactionExceptionField<>TransactionExceptionFields::none;
invariant TransactionExceptionSetReasonMethodsMustHave1ParameterOR1transactionExceptionF
ield:
self.method = TransactionExceptionMethods::setReason implies
self.parameters -
>size()=1 or self.transactionExceptionField<>TransactionExceptionFields::none;
invariant IfTransactionExceptionUseTryCatchJustAddOneField:
self.exceptionUsageType = ExceptionUsageTypes::tryCatch implies
self.fields ->size()=1;
```

```
invariant IfTransactionExceptionUseThrowNewJustAddOneField:
self.exceptionUsageType = ExceptionUsageTypes::ThrowNew implies
self.fields ->size()=1;
invariant TransactionExceptionEqualsMethodsMustHaveOneObjectTypeParameter:
self.method = TransactionExceptionMethods::equals implies
self.parameters ->size()=1 and self.parameters->forAll(x|x.dataType =
DataTypes::Object);
invariant IfTransactionExceptionUseIfElseJustAddOneFieldOR1transactionExceptionField:
self.exceptionUsageType = ExceptionUsageTypes::ifElse implies
self.fields -
>size()=1 or self.transactionExceptionField<>TransactionExceptionFields::none;
}
enum TransactionExceptionMethods { serializable }
{
literal TransactionException;
literal throwIt = 1;
literal getReason = 2;
literal setReason = 3;
literal equals = 4;
}
enum TransactionExceptionFields { serializable }
{
literal none;
literal INTERNAL_FAILURE = 1;
literal BUFFER_FULL = 2;
literal IN_PROGRESS = 3;
literal NOT_IN_PROGRESS = 4;
}
class ArithmeticException
{
property fields : Field[?] { composes };
attribute method : ArithmeticExceptionMethods[?];
attribute exceptionUsageType : ExceptionUsageTypes[?];
property parameters : Parameter[*] { ordered composes };
property instanceVariable : InstanceVariable[?] { composes };
invariant IfArithmeticExceptionUseTryCatchJustAddOneField:
self.exceptionUsageType = ExceptionUsageTypes::tryCatch implies
self.fields ->size()=1;
invariant IfArithmeticExceptionUseThrowNewJustAddOneField:
self.exceptionUsageType = ExceptionUsageTypes::ThrowNew implies
self.fields ->size()=1;
invariant ArithmeticExceptionEqualsMethodsMustHaveOneParameterAndBooleanfield:
self.method = ArithmeticExceptionMethods::equals implies
self.parameters ->size()=1 and self.fields->forAll(x|x.dataType = DataTypes::boolean);
invariant ArithmeticExceptionMethodDontNeedAnyParametersOrFields:
self.method = ArithmeticExceptionMethods::ArithmeticException implies
self.parameters->size()=0 and self.fields->size()=0;
}
enum ArithmeticExceptionMethods { serializable }
{
literal ArithmeticException;
literal equals = 1;
}
class ArrayStoreException
{
property fields : Field[?] { composes };
attribute method : ArrayStoreExceptionMethods[?];
attribute exceptionUsageType : ExceptionUsageTypes[?];
property parameters : Parameter[*] { ordered composes };
property instanceVariable : InstanceVariable[?] { composes };
invariant IfArrayStoreExceptionUseTryCatchJustAddOneField:
self.exceptionUsageType = ExceptionUsageTypes::tryCatch implies
self.fields ->size()=1;
invariant IfArrayStoreExceptionUseThrowNewJustAddOneField:
self.exceptionUsageType = ExceptionUsageTypes::ThrowNew implies
self.fields ->size()=1;
invariant ArrayStoreExceptionEqualsMethodsMustHaveOneObjectTypeParameter:
self.method = ArrayStoreExceptionMethods::equals implies
self.parameters ->size()=1 and self.parameters->forAll(x|x.dataType =
DataTypes::Object);
invariant ArrayStoreExceptionMethodDontNeedAnyParametersOrFields:
self.method = ArrayStoreExceptionMethods::ArrayStoreException implies
```

```
self.parameters->size()=0 or self.fields->size()=0;
}
enum ArrayStoreExceptionMethods { serializable }
{
literal ArrayStoreException;
literal equals = 1;
}
class ClassCastException
{
property fields : Field[?] { composes };
attribute method : ClassCastExceptionMethods[?];
attribute exceptionUsageType : ExceptionUsageTypes[?];
property parameters : Parameter[*] { ordered composes };
property instanceVariable : InstanceVariable[?] { composes };
invariant IfClassCastExceptionUseTryCatchJustAddOneField:
self.exceptionUsageType = ExceptionUsageTypes::tryCatch implies
self.fields ->size()=1;
invariant IfClassCastExceptionUseThrowNewJustAddOneField:
self.exceptionUsageType = ExceptionUsageTypes::ThrowNew implies
self.fields ->size()=1;
invariant ClassCastExceptionEqualsMethodsMustHaveOneParameterAndBooleanField:
self.method = ClassCastExceptionMethods::equals implies
self.parameters ->size()=1 and self.fields->forAll(x|x.dataType = DataTypes::boolean);
invariant ClassCastExceptionMethodDontNeedAnyParametersOrFields:
self.method = ClassCastExceptionMethods::ClassCastException implies
self.parameters->size()=0 or self.fields->size()=0;
}
enum ClassCastExceptionMethods { serializable }
{
literal ClassCastException;
literal equals = 1;
}
class IndexOutOfBoundsException
{
property fields : Field[?] { composes };
attribute method : IndexOutOfBoundsExceptionMethods[?];
attribute exceptionUsageType : ExceptionUsageTypes[?];
property parameters : Parameter[*] { ordered composes };
property instanceVariable : InstanceVariable[?] { composes };
invariant IfIndexOutOfBoundsExceptionUseTryCatchJustAddOneField:
self.exceptionUsageType = ExceptionUsageTypes::tryCatch implies
self.fields ->size()=1;
invariant IfIndexOutOfBoundsExceptionUseThrowNewJustAddOneField:
self.exceptionUsageType = ExceptionUsageTypes::ThrowNew implies
self.fields ->size()=1;
invariant IndexOutOfBoundsExceptionEqualsMethodsMustHaveOneObjectTypeParameter:
self.method = IndexOutOfBoundsExceptionMethods::equals implies
self.parameters ->size()=1 and self.parameters->forAll(x|x.dataType =
DataTypes::Object);
invariant IndexOutOfBoundsExceptionMethodsDoNotNeedAnyParameterOrFields:
self.method = IndexOutOfBoundsExceptionMethods::IndexOutOfBoundsException implies
self.parameters ->size()=0 or self.fields->size()=0;
}
enum IndexOutOfBoundsExceptionMethods { serializable }
{
literal IndexOutOfBoundsException;
literal equals = 1;
}
class ArrayIndexOutOfBoundsException
{
property fields : Field[?] { composes };
attribute method : ArrayIndexOutOfBoundsExceptionMethods[?];
attribute exceptionUsageType : ExceptionUsageTypes[?];
property parameters : Parameter[*] { ordered composes };
property instanceVariable : InstanceVariable[?] { composes };
invariant IfArrayIndexOutOfBoundsExceptionUseTryCatchJustAddOneField:
self.exceptionUsageType = ExceptionUsageTypes::tryCatch implies
self.fields ->size()=1;
invariant IfArrayIndexOutOfBoundsExceptionUseThrowNewJustAddOneField:
self.exceptionUsageType = ExceptionUsageTypes::ThrowNew implies
self.fields ->size()=1;
invariant ArrayIndexOutOfBoundsExceptionEqualsMethodsMustHaveOneObjectTypeParameter:
```

```
self.method = ArrayIndexOutOfBoundsExceptionMethods::equals implies
self.parameters ->size()=1 and self.parameters->forAll(x|x.dataType =
DataTypes::Object);
invariant ArrayIndexOutOfBoundsExceptionMethodsDoNotNeedAnyParameterOrFields:
self.method =
ArrayIndexOutOfBoundsExceptionMethods::ArrayIndexOutOfBoundsException implies
self.parameters ->size()=0 or self.fields->size()=0;
}
enum ArrayIndexOutOfBoundsExceptionMethods { serializable }
{
literal ArrayIndexOutOfBoundsException;
literal equals = 1;
}
class NegativeArraySizeException
{
property fields : Field[?] { composes };
attribute method : NegativeArraySizeExceptionMethods[?];
attribute exceptionUsageType : ExceptionUsageTypes[?];
property parameters : Parameter[*] { ordered composes };
property instanceVariable : InstanceVariable[?] { composes };
invariant IfNegativeArraySizeExceptionUseTryCatchJustAddOneField:
self.exceptionUsageType = ExceptionUsageTypes::tryCatch implies
self.fields ->size()=1;
invariant IfNegativeArraySizeExceptionUseThrowNewJustAddOneField:
self.exceptionUsageType = ExceptionUsageTypes::ThrowNew implies
self.fields ->size()=1;
invariant NegativeArraySizeExceptionEqualsMethodsMustHaveOneObjectTypeParameter:
self.method = NegativeArraySizeExceptionMethods::equals implies
self.parameters ->size()=1 and self.parameters->forAll(x|x.dataType =
DataTypes::Object);
invariant NegativeArraySizeExceptionDoNotNeedAnyParameterOrFields:
self.method = NegativeArraySizeExceptionMethods::NegativeArraySizeException implies
self.parameters ->size()=0 or self.fields->size()=0;
}
enum NegativeArraySizeExceptionMethods { serializable }
{
literal NegativeArraySizeException;
literal equals = 1;
}
class NullPointerException
{
property fields : Field[?] { composes };
attribute method : NullPointerExceptionMethods[?];
attribute exceptionUsageType : ExceptionUsageTypes[?];
property parameters : Parameter[*] { ordered composes };
property instanceVariable : InstanceVariable[?] { composes };
invariant IfNullPointerExceptionUseTryCatchJustAddOneField:
self.exceptionUsageType = ExceptionUsageTypes::tryCatch implies
self.fields ->size()=1;
invariant IfNullPointerExceptionUseThrowNewJustAddOneField:
self.exceptionUsageType = ExceptionUsageTypes::ThrowNew implies
self.fields ->size()=1;
invariant NullPointerExceptionEqualsMethodsMustHaveOneObjectTypeParameter:
self.method = NullPointerExceptionMethods::equals implies
self.parameters ->size()=1 and self.parameters->forAll(x|x.dataType =
DataTypes::Object);
invariant NullPointerExceptionDoNotNeedAnyParameterOrFields:
self.method = NullPointerExceptionMethods::NullPointerException implies
self.parameters ->size()=0 or self.fields->size()=0;
}
enum NullPointerExceptionMethods { serializable }
{
literal NullPointerException;
literal equals = 1;
}
class SecurityException
{
property fields : Field[?] { composes };
attribute method : SecurityExceptionMethods[?];
attribute exceptionUsageType : ExceptionUsageTypes[?];
property parameters : Parameter[*] { ordered composes };
property instanceVariable : InstanceVariable[?] { composes };
```

```
invariant IfSecurityExceptionUseTryCatchJustAddOneField:
self.exceptionUsageType = ExceptionUsageTypes::tryCatch implies
self.fields ->size()=1;
invariant IfSecurityExceptionUseThrowNewJustAddOneField:
self.exceptionUsageType = ExceptionUsageTypes::ThrowNew implies
self.fields ->size()=1;
invariant SecurityExceptionEqualsMethodsMustHaveOneObjectTypeParameter:
self.method = SecurityExceptionMethods::equals implies
self.parameters ->size()=1 and self.parameters->forAll(x|x.dataType =
DataTypes::Object);
invariant SecurityExceptionDoNotNeedAnyParameterOrFields:
self.method = SecurityExceptionMethods::SecurityException implies
self.parameters ->size()=0 or self.fields->size()=0;
}
enum SecurityExceptionMethods { serializable }
{
literal SecurityException;
literal equals = 1;
}
class AID
{
property fields : Field[?] { composes };
attribute method : AIDMethods[?];
property parameters : Parameter[*] { ordered composes };
property instanceVariable : InstanceVariable[?] { composes };
invariant AIDMethodNeedsThreeParameter:
self.method=AIDMethods::AID implies
self.parameters->size()=3;
invariant AIDequalsMethodNeedsOneOrThreeParameterAndNeedBooleanField:
self.method=AIDMethods::equals implies
(self.parameters->size()=1 or self.parameters->size()=3) and self.fields->forAll(e |
e.dataType <> DataTypes::boolean);
invariant AIDgetPartialBytesMethodNeeds4ParameterAndNeedByteField:
self.method=AIDMethods::getPartialBytes implies
self.parameters->size()=4 and self.fields->forAll(e | e.dataType <> DataTypes::byte);
invariant AIDRIDEqualsMethodNeeds1ParameterAndNeedBooleanFiel:
self.method=AIDMethods::RIDEquals implies
self.parameters->size()=1 and self.fields->forAll(e | e.dataType <> DataTypes::boolean);
invariant AIDpartialEqualsMethodNeeds3ParameterAndNeedBooleanField:
self.method=AIDMethods::partialEquals implies
self.parameters->size()=3 and self.fields->forAll(e | e.dataType <> DataTypes::boolean);
invariant AIDgetBytesMethodNeeds2ParameterAndNeedByteField:
self.method=AIDMethods::getBytes implies
self.parameters->size()=2 and self.fields->forAll(e | e.dataType <> DataTypes::byte);
}
enum AIDMethods { serializable }
{
literal AID;
literal equals = 1;
literal getPartialBytes = 2;
literal RIDEquals = 3;
literal partialEquals = 4;
literal getBytes = 5;
}
class Cipher
{
property fields : Field[?] { composes };
attribute cipherField : CipherFields[?];
attribute method : CipherMethods[?];
property parameters : Parameter[*] { ordered composes };
property instanceVariable : InstanceVariable[?] { composes };
invariant CipherdoFinalMethodNeeds5ParameterAndNeedShortFieldToReturnValue:
self.method=CipherMethods::doFinal implies
self.parameters->size()=5 and self.fields->size()=1 and self.fields->forAll(x|x.dataType
= DataTypes::short);
invariant CiphergetInstanceisSTATICAndMethodNeeds2ParameterAndCipherField:
self.method=CipherMethods::getInstance implies
self.parameters->size()=2 and self.fields->forAll(x|x.dataType = DataTypes::Cipher);
invariant CipherinitMethodNeeds2OR5ParameterButDontNeedAnyField:
self.method=CipherMethods::init implies
(self.parameters->size()=2 or self.parameters->size()=5) and self.fields->size()=0;
invariant CipherupdateMethodNeeds5ParameterAndNeedShortField:
```

```
self.method=CipherMethods::update implies
self.parameters->size()=5 and self.fields->forAll(x|x.dataType = DataTypes::short);
invariant CipherqualsMethodNeeds1CipherParameterToCompareAndABooleanFieldDontForgetTheIn
itializeField:
self.method=CipherMethods::equals implies
self.parameters->size()=1 and self.fields->forAll(x|x.dataType = DataTypes::boolean);
invariant CipherGetAlgorithmMethodNeeds1ByteField:
self.method=CipherMethods::getAlgorithm implies
self.fields->size()=1 and self.fields->forAll(x|x.dataType = DataTypes::byte);
}
enum CipherFields { serializable }
{
literal none;
literal ALG_DES_CBC_NOPAD = 1;
literal ALG_DES_CBC_ISO9797_M1 = 2;
literal ALG_DES_CBC_ISO9797_M2 = 3;
literal ALG_DES_CBC_PKCS5 = 4;
literal ALG_DES_ECB_NOPAD = 5;
literal ALG_DES_ECB_ISO9797_M1 = 6;
literal ALG_DES_ECB_ISO9797_M2 = 7;
literal ALG_DES_ECB_PKCS5 = 8;
literal ALG_RSA_ISO14888 = 9;
literal ALG_RSA_PKCS1 = 10;
literal ALG_RSA_ISO9796 = 11;
literal ALG_RSA_NOPAD = 12;
literal ALG_AES_BLOCK_128_CBC_NOPAD = 13;
literal ALG_AES_BLOCK_128_ECB_NOPAD = 14;
literal ALG_RSA_PKCS1_OAEP = 15;
literal ALG_KOREAN_SEED_ECB_NOPAD = 16;
literal ALG_KOREAN_SEED_CBC_NOPAD = 17;
literal MODE_DECRYPT = 18;
literal MODE_ENCRYPT = 19;
}
enum CipherMethods { serializable }
{
literal Cipher;
literal getInstance = 1;
literal init = 2;
literal getAlgorithm = 3;
literal doFinal = 4;
literal update = 5;
literal equals = 6;
}
class Checksum
{
property fields : Field[?] { composes };
attribute checksumField : ChecksumFields[?];
attribute method : ChecksumMethods[?];
property parameters : Parameter[*] { ordered composes };
property instanceVariable : InstanceVariable[?] { composes };
invariant ChecksumdoFinalMethodNeeds5ParameterAndShortField:
self.method=ChecksumMethods::doFinal implies
self.parameters->size()=5 and self.fields->forAll(x|x.dataType = DataTypes::short);
invariant ChecksumgetInstanceMethodNeeds2ParameterAndChecksumField:
self.method=ChecksumMethods::getInstance implies
self.parameters->size()=2 and self.fields->forAll(x|x.dataType = DataTypes::Checksum);
invariant ChecksuminitMethodNeeds3ParameterAndButDontNeedAnyField:
self.method=ChecksumMethods::init implies
self.parameters->size()=3 and self.fields->size()=0;
invariant ChecksumgetAlgorithmMethodDontNeedAnyParameterButNeedsByteField:
self.method=ChecksumMethods::getAlgorithm implies
self.parameters->size()=0 and self.fields->forAll(x|x.dataType = DataTypes::byte);
invariant ChecksumgetInstanceMethodNeeds2Parameter:
self.method=ChecksumMethods::getInstance implies
self.parameters->size()=2;
invariant ChecksuminitMethodNeeds2Parameter:
self.method=ChecksumMethods::init implies
self.parameters->size()=2;
invariant ChecksumupdateMethodNeeds3Parameter:
self.method=ChecksumMethods::update implies
self.parameters->size()=3;
invariant ChecksumqualsMethodNeeds1Parameter:
```

```
self.method=ChecksumMethods::equals implies
self.parameters->size()=1;
}
enum ChecksumFields { serializable }
{
literal none;
literal ALG_ISO3309_CRC16 = 1;
literal ALG_ISO3309_CRC32 = 2;
}
enum ChecksumMethods { serializable }
{
literal Checksum;
literal getInstance = 1;
literal init = 2;
literal getAlgorithm = 3;
literal doFinal = 4;
literal update = 5;
literal equals = 6;
}
class KeyAgreement
{
property fields : Field[?] { composes };
attribute keyAgreementField : KeyAgreementFields[?];
attribute method : KeyAgreementMethods[?];
property parameters : Parameter[*] { ordered composes };
property instanceVariable : InstanceVariable[?] { composes };
invariant KeyAgreementgetInstanceMethodNeeds2Parameter:
self.method=KeyAgreementMethods::getInstance implies
self.parameters->size()=2;
invariant KeyAgreementinitMethodNeeds1Parameter:
self.method=KeyAgreementMethods::init implies
self.parameters->size()=1;
invariant KeyAgreementgenerateSecretMethodNeeds5Parameter:
self.method=KeyAgreementMethods::generateSecret implies
self.parameters->size()=5;
}
enum KeyAgreementFields { serializable }
{
literal none;
literal ALG_EC_SVDP_DH = 1;
literal ALG_EC_SVDP_DH_KDF = 2;
literal ALG_EC_SVDP_DHC = 3;
literal ALG_EC_SVDP_DHC_KDF = 4;
literal ALG_EC_SVDP_DH_PLAIN = 5;
literal ALG_EC_SVDP_DHC_PLAIN = 6;
}
enum KeyAgreementMethods { serializable }
{
literal KeyAgreement;
literal getInstance = 1;
literal init = 2;
literal getAlgorithm = 3;
literal generateSecret = 4;
}
class KeyBuilder
{
property fields : Field[?] { composes };
attribute keyBuilderField : KeyBuilderFields[?];
attribute method : KeyBuilderMethods[?];
property parameters : Parameter[*] { ordered composes };
property instanceVariable : InstanceVariable[?] { composes };
invariant KeyBuilderbuildKeyMethodNeeds3Parameter:
self.method=KeyBuilderMethods::buildKey implies
self.parameters->size()=3;
invariant KeyBuilderqualsMethodNeeds1Parameter:
self.method=KeyBuilderMethods::equals implies
self.parameters->size()=1;
}
enum KeyBuilderFields { serializable }
{
literal none;
literal TYPE_DES_TRANSIENT_RESET = 1;
```

```
literal TYPE_DES_TRANSIENT_DESELECT = 2;
literal TYPE_DES = 3;
literal TYPE_RSA_PUBLIC = 4;
literal TYPE_RSA_PRIVATE = 5;
literal TYPE_RSA_CRT_PRIVATE = 6;
literal TYPE_DSA_PUBLIC = 7;
literal TYPE_DSA_PRIVATE = 8;
literal TYPE_EC_F2M_PUBLIC = 9;
literal TYPE_EC_F2M_PRIVATE = 10;
literal TYPE_EC_FP_PUBLIC = 11;
literal TYPE_EC_FP_PRIVATE = 12;
literal TYPE_AES_TRANSIENT_RESET = 13;
literal TYPE_AES_TRANSIENT_DESELECT = 14;
literal TYPE_AES = 15;
literal TYPE_KOREAN_SEED_TRANSIENT_RESET = 16;
literal TYPE_KOREAN_SEED_TRANSIENT_DESELECT = 17;
literal TYPE_KOREAN_SEED = 18;
literal TYPE_HMAC_TRANSIENT_RESET = 19;
literal TYPE_HMAC_TRANSIENT_DESELECT = 20;
literal TYPE_HMAC = 21;
literal LENGTH_DES = 22;
literal LENGTH_DES3_2KEY = 23;
literal LENGTH_DES3_3KEY = 24;
literal LENGTH_RSA_512 = 25;
literal LENGTH_RSA_736 = 26;
literal LENGTH_RSA_768 = 27;
literal LENGTH_RSA_896 = 28;
literal LENGTH_RSA_1024 = 29;
literal LENGTH_RSA_1280 = 30;
literal LENGTH_RSA_1536 = 31;
literal LENGTH_RSA_1984 = 32;
literal LENGTH_RSA_2048 = 33;
literal LENGTH_DSA_512 = 34;
literal LENGTH_DSA_768 = 35;
literal LENGTH_DSA_1024 = 36;
literal LENGTH_EC_FP_112 = 37;
literal LENGTH_EC_F2M_113 = 38;
literal LENGTH_EC_FP_128 = 39;
literal LENGTH_EC_F2M_131 = 40;
literal LENGTH_EC_FP_160 = 41;
literal LENGTH_EC_F2M_163 = 42;
literal LENGTH_EC_FP_192 = 43;
literal LENGTH_EC_F2M_193 = 44;
literal LENGTH_AES_128 = 45;
literal LENGTH_AES_192 = 46;
literal LENGTH_AES_256 = 47;
literal LENGTH_KOREAN_SEED_128 = 48;
literal LENGTH_HMAC_SHA_1_BLOCK_64 = 49;
literal LENGTH_HMAC_SHA_256_BLOCK_64 = 50;
literal LENGTH_HMAC_SHA_384_BLOCK_128 = 51;
literal LENGTH_HMAC_SHA_512_BLOCK_128 = 52;
}
enum KeyBuilderMethods { serializable }
{
literal buildKey;
literal equals = 1;
}
class KeyPair
{
property fields : Field[?] { composes };
attribute keyPairField : KeyPairFields[?];
attribute method : KeyPairMethods[?];
property parameters : Parameter[*] { ordered composes };
property instanceVariable : InstanceVariable[?] { composes };
invariant KeyPairMethodNeeds2ParameterOR1ParameterAnd1keyPairField:
self.method=KeyPairMethods::KeyPair implies
self.parameters->size()=2 or (self.parameters->size()=1 and self.keyPairField <>
KeyPairFields::none);
invariant
KeyPairqualsMethodNeeds1KeyPairParameterAndABooleanField('KeyPair Equals Method Return
Type is Boolean and Need An Object to compare'):
self.method=KeyPairMethods::equals implies
```

```
self.parameters->size()=1;
}
enum KeyPairFields { serializable }
{
literal none;
literal ALG_RSA = 1;
literal ALG_RSA_CRT = 2;
literal ALG_DSA = 3;
literal ALG_EC_F2M = 4;
literal ALG_EC_FP = 5;
}
enum KeyPairMethods { serializable }
{
literal KeyPair;
literal genKeyPair = 1;
literal getPublic = 2;
literal getPrivate = 3;
literal equals = 4;
}
class MessageDigest
{
property fields : Field[?] { composes };
attribute messageDigestField : MessageDigestFields[?];
attribute method : MessageDigestMethods[?];
property parameters : Parameter[*] { ordered composes };
property instanceVariable : InstanceVariable[?] { composes };
invariant MessageDigestdoFinalMethodNeeds5ParameterIFUSE1MessageDigestFieldSoMethodParam
eterSortingWillHasAProblem:
self.method=MessageDigestMethods::doFinal implies
self.parameters->size()=5 or (self.parameters-
>size()=4 and self.messageDigestField<>MessageDigestFields::none);
invariant MessageDigestgetInitializedMessageDigestInstanceMethodNeeds2ParameterOR1Parame
terAND1messageDigestField:
self.method=MessageDigestMethods::getInitializedMessageDigestInstance implies
self.parameters->size()=2 or (self.parameters-
>size()=1 and self.messageDigestField<>MessageDigestFields::none);
invariant MessageDigestgetInstanceMethodNeeds2ParameterOR1ParameterAND1messageDigestFiel
d:
self.method=MessageDigestMethods::getInstance implies
self.parameters->size()=2 or (self.parameters-
>size()=1 and self.messageDigestField<>MessageDigestFields::none);
invariant MessageDigestqualsMethodNeeds1Parameter:
self.method=MessageDigestMethods::equals implies
self.parameters->size()=1;
}
enum MessageDigestFields { serializable }
{
literal none;
literal ALG_SHA = 1;
literal ALG_MD5 = 2;
literal ALG_RIPEMD160 = 3;
literal ALG_SHA_256 = 4;
literal ALG_SHA_384 = 5;
literal ALG_SHA_512 = 6;
literal ALG_SHA_224 = 7;
literal LENGTH_MD5 = 8;
literal LENGTH_RIPEMD160 = 9;
literal LENGTH_SHA = 10;
literal LENGTH_SHA_256 = 11;
literal LENGTH_SHA_384 = 12;
literal LENGTH_SHA_512 = 13;
}
enum MessageDigestMethods { serializable }
{
literal MessageDigest;
literal getInstance = 1;
literal getInitializedMessageDigestInstance = 2;
literal getAlgorithm = 3;
literal getLength = 4;
literal doFinal = 5;
literal update = 6;
literal reset = 7;
```

```
literal equals = 8;
}
class InitializedMessageDigest
{
property fields : Field[?] { composes };
attribute initializedMessageDigestField : InitializedMessageDigestFields[?];
attribute method : InitializedMessageDigestMethods[?];
property parameters : Parameter[*] { ordered composes };
property instanceVariable : InstanceVariable[?] { composes };
invariant InitializedMessageDigestdoFinalMethodNeeds5ParameterAndShortField:
self.method=InitializedMessageDigestMethods::doFinal implies
self.parameters->size()=5 and self.fields->forAll(x|x.dataType = DataTypes::short);
invariant
InitializedMessageDigestgetInitializedMessageDigestInstanceMethodNeeds2ParameterAndIniti
alizedMessageDigestField:
self.method=InitializedMessageDigestMethods::getInitializedMessageDigestInstance implies
self.parameters->size()=2 and self.fields->forAll(x|x.dataType =
DataTypes::InitializedMessageDigest);
invariant InitializedMessageDigestgetInstanceMethodNeeds2ParameterAndMessageDigestField:

self.method=InitializedMessageDigestMethods::getInstance implies
self.parameters->size()=2 and self.fields->forAll(x|x.dataType =
DataTypes::MessageDigest);
invariant InitializedMessageDigestsetInitialDigestMethodNeeds6ParameterButDontNeedAnyFie
ld:
self.method=InitializedMessageDigestMethods::setInitialDigest implies
self.parameters->size()=6 and self.fields->size()=0;
invariant InitializedMessageDigestqualsMethodNeeds1Parameter:
self.method=InitializedMessageDigestMethods::equals implies
self.parameters->size()=1;
invariant InitializedMessageDigestgetAlgorithmMethodNeedByteFieldButDontNeedAnyParameter
:
self.method=InitializedMessageDigestMethods::getAlgorithm implies
self.parameters->size()=0 and self.fields->forAll(x|x.dataType = DataTypes::byte);
invariant InitializedMessageDigestgetLengthMethodNeedsShortFieldButDontNeedAnyParameters
:
self.method=InitializedMessageDigestMethods::getLength implies
self.parameters->size()=0 and self.fields->forAll(x|x.dataType = DataTypes::short);
invariant InitializedMessageDigestresetMethodDontNeedAnyParameterOrField:
self.method=InitializedMessageDigestMethods::reset implies
self.parameters->size()=0 and self.fields->size()=0;
invariant InitializedMessageDigestupdateMethodNeeds3ParameterButDontNeedAnyField:
self.method=InitializedMessageDigestMethods::update implies
self.parameters->size()=3 and self.fields->size()=0;
}
enum InitializedMessageDigestFields { serializable }
{
literal none;
literal ALG_SHA = 1;
literal ALG_MD5 = 2;
literal ALG_RIPEMD160 = 3;
literal ALG_SHA_256 = 4;
literal ALG_SHA_384 = 5;
literal ALG_SHA_512 = 6;
literal ALG_SHA_224 = 7;
literal LENGTH_MD5 = 8;
literal LENGTH_RIPEMD160 = 9;
literal LENGTH_SHA = 10;
literal LENGTH_SHA_256 = 11;
literal LENGTH_SHA_384 = 12;
literal LENGTH_SHA_512 = 13;
}
enum InitializedMessageDigestMethods { serializable }
{
literal InitializedMessageDigest;
literal getInstance = 1;
literal getInitializedMessageDigestInstance = 2;
literal getAlgorithm = 3;
literal getLength = 4;
literal doFinal = 5;
literal update = 6;
literal reset = 7;
```

```
literal equals = 8;
literal setInitialDigest = 9;
}
class RandomData
{
property fields : Field[?] { composes };
attribute randomDataField : RandomDataFields[?];
attribute method : RandomDataMethods[?];
property parameters : Parameter[*] { ordered composes };
property instanceVariable : InstanceVariable[?] { composes };
invariant RandomDataequalsMethodNeeds1Parameter:
self.method=RandomDataMethods::equals implies
self.parameters->size()=1;
invariant RandomDatagetInstanceMethodNeeds1ParameterOR1randomDataField:
self.method=RandomDataMethods::getInstance implies
self.parameters->size()=1 or self.randomDataField <> RandomDataFields::none;
invariant RandomDatagenerateDataMethodNeeds3Parameter:
self.method=RandomDataMethods::generateData implies
self.parameters->size()=3;
invariant RandomDatasetSeedMethodNeeds3Parameter:
self.method=RandomDataMethods::setSeed implies
self.parameters->size()=3;
}
enum RandomDataFields { serializable }
{
literal none;
literal ALG_PSEUDO_RANDOM = 1;
literal ALG_SECURE_RANDOM = 2;
}
enum RandomDataMethods { serializable }
{
literal RandomData;
literal getInstance = 1;
literal generateData = 2;
literal setSeed = 3;
literal equals = 4;
}
class Signature
{
property fields : Field[?] { composes };
attribute signatureField : SignatureFields[?];
attribute method : SignatureMethods[?];
property parameters : Parameter[*] { ordered composes };
property instanceVariable : InstanceVariable[?] { composes };
invariant SignatureequalsMethodNeeds1Parameter:
self.method=SignatureMethods::equals implies
self.parameters->size()=1;
invariant SignaturegetInstanceMethodNeeds2ParameterOR1ParameterAnd1signatureField:
self.method=SignatureMethods::getInstance implies
self.parameters->size()=2 or (self.parameters-
>size()=1 and self.signatureField<>SignatureFields::none);
invariant SignatureinitMethodNeeds2OR5ParameterOR1ParameterAND1signatureFieldOR4Paramete
AND1signatureField:
self.method=SignatureMethods::init implies
self.parameters->size()=2 or self.parameters->size()=5 or (self.parameters-
>size()=1 and self.signatureField<>SignatureFields::none) or (self.parameters-
>size()=4 and self.signatureField<>SignatureFields::none);
invariant SignatureupdateMethodNeeds3Parameter:
self.method=SignatureMethods::update implies
self.parameters->size()=3;
invariant SignaturesignMethodNeeds5Parameter:
self.method=SignatureMethods::sign implies
self.parameters->size()=5;
invariant SignatureverifyMethodNeeds6Parameter:
self.method=SignatureMethods::verify implies
self.parameters->size()=6;
}
enum SignatureFields { serializable }
{
literal none;
literal ALG_DES_MAC4_NOPAD = 1;
literal ALG_DES_MAC8_NOPAD = 2;
```

```
literal ALG_DES_MAC4_ISO9797_M1 = 3;
literal ALG_DES_MAC8_ISO9797_M1 = 4;
literal ALG_DES_MAC4_ISO9797_M2 = 5;
literal ALG_DES_MAC8_ISO9797_M2 = 6;
literal ALG_DES_MAC4_PKCS5 = 7;
literal ALG_DES_MAC8_PKCS5 = 8;
literal ALG_RSA_SHA_ISO9796 = 9;
literal ALG_RSA_SHA_PKCS1 = 10;
literal ALG_RSA_MD5_PKCS1 = 11;
literal ALG_RSA_RIPEMD160_ISO9796 = 12;
literal ALG_RSA_RIPEMD160_PKCS1 = 11;
literal ALG_DSA_SHA = 13;
literal ALG_RSA_SHA_RFC2409 = 14;
literal ALG_RSA_MD5_RFC2409 = 15;
literal ALG_ECDSA_SHA = 16;
literal ALG_AES_MAC_128_NOPAD = 17;
literal ALG_DES_MAC4_ISO9797_1_M2_ALG3 = 18;
literal ALG_DES_MAC8_ISO9797_1_M2_ALG3 = 19;
literal ALG_RSA_SHA_PKCS1_PSS = 20;
literal ALG_RSA_MD5_PKCS1_PSS = 21;
literal ALG_RSA_RIPEMD160_PKCS1_PSS = 22;
literal ALG_HMAC_SHA1 = 23;
literal ALG_HMAC_SHA_256 = 24;
literal ALG_HMAC_SHA_384 = 25;
literal ALG_HMAC_SHA_512 = 26;
literal ALG_HMAC_MD5 = 27;
literal ALG_HMAC_RIPEMD160 = 28;
literal ALG_RSA_SHA_ISO9796_MR = 29;
literal ALG_RSA_RIPEMD160_ISO9796_MR = 30;
literal ALG_KOREAN_SEED_MAC_NOPAD = 31;
literal ALG_ECDSA_SHA_256 = 32;
literal ALG_ECDSA_SHA_384 = 33;
literal ALG_AES_MAC_192_NOPAD = 34;
literal ALG_AES_MAC_256_NOPAD = 35;
literal ALG_ECDSA_SHA_224 = 36;
literal ALG_ECDSA_SHA_512 = 37;
literal ALG_RSA_SHA_224_PKCS1 = 38;
literal ALG_RSA_SHA_256_PKCS1 = 39;
literal ALG_RSA_SHA_384_PKCS1 = 40;
literal ALG_RSA_SHA_512_PKCS1 = 41;
literal ALG_RSA_SHA_224_PKCS1_PSS = 42;
literal ALG_RSA_SHA_256_PKCS1_PSS = 43;
literal ALG_RSA_SHA_384_PKCS1_PSS = 44;
literal ALG_RSA_SHA_512_PKCS1_PSS = 45;
literal MODE_SIGN = 46;
literal MODE_VERIFY = 47;
}
enum SignatureMethods { serializable }
{
literal Signature;
literal getInstance = 1;
literal init = 2;
literal getAlgorithm = 3;
literal getLength = 4;
literal update = 5;
literal sign = 6;
literal verify = 7;
literal equals = 8;
}
class DSAKey
{
property fields : Field[?] { composes };
attribute method : DSAKeyMethods[?];
property parameters : Parameter[*] { ordered composes };
property instanceVariable : InstanceVariable[?] { composes };
invariant DSAKeysetPMethodNeeds3ParameterButDontNeedAnyField:
self.method=DSAKeyMethods::setP implies
self.parameters->size()=3 and self.fields->size()=0;
invariant DSAKeysetQMethodNeeds3ParameterButDontNeedAnyField:
self.method=DSAKeyMethods::setQ implies
self.parameters->size()=3 and self.fields->size()=0;
invariant DSAKeysetGMethodNeeds3ParameterButDontNeedAnyField:
```

```
self.method=DSAKeyMethods::setG implies
self.parameters->size()=3 and self.fields->size()=0;
invariant DSAKeygetPMethodNeeds2ParameterAndShortField:
self.method=DSAKeyMethods::getP implies
self.parameters->size()=2 and self.fields->forAll(x|x.dataType = DataTypes::short);
invariant DSAKeygetQMethodNeeds2ParameterAndShortField:
self.method=DSAKeyMethods::getQ implies
self.parameters->size()=2 and self.fields->forAll(x|x.dataType = DataTypes::short);
invariant DSAKeygetGMethodNeeds2ParameterAndNeedShortField:
self.method=DSAKeyMethods::getG implies
self.parameters->size()=2 and self.fields->forAll(x|x.dataType = DataTypes::short);
}
enum DSAKeyMethods { serializable }
{
literal setP;
literal setQ = 1;
literal setG = 2;
literal getP = 3;
literal getQ = 4;
literal getG = 5;
}
class DSAPrivateKey
{
property fields : Field[?] { composes };
attribute method : DSAPrivateKeyMethods[?];
property parameters : Parameter[*] { ordered composes };
property instanceVariable : InstanceVariable[?] { composes };
invariant DSAPrivateKeysetPMethodNeeds3ParameterButDontNeedAnyField:
self.method=DSAPrivateKeyMethods::setP implies
self.parameters->size()=3 and self.fields->size()=0;
invariant DSAPrivateKeysetQMethodNeeds3ParameterButDontNeedAnyField:
self.method=DSAPrivateKeyMethods::setQ implies
self.parameters->size()=3 and self.fields->size()=0;
invariant DSAPrivateKeysetGMethodNeeds3ParameterButDontNeedAnyField:
self.method=DSAPrivateKeyMethods::setG implies
self.parameters->size()=3 and self.fields->size()=0;
invariant DSAPrivateKeygetPMethodNeeds2ParameterAndShortField:
self.method=DSAPrivateKeyMethods::getP implies
self.parameters->size()=2 and self.fields->forAll(x|x.dataType = DataTypes::short);
invariant DSAPrivateKeygetQMethodNeeds2ParameterAndShortField:
self.method=DSAPrivateKeyMethods::getQ implies
self.parameters->size()=2 and self.fields->forAll(x|x.dataType = DataTypes::short);
invariant DSAPrivateKeygetGMethodNeeds2ParameterAndShortField:
self.method=DSAPrivateKeyMethods::getG implies
self.parameters->size()=2 and self.fields->forAll(x|x.dataType = DataTypes::short);
invariant DSAPrivateKeygsetXMethodNeeds3ParameterButDontNeedField:
self.method=DSAPrivateKeyMethods::setX implies
self.parameters->size()=3 and self.fields->size()=0;
invariant DSAPrivateKeygetXMethodNeeds2ParameterAndShortField:
self.method=DSAPrivateKeyMethods::getX implies
self.parameters->size()=2 and self.fields->forAll(x|x.dataType = DataTypes::short);
}
enum DSAPrivateKeyMethods { serializable }
{
literal setX;
literal getX = 1;
literal clearKey = 2;
literal getSize = 3;
literal getType = 4;
literal isInitialized = 5;
literal setP = 6;
literal setQ = 7;
literal setG = 8;
literal getP = 9;
literal getQ = 10;
literal getG = 11;
}
class DSAPublicKey
{
property fields : Field[?] { composes };
attribute method : DSAPublicKeyMethods[?];
property parameters : Parameter[*] { ordered composes };
```

```
property instanceVariable : InstanceVariable[?] { composes };
invariant DSAPublicKeyMethodssetPMethodNeeds3ParameterButDontNeedAnyField:
self.method=DSAPublicKeyMethods::setP implies
self.parameters->size()=3 and self.fields->size()=0;
invariant DSAPublicKeyMethodssetQMethodNeeds3ParameterButDontNeedAnyField:
self.method=DSAPublicKeyMethods::setQ implies
self.parameters->size()=3 and self.fields->size()=0;
invariant DSAPublicKeyMethodssetGMethodNeeds3ParameterButDontNeedAnyField:
self.method=DSAPublicKeyMethods::setG implies
self.parameters->size()=3 and self.fields->size()=0;
invariant DSAPublicKeyMethodsgetPMethodNeeds2ParameterAndShortField:
self.method=DSAPublicKeyMethods::getP implies
self.parameters->size()=2 and self.fields->forAll(x|x.dataType = DataTypes::short);
invariant DSAPublicKeyMethodsgetQMethodNeeds2ParameterAndShortField:
self.method=DSAPublicKeyMethods::getQ implies
self.parameters->size()=2 and self.fields->forAll(x|x.dataType = DataTypes::short);
invariant DSAPublicKeyMethodsgetGMethodNeeds2ParameterAndShortField:
self.method=DSAPublicKeyMethods::getG implies
self.parameters->size()=2 and self.fields->forAll(x|x.dataType = DataTypes::short);
invariant DSAPublicKeygsetXMethodNeeds3ParameterButDontNeedAnyField:
self.method=DSAPublicKeyMethods::setY implies
self.parameters->size()=3 and self.fields->size()=0;
invariant DSAPublicKeygetXMethodNeeds2ParameterAndNeedShortField:
self.method=DSAPublicKeyMethods::getY implies
self.parameters->size()=2 and self.fields->forAll(x|x.dataType = DataTypes::short);
}
enum DSAPublicKeyMethods { serializable }
{
literal setY;
literal getY = 1;
literal clearKey = 2;
literal getSize = 3;
literal getType = 4;
literal isInitialized = 5;
literal setP = 6;
literal setQ = 7;
literal setG = 8;
literal getP = 9;
literal getQ = 10;
literal getG = 11;
}
class ECKey
{
property fields : Field[?] { composes };
attribute method : ECKeyMethods[?];
property parameters : Parameter[*] { ordered composes };
property instanceVariable : InstanceVariable[?] { composes };
invariant ECKeysetFieldFPMethodNeeds3ParameterButDontNeedAnyField:
self.method=ECKeyMethods::setFieldFP implies
self.parameters->size()=3 and self.fields->size()=0;
invariant ECKeysetsetFieldF2MMethodNeeds1OR3ParameterButDontNeedAnyField:
self.method=ECKeyMethods::setFieldF2M implies
(self.parameters->size()=1 or self.parameters->size()=3) and self.fields->size()=0;
invariant ECKeysetAMethodNeeds3ParameterButDontNeedAnyField:
self.method=ECKeyMethods::setA implies
self.parameters->size()=3 and self.fields->size()=0;
invariant ECKeysetBMethodNeeds3ParameterButDontNeedAnyField:
self.method=ECKeyMethods::setB implies
self.parameters->size()=3 and self.fields->size()=0;
invariant ECKeysetGMethodNeeds3ParameterButDontNeedAnyField:
self.method=ECKeyMethods::setG implies
self.parameters->size()=3 and self.fields->size()=0;
invariant ECKeysetRMethodNeeds3ParameterButDontNeedAnyField:
self.method=ECKeyMethods::setR implies
self.parameters->size()=3 and self.fields->size()=0;
invariant ECKeysetKMethodNeeds1ParameterButDontNeedAnyField:
self.method=ECKeyMethods::setK implies
self.parameters->size()=1 and self.fields->size()=0;
invariant ECKeygetFieldMethodNeeds2ParameterAndNeedShortField:
self.method=ECKeyMethods::getField implies
self.parameters->size()=2 and self.fields->forAll(x|x.dataType = DataTypes::short);
invariant ECKeygetAMethodNeeds2ParameterAndNeedShortField:
```

```
self.method=ECKeyMethods::getA implies
self.parameters->size()=2 and self.fields->forAll(x|x.dataType = DataTypes::short);
invariant ECKeygetBMethodNeeds2ParameterAndNeedShortField:
self.method=ECKeyMethods::getB implies
self.parameters->size()=2 and self.fields->forAll(x|x.dataType = DataTypes::short);
invariant ECKeygetGMethodNeeds2ParameterAndNeedShortField:
self.method=ECKeyMethods::getG implies
self.parameters->size()=2 and self.fields->forAll(x|x.dataType = DataTypes::short);
invariant ECKeygetRMethodNeeds2ParameterAndNeedShortField:
self.method=ECKeyMethods::getR implies
self.parameters->size()=2 and self.fields->forAll(x|x.dataType = DataTypes::short);
invariant ECKeygetKMethodNeedsShortFieldButDontNeedAnyParameters:
self.method=ECKeyMethods::getK implies
self.parameters->size()=0 and self.fields->forAll(x|x.dataType = DataTypes::short);
}
enum ECKeyMethods { serializable }
{
literal setFieldFP;
literal setFieldF2M = 1;
literal setA = 2;
literal setB = 3;
literal setG = 4;
literal setR = 5;
literal setK = 6;
literal getField = 7;
literal getA = 8;
literal getB = 9;
literal getG = 10;
literal getR = 11;
literal getK = 12;
}
class ECPrivateKey
{
property fields : Field[?] { composes };
attribute method : ECPrivateKeyMethods[?];
property parameters : Parameter[*] { ordered composes };
property instanceVariable : InstanceVariable[?] { composes };
invariant ECPrivateKeysetSMethodNeeds3ParameterButDontNeedAnyField:
self.method=ECPrivateKeyMethods::setS implies
self.parameters->size()=3 and self.fields->size()=0;
invariant ECPrivateKeygetSMethodNeeds2ParameterAndNeedShortField:
self.method=ECPrivateKeyMethods::getS implies
self.parameters->size()=2 and self.fields->forAll(x|x.dataType = DataTypes::short);
invariant ECPrivateKeysetFieldFPMethodNeeds3ParameterButDontNeedAnyField:
self.method=ECPrivateKeyMethods::setFieldFP implies
self.parameters->size()=3 and self.fields->size()=0;
invariant ECPrivateKeysetsetFieldF2MMethodNeeds1OR3ParameterButDontNeedAnyField:
self.method=ECPrivateKeyMethods::setFieldF2M implies
(self.parameters->size()=1 or self.parameters->size()=3) and self.fields->size()=0;
invariant ECPrivateKeysetAMethodNeeds3ParameterButDontNeedAnyField:
self.method=ECPrivateKeyMethods::setA implies
self.parameters->size()=3 and self.fields->size()=0;
invariant ECPrivateKeysetBMethodNeeds3ParameterButDontNeedAnyField:
self.method=ECPrivateKeyMethods::setB implies
self.parameters->size()=3 and self.fields->size()=0;
invariant ECPrivateKeysetGMethodNeeds3ParameterButDontNeedAnyField:
self.method=ECPrivateKeyMethods::setG implies
self.parameters->size()=3 and self.fields->size()=0;
invariant ECPrivateKeysetRMethodNeeds3ParameterButDontNeedAnyField:
self.method=ECPrivateKeyMethods::setR implies
self.parameters->size()=3 and self.fields->size()=0;
invariant ECPrivateKeysetKMethodNeeds1ParameterButDontNeedAnyField:
self.method=ECPrivateKeyMethods::setK implies
self.parameters->size()=1 and self.fields->size()=0;
invariant ECPrivateKeygetFieldMethodNeeds2ParameterAndNeedShortField:
self.method=ECPrivateKeyMethods::getField implies
self.parameters->size()=2 and self.fields->forAll(x|x.dataType = DataTypes::short);
invariant ECPrivateKeyNeeds2ParameterAndNeedShortField:
self.method=ECPrivateKeyMethods::getA implies
self.parameters->size()=2 and self.fields->forAll(x|x.dataType = DataTypes::short);
invariant ECPrivateKeygetBMethodNeeds2ParameterAndNeedShortField:
self.method=ECPrivateKeyMethods::getB implies
```

```
self.parameters->size()=2 and self.fields->forAll(x|x.dataType = DataTypes::short);
invariant ECPrivateKeygetGMethodNeeds2ParameterAndNeedShortField:
self.method=ECPrivateKeyMethods::getG implies
self.parameters->size()=2 and self.fields->forAll(x|x.dataType = DataTypes::short);
invariant ECPrivateKeygetRMethodNeeds2ParameterAndNeedShortField:
self.method=ECPrivateKeyMethods::getR implies
self.parameters->size()=2 and self.fields->forAll(x|x.dataType = DataTypes::short);
invariant ECPrivateKeygetKMethodNeedsShortFieldButDontNeedAnyParameters:
self.method=ECPrivateKeyMethods::getK implies
self.parameters->size()=0 and self.fields->forAll(x|x.dataType = DataTypes::short);
}
enum ECPrivateKeyMethods { serializable }
{
literal setS;
literal getS = 1;
literal clearKey = 2;
literal getSize = 3;
literal getType = 4;
literal isInitialized = 5;
literal setFieldFP = 6;
literal setFieldF2M = 7;
literal setA = 8;
literal setB = 9;
literal setG = 10;
literal setR = 11;
literal setK = 12;
literal getField = 13;
literal getA = 14;
literal getB = 15;
literal getG = 16;
literal getR = 17;
literal getK = 18;
}
class ECPublicKey
{
property fields : Field[?] { composes };
attribute method : ECPublicKeyMethods[?];
property parameters : Parameter[*] { ordered composes };
property instanceVariable : InstanceVariable[?] { composes };
invariant ECPublicKeysetWMethodNeeds3ParameterButDontNeedAnyField:
self.method=ECPublicKeyMethods::setW implies
self.parameters->size()=3 and self.fields->size()=0;
invariant ECPublicKeygetWMethodNeeds2ParameterAndNeedShortField:
self.method=ECPublicKeyMethods::getW implies
self.parameters->size()=2 and self.fields->forAll(x|x.dataType = DataTypes::short);
invariant ECPrivateKeysetFieldFPMethodNeeds3ParameterButDontNeedAnyField:
self.method=ECPrivateKeyMethods::setFieldFP implies
self.parameters->size()=3 and self.fields->size()=0;
invariant ECPublicKeysetsetFieldF2MMethodNeeds1OR3ParameterButDontNeedAnyField:
self.method=ECPublicKeyMethods::setFieldF2M implies
(self.parameters->size()=1 or self.parameters->size()=3) and self.fields->size()=0;
invariant ECPublicKeysetAMethodNeeds3ParameterButDontNeedAnyField:
self.method=ECPublicKeyMethods::setA implies
self.parameters->size()=3 and self.fields->size()=0;
invariant ECPublicKeysetBMethodNeeds3ParameterButDontNeedAnyField:
self.method=ECPublicKeyMethods::setB implies
self.parameters->size()=3 and self.fields->size()=0;
invariant ECPublicKeysetGMethodNeeds3ParameterButDontNeedAnyField:
self.method=ECPublicKeyMethods::setG implies
self.parameters->size()=3 and self.fields->size()=0;
invariant ECPublicKeysetRMethodNeeds3ParameterButDontNeedAnyField:
self.method=ECPublicKeyMethods::setR implies
self.parameters->size()=3 and self.fields->size()=0;
invariant ECPublicKeysetKMethodNeeds1ParameterButDontNeedAnyField:
self.method=ECPublicKeyMethods::setK implies
self.parameters->size()=1 and self.fields->size()=0;
invariant ECPublicKeygetFieldMethodNeeds2ParameterAndNeedShortField:
self.method=ECPublicKeyMethods::getField implies
self.parameters->size()=2 and self.fields->forAll(x|x.dataType = DataTypes::short);
invariant ECPublicKeyNeeds2ParameterAndNeedShortField:
self.method=ECPublicKeyMethods::getA implies
self.parameters->size()=2 and self.fields->forAll(x|x.dataType = DataTypes::short);
```

```
invariant ECPublicKeygetBMethodNeeds2ParameterAndNeedShortField:
self.method=ECPublicKeyMethods::getB implies
self.parameters->size()=2 and self.fields->forAll(x|x.dataType = DataTypes::short);
invariant ECPublicKeygetGMethodNeeds2ParameterAndNeedShortField:
self.method=ECPublicKeyMethods::getG implies
self.parameters->size()=2 and self.fields->forAll(x|x.dataType = DataTypes::short);
invariant ECPublicKeygetRMethodNeeds2ParameterAndNeedShortField:
self.method=ECPublicKeyMethods::getR implies
self.parameters->size()=2 and self.fields->forAll(x|x.dataType = DataTypes::short);
invariant ECPublicKeygetKMethodNeedsShortFieldButDontNeedAnyParameters:
self.method=ECPublicKeyMethods::getK implies
self.parameters->size()=0 and self.fields->forAll(x|x.dataType = DataTypes::short);
}
enum ECPublicKeyMethods { serializable }
{
literal setW;
literal getW = 1;
literal clearKey = 2;
literal getSize = 3;
literal getType = 4;
literal isInitialized = 5;
literal setFieldFP = 6;
literal setFieldF2M = 7;
literal setA = 8;
literal setB = 9;
literal setG = 10;
literal setR = 11;
literal setK = 12;
literal getField = 13;
literal getA = 14;
literal getB = 15;
literal getG = 16;
literal getR = 17;
literal getK = 18;
}
class Key
{
property fields : Field[?] { composes };
attribute method : KeyMethods[?];
property parameters : Parameter[*] { ordered composes };
property instanceVariable : InstanceVariable[?] { composes };
invariant KeyclearKeyMethodDontNeedAnyParameter:
self.method=KeyMethods::clearKey implies
self.parameters->size()=0;
invariant KeygetSizeMethodDontNeedAnyParameter:
self.method=KeyMethods::getSize implies
self.parameters->size()=0;
invariant KeygetTypeMethodDontNeedAnyParameter:
self.method=KeyMethods::getType implies
self.parameters->size()=0;
invariant KeyisInitializedMethodDontNeedAnyParameter:
self.method=KeyMethods::isInitialized implies
self.parameters->size()=0;
}
enum KeyMethods { serializable }
{
literal clearKey;
literal getSize = 1;
literal getType = 2;
literal isInitialized = 3;
}
class PrivateKey
{
property fields : Field[?] { composes };
attribute method : PrivateKeyMethods[?];
property parameters : Parameter[*] { ordered composes };
property instanceVariable : InstanceVariable[?] { composes };
invariant PrivateKeyclearKeyMethodDontNeedAnyParameter:
self.method=PrivateKeyMethods::clearKey implies
self.parameters->size()=0;
invariant PrivateKeygetSizeMethodDontNeedAnyParameter:
self.method=PrivateKeyMethods::getSize implies
```

```
self.parameters->size()=0;
invariant PrivateKeygetTypeMethodDontNeedAnyParameter:
self.method=PrivateKeyMethods::getType implies
self.parameters->size()=0;
invariant PrivateKeyisInitializedMethodDontNeedAnyParameter:
self.method=PrivateKeyMethods::isInitialized implies
self.parameters->size()=0;
}
enum PrivateKeyMethods { serializable }
{
literal clearKey;
literal getSize = 1;
literal getType = 2;
literal isInitialized = 3;
}
class RSAPrivateKey
{
property fields : Field[?] { composes };
attribute method : RSAPrivateKeyMethods[?];
property parameters : Parameter[*] { ordered composes };
property instanceVariable : InstanceVariable[?] { composes };
invariant RSAPrivateKeysetModulusMethodNeeds3Parameter:
self.method=RSAPrivateKeyMethods::setModulus implies
self.parameters->size()=3;
invariant RSAPrivateKeysetExponentMethodNeeds3Parameter:
self.method=RSAPrivateKeyMethods::setExponent implies
self.parameters->size()=3;
invariant RSAPrivateKeygetModulusMethodNeeds2Parameter:
self.method=RSAPrivateKeyMethods::getModulus implies
self.parameters->size()=2;
invariant RSAPrivateKeygetExponentMethodNeeds2Parameter:
self.method=RSAPrivateKeyMethods::getExponent implies
self.parameters->size()=2;
invariant RSAPrivateKeyclearKeyMethodDontNeedAnyParameter:
self.method=RSAPrivateKeyMethods::clearKey implies
self.parameters->size()=0;
invariant RSAPrivateKeygetSizeMethodDontNeedAnyParameter:
self.method=RSAPrivateKeyMethods::getSize implies
self.parameters->size()=0;
invariant RSAPrivateKeygetTypeMethodDontNeedAnyParameter:
self.method=RSAPrivateKeyMethods::getType implies
self.parameters->size()=0;
invariant RSAPrivateKeyisInitializedMethodDontNeedAnyParameter:
self.method=RSAPrivateKeyMethods::isInitialized implies
self.parameters->size()=0;
}
enum RSAPrivateKeyMethods { serializable }
{
literal setModulus;
literal setExponent = 1;
literal getModulus = 2;
literal getExponent = 3;
literal clearKey = 4;
literal getSize = 5;
literal getType = 6;
literal isInitialized = 7;
}
class RSAPrivateCrtKey
{
property fields : Field[?] { composes };
attribute method : RSAPrivateCrtKeyMethods[?];
property parameters : Parameter[*] { ordered composes };
property instanceVariable : InstanceVariable[?] { composes };
invariant RSAPrivateCrtKeysetPMethodNeeds3Parameter:
self.method=RSAPrivateCrtKeyMethods::setP implies
self.parameters->size()=3;
invariant RSAPrivateCrtKeysetQMethodNeeds3Parameter:
self.method=RSAPrivateCrtKeyMethods::setQ implies
self.parameters->size()=3;
invariant RSAPrivateCrtKeysetDP1MethodNeeds3Parameter:
self.method=RSAPrivateCrtKeyMethods::setDP1 implies
self.parameters->size()=3;
```

```
invariant RSAPrivateCrtKeysetDQ1MethodNeeds3Parameter:
self.method=RSAPrivateCrtKeyMethods::setDQ1 implies
self.parameters->size()=3;
invariant RSAPrivateCrtKeysetPQMethodNeeds3Parameter:
self.method=RSAPrivateCrtKeyMethods::setPQ implies
self.parameters->size()=3;
invariant RSAPrivateCrtKeygetPMethodNeeds2Parameter:
self.method=RSAPrivateCrtKeyMethods::getP implies
self.parameters->size()=2;
invariant RSAPrivateCrtKeygetQMethodNeeds2Parameter:
self.method=RSAPrivateCrtKeyMethods::getQ implies
self.parameters->size()=2;
invariant RSAPrivateCrtKeygetDP1MethodNeeds2Parameter:
self.method=RSAPrivateCrtKeyMethods::getDP1 implies
self.parameters->size()=2;
invariant RSAPrivateCrtKeygetDQ1MethodNeeds2Parameter:
self.method=RSAPrivateCrtKeyMethods::getDQ1 implies
self.parameters->size()=2;
invariant RSAPrivateCrtKeygetPQMethodNeeds2Parameter:
self.method=RSAPrivateCrtKeyMethods::getPQ implies
self.parameters->size()=2;
invariant RSAPrivateCrtKeyclearKeyMethodDontNeedAnyParameter:
self.method=RSAPrivateCrtKeyMethods::clearKey implies
self.parameters->size()=0;
invariant RSAPrivateCrtKeygetSizeMethodDontNeedAnyParameter:
self.method=RSAPrivateCrtKeyMethods::getSize implies
self.parameters->size()=0;
invariant RSAPrivateCrtKeygetTypeMethodDontNeedAnyParameter:
self.method=RSAPrivateCrtKeyMethods::getType implies
self.parameters->size()=0;
invariant RSAPrivateCrtKeyisInitializedMethodDontNeedAnyParameter:
self.method=RSAPrivateCrtKeyMethods::isInitialized implies
self.parameters->size()=0;
}
enum RSAPrivateCrtKeyMethods { serializable }
{
literal setP;
literal setQ = 1;
literal setDP1 = 2;
literal setDQ1 = 3;
literal setPQ = 4;
literal getP = 5;
literal getQ = 6;
literal getDP1 = 7;
literal getDQ1 = 8;
literal getPQ = 9;
literal clearKey = 10;
literal getSize = 11;
literal getType = 12;
literal isInitialized = 13;
}
class PublicKey
{
property fields : Field[?] { composes };
attribute method : PublicKeyMethods[?];
property parameters : Parameter[*] { ordered composes };
property instanceVariable : InstanceVariable[?] { composes };
invariant PublicKeyclearKeyMethodDontNeedAnyParameter:
self.method=PublicKeyMethods::clearKey implies
self.parameters->size()=0;
invariant PublicKeygetSizeMethodDontNeedAnyParameter:
self.method=PublicKeyMethods::getSize implies
self.parameters->size()=0;
invariant PublicKeygetTypeMethodDontNeedAnyParameter:
self.method=PublicKeyMethods::getType implies
self.parameters->size()=0;
invariant PublicKeyisInitializedMethodDontNeedAnyParameter:
self.method=PublicKeyMethods::isInitialized implies
self.parameters->size()=0;
}
enum PublicKeyMethods { serializable }
{
```

```
literal clearKey;
literal getSize = 1;
literal getType = 2;
literal isInitialized = 3;
}
class RSAPublicKey
{
property fields : Field[?] { composes };
attribute method : RSAPublicKeyMethods[?];
property parameters : Parameter[*] { ordered composes };
property instanceVariable : InstanceVariable[?] { composes };
invariant RSAPublicKeysetModulusMethodNeeds3Parameter:
self.method=RSAPublicKeyMethods::setModulus implies
self.parameters->size()=3;
invariant RSAPublicKeysetExponentMethodNeeds3Parameter:
self.method=RSAPublicKeyMethods::setExponent implies
self.parameters->size()=3;
invariant RSAPublicKeygetModulusMethodNeeds2ParameterDontForgetTheInitializeTheseValues:

self.method=RSAPublicKeyMethods::getModulus implies
self.parameters->size()=2;
invariant RSAPublicKeygetExponentMethodNeeds2ParameterDontForgetTheInitializeTheseValues
:
self.method=RSAPublicKeyMethods::getExponent implies
self.parameters->size()=2;
invariant RSAPublicKeyclearKeyMethodDontNeedAnyParameterDontForgetTheInitializeTheseValu
es:
self.method=RSAPublicKeyMethods::clearKey implies
self.parameters->size()=0;
invariant RSAPublicKeygetSizeMethodDontNeedAnyParameterDontForgetTheInitializeTheseValue
s:
self.method=RSAPublicKeyMethods::getSize implies
self.parameters->size()=0;
invariant RSAPublicKeygetTypeMethodDontNeedAnyParameterDontForgetTheInitializeTheseValue
s:
self.method=RSAPublicKeyMethods::getType implies
self.parameters->size()=0;
invariant RSAPublicKeyisInitializedMethodDontNeedAnyParameterDontForgetTheInitializeThes
eValues:
self.method=RSAPublicKeyMethods::isInitialized implies
self.parameters->size()=0;
}
enum RSAPublicKeyMethods { serializable }
{
literal setModulus;
literal setExponent = 1;
literal getModulus = 2;
literal getExponent = 3;
literal clearKey = 4;
literal getSize = 5;
literal getType = 6;
literal isInitialized = 7;
}
class SecretKey
{
property fields : Field[?] { composes };
attribute method : SecretKeyMethods[?];
property parameters : Parameter[*] { ordered composes };
property instanceVariable : InstanceVariable[?] { composes };
invariant SecretKeyclearKeyMethodDontNeedAnyParameter:
self.method=SecretKeyMethods::clearKey implies
self.parameters->size()=0;
invariant SecretKeygetSizeMethodDontNeedAnyParameter:
self.method=SecretKeyMethods::getSize implies
self.parameters->size()=0;
invariant SecretKeygetTypeMethodDontNeedAnyParameter:
self.method=SecretKeyMethods::getType implies
self.parameters->size()=0;
invariant SecretKeyisInitializedMethodDontNeedAnyParameter:
self.method=SecretKeyMethods::isInitialized implies
self.parameters->size()=0;
}
```

```
enum SecretKeyMethods { serializable }
{
literal clearKey;
literal getSize = 1;
literal getType = 2;
literal isInitialized = 3;
}
class AESKey
{
property fields : Field[?] { composes };
attribute method : AESKeyMethods[?];
property parameters : Parameter[*] { ordered composes };
property instanceVariable : InstanceVariable[?] { composes };
invariant AESKeysetKeyMethodNeeds2ParameterButDontNeedAnyField:
self.method=AESKeyMethods::setKey implies
self.parameters->size()=2 and self.fields->size()=0;
invariant AESKeygetKeyMethodNeeds2ParameterAndNeedByteField:
self.method=AESKeyMethods::getKey implies
self.parameters->size()=2 and self.fields->forAll(e | e.dataType <> DataTypes::byte);
invariant AESKeyclearKeyMethodDontNeedAnyParameterAndField:
self.method=AESKeyMethods::clearKey implies
self.parameters->size()=0 and self.fields->size()=0;
invariant AESKeygetSizeMethodDontNeedAnyParameterButNeedShortField:
self.method=AESKeyMethods::getSize implies
self.parameters->size()=0 and self.fields->forAll(e | e.dataType <> DataTypes::short);
invariant AESKeygetTypeMethodDontNeedAnyParameterButNeedByteField:
self.method=AESKeyMethods::getType implies
self.parameters->size()=0 and self.fields->forAll(e | e.dataType <> DataTypes::byte);
invariant AESKeyisInitializedMethodDontNeedAnyParameterButNeedBooleanField:
self.method=AESKeyMethods::isInitialized implies
self.parameters->size()=0 and self.fields->forAll(e | e.dataType <> DataTypes::boolean);
}
enum AESKeyMethods { serializable }
{
literal setKey;
literal getKey = 1;
literal clearKey = 2;
literal getSize = 3;
literal getType = 4;
literal isInitialized = 5;
}
class DESKey
{
property fields : Field[?] { composes };
attribute method : DESKeyMethods[?];
property parameters : Parameter[*] { ordered composes };
property instanceVariable : InstanceVariable[?] { composes };
invariant DESKeysetKeyMethodNeeds2ParameterButDontNeedAnyField:
self.method=DESKeyMethods::setKey implies
self.parameters->size()=2 and self.fields->size()=0;
invariant DESKeygetKeyMethodNeeds2ParameterAndByteField:
self.method=DESKeyMethods::getKey implies
self.parameters->size()=2 and self.fields->forAll(x|x.dataType = DataTypes::byte);
invariant DESKeyclearKeyMethodDontNeedAnyParameterAndField:
self.method=DESKeyMethods::clearKey implies
self.parameters->size()=0 and self.fields->size()=0;
invariant DESKeygetSizeMethodDontNeedAnyParameterButNeedShortField:
self.method=DESKeyMethods::getSize implies
self.parameters->size()=0 and self.fields->forAll(x|x.dataType = DataTypes::short);
invariant DESKeygetTypeMethodDontNeedAnyParameterButNeedByteField:
self.method=DESKeyMethods::getType implies
self.parameters->size()=0 and self.fields->forAll(x|x.dataType = DataTypes::byte);
invariant DESKeyisInitializedMethodDontNeedAnyParameterButNeedBooleanField:
self.method=DESKeyMethods::isInitialized implies
self.parameters->size()=0 and self.fields->forAll(x|x.dataType = DataTypes::boolean);
}
enum DESKeyMethods { serializable }
{
literal setKey;
literal getKey = 1;
literal clearKey = 2;
literal getSize = 3;
```

```
literal getType = 4;
literal isInitialized = 5;
}
class HMACKey
{
property fields : Field[?] { composes };
attribute method : HMACKeyMethods[?];
property parameters : Parameter[*] { ordered composes };
property instanceVariable : InstanceVariable[?] { composes };
invariant HMACKeysetKeyMethodNeeds3ParameterButDontNeedAnyField:
self.method=HMACKeyMethods::setKey implies
self.parameters->size()=3 and self.fields->size()=0;
invariant HMACKeygetKeyMethodNeeds2ParameterAndNeedShortField:
self.method=HMACKeyMethods::getKey implies
self.parameters->size()=2 and self.fields->forAll(x|x.dataType = DataTypes::short);
invariant HMACKeyclearKeyMethodDontNeedAnyParameterAndField:
self.method=HMACKeyMethods::clearKey implies
self.parameters->size()=0 and self.fields->size()=0;
invariant HMACKeygetSizeMethodDontNeedAnyParameterButNeedShortField:
self.method=HMACKeyMethods::getSize implies
self.parameters->size()=0 and self.fields->forAll(x|x.dataType = DataTypes::short);
invariant HMACKeygetTypeMethodDontNeedAnyParameterButNeedByteField:
self.method=HMACKeyMethods::getType implies
self.parameters->size()=0 and self.fields->forAll(x|x.dataType = DataTypes::byte);
invariant HMACKeyisInitializedMethodDontNeedAnyParameterButNeedBooleanField:
self.method=HMACKeyMethods::isInitialized implies
self.parameters->size()=0 and self.fields->forAll(x|x.dataType = DataTypes::boolean);
}
enum HMACKeyMethods { serializable }
{
literal setKey;
literal getKey = 1;
literal clearKey = 2;
literal getSize = 3;
literal getType = 4;
literal isInitialized = 5;
}
class KoreanSEEDKey
{
property fields : Field[?] { composes };
attribute method : KoreanSEEDKeyMethods[?];
property parameters : Parameter[*] { ordered composes };
property instanceVariable : InstanceVariable[?] { composes };
invariant KoreanSEEDKeysetKeyMethodNeeds2Parameter:
self.method=KoreanSEEDKeyMethods::setKey implies
self.parameters->size()=2;
invariant KoreanSEEDKeygetKeyMethodNeeds2Parameter:
self.method=KoreanSEEDKeyMethods::getKey implies
self.parameters->size()=2;
invariant KoreanSEEDKeyclearKeyMethodDontNeedAnyParameter:
self.method=KoreanSEEDKeyMethods::clearKey implies
self.parameters->size()=0;
invariant KoreanSEEDKeygetSizeMethodDontNeedAnyParameter:
self.method=KoreanSEEDKeyMethods::getSize implies
self.parameters->size()=0;
invariant KoreanSEEDKeygetTypeMethodDontNeedAnyParameter:
self.method=KoreanSEEDKeyMethods::getType implies
self.parameters->size()=0;
invariant KoreanSEEDKeyisInitializedMethodDontNeedAnyParameter:
self.method=KoreanSEEDKeyMethods::isInitialized implies
self.parameters->size()=0;
}
enum KoreanSEEDKeyMethods { serializable }
{
literal setKey;
literal getKey = 1;
literal clearKey = 2;
literal getSize = 3;
literal getType = 4;
literal isInitialized = 5;
}
class SignatureMessageRecovery
```

```
{
property fields : Field[?] { composes };
attribute method : SignatureMessageRecoveryMethods[?];
property parameters : Parameter[*] { ordered composes };
property instanceVariable : InstanceVariable[?] { composes };
invariant SignatureMessageRecoveryinitMethodNeeds2Parameter:
self.method=SignatureMessageRecoveryMethods::init implies
self.parameters->size()=2;
invariant SignatureMessageRecoverybeginVerifyMethodNeeds3Parameter:
self.method=SignatureMessageRecoveryMethods::beginVerify implies
self.parameters->size()=3;
invariant SignatureMessageRecoverysignMethodNeeds7Parameter:
self.method=SignatureMessageRecoveryMethods::sign implies
self.parameters->size()=7;
invariant SignatureMessageRecoveryverifyMethodNeeds3Parameter:
self.method=SignatureMessageRecoveryMethods::verify implies
self.parameters->size()=3;
invariant SignatureMessageRecoveryupdateMethodNeeds3Parameter:
self.method=SignatureMessageRecoveryMethods::update implies
self.parameters->size()=3;
}
enum SignatureMessageRecoveryMethods { serializable }
{
literal init;
literal beginVerify = 1;
literal sign = 2;
literal verify = 3;
literal getAlgorithm = 4;
literal getLength = 5;
literal update = 6;
}
class ISO7816
{
property fields : Field[?] { composes };
attribute iso7816Field : ISO7816Fields[?];
attribute spiralFieldName : String[?];
attribute ISOExceptionthrowIt : Boolean[?];
}
enum ISO7816Fields { serializable }
{
literal none;
literal SW_NO_ERROR = 1;
literal SW_BYTES_REMAINING_00 = 2;
literal SW_WRONG_LENGTH = 3;
literal SW_SECURITY_STATUS_NOT_SATISFIED = 4;
literal SW_FILE_INVALID = 5;
literal SW_DATA_INVALID = 6;
literal SW_CONDITIONS_NOT_SATISFIED = 7;
literal SW_COMMAND_NOT_ALLOWED = 8;
literal SW_APPLET_SELECT_FAILED = 9;
literal SW_WRONG_DATA = 10;
literal SW_FUNC_NOT_SUPPORTED = 11;
literal SW_FILE_NOT_FOUND = 12;
literal SW_RECORD_NOT_FOUND = 13;
literal SW_INCORRECT_P1P2 = 14;
literal SW_WRONG_P1P2 = 15;
literal SW_CORRECT_LENGTH_00 = 16;
literal SW_INS_NOT_SUPPORTED = 17;
literal SW_CLA_NOT_SUPPORTED = 18;
literal SW_UNKNOWN = 19;
literal SW_FILE_FULL = 20;
literal SW_LOGICAL_CHANNEL_NOT_SUPPORTED = 21;
literal SW_SECURE_MESSAGING_NOT_SUPPORTED = 22;
literal SW_WARNING_STATE_UNCHANGED = 23;
literal SW_LAST_COMMAND_EXPECTED = 24;
literal SW_COMMAND_CHAINING_NOT_SUPPORTED = 25;
literal OFFSET_CLA = 26;
literal OFFSET_INS = 27;
literal OFFSET_P1 = 28;
literal OFFSET_P2 = 29;
literal OFFSET_LC = 30;
literal OFFSET_CDATA = 31;
```

```
literal OFFSET_EXT_CDATA = 32;
literal CLA_ISO7816 = 33;
literal INS_SELECT = 34;
literal INS_EXTERNAL_AUTHENTICATE = 35;
}
class KeyEncryption
{
property fields : Field[?] { composes };
attribute method : KeyEncryptionMethods[?];
property parameters : Parameter[*] { ordered composes };
property instanceVariable : InstanceVariable[?] { composes };
invariant KeyEncryptionsetKeyCipherMethodNeeds1Parameter:
self.method=KeyEncryptionMethods::setKeyCipher implies
self.parameters->size()=1;
}
enum KeyEncryptionMethods { serializable }
{
literal setKeyCipher;
literal getKeyCipher = 1;
}
class Util
{
property fields : Field[?] { composes };
attribute method : UtilMethods[?];
property parameters : Parameter[*] { ordered composes };
property instanceVariable : InstanceVariable[?] { composes };
invariant UtilarrayCopyMethodNeeds5Parameter:
self.method=UtilMethods::arrayCopy implies
self.parameters->size()=5;
invariant UtilarrayCopyNonAtomicMethodNeeds5Parameter:
self.method=UtilMethods::arrayCopyNonAtomic implies
self.parameters->size()=5;
invariant UtilarrayFillNonAtomicMethodNeeds4Parameter:
self.method=UtilMethods::arrayFillNonAtomic implies
self.parameters->size()=4;
invariant UtilarrayCompareMethodNeeds5Parameter:
self.method=UtilMethods::arrayCompare implies
self.parameters->size()=5;
invariant UtilmakeShortMethodNeeds2Parameter:
self.method=UtilMethods::makeShort implies
self.parameters->size()=2;
invariant UtilgetShortMethodNeeds2Parameter:
self.method=UtilMethods::getShort implies
self.parameters->size()=2;
invariant UtilsetShortMethodNeeds3Parameter:
self.method=UtilMethods::setShort implies
self.parameters->size()=3;
}
enum UtilMethods { serializable }
{
literal arrayCopy;
literal arrayCopyNonAtomic = 1;
literal arrayFillNonAtomic = 2;
literal arrayCompare = 3;
literal makeShort = 4;
literal getShort = 5;
literal setShort = 6;
literal equals = 7;
}
class InstanceVariable
{
attribute instanceVariableName : String[?];
invariant InstanceVariableNameMustGreaterThanZeroANDWordsMustBeUnited:
self.instanceVariableName.size()>0;
invariant Warning_DontForgetTheDefineInstanceVariable:
self.instanceVariableName.size()>0;
}
}
```

EK-3

DSL4JavaCard XPAND Kodları

```
«IMPORT metamodel»

«DEFINE Root FOR javaCardModel::Object»
«EXPAND ApplicationClass FOREACH applicationClasses»
«ENDDEFINE»
«DEFINE ApplicationClass FOR javaCardModel::ApplicationClass»
   «FILE className + ".java"»
import javacard.framework.*;
import javacard.security.*;
import javacardx.crypto.*;
   public class «className» extends Applet {
   «FOREACH methods AS m ITERATOR metIt»
     «FOREACH m.keyPairs AS k ITERATOR keyIt»«IF keyIt.counter1 == 1»«LET "KeyPair
_" + m.methodName
+ "KeyPair" + ";" AS keyPairVar»«keyPairVar»//You may delete or rename to use this va
riable in your methods«ENDLET»«ENDIF»«ENDFOREACH»
«FOREACH m.ciphers AS c ITERATOR cipIt»«IF cipIt.counter1 == 1»«LET "Cipher _" +
m.methodName
+ "Cipher" + ";" AS cipherVar»«cipherVar»//You may delete or rename to use this varia
ble in your methods«ENDLET»«ENDIF»«ENDFOREACH»
«FOREACH m.apdus AS apdu ITERATOR apduIt»«IF apduIt.counter1 == 1»«LET "APDU _" +
m.methodName
+ "APDU" + ";" AS apduVar»«apduVar»//You may delete or rename to use this variable in
 your methods«ENDLET»«ENDIF»«ENDFOREACH»
«FOREACH m.ownerPins AS ownerPin ITERATOR oPinIt»«IF oPinIt.counter1 ==
1»«LET "OwnerPIN _" + m.methodName
+ "OwnerPIN" + ";" AS oPINVar»«oPINVar»//You may delete or rename to use this variabl
e in your methods«ENDLET»«ENDIF»«ENDFOREACH»
«FOREACH m.cardExceptions AS cardEx ITERATOR cardExIt»«IF cardExIt.counter1 ==
1»«LET "CardException _" + m.methodName
+ "CardException" + ";" AS cardExVar»«cardExVar»//You may delete or rename to use thi
s variable in yourmethods«ENDLET»«ENDIF»«ENDFOREACH»
«FOREACH m.aids AS aid ITERATOR aidIt»«IF aidIt.counter1 == 1»«LET "AID _" +
m.methodName
+ "AID" + ";" AS aidVar»«aidVar»//You may delete or rename to use this variable in yo
ur methods«ENDLET»«ENDIF»«ENDFOREACH»
«FOREACH m.checksums AS cSum ITERATOR cSumIt»«IF cSumIt.counter1 == 1»«LET "Checksum
_" + m.methodName
+ "Checksum" + ";" AS cSumVar»«cSumVar»//You may delete or rename to use this variabl
e in your methods«ENDLET»«ENDIF»«ENDFOREACH»
        «FOREACH m.keyAgreements AS kAg ITERATOR kAgIt»«IF kAgIt.counter1 ==
1»«LET "KeyAgreement _" + m.methodName
+ "KeyAgreement" + ";" AS kAgVar»«kAgVar»//You may delete or rename to use this varia
ble in your methods«ENDLET»«ENDIF»«ENDFOREACH»
        «FOREACH m.keyBuilders AS kBuild ITERATOR kBuildIt»«IF kBuildIt.counter1 ==
1»«LET "KeyBuilder _" + m.methodName
+ "KeyBuilder" + ";" AS kBuildVar»«kBuildVar»//You may delete or rename to use this v
ariable in your methods«ENDLET»«ENDIF»«ENDFOREACH»
        «FOREACH m.messageDigests AS mDig ITERATOR mDigIt»«IF mDigIt.counter1 ==
1»«LET "MessageDigest _" + m.methodName
+ "MessageDigest" + ";" AS mdVar»«mdVar»//You may delete or rename to use this variab
le in your methods«ENDLET»«ENDIF»«ENDFOREACH»
   «FOREACH m.initializeMessageDigests AS mInDig ITERATOR mInDigIt»«IF mInDigIt.count
er1 == 1»«LET "InitializedMessageDigest _" + m.methodName
+ "InitializedMessageDigest" + ";" AS mIdVar»«mIdVar»//You may delete or rename to us
e this variable in yourmethods«ENDLET»«ENDIF»«ENDFOREACH»
    «FOREACH m.randomData AS rData ITERATOR rDataIt»«IF rDataIt.counter1 ==
1»«LET "RandomData _" + m.methodName
+ "RandomData" + ";" AS rVar»«rVar»//You may delete or rename to use this variable in
 your methods«ENDLET»«ENDIF»«ENDFOREACH»
    «FOREACH m.signatures AS sig ITERATOR sigIt»«IF sigIt.counter1 ==
1»«LET "Signature _" + m.methodName
+ "Signature" + ";" AS sVar»«sVar»//You may delete or rename to use this variable in
your methods«ENDLET»«ENDIF»«ENDFOREACH»
    «FOREACH m.dsaKeys AS dsaKey ITERATOR dsaKeyIt»«IF dsaKeyIt.counter1 ==
```

```
1»«LET "DSAKey _" + m.methodName
+ "DSAKey" + "=" AS dsaVar»«dsaVar» (DSAKey) KeyBuilder.buildKey(KeyBuilder.TYPE_DSA_
PUBLIC,KeyBuilder.LENGTH_DSA_768,true);//You may delete or rename to use this variabl
e in your methods«ENDLET»«ENDIF»«ENDFOREACH»
        «FOREACH m.dsaPublicKeys AS dsaPubKey ITERATOR dsaPubKeyIt»«IF dsaPubKeyIt.co
unter1 == 1»«LET "DSAPublicKey _" + m.methodName
+ "DSAPublicKey" + "=" AS dsaVar»«dsaVar» (DSAPublicKey) KeyBuilder.buildKey(KeyBuild
er.TYPE_DSA_PUBLIC,KeyBuilder.LENGTH_DSA_768, true); //You may delete or rename to us
e this variable in your methods«ENDLET»«ENDIF»«ENDFOREACH»
    «FOREACH m.ecKeys AS ecKey ITERATOR ecKeyIt»«IF ecKeyIt.counter1 == 1»«LET "ECKey
_" + m.methodName
+ "ECKey"  + "=" AS ecVar»«ecVar» (ECKey) KeyBuilder.buildKey(KeyBuilder.TYPE_EC_F2M_
PRIVATE_TRANSIENT_DESELECT,KeyBuilder.LENGTH_EC_FP_384, false);//You may delete or re
name to use this variable in your methods«ENDLET»«ENDIF»«ENDFOREACH»«FOREACH m.ecPriv
ateKeys AS ecPriKey ITERATOR ecPriKeyIt»«IF ecPriKeyIt.counter1 ==
1»«LET "ECPrivateKey _" + m.methodName
+ "ECPrivateKey" + ";" AS ECPriKeyVar»«ECPriKeyVar»//You may delete or rename to use
this variable in your methods«ENDLET»«ENDIF»«ENDFOREACH»
    «FOREACH m.ecPublicKeys AS ecPubKey ITERATOR ecPubKeyIt»«IF ecPubKeyIt.counter1
== 1»«LET "ECPublicKey _" + m.methodName
+ "ECPublicKey"   + "=" AS ECPubKeyVar»«ECPubKeyVar» (ECPublicKey)KeyBuilder.buildKey
(KeyBuilder.TYPE_EC_F2M_PUBLIC, KeyBuilder.LENGTH_EC_FP_112, false);//You may delete
or rename to use this variable in your methods«ENDLET»«ENDIF»«ENDFOREACH»
    «FOREACH m.keys AS key ITERATOR keyIt»«IF keyIt.counter1 == 1»«LET "Key _" +
m.methodName
+ "Key"  + "=" AS KeyVar»«KeyVar» (Key)KeyBuilder.buildKey(KeyBuilder.TYPE_DES, KeyBu
ilder.LENGTH_DES, false); //You may delete or rename to use thisvariable in your meth
ods«ENDLET»«ENDIF»«ENDFOREACH»
        «FOREACH m.privateKeys AS privateKey ITERATOR privateKeysIt»«IF privateKeysIt
.counter1 == 1»«LET "PrivateKey _" + m.methodName
+ "PrivateKey"  + "=" AS PrKeyVar»«PrKeyVar» (PrivateKey) KeyBuilder.buildKey(KeyBuil
der.TYPE_DSA_PRIVATE,KeyBuilder.LENGTH_DSA_512, false);//You may delete or rename to
use this variable in your methods«ENDLET»«ENDIF»«ENDFOREACH»
        «FOREACH m.publicKeys AS publicKey ITERATOR publicKeyIt»«IF publicKeyIt.count
er1 == 1»«LET "PublicKey _" + m.methodName
+ "PublicKey"  + "=" AS PubKeyVar»«PubKeyVar» (PublicKey) KeyBuilder.buildKey(KeyBuil
der.TYPE_DSA_PUBLIC,KeyBuilder.LENGTH_DSA_512, false);//You may delete or rename to u
se this variable in your methods«ENDLET»«ENDIF»«ENDFOREACH»
        «FOREACH m.rsaPrivateKeys AS rsaPrivateKey ITERATOR iterator»«IF iterator.cou
nter1 == 1»«LET "RSAPrivateKey _" + m.methodName
+ "RSAPrivateKey"  + "=" AS Var»«Var» (RSAPrivateKey) KeyBuilder.buildKey(KeyBuilder.
TYPE_RSA_PRIVATE,KeyBuilder.LENGTH_RSA_512, false);//You may delete or rename to use
this variable in your methods«ENDLET»«ENDIF»«ENDFOREACH»
        «FOREACH m.rsaPrivateCrtKeys AS rsaPrivateCrtKey ITERATOR iterator»«IF iterat
or.counter1 == 1»«LET "RSAPrivateCrtKey _" + m.methodName
+ "RSAPrivateCrtKey"  + "=" AS Var»«Var» (RSAPrivateCrtKey) KeyBuilder.buildKey(KeyBu
ilder.TYPE_RSA_CRT_PRIVATE, KeyBuilder.LENGTH_RSA_512, false);//You may delete or ren
ame to use this variable in your methods«ENDLET»«ENDIF»«ENDFOREACH»
    «FOREACH m.rsaPublicKeys AS rsaPublicKey ITERATOR iterator»«IF iterator.counter1
== 1»«LET "RSAPublicKey _" + m.methodName
+ "RSAPublicKey"  + "=" AS Var»«Var» (RSAPublicKey) KeyBuilder.buildKey(KeyBuilder.TY
PE_RSA_PUBLIC,KeyBuilder.LENGTH_RSA_512, false); //You may delete or rename to use th
is variable in your methods«ENDLET»«ENDIF»«ENDFOREACH»
    «FOREACH m.secretKeys AS secretKey ITERATOR iterator»«IF iterator.counter1 ==
1»«LET "SecretKey _" + m.methodName
+ "SecretKey"  + "=" AS Var»«Var»//You may delete or rename to use this variable in y
our methods«ENDLET»«ENDIF»«ENDFOREACH»
    «FOREACH m.aesKeys AS aesKey ITERATOR iterator»«IF iterator.counter1 ==
1»«LET "AESKey _" + m.methodName
+ "AESKey"  + "=" AS Var»«Var» (AESKey) KeyBuilder.buildKey(KeyBuilder.TYPE_AES, KeyB
uilder.LENGTH_AES_128, false); //You may delete orrename to use this variable in your
 methods«ENDLET»«ENDIF»«ENDFOREACH»
    «FOREACH m.desKeys AS desKey ITERATOR iterator»«IF iterator.counter1 ==
1»«LET "DESKey _" + m.methodName
+ "DESKey"  + "=" AS Var»«Var» (DESKey) KeyBuilder.buildKey(KeyBuilder.TYPE_DES, KeyB
uilder.LENGTH_DES, false); //You may delete orrename to use this variable in your met
hods«ENDLET»«ENDIF»«ENDFOREACH»
    «FOREACH m.hmacKeys AS hmacKey ITERATOR iterator»«IF iterator.counter1 ==
1»«LET "HMACKey _" + m.methodName
+ "HMACKey"  + "=" AS Var»«Var» (HMACKey) KeyBuilder.buildKey(KeyBuilder.TYPE_HMAC, K
```

```
eyBuilder.TYPE_HMAC, false); //You maydelete or rename to use this variable in your m
ethods«ENDLET»«ENDIF»«ENDFOREACH»
    «FOREACH m.koreanSEEDKeys AS koreanSEEDKey ITERATOR iterator»«IF iterator.counter
1 == 1»«LET "KoreanSEEDKey _" + m.methodName
+ "KoreanSEEDKey"  + "=" AS Var»«Var» (KoreanSEEDKey) KeyBuilder.buildKey(KeyBuilder.
TYPE_KOREAN_SEED,KeyBuilder.LENGTH_KOREAN_SEED_128, false); //You may delete or renam
e to use this variable in your methods«ENDLET»«ENDIF»«ENDFOREACH»
   «FOREACH m.signatureMessageRecoveries AS signatureMessageRecovery ITERATOR iterato
r»«IF iterator.counter1 == 1»«LET "SignatureMessageRecovery _" + m.methodName
+ "SignatureMessageRecovery" + ";" AS Var»«Var»//You may delete or rename to use this
variable in your methods«ENDLET»«ENDIF»«ENDFOREACH»
«FOREACH m.keyEncryotions AS keyEncryotion ITERATOR iterator»«IF iterator.counter1 ==
1»«LET "KeyEncryption _" + m.methodName
+ "KeyEncryption" + ";" AS Var»«Var»//You may delete or rename to use this variable i
n yourmethods«ENDLET»«ENDIF»«ENDFOREACH»
   «FOREACH m.utils AS util ITERATOR iterator»«IF iterator.counter1 == 1»«LET "Util
_" + m.methodName
+ "Util" + ";" AS Var»«Var»//You may delete or rename to use this variable in your me
thods«ENDLET»«ENDIF»«ENDFOREACH»
«ENDFOREACH»
   «EXPAND Impl FOREACH fields»
      public static void install(byte[] bArray, short bOffset, byte bLength) {
      //install method automatically generated
        new «className»(bArray,bOffset,bLength);
    }
     protected «className»(byte[] bArray, short bOffset, byte bLength){
     //constructor method automatically generated
 register();
 }
     «EXPAND Meth FOREACH methods»
     «FOREACH applets AS app»
    «IF app.method.toString() != "install".toString()»
    «IF app.method.toString() == "process".toString()»
    public void «app.method»(APDU apdu){
    byte[] buffer = apdu.getBuffer();
switch(buffer[ISO7816.OFFSET_INS])
{//in the process method all functions are called
   «FOREACH methods AS m ITERATOR iter»
    case «iter.counter1» : «m.methodName»(«FOREACH m.parameters AS mpar SEPARATOR ','
»«mpar.name»«ENDFOREACH»);
    return;
   «ENDFOREACH»
   default: ISOException.throwIt (ISO7816.SW_INS_NOT_SUPPORTED);
}}«ELSEIF app.method.toString() == "select".toString()»
public boolean «app.method»(){//  Perform any applet-
specific session initialization.
return true;}
«ELSEIF app.method.toString() == "deselect".toString()»
public void «app.method»(){
// Perform appropriate cleanup.
}«ELSE»«ENDIF»«ENDIF»«ENDFOREACH»
 }«ENDFILE»
«ENDDEFINE»
«DEFINE Impl FOR javaCardModel::Field»
«IF accessSpecifier.toString() != "none".toString()»
«accessSpecifier»«ELSE»«ENDIF»«IF isStatic» static«ELSE»«ENDIF»
«IF isFinal»final«ELSE»«ENDIF» «IF dataType.toString()!="none".toString()»
«dataType»«ELSE»«ENDIF»«IF isArray» «name»[]«ELSE» «name»
«ENDIF»«IF initialValue!=null» «IF dataType.toString()=="byte".toString()»
«IF isArray» = «initialValue»;«ELSE» = (byte)«initialValue»;«ENDIF»«ELSEIF dataType.t
oString()=="short".toString()»
«IF isArray» = «initialValue»; «ELSE» = (short)«initialValue»;«ENDIF»«ELSE»=«initialV
alue»;«ENDIF»«ELSE»;«ENDIF»«ENDDEFINE»
«DEFINE Appl FOR javaCardModel::Applet»
«ENDDEFINE»
«DEFINE Meth FOR javaCardModel::Method»
«accessSpecifier» «IF isStatic»static«ELSE»«ENDIF»
«IF returnType.toString()!="none".toString()»«returnType»«ELSE»«ENDIF»
«methodName»(«FOREACH parameters AS mpar SEPARATOR ','»«IF mpar.isArray»«mpar.dataTyp
e»[] «mpar.name»«ELSE»«mpar.dataType»
```

```
«mpar.name»«ENDIF»«ENDFOREACH»)«IF !exceptions.isEmpty»throws «FOREACH exceptions AS
mex SEPARATOR ','»«mex.exceptionType»«ENDFOREACH»«ELSE»«ENDIF»
{
    «EXPAND field FOREACH fields»
«EXPAND comment FOREACH comments»
«EXPAND keyp(methodName) FOREACH keyPairs»
«EXPAND cipher(methodName) FOREACH ciphers»
«EXPAND apdu(methodName) FOREACH apdus»
«EXPAND ownerPin(methodName) FOREACH ownerPins»
«EXPAND aid(methodName) FOREACH aids»
«EXPAND checksum(methodName) FOREACH checksums»
«EXPAND keyAgreement(methodName) FOREACH keyAgreements»
«EXPAND keyBuilder(methodName) FOREACH keyBuilders»
«EXPAND messageDigest(methodName) FOREACH messageDigests»
«EXPAND initializeMessageDigest(methodName) FOREACH initializeMessageDigests»
«EXPAND randomD(methodName) FOREACH randomData»
«EXPAND signature(methodName) FOREACH signatures»
«EXPAND dsaKey(methodName) FOREACH dsaKeys»
«EXPAND dsaPrivateKey(methodName) FOREACH dsaPrivateKeys»
«EXPAND dsaPublicKey(methodName) FOREACH dsaPublicKeys»
«EXPAND ecKey(methodName) FOREACH ecKeys»
«EXPAND ecPrivateKey(methodName) FOREACH ecPrivateKeys»
«EXPAND ecPublicKey(methodName) FOREACH ecPublicKeys»
«EXPAND key(methodName) FOREACH keys»
«EXPAND privateKey(methodName) FOREACH privateKeys»
«EXPAND publicKey(methodName) FOREACH publicKeys»
«EXPAND rsaPrivateKey(methodName) FOREACH rsaPrivateKeys»
«EXPAND rsaPrivateCrtKey(methodName) FOREACH rsaPrivateCrtKeys»
«EXPAND rsaPublicKey(methodName) FOREACH rsaPublicKeys»
«EXPAND secretKey(methodName) FOREACH secretKeys»
«EXPAND aesKey(methodName) FOREACH aesKeys»
«EXPAND desKey(methodName) FOREACH desKeys»
«EXPAND hmacKey(methodName) FOREACH hmacKeys»
«EXPAND koreanSEEDKey(methodName) FOREACH koreanSEEDKeys»
«EXPAND signatureMessageRecovery(methodName) FOREACH signatureMessageRecoveries»
«EXPAND iso7816(methodName) FOREACH iso7816s»
«EXPAND keyEncryotion(methodName) FOREACH keyEncryotions»
«EXPAND util(methodName) FOREACH utils»
«EXPAND apduEx(methodName) FOREACH apduExceptions»
«EXPAND crtEx(methodName) FOREACH cardExceptions»
«EXPAND usrEx(methodName) FOREACH userExceptions»
«EXPAND cryEx(methodName) FOREACH cryptoExceptions»
«EXPAND pinEx(methodName) FOREACH pinExceptions»
«EXPAND isoEx(methodName) FOREACH isoExceptions»
«EXPAND srvEx(methodName) FOREACH serviceExceptions»
«EXPAND sysEx(methodName) FOREACH systemExceptions»
«EXPAND tranEx(methodName) FOREACH transactionExceptions»
«EXPAND arEx(methodName) FOREACH arithmaticExceptions»
«EXPAND arrStoEx(methodName) FOREACH arrayStoreExceptions»
«EXPAND cllCastEx(methodName) FOREACH classCastExceptions»
«EXPAND inBEx(methodName) FOREACH indexOutOfBoundsExceptions»
«EXPAND arBoundsEx(methodName) FOREACH arrayIndexOutOfBoundsExceptions»
«EXPAND negativeArraySizeEx(methodName) FOREACH negativeArraySizeExceptions»
«EXPAND nullPointerEx(methodName) FOREACH nullPointerExceptions»
«EXPAND securityEx(methodName) FOREACH securityExceptions»
«EXPAND utilEx(methodName) FOREACH utilExceptions»
}
«ENDDEFINE»
«DEFINE field FOR javaCardModel::Field»«IF accessSpecifier.toString()!="none".toStrin
g()»«accessSpecifier»«ELSE»«ENDIF»«IF isStatic»static«ELSE»«ENDIF»
«IF isFinal»final«ENDIF»«IF dataType.toString()!="none".toString()»
«dataType»«ELSE»«ENDIF»«IFisArray»[]«ENDIF» «name»
«IF initialValue!=null»= «initialValue»;«ELSE»;«ENDIF»«ENDDEFINE»
«DEFINE comment FOR javaCardModel::Comment»//«commentLine»
«ENDDEFINE»
«DEFINE ownerPin(String methodName) FOR javaCardModel::OwnerPIN»
    «IF fields!=null»«IF fields.dataType.toString()!="none".toString()»«fields.dataTy
pe»
«ELSE»«ENDIF»«IF fields.isArray»«fields.name»[]= «ELSE»«fields.name» = «ENDIF»«ELSE»«ENDIF
»«IF method.toString()=="OwnerPIN".toString()»new «method»(«FOREACHparameters AS owne
```

```
rPINP SEPARATOR ','»«ownerPINP.name»«ENDFOREACH»);«ELSEIF method.toString()=="check".
toString()»«IF instanceVariable!=null»«instanceVariable.instanceVariableName».«method
»«ELSE»_«methodName»OwnerPIN.«method»«ENDIF»(«FOREACH parameters AS ownerPINP SEPARAT
OR ','»«ownerPINP.name»«ENDFOREACH»);«ELSEIF method.toString()=="getTriesRemaining".t
oString()»«IF instanceVariable!=null»«instanceVariable.instanceVariableName».«method»
«ELSE»_«methodName»OwnerPIN.«method»«ENDIF»();//this method dont need any parameter«E
LSEIF method.toString()=="getValidatedFlag".toString()»//«IF instanceVariable!=null»«
instanceVariable.instanceVariableName».«method»«ELSE»_«methodName»OwnerPIN.«method»«E
NDIF»();//this method is protected method«ELSEIF method.toString()=="isValidated".toS
tring()»«IF instanceVariable!=null»«instanceVariable.instanceVariableName».«method»«E
LSE»_«methodName»OwnerPIN.«method»«ENDIF»();//this method dont need anyparameter«ELSE
IF method.toString()=="reset".toString()»«IF instanceVariable!=null»«instanceVariable
.instanceVariableName».«method»«ELSE»_«methodName»OwnerPIN.«method»«ENDIF»();//this m
ethod dont need any parameter and Field«ELSEIF method.toString()=="resetAndUnblock".t
oString()»«IF instanceVariable!=null»«instanceVariable.instanceVariableName».«method»
«ELSE»_«methodName»OwnerPIN.«method»«ENDIF»();//this method dont need any parameter a
nd Field«ELSEIF method.toString()=="setValidatedFlag".toString()»//«IF instanceVariab
le!=null»«instanceVariable.instanceVariableName».«method»«ELSE»_«methodName»OwnerPIN.
«method»«ENDIF»(«FOREACH parameters AS ownerPINP SEPARATOR ','»«ownerPINP.name»«ENDFO
REACH»);//this method is protected method«ELSEIFmethod.toString()=="update".toString(
)»«IF instanceVariable!=null»«instanceVariable.instanceVariableName».«method»«ELSE»_«
methodName»OwnerPIN.«method»«ENDIF»(«FOREACH parameters AS ownerPINP SEPARATOR ','»«o
wnerPINP.name»«ENDFOREACH»);//thismethod dont need Field«ENDIF»
«ENDDEFINE»
«DEFINE pinEx(String methodName) FOR javaCardModel::PINException»
«IF exceptionUsageType.toString()
== "tryCatch".toString()»try{}catch(PINException «fields.name»){} «ELSEIF exceptionUsag
eType.toString()
== "ThrowNew".toString()»throw new PINException «fields.name»; «ELSEIF exceptionUsageTy
pe.toString()
=="ifElse".toString()»if() {«IF fields!=null»«IF fields.dataType.toString()!="none".t
oString()»«fields.dataType»
«ELSE»«ENDIF»«IF fields.isArray»«fields.name»[]= «ELSE»«fields.name» = «ENDIF»«ELSE»«ENDIF
»«IF method.toString()=="PINException".toString()»PINException.throwIt(«IF pinExcepti
onField.toString()=="none".toString()»«FOREACH parameters AS par»«par.name»«ENDFOREACH
»«ELSE» PINException.«pinExceptionField»«ENDIF»);}else{}«ELSEIF method.toString()=="t
hrowIt".toString()»PINException.throwIt(«IFpinExceptionField.toString()=="none".toStr
ing()»«FOREACH parameters AS par»«par.name»«ENDFOREACH»«ELSE» PINException.«pinExcepti
onField»«ENDIF»);}else{}«ELSEIF method.toString()=="getReason".toString()»«IF instanc
eVariable!=null»«instanceVariable.instanceVariableName».«method»«ELSE»_«methodName»PI
NException.«method»«ENDIF»(«IF pinExceptionField.toString()=="none".toString()»«FOREA
CH parameters AS par»«par.name»«ENDFOREACH»«ELSE» PINException.«pinExceptionField»«END
IF»);}else{}«ELSEIF method.toString()=="setReason".toString()»«IF instanceVariable!=n
ull»«instanceVariable.instanceVariableName».«method»«ELSE»_«methodName»PINException.«
method»«ENDIF»(«IF pinExceptionField.toString()=="none".toString()»«FOREACH parameter
s ASpar»«par.name»«ENDFOREACH»«ELSE» PINException.«pinExceptionField»«ENDIF»);}else{}«
ENDIF»«ELSE»«IF fields!=null»«IF fields.dataType.toString()!="none".toString()»«field
s.dataType»
«ELSE»«ENDIF»«IF fields.isArray»«fields.name»[]= «ELSE»«fields.name» =«ENDIF»«ELSE»«ENDIF»
«IF method.toString()=="PINException".toString()»PINException.throwIt(«IF pinExceptio
nField.toString()=="none".toString()»«FOREACH parameters AS par»«par.name»«ENDFOREACH»
«ELSE» PINException.«pinExceptionField»«ENDIF»);«ELSEIF method.toString()=="throwIt".
toString()»PINException.throwIt(«IF pinExceptionField.toString()=="none".toString()»«
FOREACH parameters AS par»«par.name»«ENDFOREACH»«ELSE» PINException.«pinExceptionField
»«ENDIF»);«ELSEIFmethod.toString()=="getReason".toString()»«IF instanceVariable!=null
»«instanceVariable.instanceVariableName».«method»«ELSE»_«methodName»PINException.«met
hod»«ENDIF»(«IF pinExceptionField.toString()=="none".toString()»«FOREACH parameters A
Spar»«par.name»«ENDFOREACH»«ELSE» PINException.«pinExceptionField»«ENDIF»);«ELSEIF met
hod.toString()=="setReason".toString()»«IF instanceVariable!=null»«instanceVariable.i
nstanceVariableName».«method»«ELSE»_«methodName»PINException.«method»«ENDIF»(«IF pinE
xceptionField.toString()=="none".toString()»«FOREACH parameters AS par»«par.name»«ENDF
OREACH»«ELSE» PINException.«pinExceptionField»«ENDIF»);«ENDIF» «ENDIF»
«ENDDEFINE»
«DEFINE apdu(String methodName) FOR javaCardModel::APDU»
«IF fields!=null»«IF fields.dataType.toString()!="none".toString()»«fields.dataType»
«ELSE»«ENDIF»«IF fields.isArray»«fields.name»[]= «ELSE»«fields.name» = «ENDIF»«ELSE»«ENDIF
»«IF method.toString()=="getCLAChannel".toString()»APDU.«method»();//staticmethod cal
l«ELSEIF method.toString()=="getCurrentAPDU".toString()»APDU.«method»();//static meth
od call«ELSEIF method.toString()=="getCurrentAPDUBuffer".toString()»APDU.«method»();/
```

```
/static method call«ELSEIF method.toString()=="getInBlockSize".toString()»APDU.«metho
d»();//static method call«ELSEIF method.toString()=="getOutBlockSize".toString()»APDU
.«method»();//static method call«ELSEIF method.toString()=="getProtocol".toString()»A
PDU.«method»();//static method call«ELSEIF method.toString()=="waitExtension".toStrin
g()»APDU.«method»();//static method call«ELSEIF method.toString()=="getBuffer".toStri
ng()»«IF instanceVariable!=null»«instanceVariable.instanceVariableName».«method»«ELSE
»_«methodName»APDU.«method»«ENDIF»();//this method need byteArray Field«ELSEIF method
.toString()=="getCurrentState".toString()»«IF instanceVariable!=null»«instanceVariabl
e.instanceVariableName».«method»«ELSE»_«methodName»APDU.«method»«ENDIF»();//this meth
od need byte Field«ELSEIF method.toString()=="getIncomingLength".toString()»«IF insta
nceVariable!=null»«instanceVariable.instanceVariableName».«method»«ELSE»_«methodName»
APDU.«method»«ENDIF»();//this method need short Field«ELSEIF method.toString()=="getN
AD".toString()»«IF instanceVariable!=null»«instanceVariable.instanceVariableName».«me
thod»«ELSE»_«methodName»APDU.«method»«ENDIF»();//this method need byte Field«ELSEIF m
ethod.toString()=="getOffsetCdata".toString()»«IF instanceVariable!=null»«instanceVar
iable.instanceVariableName».«method»«ELSE»_«methodName»APDU.«method»«ENDIF»();//this
method need short Field«ELSEIF method.toString()=="isCommandChainingCLA".toString()»«
IF instanceVariable!=null»«instanceVariable.instanceVariableName».«method»«ELSE»_«met
hodName»APDU.«method»«ENDIF»();//this method need boolean Field«ELSEIF method.toStrin
g()=="isISOInterindustryCLA".toString()»«IF instanceVariable!=null»«instanceVariable.
instanceVariableName».«method»«ELSE»_«methodName»APDU.«method»«ENDIF»();//this method
 needboolean Field«ELSEIF method.toString()=="isSecureMessagingCLA".toString()»«IF in
stanceVariable!=null»«instanceVariable.instanceVariableName».«method»«ELSE»_«methodNa
me»APDU.«method»«ENDIF»();//this method need boolean Field«ELSEIF method.toString()==
"isValidCLA".toString()»«IF instanceVariable!=null»«instanceVariable.instanceVariable
Name».«method»«ELSE»_«methodName»APDU.«method»«ENDIF»();//this method need boolean Fi
eld«ELSEIF method.toString()=="receiveBytes".toString()»«IF instanceVariable!=null»«i
nstanceVariable.instanceVariableName».«method»«ELSE»_«methodName»APDU.«method»«ENDIF»
(«FOREACH parameters AS apduP SEPARATOR ','»«apduP.name»«ENDFOREACH»);//this method n
eed short Field«ELSEIF method.toString()=="sendBytes".toString()»«IFinstanceVariable!
=null»«instanceVariable.instanceVariableName».«method»«ELSE»_«methodName»APDU.«method
»«ENDIF»(«FOREACH parameters AS apduP SEPARATOR ','»«apduP.name»«ENDFOREACH»);//this
method dont need any Field«ELSEIF method.toString()=="sendBytesLong".toString()»«IF i
nstanceVariable!=null»«instanceVariable.instanceVariableName».«method»«ELSE»_«methodN
ame»APDU.«method»«ENDIF»(«FOREACH parameters AS apduP SEPARATOR ','»«apduP.name»«ENDF
OREACH»);//this method dont need anyField«ELSEIF method.toString()=="setIncomingAndRe
ceive".toString()»«IF instanceVariable!=null»«instanceVariable.instanceVariableName».
«method»«ELSE»_«methodName»APDU.«method»«ENDIF»();//this method need short Field«ELSE
IF method.toString()=="setOutgoing".toString()»«IF instanceVariable!=null»«instanceVa
riable.instanceVariableName».«method»«ELSE»_«methodName»APDU.«method»«ENDIF»();//this
 method need short Field«ELSEIF method.toString()=="setOutgoingAndSend".toString()»«I
FinstanceVariable!=null»«instanceVariable.instanceVariableName».«method»«ELSE»_«metho
dName»APDU.«method»«ENDIF»(«FOREACH parameters AS apduP SEPARATOR ','»«apduP.name»«EN
DFOREACH»);//this method dont need any Field«ELSEIF method.toString()=="setOutgoingNo
Chaining".toString()»«IF instanceVariable!=null»«instanceVariable.instanceVariableNam
e».«method»«ELSE»_«methodName»APDU.«method»«ENDIF»();//this method need short Field«E
LSEIF method.toString()=="getIncomingLength".toString()»«IFinstanceVariable!=null»«in
stanceVariable.instanceVariableName».«method»«ELSE»_«methodName»APDU.«method»«ENDIF»(
);//this method need short Field«ELSEIF method.toString()=="getOffsetCdata".toString(
)»«IF instanceVariable!=null»«instanceVariable.instanceVariableName».«method»«ELSE»_«
methodName»APDU.«method»«ENDIF»();//this method need short Field«ELSEIF method.toStri
ng()=="isISOInterindustryCLA".toString()»«IF instanceVariable!=null»«instanceVariable
.instanceVariableName».«method»«ELSE»_«methodName»APDU.«method»«ENDIF»();//this metho
d need booelan Field«ELSEIF method.toString()=="isSecureMessagingCLA".toString()»«IF
instanceVariable!=null»«instanceVariable.instanceVariableName».«method»«ELSE»_«method
Name»APDU.«method»«ENDIF»();//this method need boolean Field«ELSEIF method.toString()
=="isValidCLA".toString()»«IF instanceVariable!=null»«instanceVariable.instanceVariab
leName».«method»«ELSE»_«methodName»APDU.«method»«ENDIF»();//this method need booleanF
ield«ENDIF»
«ENDDEFINE»
«DEFINE apduEx(String methodName) FOR javaCardModel::APDUException»
«IF exceptionUsageType.toString()
== "tryCatch".toString()»try{}catch(APDUException «fields.name»){}«ELSEIF exceptionUsag
eType.toString()
== "ThrowNew".toString()»throw new APDUException «fields.name»;«ELSEIF exceptionUsageTy
pe.toString()
=="ifElse".toString()»if() {«IF fields!=null»«IF fields.dataType.toString()!="none".t
oString()»«fields.dataType»
«ELSE»«ENDIF»«IF fields.isArray»«fields.name»[]= «ELSE»«fields.name» = «ENDIF»«ELSE»«ENDIF
»«IF method.toString()=="APDUException".toString()»APDUException.throwIt(«IF apduExce
```

```
ptionField.toString()=="none".toString()»«FOREACH parameters AS par»«par.name»«ENDFORE
ACH»«ELSE» APDUException.«apduExceptionField»«ENDIF»);}else{}«ELSEIF method.toString(
)=="throwIt".toString()»APDUException.throwIt(«IF apduExceptionField.toString()=="non
e".toString()»«FOREACH parameters AS par»«par.name»«ENDFOREACH»«ELSE» APDUException.«a
pduExceptionField»«ENDIF»);}else{}«ELSEIF method.toString()=="getReason".toString()»«
IF instanceVariable!=null»«instanceVariable.instanceVariableName».«method»«ELSE»_«met
hodName»APDUException.«method»«ENDIF»(«IF apduExceptionField.toString()=="none".toStr
ing()»«FOREACH parameters AS par»«par.name»«ENDFOREACH»«ELSE» APDUException.«apduExcep
tionField»«ENDIF»);}else{}«ELSEIF method.toString()=="setReason".toString()»«IF insta
nceVariable!=null»«instanceVariable.instanceVariableName».«method»«ELSE»_«methodName»
APDUException.«method»«ENDIF»(«IF apduExceptionField.toString()=="none".toString()»«F
OREACH parameters AS par»«par.name»«ENDFOREACH»«ELSE» APDUException.«apduExceptionFiel
d»«ENDIF»);}else{}«ENDIF»«ELSE»«IF fields!=null»«IF fields.dataType.toString()!="none
".toString()»«fields.dataType»
«ELSE»«ENDIF»«IFfields.isArray»«fields.name»[]= «ELSE»«fields.name» = «ENDIF»«ELSE»«ENDIF»
     «IF method.toString()=="APDUException".toString()»APDUException.throwIt(«IF apdu
ExceptionField.toString()=="none".toString()»«FOREACH parameters ASpar»«par.name»«ENDF
OREACH»«ELSE» APDUException.«apduExceptionField»«ENDIF»);«ELSEIF method.toString()=="
throwIt".toString()»APDUException.throwIt(«IF apduExceptionField.toString()=="none".t
oString()»«FOREACH parameters ASpar»«par.name»«ENDFOREACH»«ELSE» APDUException.«apduEx
ceptionField»«ENDIF»);«ELSEIF method.toString()=="getReason".toString()»«IF instanceV
ariable!=null»«instanceVariable.instanceVariableName».«method»«ELSE»_«methodName»APDU
Exception.«method»«ENDIF»(«IF apduExceptionField.toString()=="none".toString()»«FOREA
CH parameters AS par»«par.name»«ENDFOREACH»«ELSE» APDUException.«apduExceptionField»«E
NDIF»);«ELSEIF method.toString()=="setReason".toString()»«IFinstanceVariable!=null»«i
nstanceVariable.instanceVariableName».«method»«ELSE»_«methodName»APDUException.«metho
d»«ENDIF»(«IF apduExceptionField.toString()=="none".toString()»«FOREACH parameters AS
 par»«par.name»«ENDFOREACH»«ELSE»APDUException.«apduExceptionField»«ENDIF»);«ENDIF»«EN
DIF»
«ENDDEFINE»
«DEFINE aid(String methodName) FOR javaCardModel::AID»
    «IF fields!=null»«IF fields.dataType.toString()!="none".toString()»«fields.dataTy
pe»
«ELSE»«ENDIF»«IF fields.isArray»«fields.name»[]= «ELSE»«fields.name» = «ENDIF»«ELSE»«ENDIF
»«IF method.toString()=="AID".toString()»new «method»(«FOREACH parameters ASaidP SEPA
RATOR ','»«aidP.name»«ENDFOREACH»);«ELSEIF method.toString()=="equals".toString()»«IF
 instanceVariable!=null»«instanceVariable.instanceVariableName».«method»«ELSE»_«metho
dName»AID.«method»«ENDIF»(«FOREACH parameters AS aidPSEPARATOR ','»«aidP.name»«ENDFOR
EACH»);//this method need boolean field«ELSEIF method.toString()=="getBytes".toString
()»«IF instanceVariable!=null»«instanceVariable.instanceVariableName».«method»«ELSE»_
«methodName»AID.«method»«ENDIF»(«FOREACHparameters AS aidP SEPARATOR ','»«aidP.name»«
ENDFOREACH»);//this method needs parameter and  byte field «ELSEIF method.toString()=
="partialEquals".toString()»«IF instanceVariable!=null»«instanceVariable.instanceVari
ableName».«method»«ELSE»_«methodName»AID.«method»«ENDIF»(«FOREACH parameters AS aidP
SEPARATOR ','»«aidP.name»«ENDFOREACH»);//this method need parameter and boolean field
«ELSEIF method.toString()=="RIDEquals".toString()»«IF instanceVariable!=null»«instanc
eVariable.instanceVariableName».«method»«ELSE»_«methodName»AID.«method»«ENDIF»(«FOREA
CH parameters AS aidP SEPARATOR ','»«aidP.name»«ENDFOREACH»);//this method need boole
an parameter«ELSEIF method.toString()=="getPartialBytes".toString()»«IFinstanceVariab
le!=null»«instanceVariable.instanceVariableName».«method»«ELSE»_«methodName»AID.«meth
od»«ENDIF»(«FOREACH parameters AS aidP SEPARATOR ','»«aidP.name»«ENDFOREACH»);//this
method dont byte parameter«ENDIF»
«ENDDEFINE»
«DEFINE cipher(String methodName) FOR javaCardModel::Cipher»
    «IF fields!=null»«IF fields.dataType.toString()!="none".toString()»«fields.dataTy
pe»
«ELSE»«ENDIF»«IF fields.isArray»«fields.name»[]= «ELSE»«fields.name» = «ENDIF»«ELSE»«ENDIF
»«IF method.toString()=="doFinal".toString()»«IF instanceVariable!=null»«instanceVari
able.instanceVariableName».«method»«ELSE»_«methodName»Cipher.«method»«ENDIF»(«FOREACH
 parameters AS CipP SEPARATOR ','»«CipP.name»«ENDFOREACH»);//this method need paramet
ers and short field    «ELSEIF method.toString()=="getAlgorithm".toString()»«IF insta
nceVariable!=null»«instanceVariable.instanceVariableName».«method»«ELSE»_«methodName»
AID.«method»«ENDIF»();//this method byte field «ELSEIF method.toString()=="getInstanc
e".toString()»Cipher.«method»(«FOREACH parametersAS CipP SEPARATOR ','»«CipP.name»«EN
DFOREACH»);//this method need parameter and Cipher field«ELSEIF method.toString()=="i
nit".toString()»«IF instanceVariable!=null»«instanceVariable.instanceVariableName».«m
ethod»«ELSE»_«methodName»Cipher.«method»«ENDIF»(«IF cipherField.toString()=="none".to
String()»«FOREACH parameters AS cipherP SEPARATOR ','»«cipherP.name»«ENDFOREACH»«ELSE
»«FOREACH parameters AS cipherP SEPARATOR ','»«cipherP.name»«ENDFOREACH»,«cipherField
```

```
»«ENDIF»);//thismethod dont need any field«ELSEIF method.toString()=="update".toStrin
g()»«IF instanceVariable!=null»«instanceVariable.instanceVariableName».«method»«ELSE»
_«methodName»Cipher.«method»«ENDIF»(«FOREACH parameters AS CipP SEPARATOR','»«CipP.na
me»«ENDFOREACH»);//this method needs parameter and Short field«ELSEIF method.toString
()=="equals".toString()»«IF instanceVariable!=null»«instanceVariable.instanceVariable
Name».«method»«ELSE»_«methodName»Cipher.«method»«ENDIF»(«FOREACHparameters AS cipP SE
PARATOR ','»«cipP.name»«ENDFOREACH»);//this method need boolean field«ENDIF»
«ENDDEFINE»
«DEFINE checksum(String methodName) FOR javaCardModel::Checksum»
    «IF fields!=null»«IF fields.dataType.toString()!="none".toString()»«fields.dataTy
pe»
«ELSE»«ENDIF»«IF fields.isArray»«fields.name»[]= «ELSE»«fields.name» = «ENDIF»«ELSE»«ENDIF
»«IF method.toString()=="doFinal".toString()»«IF instanceVariable!=null»«instanceVari
able.instanceVariableName».«method»«ELSE»_«methodName»Checksum.«method»«ENDIF»(«FOREA
CH parameters AS ckSumP SEPARATOR ','»«ckSumP.name»«ENDFOREACH»);//this method need p
arameters and short field«ELSEIF method.toString()=="getAlgorithm".toString()»«IF ins
tanceVariable!=null»«instanceVariable.instanceVariableName».«method»«ELSE»_«methodNam
e»Checksum.«method»«ENDIF»();//this method byte field«ELSEIF method.toString()=="getI
nstance".toString()»Checksum.«method»(«IFchecksumField.toString()=="none".toString()»
«FOREACH parameters AS cipherP SEPARATOR ','»«cipherP.name»«ENDFOREACH»«ELSE»«FOREACH
 parameters AS ckSumP SEPARATOR ','»«ckSumP.name»«ENDFOREACH»,«checksumField»«ENDIF»)
;//this method needparameter and Checksum field«ELSEIF method.toString()=="init".toSt
ring()»«IF instanceVariable!=null»«instanceVariable.instanceVariableName».«method»«EL
SE»_«methodName»Checksum.«method»«ENDIF»(«FOREACH parameters AS ckSumP SEPARATOR','»«
ckSumP.name»«ENDFOREACH»);//this method dont need any field«ELSEIF method.toString()=
="update".toString()»«IF instanceVariable!=null»«instanceVariable.instanceVariableNam
e».«method»«ELSE»_«methodName»Checksum.«method»«ENDIF»(«FOREACHparameters AS ckSumP S
EPARATOR ','»«ckSumP.name»«ENDFOREACH»);//this method needs parameter and Short field
«ELSEIF method.toString()=="equals".toString()»«IF instanceVariable!=null»«instanceVa
riable.instanceVariableName».«method»«ELSE»_«methodName»Checksum.«method»«ENDIF»(«FOR
EACH parameters AS ckSumP SEPARATOR ','»«ckSumP.name»«ENDFOREACH»);//this method need
 boolean field«ENDIF»
«ENDDEFINE»
«DEFINE keyAgreement(String methodName) FOR javaCardModel::KeyAgreement»
    «IF fields!=null»«IF fields.dataType.toString()!="none".toString()»«fields.dataTy
pe»
«ELSE»«ENDIF»«IF fields.isArray»«fields.name»[]= «ELSE»«fields.name» = «ENDIF»«ELSE»«ENDIF
»«IF method.toString()=="generateSecret".toString()»«IF instanceVariable!=null»«insta
nceVariable.instanceVariableName».«method»«ELSE»_«methodName»KeyAgreement.«method»«EN
DIF»(«FOREACH parameters AS keyAgP SEPARATOR ','»«keyAgP.name»«ENDFOREACH»);//this me
thod need parameters and short field«ELSEIF method.toString()=="getAlgorithm".toStrin
g()»«IF instanceVariable!=null»«instanceVariable.instanceVariableName».«method»«ELSE»
_«methodName»KeyAgreement.«method»«ENDIF»();//this method byte field«ELSEIF method.to
String()=="getInstance".toString()»KeyAgreement.«method»(«IFkeyAgreementField.toStrin
g()=="none".toString()»«FOREACH parameters AS keyAgP SEPARATOR ','»«keyAgP.name»«ENDF
OREACH»«ELSE»«FOREACH parameters AS keyAgP SEPARATOR ','»«keyAgP.name»«ENDFOREACH»,«k
eyAgreementField»«ENDIF»);//thismethod need parameter and KeyAgreement field«ELSEIF m
ethod.toString()=="init".toString()»«IF instanceVariable!=null»«instanceVariable.inst
anceVariableName».«method»«ELSE»_«methodName»KeyAgreement.«method»«ENDIF»(«FOREACH pa
rameters AS keyAgPSEPARATOR ','»«keyAgP.name»«ENDFOREACH»);//this method dont need an
y field«ELSEIF method.toString()=="equals".toString()»«IF instanceVariable!=null»«ins
tanceVariable.instanceVariableName».«method»«ELSE»_«methodName»KeyAgreement.«method»«
ENDIF»(«FOREACH parameters AS keyAgP SEPARATOR ','»«keyAgP.name»«ENDFOREACH»);//this
method need boolean field«ENDIF»
«ENDDEFINE»
«DEFINE securityEx(String methodName) FOR javaCardModel::SecurityException»
«IF exceptionUsageType.toString()
== "tryCatch".toString()»try{}catch(SecurityException «fields.name»){}«ELSEIF exception
UsageType.toString()
== "ThrowNew".toString()»throw new SecurityException «fields.name»; «ELSEIF exceptionUs
ageType.toString()
=="ifElse".toString()»if() {«IF fields!=null»«IF fields.dataType.toString()!="none".t
oString()»«fields.dataType»
«ELSE»«ENDIF»«IF fields.isArray»«fields.name»[]= «ELSE»«fields.name» = «ENDIF»«ELSE»«ENDIF
»«IF method.toString()=="SecurityException".toString()»SecurityException.throwIt(«FOR
EACH parameters AS par»«par.name»«ENDFOREACH»);}else{}«ELSE »«IF instanceVariable!=nul
l»«instanceVariable.instanceVariableName».«method»«ELSE»_«methodName»SecurityExceptio
n.«method»«ENDIF»(«FOREACH parameters ASpar»«par.name»«ENDFOREACH»);}else{}«ENDIF»«ELS
E»«IF fields!=null»«IF fields.dataType.toString()!="none".toString()»«fields.dataType
»
```

```
«ELSE»«ENDIF»«IF fields.isArray»«fields.name»[]= «ELSE»«fields.name» = «ENDIF»«ELSE»«ENDIF
»«IF method.toString()=="SecurityException".toString()»SecurityException.throwIt(«FOR
EACH parameters AS par»«par.name»«ENDFOREACH»);«ELSE»«IF instanceVariable!=null»«insta
nceVariable.instanceVariableName».«method»«ELSE»_«methodName»SecurityException.«metho
d»«ENDIF»(«FOREACH parameters AS par»«par.name»«ENDFOREACH»);«ENDIF»«ENDIF»
«ENDDEFINE»
«DEFINE keyBuilder(String methodName) FOR javaCardModel::KeyBuilder»
«IF fields!=null»«IF fields.dataType.toString()!="none".toString()»«fields.dataType»
«ELSE»«ENDIF»«IF fields.isArray»«fields.name»[]= «ELSE»«fields.name» = «ENDIF»«ELSE»«ENDIF
»«IF method.toString()=="buildKey".toString()»KeyBuilder.«method»(«IFkeyBuilderField.
toString()=="none".toString()»KeyBuilder.«keyBuilderField»,«FOREACH parameters AS key
BP SEPARATOR ','»«keyBP.name»«ENDFOREACH»«ELSE»«FOREACH parameters AS keyBP SEPARATOR
 ','»«keyBP.name»«ENDFOREACH»«ENDIF»);//thismethod need parameter and KeyBuilder fiel
d //If you use Lenght field you have to remove your fields space with eachother«ELSEI
F method.toString()=="equals".toString()»«IF instanceVariable!=null»«instanceVariable
.instanceVariableName».«method»«ELSE»_«methodName»KeyBuilder.«method»«ENDIF»(«FOREACH
 parameters AS keyAgP SEPARATOR ','»«keyAgP.name»«ENDFOREACH»);//this method need boo
lean field«ENDIF»
«ENDDEFINE»
«DEFINE keyp(String methodName) FOR javaCardModel::KeyPair»
«IF fields!=null»«IF fields.dataType.toString()!="none".toString()»«fields.dataType»
«ELSE»«ENDIF»«IF fields.isArray»«fields.name»[]= «ELSE»«fields.name» = «ENDIF»«ELSE»«ENDIF
»«IF method.toString()=="KeyPair".toString()»new «method»(«IFkeyPairField.toString()!
="none".toString()»KeyPair.«keyPairField»,«FOREACH parameters AS keyPairP SEPARATOR '
,'»«keyPairP.name»«ENDFOREACH»«ELSE»«FOREACH parameters AS keyPairP SEPARATOR ','»«ke
yPairP.name»«ENDFOREACH»«ENDIF»);«ELSEIFmethod.toString()=="genKeyPair".toString()»«I
F instanceVariable!=null»«instanceVariable.instanceVariableName».«method»«ELSE»_«meth
odName»KeyPair.«method»«ENDIF»();//this method dont need any parameter and field.«ELS
EIF method.toString()=="getPrivate".toString()»«IF instanceVariable!=null»«instanceVa
riable.instanceVariableName».«method»«ELSE»_«methodName»KeyPair.«method»«ENDIF»();//t
his method need PrivateKey field.«ELSEIF method.toString()=="getPublic".toString()»«I
F instanceVariable!=null»«instanceVariable.instanceVariableName».«method»«ELSE»_«meth
odName»KeyPair.«method»«ENDIF»();//this method need PublicKey field.«ELSEIF method.to
String()=="equals".toString()»«IF instanceVariable!=null»«instanceVariable.instanceVa
riableName».«method»«ELSE»_«methodName»KeyPair.«method»«ENDIF»(«FOREACH parameters AS
 keyPairP SEPARATOR ','»«keyPairP.name»«ENDFOREACH»);//this method need boolean field
«ENDIF»
«ENDDEFINE»
«DEFINE messageDigest(String methodName) FOR javaCardModel::MessageDigest»
«IF fields!=null»«IF fields.dataType.toString()!="none".toString()»«fields.dataType»
«ELSE»«ENDIF»«IF fields.isArray»«fields.name»[]= «ELSE»«fields.name» = «ENDIF»«ELSE»«ENDIF
»«IF method.toString()=="doFinal".toString()»«IF instanceVariable!=null»«instanceVari
able.instanceVariableName».«method»«ELSE»_«methodName»MessageDigest.«method»«ENDIF»(«
FOREACH parameters AS mDigP SEPARATOR ','»«mDigP.name»«ENDFOREACH»);//this method nee
ds paramters and short field.«ELSEIF method.toString()=="getAlgorithm".toString()»«IF
 instanceVariable!=null»«instanceVariable.instanceVariableName».«method»«ELSE»_«metho
dName»MessageDigest.«method»«ENDIF»();//this method need byte field.«ELSEIF method.to
String()=="getInitializedMessageDigestInstance".toString()»MessageDigest.«method»(«IF
 messageDigestField.toString()!="none".toString()»MessageDigest.«messageDigestField»,
«FOREACH parameters AS mDigP SEPARATOR ','»«mDigP.name»«ENDFOREACH»«ELSE»«FOREACH par
ameters AS mDigP SEPARATOR','»«mDigP.name»«ENDFOREACH»«ENDIF»);//this method need Ini
tializedMessageDigest field.«ELSEIF method.toString()=="getInstance".toString()»Messa
geDigest.«method»(«IF messageDigestField.toString()!="none".toString()»MessageDigest.
«messageDigestField»,«FOREACH parameters AS mDigP SEPARATOR ','»«mDigP.name»«ENDFOREA
CH»«ELSE»«FOREACH parameters AS mDigP SEPARATOR ','»«mDigP.name»«ENDFOREACH»«ENDIF»);
//this method need MessageDigest field.«ELSEIF method.toString()=="getLength".toStrin
g()»«IFinstanceVariable!=null»«instanceVariable.instanceVariableName».«method»«ELSE»_
«methodName»MessageDigest.«method»«ENDIF»();//this method need byte field.«ELSEIF met
hod.toString()=="reset".toString()»«IF instanceVariable!=null»«instanceVariable.insta
nceVariableName».«method»«ELSE»_«methodName»MessageDigest.«method»«ENDIF»();//this me
thod dont need parameter or field.«ELSEIF method.toString()=="update".toString()»«IF
instanceVariable!=null»«instanceVariable.instanceVariableName».«method»«ELSE»_«method
Name»MessageDigest.«method»«ENDIF»(«FOREACH parameters AS mDigP SEPARATOR ','»«mDigP.
name»«ENDFOREACH»);//this method dont need any field.«ELSEIF method.toString()=="equa
ls".toString()»«IF instanceVariable!=null»«instanceVariable.instanceVariableName».«me
thod»«ELSE»_«methodName»MessageDigest.«method»«ENDIF»(«FOREACH parameters AS mDigP SE
PARATOR ','»«mDigP.name»«ENDFOREACH»);//this method dont need any field.«ENDIF»
«ENDDEFINE»
«DEFINE initializeMessageDigest(String
methodName) FOR javaCardModel::InitializedMessageDigest»
```

```
«IF fields!=null»«IF fields.dataType.toString()!="none".toString()»«fields.dataType»
«ELSE»«ENDIF»«IF fields.isArray»«fields.name»[]= «ELSE»«fields.name» = «ENDIF»«ELSE»«ENDIF
»«IF method.toString()=="setInitialDigest".toString()»«IF instanceVariable!=null»«ins
tanceVariable.instanceVariableName».«method»«ELSE»_«methodName»InitializedMessageDige
st.«method»«ENDIF»(«FOREACH parameters AS mDigP SEPARATOR ','»«mDigP.name»«ENDFOREACH
»);//this method needs paramters.«ELSEIF method.toString()=="doFinal".toString()»«IF
instanceVariable!=null»«instanceVariable.instanceVariableName».«method»«ELSE»_«method
Name»InitializedMessageDigest.«method»«ENDIF»(«FOREACH parameters AS mDigP SEPARATOR
','»«mDigP.name»«ENDFOREACH»);//this method needs paramtersand short field.«ELSEIF me
thod.toString()=="getAlgorithm".toString()»«IF instanceVariable!=null»«instanceVariab
le.instanceVariableName».«method»«ELSE»_«methodName»InitializedMessageDigest.«method»
«ENDIF»();//this method need byte field.«ELSEIFmethod.toString()=="getInitializedMess
ageDigestInstance".toString()»InitializedMessageDigest.«method»(«IF initializedMessag
eDigestField.toString()!="none".toString()»InitializedMessageDigest.«initializedMessa
geDigestField»,«FOREACH parameters AS mDigP SEPARATOR','»«mDigP.name»«ENDFOREACH»«ELS
E»«FOREACH parameters AS mDigP SEPARATOR ','»«mDigP.name»«ENDFOREACH»«ENDIF»);//this
method need InitializedMessageDigest field.«ELSEIF method.toString()=="getInstance".t
oString()»InitializedMessageDigest.«method»(«IF initializedMessageDigestField.toStrin
g()!="none".toString()»InitializedMessageDigest.«initializedMessageDigestField»,«FORE
ACH parameters AS mDigP SEPARATOR ','»«mDigP.name»«ENDFOREACH»«ELSE»«FOREACH paramete
rs ASmDigP SEPARATOR ','»«mDigP.name»«ENDFOREACH»«ENDIF»);//this method need MessageD
igest field.«ELSEIF method.toString()=="getLength".toString()»«IF instanceVariable!=n
ull»«instanceVariable.instanceVariableName».«method»«ELSE»_«methodName»InitializedMes
sageDigest.«method»«ENDIF»();//this method need byte field.«ELSEIF method.toString()=
="reset".toString()»«IF instanceVariable!=null»«instanceVariable.instanceVariableName
».«method»«ELSE»_«methodName»InitializedMessageDigest.«method»«ENDIF»();//this method
dont need parameter or field.«ELSEIF method.toString()=="update".toString()»«IF insta
nceVariable!=null»«instanceVariable.instanceVariableName».«method»«ELSE»_«methodName»
InitializedMessageDigest.«method»«ENDIF»(«FOREACH parameters AS mDigP SEPARATOR','»«m
DigP.name»«ENDFOREACH»);//this method dont need any field.«ELSEIF method.toString()==
"equals".toString()»«IF instanceVariable!=null»«instanceVariable.instanceVariableName
».«method»«ELSE»_«methodName»InitializedMessageDigest.«method»«ENDIF»(«FOREACH parame
ters AS mDigP SEPARATOR ','»«mDigP.name»«ENDFOREACH»);//this method dont need any fie
ld.«ENDIF»
«ENDDEFINE»
«DEFINE randomD(String methodName) FOR javaCardModel::RandomData»
«IF fields!=null»«IF fields.dataType.toString()!="none".toString()»«fields.dataType»
«ELSE»«ENDIF»«IF fields.isArray»«fields.name»[]= «ELSE»«fields.name» = «ENDIF»«ELSE»«ENDIF
»«IF method.toString()=="generateData".toString()»«IF instanceVariable!=null»«instanc
eVariable.instanceVariableName».«method»«ELSE»_«methodName»RandomData.«method»«ENDIF»
(«FOREACH parameters AS rDataP SEPARATOR ','»«rDataP.name»«ENDFOREACH»);//this method
 needs paramters but dont need field.«ELSEIFmethod.toString()=="setSeed".toString()»«
IF instanceVariable!=null»«instanceVariable.instanceVariableName».«method»«ELSE»_«met
hodName»RandomData.«method»«ENDIF»(«FOREACH parameters AS rDataP SEPARATOR ','»«rData
P.name»«ENDFOREACH»);//thismethod needs paramters but dont need field.«ELSEIF method.
toString()=="getInstance".toString()»RandomData.«method»(«IF randomDataField.toString
()!="none".toString()»RandomData.«randomDataField»«ELSE»«FOREACH parameters AS rDataP
 SEPARATOR','»«rDataP.name»«ENDFOREACH»«ENDIF»);//this method need RandomData field.«
ELSEIF method.toString()=="equals".toString()»«IF instanceVariable!=null»«instanceVar
iable.instanceVariableName».«method»«ELSE»_«methodName»RandomData.«method»«ENDIF»(«FO
REACH parameters AS rDataP SEPARATOR ','»«rDataP.name»«ENDFOREACH»);//this method don
t need any field.«ENDIF»
«ENDDEFINE»
«DEFINE signature(String methodName) FOR javaCardModel::Signature»
«IF fields!=null»«IF fields.dataType.toString()!="none".toString()»«fields.dataType»
«ELSE»«ENDIF»«IF fields.isArray»«fields.name»[]= «ELSE»«fields.name» = «ENDIF»«ELSE»«ENDIF
»«IF method.toString()=="getAlgorithm".toString()»«IF instanceVariable!=null»«instanc
eVariable.instanceVariableName».«method»«ELSE»_«methodName»Signature.«method»«ENDIF»(
);//this method dont need any field.«ELSEIF method.toString()=="getInstance".toString
()»Signature.«method»(«IF signatureField.toString()!="none".toString()»Signature.«sig
natureField»,«FOREACH parameters AS sigP SEPARATOR ','»«sigP.name»«ENDFOREACH»«ELSE»«
FOREACH parameters AS sigP SEPARATOR ','»«sigP.name»«ENDFOREACH»«ENDIF»);//this metho
d need RandomData field.«ELSEIF method.toString()=="getLength".toString()»«IF instanc
eVariable!=null»«instanceVariable.instanceVariableName».«method»«ELSE»_«methodName»Si
gnature.«method»«ENDIF»();//this method needs short field.«ELSEIF method.toString()==
"init".toString()»«IF instanceVariable!=null»«instanceVariable.instanceVariableName».
«method»«ELSE»_«methodName»Signature.«method»«ENDIF»(«FOREACH parameters AS sigP SEPA
RATOR ','»«sigP.name»«ENDFOREACH»);//this method dont need any field.«ELSEIF method.t
oString()=="sign".toString()»«IFinstanceVariable!=null»«instanceVariable.instanceVari
ableName».«method»«ELSE»_«methodName»Signature.«method»«ENDIF»(«FOREACH parameters AS
```

```
 sigP SEPARATOR ','»«sigP.name»«ENDFOREACH»);//this method need short field.«ELSEIFme
thod.toString()=="update".toString()»«IF instanceVariable!=null»«instanceVariable.ins
tanceVariableName».«method»«ELSE»_«methodName»Signature.«method»«ENDIF»(«FOREACH para
meters AS sigP SEPARATOR ','»«sigP.name»«ENDFOREACH»);//this method dontneed any fiel
d.«ELSEIF method.toString()=="verify".toString()»«IF instanceVariable!=null»«instance
Variable.instanceVariableName».«method»«ELSE»_«methodName»Signature.«method»«ENDIF»(«
FOREACH parameters AS sigP SEPARATOR','»«sigP.name»«ENDFOREACH»);//this method need b
oolean field.«ELSEIF method.toString()=="equals".toString()»«IF instanceVariable!=nul
l»«instanceVariable.instanceVariableName».«method»«ELSE»_«methodName»Signature.«metho
d»«ENDIF»(«FOREACH parametersAS sigP SEPARATOR ','»«sigP.name»«ENDFOREACH»);//this me
thod dont need any field.«ENDIF»
«ENDDEFINE»
«DEFINE dsaKey(String methodName) FOR javaCardModel::DSAKey»
«IF fields!=null»«IF fields.dataType.toString()!="none".toString()»«fields.dataType»
«ELSE»«ENDIF»«IF fields.isArray»«fields.name»[]= «ELSE»«fields.name» = «ENDIF»«ELSE»«ENDIF
»«IF method.toString()=="getG".toString()»«IF instanceVariable!=null»«instanceVariabl
e.instanceVariableName».«method»«ELSE»_«methodName»DSAKey.«method»«ENDIF»(«FOREACH pa
rameters AS dsaKP SEPARATOR ','»«dsaKP.name»«ENDFOREACH»);//This method needs paramet
ers and short field.«ELSEIF method.toString()=="getP".toString()»«IF instanceVariable
!=null»«instanceVariable.instanceVariableName».«method»«ELSE»_«methodName»DSAKey.«met
hod»«ENDIF»(«FOREACH parameters AS dsaKP SEPARATOR ','»«dsaKP.name»«ENDFOREACH»);//Th
is method needs parameters and shortfield.«ELSEIF method.toString()=="getQ".toString(
)»«IF instanceVariable!=null»«instanceVariable.instanceVariableName».«method»«ELSE»_«
methodName»DSAKey.«method»«ENDIF»(«FOREACH parameters AS dsaKP SEPARATOR ','»«dsaKP.n
ame»«ENDFOREACH»);//This method needs parameters and short field.«ELSEIF method.toStr
ing()=="setG".toString()»«IF instanceVariable!=null»«instanceVariable.instanceVariabl
eName».«method»«ELSE»_«methodName»DSAKey.«method»«ENDIF»(«FOREACH parameters AS dsaKP
 SEPARATOR','»«dsaKP.name»«ENDFOREACH»);//this method dont need any field.«ELSEIF met
hod.toString()=="setP".toString()»«IF instanceVariable!=null»«instanceVariable.instan
ceVariableName».«method»«ELSE»_«methodName»DSAKey.«method»«ENDIF»(«FOREACH parameters
AS dsaKP SEPARATOR ','»«dsaKP.name»«ENDFOREACH»);//this method dont need any field.«E
LSEIF method.toString()=="setQ".toString()»«IF instanceVariable!=null»«instanceVariab
le.instanceVariableName».«method»«ELSE»_«methodName»DSAKey.«method»«ENDIF»(«FOREACH p
arameters AS dsaKP SEPARATOR ','»«dsaKP.name»«ENDFOREACH»);//this method dont need an
y field.«ENDIF»
«ENDDEFINE»
«DEFINE dsaPrivateKey(String methodName) FOR javaCardModel::DSAPrivateKey»
«IF fields!=null»«IF fields.dataType.toString()!="none".toString()»«fields.dataType»
«ELSE»«ENDIF»«IF fields.isArray»«fields.name»[]= «ELSE»«fields.name» = «ENDIF»«ELSE»«ENDIF
»«IF method.toString()=="getG".toString()»«IF instanceVariable!=null»«instanceVariabl
e.instanceVariableName».«method»«ELSE»_«methodName»DSAPrivateKey.«method»«ENDIF»(«FOR
EACH parameters AS dsaKP SEPARATOR ','»«dsaKP.name»«ENDFOREACH»);//This method needs
parameters and short field.«ELSEIFmethod.toString()=="getP".toString()»«IF instanceVa
riable!=null»«instanceVariable.instanceVariableName».«method»«ELSE»_«methodName»DSAPr
ivateKey.«method»«ENDIF»(«FOREACH parameters AS dsaKP SEPARATOR ','»«dsaKP.name»«ENDF
OREACH»);//This methodneeds parameters and short field.«ELSEIF method.toString()=="ge
tQ".toString()»«IF instanceVariable!=null»«instanceVariable.instanceVariableName».«me
thod»«ELSE»_«methodName»DSAPrivateKey.«method»«ENDIF»(«FOREACH parameters AS dsaKP SE
PARATOR','»«dsaKP.name»«ENDFOREACH»);//This method needs parameters and short field.«
ELSEIF method.toString()=="setG".toString()»«IF instanceVariable!=null»«instanceVaria
ble.instanceVariableName».«method»«ELSE»_«methodName»DSAPrivateKey.«method»«ENDIF»(«F
OREACH parameters AS dsaKP SEPARATOR ','»«dsaKP.name»«ENDFOREACH»);//this method dont
 need any field.«ELSEIF method.toString()=="setP".toString()»«IF instanceVariable!=nu
ll»«instanceVariable.instanceVariableName».«method»«ELSE»_«methodName»DSAPrivateKey.«
method»«ENDIF»(«FOREACH parameters AS dsaKP SEPARATOR ','»«dsaKP.name»«ENDFOREACH»);/
/this method dont need any field.«ELSEIF method.toString()=="setQ".toString()»«IF ins
tanceVariable!=null»«instanceVariable.instanceVariableName».«method»«ELSE»_«methodNam
e»DSAPrivateKey.«method»«ENDIF»(«FOREACH parameters AS dsaKP SEPARATOR ','»«dsaKP.nam
e»«ENDFOREACH»);//this method dont need any field.
«ELSEIF method.toString()=="getX".toString()»«IFinstanceVariable!=null»«instanceVaria
ble.instanceVariableName».«method»«ELSE»_«methodName»DSAPrivateKey.«method»«ENDIF»(«F
OREACH parameters AS dsaKP SEPARATOR ','»«dsaKP.name»«ENDFOREACH»);//This method need
s parameters and shortfield.«ELSEIF method.toString()=="setX".toString()»«IF instance
Variable!=null»«instanceVariable.instanceVariableName».«method»«ELSE»_«methodName»DSA
PrivateKey.«method»«ENDIF»(«FOREACH parameters AS dsaKP SEPARATOR','»«dsaKP.name»«END
FOREACH»);//this method dont need any field.
«ELSEIF method.toString()=="clearKey".toString()»«IF instanceVariable!=null»«instance
Variable.instanceVariableName».«method»«ELSE»_«methodName»DSAPrivateKey.«method»«ENDI
F»();//thismethod dont need any parameters or fields.«ELSEIF method.toString()=="getS
ize".toString()»«IF instanceVariable!=null»«instanceVariable.instanceVariableName».«m
```

```
ethod»«ELSE»_«methodName»DSAPrivateKey.«method»«ENDIF»();//this method need short fie
ld.«ELSEIFmethod.toString()=="getType".toString()»«IF instanceVariable!=null»«instanc
eVariable.instanceVariableName».«method»«ELSE»_«methodName»DSAPrivateKey.«method»«END
IF»();//this method need byte field.«ELSEIF method.toString()=="isInitialized".toStri
ng()»«IFinstanceVariable!=null»«instanceVariable.instanceVariableName».«method»«ELSE»
_«methodName»DSAPrivateKey.«method»«ENDIF»();//this method need boolean field.«ENDIF»
«ENDDEFINE»
«DEFINE dsaPublicKey(String methodName) FOR javaCardModel::DSAPublicKey»
«IF fields!=null»«IF fields.dataType.toString()!="none".toString()»«fields.dataType»
«ELSE»«ENDIF»«IF fields.isArray»«fields.name»[]= «ELSE»«fields.name» = «ENDIF»«ELSE»«ENDIF
»«IF method.toString()=="getG".toString()»«IF instanceVariable!=null»«instanceVariabl
e.instanceVariableName».«method»«ELSE»_«methodName»DSAPublicKey.«method»«ENDIF»(«FORE
ACH parameters AS dsaKP SEPARATOR ','»«dsaKP.name»«ENDFOREACH»);//This method needs p
arameters and short field.«ELSEIFmethod.toString()=="getP".toString()»«IF instanceVar
iable!=null»«instanceVariable.instanceVariableName».«method»«ELSE»_«methodName»DSAPub
licKey.«method»«ENDIF»(«FOREACH parameters AS dsaKP SEPARATOR ','»«dsaKP.name»«ENDFOR
EACH»);//This methodneeds parameters and short field.«ELSEIF method.toString()=="getQ
".toString()»«IF instanceVariable!=null»«instanceVariable.instanceVariableName».«meth
od»«ELSE»_«methodName»DSAPublicKey.«method»«ENDIF»(«FOREACH parameters AS dsaKP SEPAR
ATOR','»«dsaKP.name»«ENDFOREACH»);//This method needs parameters and short field.«ELS
EIF method.toString()=="setG".toString()»«IF instanceVariable!=null»«instanceVariable
.instanceVariableName».«method»«ELSE»_«methodName»DSAPublicKey.«method»«ENDIF»(«FOREA
CH parameters AS dsaKP SEPARATOR ','»«dsaKP.name»«ENDFOREACH»);//this method dont nee
d any field.«ELSEIF method.toString()=="setP".toString()»«IF instanceVariable!=null»«
instanceVariable.instanceVariableName».«method»«ELSE»_«methodName»DSAPublicKey.«metho
d»«ENDIF»(«FOREACH parameters AS dsaKP SEPARATOR ','»«dsaKP.name»«ENDFOREACH»);//this
 method dont need any field.«ELSEIF method.toString()=="setQ".toString()»«IF instance
Variable!=null»«instanceVariable.instanceVariableName».«method»«ELSE»_«methodName»DSA
PublicKey.«method»«ENDIF»(«FOREACH parameters AS dsaKP SEPARATOR ','»«dsaKP.name»«END
FOREACH»);//this method dont need any field.
«ELSEIF method.toString()=="getY".toString()»«IFinstanceVariable!=null»«instanceVaria
ble.instanceVariableName».«method»«ELSE»_«methodName»DSAPublicKey.«method»«ENDIF»(«FO
REACH parameters AS dsaKP SEPARATOR ','»«dsaKP.name»«ENDFOREACH»);//This method needs
 parameters and short field.«ELSEIFmethod.toString()=="setY".toString()»«IF instanceV
ariable!=null»«instanceVariable.instanceVariableName».«method»«ELSE»_«methodName»DSAP
ublicKey.«method»«ENDIF»(«FOREACH parameters AS dsaKP SEPARATOR ','»«dsaKP.name»«ENDF
OREACH»);//this methoddont need any field.
«ELSEIF method.toString()=="clearKey".toString()»«IF instanceVariable!=null»«instance
Variable.instanceVariableName».«method»«ELSE»_«methodName»DSAPublicKey.«method»«ENDIF
»();//this method dont need any parameters or fields.«ELSEIFmethod.toString()=="getSi
ze".toString()»«IF instanceVariable!=null»«instanceVariable.instanceVariableName».«me
thod»«ELSE»_«methodName»DSAPublicKey.«method»«ENDIF»();//this method need short field
.«ELSEIF method.toString()=="getType".toString()»«IFinstanceVariable!=null»«instanceV
ariable.instanceVariableName».«method»«ELSE»_«methodName»DSAPublicKey.«method»«ENDIF»
();//this method need byte field.«ELSEIF method.toString()=="isInitialized".toString(
)»«IF instanceVariable!=null»«instanceVariable.instanceVariableName».«method»«ELSE»_«
methodName»DSAPublicKey.«method»«ENDIF»();//this method need boolean field.«ENDIF»
«ENDDEFINE»
«DEFINE ecKey(String methodName) FOR javaCardModel::ECKey»
«IF fields!=null»«IF fields.dataType.toString()!="none".toString()»«fields.dataType»
«ELSE»«ENDIF»«IF fields.isArray»«fields.name»[]= «ELSE»«fields.name» = «ENDIF»«ELSE»«ENDIF
»«IF method.toString()=="getA".toString()»«IF instanceVariable!=null»«instanceVariabl
e.instanceVariableName».«method»«ELSE»_«methodName»ECKey.«method»«ENDIF»(«FOREACH par
ameters AS ecKP SEPARATOR ','»«ecKP.name»«ENDFOREACH»);//This method needs parameters
 and short field.«ELSEIF method.toString()=="getB".toString()»«IF instanceVariable!=n
ull»«instanceVariable.instanceVariableName».«method»«ELSE»_«methodName»ECKey.«method»
«ENDIF»(«FOREACH parameters AS ecKP SEPARATOR ','»«ecKP.name»«ENDFOREACH»);//This met
hod needs parameters and short field.
«ELSEIF method.toString()=="getField".toString()»«IF instanceVariable!=null»«instance
Variable.instanceVariableName».«method»«ELSE»_«methodName»ECKey.«method»«ENDIF»(«FORE
ACH parameters AS ecKP SEPARATOR ','»«ecKP.name»«ENDFOREACH»);//Thismethod needs para
meters and short field.«ELSEIF method.toString()=="getG".toString()»«IF instanceVaria
ble!=null»«instanceVariable.instanceVariableName».«method»«ELSE»_«methodName»ECKey.«m
ethod»«ENDIF»(«FOREACH parameters AS ecKP SEPARATOR','»«ecKP.name»«ENDFOREACH»);//Thi
s method needs parameters and short field.«ELSEIF method.toString()=="getK".toString(
)»«IF instanceVariable!=null»«instanceVariable.instanceVariableName».«method»«ELSE»_«
methodName»ECKey.«method»«ENDIF»();//this methoddont need any parameter, but need sho
rt field.«ELSEIF method.toString()=="getR".toString()»«IF instanceVariable!=null»«ins
tanceVariable.instanceVariableName».«method»«ELSE»_«methodName»ECKey.«method»«ENDIF»(
«FOREACH parameters AS ecKP SEPARATOR','»«ecKP.name»«ENDFOREACH»);//This method needs
```

```
 parameters and short field.
«ELSEIF method.toString()=="setA".toString()»«IF instanceVariable!=null»«instanceVari
able.instanceVariableName».«method»«ELSE»_«methodName»ECKey.«method»«ENDIF»(«FOREACH
parameters AS ecKP SEPARATOR ','»«ecKP.name»«ENDFOREACH»);//this method dont need any
 field.
«ELSEIF method.toString()=="setB".toString()»«IF instanceVariable!=null»«instanceVari
able.instanceVariableName».«method»«ELSE»_«methodName»ECKey.«method»«ENDIF»(«FOREACH
parameters AS ecKP SEPARATOR ','»«ecKP.name»«ENDFOREACH»);//this method dont need any
 field.
«ELSEIF method.toString()=="setFieldF2M".toString()»«IF instanceVariable!=null»«insta
nceVariable.instanceVariableName».«method»«ELSE»_«methodName»ECKey.«method»«ENDIF»(«F
OREACH parameters AS ecKP SEPARATOR ','»«ecKP.name»«ENDFOREACH»);//this method dont n
eed any field.«ELSEIF method.toString()=="setFieldFP".toString()»«IFinstanceVariable!
=null»«instanceVariable.instanceVariableName».«method»«ELSE»_«methodName»ECKey.«metho
d»«ENDIF»(«FOREACH parameters AS ecKP SEPARATOR ','»«ecKP.name»«ENDFOREACH»);//this m
ethod dont need any field.«ELSEIFmethod.toString()=="setG".toString()»«IF instanceVar
iable!=null»«instanceVariable.instanceVariableName».«method»«ELSE»_«methodName»ECKey.
«method»«ENDIF»(«FOREACH parameters AS ecKP SEPARATOR ','»«ecKP.name»«ENDFOREACH»);//
this method dont needany field.«ELSEIF method.toString()=="setK".toString()»«IF insta
nceVariable!=null»«instanceVariable.instanceVariableName».«method»«ELSE»_«methodName»
ECKey.«method»«ENDIF»(«FOREACH parameters AS ecKP SEPARATOR ','»«ecKP.name»«ENDFOREAC
H»);//thismethod dont need any field.«ELSEIF method.toString()=="setR".toString()»«IF
 instanceVariable!=null»«instanceVariable.instanceVariableName».«method»«ELSE»_«metho
dName»ECKey.«method»«ENDIF»(«FOREACH parameters AS ecKP SEPARATOR','»«ecKP.name»«ENDF
OREACH»);//this method dont need any field.«ENDIF»
«ENDDEFINE»
«DEFINE ecPrivateKey(String methodName) FOR javaCardModel::ECPrivateKey»
«IF fields!=null»«IF fields.dataType.toString()!="none".toString()»«fields.dataType»
«ELSE»«ENDIF»«IF fields.isArray»«fields.name»[]= «ELSE»«fields.name» = «ENDIF»«ELSE»«ENDIF
»«IF method.toString()=="getS".toString()»«IF instanceVariable!=null»«instanceVariabl
e.instanceVariableName».«method»«ELSE»_«methodName»ECPrivateKey.«method»«ENDIF»(«FORE
ACH parameters AS ecKP SEPARATOR ','»«ecKP.name»«ENDFOREACH»);//This method needs par
ameters and short field.«ELSEIF method.toString()=="setS".toString()»«IF instanceVari
able!=null»«instanceVariable.instanceVariableName».«method»«ELSE»_«methodName»ECPriva
teKey.«method»«ENDIF»(«FOREACH parameters AS ecKP SEPARATOR ','»«ecKP.name»«ENDFOREAC
H»);//this method dont need any field.«ELSEIFmethod.toString()=="getA".toString()»«IF
 instanceVariable!=null»«instanceVariable.instanceVariableName».«method»«ELSE»_«metho
dName»ECPrivateKey.«method»«ENDIF»(«FOREACH parameters AS ecKP SEPARATOR ','»«ecKP.na
me»«ENDFOREACH»);//This methodneeds parameters and short field.«ELSEIF method.toStrin
g()=="getB".toString()»«IF instanceVariable!=null»«instanceVariable.instanceVariableN
ame».«method»«ELSE»_«methodName»ECPrivateKey.«method»«ENDIF»(«FOREACH parameters AS e
cKP SEPARATOR','»«ecKP.name»«ENDFOREACH»);//This method needs parameters and short fi
eld.«ELSEIF method.toString()=="getField".toString()»«IF instanceVariable!=null»«inst
anceVariable.instanceVariableName».«method»«ELSE»_«methodName»ECPrivateKey.«method»«E
NDIF»(«FOREACH parameters AS ecKP SEPARATOR ','»«ecKP.name»«ENDFOREACH»);//This metho
d needs parameters and short field.«ELSEIF method.toString()=="getG".toString()»«IF i
nstanceVariable!=null»«instanceVariable.instanceVariableName».«method»«ELSE»_«methodN
ame»ECPrivateKey.«method»«ENDIF»(«FOREACH parameters AS ecKP SEPARATOR ','»«ecKP.name
»«ENDFOREACH»);//This method needs parameters and short field.«ELSEIF method.toString
()=="getK".toString()»«IFinstanceVariable!=null»«instanceVariable.instanceVariableNam
e».«method»«ELSE»_«methodName»ECPrivateKey.«method»«ENDIF»();//this method dont need
 any parameter, but need short field.«ELSEIF method.toString()=="getR".toString()»«IF
instanceVariable!=null»«instanceVariable.instanceVariableName».«method»«ELSE»_«method
Name»ECPrivateKey.«method»«ENDIF»(«FOREACH parameters AS ecKP SEPARATOR ','»«ecKP.nam
e»«ENDFOREACH»);//This method needs parameters and short field.«ELSEIF method.toStrin
g()=="setA".toString()»«IF instanceVariable!=null»«instanceVariable.instanceVariableN
ame».«method»«ELSE»_«methodName»ECPrivateKey.«method»«ENDIF»(«FOREACH parameters AS e
cKP SEPARATOR ','»«ecKP.name»«ENDFOREACH»);//this method dont need any field.
«ELSEIF method.toString()=="setB".toString()»«IF instanceVariable!=null»«instanceVari
able.instanceVariableName».«method»«ELSE»_«methodName»ECPrivateKey.«method»«ENDIF»(«F
OREACH parameters AS ecKP SEPARATOR ','»«ecKP.name»«ENDFOREACH»);//this method dont n
eed any field.
«ELSEIF method.toString()=="setFieldF2M".toString()»«IF instanceVariable!=null»«insta
nceVariable.instanceVariableName».«method»«ELSE»_«methodName»ECPrivateKey.«method»«EN
DIF»(«FOREACH parameters AS ecKP SEPARATOR','»«ecKP.name»«ENDFOREACH»);//this method
 dont need any field.«ELSEIF method.toString()=="setFieldFP".toString()»«IF instanceVa
riable!=null»«instanceVariable.instanceVariableName».«method»«ELSE»_«methodName»ECPri
vateKey.«method»«ENDIF»(«FOREACHparameters AS ecKP SEPARATOR ','»«ecKP.name»«ENDFOREA
CH»);//this method dont need any field.«ELSEIF method.toString()=="setG".toString()»«
IF instanceVariable!=null»«instanceVariable.instanceVariableName».«method»«ELSE»_«met
```

```
hodName»ECPrivateKey.«method»«ENDIF»(«FOREACH parameters AS ecKP SEPARATOR ','»«ecKP.
name»«ENDFOREACH»);//this method dont need any field.«ELSEIF method.toString()=="setK
".toString()»«IF instanceVariable!=null»«instanceVariable.instanceVariableName».«meth
od»«ELSE»_«methodName»ECPrivateKey.«method»«ENDIF»(«FOREACH parameters AS ecKP SEPARA
TOR ','»«ecKP.name»«ENDFOREACH»);//this method dont need any field.«ELSEIF method.toS
tring()=="setR".toString()»«IF instanceVariable!=null»«instanceVariable.instanceVaria
bleName».«method»«ELSE»_«methodName»ECPrivateKey.«method»«ENDIF»(«FOREACH parameters
AS ecKP SEPARATOR ','»«ecKP.name»«ENDFOREACH»);//this method dont need any field.«ELS
EIF method.toString()=="clearKey".toString()»«IFinstanceVariable!=null»«instanceVaria
ble.instanceVariableName».«method»«ELSE»_«methodName»ECPrivateKey.«method»«ENDIF»();/
/this method dont need any parameters or fields.«ELSEIF method.toString()=="getSize".
toString()»«IF instanceVariable!=null»«instanceVariable.instanceVariableName».«method
»«ELSE»_«methodName»ECPrivateKey.«method»«ENDIF»();//this method need short field.«EL
SEIF method.toString()=="getType".toString()»«IF instanceVariable!=null»«instanceVari
able.instanceVariableName».«method»«ELSE»_«methodName»ECPrivateKey.«method»«ENDIF»();
//this method need byte field.«ELSEIF method.toString()=="isInitialized".toString()»«
IF instanceVariable!=null»«instanceVariable.instanceVariableName».«method»«ELSE»_«met
hodName»ECPrivateKey.«method»«ENDIF»();//this method need boolean field.«ENDIF»
«ENDDEFINE»
«DEFINE ecPublicKey(String methodName) FOR javaCardModel::ECPublicKey»
«IF fields!=null»«IF fields.dataType.toString()!="none".toString()»«fields.dataType»
«ELSE»«ENDIF»«IF fields.isArray»«fields.name»[]= «ELSE»«fields.name» = «ENDIF»«ELSE»«ENDIF
»«IF method.toString()=="getW".toString()»«IF instanceVariable!=null»«instanceVariabl
e.instanceVariableName».«method»«ELSE»_«methodName»ECPublicKey.«method»«ENDIF»(«FOREA
CH parameters AS ecKP SEPARATOR ','»«ecKP.name»«ENDFOREACH»);//This method needs para
meters and short field.
«ELSEIFmethod.toString()=="setW".toString()»«IF instanceVariable!=null»«instanceVaria
ble.instanceVariableName».«method»«ELSE»_«methodName»ECPublicKey.«method»«ENDIF»(«FOR
EACH parameters AS ecKP SEPARATOR ','»«ecKP.name»«ENDFOREACH»);//this methoddont need
 any field.
«ELSEIF method.toString()=="getA".toString()»«IF instanceVariable!=null»«instanceVari
able.instanceVariableName».«method»«ELSE»_«methodName»ECPublicKey.«method»«ENDIF»(«FO
REACH parameters AS ecKP SEPARATOR','»«ecKP.name»«ENDFOREACH»);//This method needs pa
rameters and short field.«ELSEIF method.toString()=="getB".toString()»«IF instanceVar
iable!=null»«instanceVariable.instanceVariableName».«method»«ELSE»_«methodName»ECPubl
icKey.«method»«ENDIF»(«FOREACH parameters AS ecKP SEPARATOR ','»«ecKP.name»«ENDFOREAC
H»);//This method needs parameters and short field.
«ELSEIF method.toString()=="getField".toString()»«IF instanceVariable!=null»«instance
Variable.instanceVariableName».«method»«ELSE»_«methodName»ECPublicKey.«method»«ENDIF»
(«FOREACH parameters AS ecKP SEPARATOR ','»«ecKP.name»«ENDFOREACH»);//This method nee
ds parameters and short field.«ELSEIF method.toString()=="getG".toString()»«IFinstanc
eVariable!=null»«instanceVariable.instanceVariableName».«method»«ELSE»_«methodName»EC
PublicKey.«method»«ENDIF»(«FOREACH parameters AS ecKP SEPARATOR ','»«ecKP.name»«ENDFO
REACH»);//This method needs parameters and short field.«ELSEIFmethod.toString()=="get
K".toString()»«IF instanceVariable!=null»«instanceVariable.instanceVariableName».«met
hod»«ELSE»_«methodName»ECPublicKey.«method»«ENDIF»();//this method dont need any para
meter, but need short field.«ELSEIF method.toString()=="getR".toString()»«IF instance
Variable!=null»«instanceVariable.instanceVariableName».«method»«ELSE»_«methodName»ECP
ublicKey.«method»«ENDIF»(«FOREACH parameters AS ecKP SEPARATOR ','»«ecKP.name»«ENDFOR
EACH»);//This method needs parameters and shortfield.
«ELSEIF method.toString()=="setA".toString()»«IF instanceVariable!=null»«instanceVari
able.instanceVariableName».«method»«ELSE»_«methodName»ECPublicKey.«method»«ENDIF»(«FO
REACH parameters AS ecKP SEPARATOR','»«ecKP.name»«ENDFOREACH»);//this method dont nee
d any field.
«ELSEIF method.toString()=="setB".toString()»«IF instanceVariable!=null»«instanceVari
able.instanceVariableName».«method»«ELSE»_«methodName»ECPublicKey.«method»«ENDIF»(«FO
REACHparameters AS ecKP SEPARATOR ','»«ecKP.name»«ENDFOREACH»);//this method dont nee
d any field.
«ELSEIF method.toString()=="setFieldF2M".toString()»«IF instanceVariable!=null»«insta
nceVariable.instanceVariableName».«method»«ELSE»_«methodName»ECPublicKey.«method»«END
IF»(«FOREACH parameters AS ecKP SEPARATOR ','»«ecKP.name»«ENDFOREACH»);//this method
dont need any field.«ELSEIF method.toString()=="setFieldFP".toString()»«IF instanceVa
riable!=null»«instanceVariable.instanceVariableName».«method»«ELSE»_«methodName»ECPub
licKey.«method»«ENDIF»(«FOREACH parameters AS ecKP SEPARATOR ','»«ecKP.name»«ENDFOREA
CH»);//this method dont need any field.«ELSEIF method.toString()=="setG".toString()»«
IFinstanceVariable!=null»«instanceVariable.instanceVariableName».«method»«ELSE»_«meth
odName»ECPublicKey.«method»«ENDIF»(«FOREACH parameters AS ecKP SEPARATOR ','»«ecKP.na
me»«ENDFOREACH»);//this method dont need any field.«ELSEIFmethod.toString()=="setK".t
oString()»«IF instanceVariable!=null»«instanceVariable.instanceVariableName».«method»
«ELSE»_«methodName»ECPublicKey.«method»«ENDIF»(«FOREACH parameters AS ecKP SEPARATOR
```

```
','»«ecKP.name»«ENDFOREACH»);//this method dontneed any field.«ELSEIF method.toString
()=="setR".toString()»«IF instanceVariable!=null»«instanceVariable.instanceVariableNa
me».«method»«ELSE»_«methodName»ECPublicKey.«method»«ENDIF»(«FOREACH parameters AS ecK
P SEPARATOR','»«ecKP.name»«ENDFOREACH»);//this method dont need any field.«ELSEIF met
hod.toString()=="clearKey".toString()»«IF instanceVariable!=null»«instanceVariable.in
stanceVariableName».«method»«ELSE»_«methodName»ECPublicKey.«method»«ENDIF»();//this m
ethoddont need any parameters or fields.«ELSEIF method.toString()=="getSize".toString
()»«IF instanceVariable!=null»«instanceVariable.instanceVariableName».«method»«ELSE»_
«methodName»ECPublicKey.«method»«ENDIF»();//this method need short field.«ELSEIFmetho
d.toString()=="getType".toString()»«IF instanceVariable!=null»«instanceVariable.insta
nceVariableName».«method»«ELSE»_«methodName»ECPublicKey.«method»«ENDIF»();//this meth
od need byte field.«ELSEIF method.toString()=="isInitialized".toString()»«IFinstanceV
ariable!=null»«instanceVariable.instanceVariableName».«method»«ELSE»_«methodName»ECPu
blicKey.«method»«ENDIF»();//this method need boolean field.«ENDIF»
«ENDDEFINE»
«DEFINE key(String methodName) FOR javaCardModel::Key»
«IF fields!=null»«IF fields.dataType.toString()!="none".toString()»«fields.dataType»
«ELSE»«ENDIF»«IF fields.isArray»«fields.name»[]= «ELSE»«fields.name» = «ENDIF»«ELSE»«ENDIF
»«IF method.toString()=="clearKey".toString()»«IF instanceVariable!=null»«instanceVar
iable.instanceVariableName».«method»«ELSE»_«methodName»Key.«method»«ENDIF»();//this m
ethod dont need any parameters or fields.«ELSEIF method.toString()=="getSize".toStrin
g()»«IF instanceVariable!=null»«instanceVariable.instanceVariableName».«method»«ELSE»
_«methodName»Key.«method»«ENDIF»();//this method need short field.«ELSEIF method.toSt
ring()=="getType".toString()»«IF instanceVariable!=null»«instanceVariable.instanceVar
iableName».«method»«ELSE»_«methodName»Key.«method»«ENDIF»();//this method need byte f
ield.«ELSEIF method.toString()=="isInitialized".toString()»«IF instanceVariable!=null
»«instanceVariable.instanceVariableName».«method»«ELSE»_«methodName»Key.«method»«ENDI
F»();//this method need boolean field.«ENDIF»
«ENDDEFINE»
«DEFINE privateKey(String methodName) FOR javaCardModel::PrivateKey»
«IF fields!=null»«IF fields.dataType.toString()!="none".toString()»«fields.dataType»
«ELSE»«ENDIF»«IF fields.isArray»«fields.name»[]= «ELSE»«fields.name» = «ENDIF»«ELSE»«ENDIF
»«IF method.toString()=="clearKey".toString()»«IF instanceVariable!=null»«instanceVar
iable.instanceVariableName».«method»«ELSE»_«methodName»PrivateKey.«method»«ENDIF»();/
/this method dont need any parameters or fields.«ELSEIF method.toString()=="getSize".
toString()»«IF instanceVariable!=null»«instanceVariable.instanceVariableName».«method
»«ELSE»_«methodName»PrivateKey.«method»«ENDIF»();//this method need short field.«ELSE
IF method.toString()=="getType".toString()»«IF instanceVariable!=null»«instanceVariab
le.instanceVariableName».«method»«ELSE»_«methodName»PrivateKey.«method»«ENDIF»();//th
is method need byte field.«ELSEIF method.toString()=="isInitialized".toString()»«IF i
nstanceVariable!=null»«instanceVariable.instanceVariableName».«method»«ELSE»_«methodN
ame»PrivateKey.«method»«ENDIF»();//this method need boolean field.«ENDIF»
«ENDDEFINE»
«DEFINE publicKey(String methodName) FOR javaCardModel::PublicKey»
«IF fields!=null»«IF fields.dataType.toString()!="none".toString()»«fields.dataType»
«ELSE»«ENDIF»«IF fields.isArray»«fields.name»[]= «ELSE»«fields.name» = «ENDIF»«ELSE»«ENDIF
»«IF method.toString()=="clearKey".toString()»«IF instanceVariable!=null»«instanceVar
iable.instanceVariableName».«method»«ELSE»_«methodName»PublicKey.«method»«ENDIF»();//
this method dont need any parameters or fields.«ELSEIF method.toString()=="getSize".t
oString()»«IF instanceVariable!=null»«instanceVariable.instanceVariableName».«method»
«ELSE»_«methodName»PublicKey.«method»«ENDIF»();//this method need short field.«ELSEIF
 method.toString()=="getType".toString()»«IF instanceVariable!=null»«instanceVariable
.instanceVariableName».«method»«ELSE»_«methodName»PublicKey.«method»«ENDIF»();//this
method need byte field.«ELSEIF method.toString()=="isInitialized".toString()»«IF inst
anceVariable!=null»«instanceVariable.instanceVariableName».«method»«ELSE»_«methodName
»PublicKey.«method»«ENDIF»();//this method need boolean field.«ENDIF»
«ENDDEFINE»
«DEFINE rsaPrivateKey(String methodName) FOR javaCardModel::RSAPrivateKey»
«IF fields!=null»«IF fields.dataType.toString()!="none".toString()»«fields.dataType»
«ELSE»«ENDIF»«IF fields.isArray»«fields.name»[]= «ELSE»«fields.name» = «ENDIF»«ELSE»«ENDIF
»«IF method.toString()=="getExponent".toString()»«IF instanceVariable!=null»«instance
Variable.instanceVariableName».«method»«ELSE»_«methodName»RSAPrivateKey.«method»«ENDI
F»(«FOREACH parameters AS par SEPARATOR ','»«par.name»«ENDFOREACH»);//This method need
s parameters and short field.«ELSEIF method.toString()=="getModulus(".toString()»«IF
instanceVariable!=null»«instanceVariable.instanceVariableName».«method»«ELSE»_«method
Name»RSAPrivateKey.«method»«ENDIF»(«FOREACH parameters AS par SEPARATOR ','»«par.name»
«ENDFOREACH»);//This method needs parametersand short field.«ELSEIF method.toString()
=="setExponent".toString()»«IF instanceVariable!=null»«instanceVariable.instanceVaria
bleName».«method»«ELSE»_«methodName»RSAPrivateKey.«method»«ENDIF»(«FOREACH parameters
 AS par SEPARATOR','»«par.name»«ENDFOREACH»);//this method dont need any field.«ELSEIF
```

```
 method.toString()=="setModulus".toString()»«IF instanceVariable!=null»«instanceVaria
ble.instanceVariableName».«method»«ELSE»_«methodName»RSAPrivateKey.«method»«ENDIF»(«F
OREACHparameters AS par SEPARATOR ','»«par.name»«ENDFOREACH»);//this method dont need
any field.«ELSEIF method.toString()=="clearKey".toString()»«IF instanceVariable!=null
»«instanceVariable.instanceVariableName».«method»«ELSE»_«methodName»RSAPrivateKey.«me
thod»«ENDIF»();//this method dont need any parameters or fields.«ELSEIF method.toStri
ng()=="getSize".toString()»«IF instanceVariable!=null»«instanceVariable.instanceVaria
bleName».«method»«ELSE»_«methodName»RSAPrivateKey.«method»«ENDIF»();//this methodneed
 short field.«ELSEIF method.toString()=="getType".toString()»«IF instanceVariable!=nu
ll»«instanceVariable.instanceVariableName».«method»«ELSE»_«methodName»RSAPrivateKey.«
method»«ENDIF»();//this method need byte field.«ELSEIF method.toString()=="isInitiali
zed".toString()»«IF instanceVariable!=null»«instanceVariable.instanceVariableName».«m
ethod»«ELSE»_«methodName»RSAPrivateKey.«method»«ENDIF»();//this method need boolean f
ield.«ENDIF»
«ENDDEFINE»
«DEFINE rsaPrivateCrtKey(String methodName) FOR javaCardModel::RSAPrivateCrtKey»
«IF fields!=null»«IF fields.dataType.toString()!="none".toString()»«fields.dataType»
«ELSE»«ENDIF»«IF fields.isArray»«fields.name»[]= «ELSE»«fields.name» = «ENDIF»«ELSE»«ENDIF
»«IF method.toString()=="getDP1".toString()»«IF instanceVariable!=null»«instanceVaria
ble.instanceVariableName».«method»«ELSE»_«methodName»RSAPrivateCrtKey.«method»«ENDIF»
(«FOREACH parameters AS par SEPARATOR ','»«par.name»«ENDFOREACH»);//This method needs
parameters and short field.«ELSEIFmethod.toString()=="getDQ1".toString()»«IF instance
Variable!=null»«instanceVariable.instanceVariableName».«method»«ELSE»_«methodName»RSA
PrivateCrtKey.«method»«ENDIF»(«FOREACH parameters AS par SEPARATOR ','»«par.name»«ENDF
OREACH»);//Thismethod needs parameters and short field.«ELSEIF method.toString()=="ge
tP".toString()»«IF instanceVariable!=null»«instanceVariable.instanceVariableName».«me
thod»«ELSE»_«methodName»RSAPrivateCrtKey.«method»«ENDIF»(«FOREACH parameters AS par S
EPARATOR','»«par.name»«ENDFOREACH»);//This method needs parameters and short field.«EL
SEIF method.toString()=="getPQ".toString()»«IF instanceVariable!=null»«instanceVariab
le.instanceVariableName».«method»«ELSE»_«methodName»RSAPrivateCrtKey.«method»«ENDIF»(
«FOREACH parameters AS par SEPARATOR ','»«par.name»«ENDFOREACH»);//This method needs p
arameters and short field.
«ELSEIF method.toString()=="getQ".toString()»«IF instanceVariable!=null»«instanceVari
able.instanceVariableName».«method»«ELSE»_«methodName»RSAPrivateCrtKey.«method»«ENDIF
»(«FOREACH parameters AS par SEPARATOR ','»«par.name»«ENDFOREACH»);//This method needs
 parameters and short field.«ELSEIF method.toString()=="setDP1".toString()»«IF instan
ceVariable!=null»«instanceVariable.instanceVariableName».«method»«ELSE»_«methodName»R
SAPrivateCrtKey.«method»«ENDIF»(«FOREACH parameters AS par SEPARATOR ','»«par.name»«EN
DFOREACH»);//this method dont need any field.«ELSEIFmethod.toString()=="setDQ1".toStr
ing()»«IF instanceVariable!=null»«instanceVariable.instanceVariableName».«method»«ELS
E»_«methodName»RSAPrivateCrtKey.«method»«ENDIF»(«FOREACH parameters AS par SEPARATOR
','»«par.name»«ENDFOREACH»);//this methoddont need any field.«ELSEIF method.toString()
=="setP".toString()»«IF instanceVariable!=null»«instanceVariable.instanceVariableName
».«method»«ELSE»_«methodName»RSAPrivateCrtKey.«method»«ENDIF»(«FOREACH parameters AS
par SEPARATOR','»«par.name»«ENDFOREACH»);//this method dont need any field.
«ELSEIF method.toString()=="setPQ".toString()»«IF instanceVariable!=null»«instanceVar
iable.instanceVariableName».«method»«ELSE»_«methodName»RSAPrivateCrtKey.«method»«ENDI
F»(«FOREACHparameters AS par SEPARATOR ','»«par.name»«ENDFOREACH»);//this method dont
need any field.
«ELSEIF method.toString()=="setQ".toString()»«IF instanceVariable!=null»«instanceVari
able.instanceVariableName».«method»«ELSE»_«methodName»RSAPrivateCrtKey.«method»«ENDIF
»(«FOREACH parameters AS par SEPARATOR ','»«par.name»«ENDFOREACH»);//this method dont
need any field.
«ELSEIF method.toString()=="clearKey".toString()»«IF instanceVariable!=null»«instance
Variable.instanceVariableName».«method»«ELSE»_«methodName»RSAPrivateCrtKey.«method»«E
NDIF»();//this method dont need any parameters or fields.«ELSEIF method.toString()=="
getSize".toString()»«IF instanceVariable!=null»«instanceVariable.instanceVariableName
».«method»«ELSE»_«methodName»RSAPrivateCrtKey.«method»«ENDIF»();//this method need sh
ort field.«ELSEIF method.toString()=="getType".toString()»«IF instanceVariable!=null»
«instanceVariable.instanceVariableName».«method»«ELSE»_«methodName»RSAPrivateCrtKey.«
method»«ENDIF»();//this method need byte field.«ELSEIF method.toString()=="isInitiali
zed".toString()»«IF instanceVariable!=null»«instanceVariable.instanceVariableName».«m
ethod»«ELSE»_«methodName»RSAPrivateCrtKey.«method»«ENDIF»();//this method needboolean
 field.«ENDIF»
«ENDDEFINE»
«DEFINE rsaPublicKey(String methodName) FOR javaCardModel::RSAPublicKey»
«IF fields!=null»«IF fields.dataType.toString()!="none".toString()»«fields.dataType»
«ELSE»«ENDIF»«IF fields.isArray»«fields.name»[]= «ELSE»«fields.name» = «ENDIF»«ELSE»«ENDIF
»«IF method.toString()=="getExponent".toString()»«IF instanceVariable!=null»«instance
```

```
Variable.instanceVariableName».«method»«ELSE»_«methodName»RSAPublicKey.«method»«ENDIF
»(«FOREACH parameters AS par SEPARATOR ','»«par.name»«ENDFOREACH»);//This method needs
 parameters and short field.«ELSEIF method.toString()=="getModulus(".toString()»«IF i
nstanceVariable!=null»«instanceVariable.instanceVariableName».«method»«ELSE»_«methodN
ame»RSAPublicKey.«method»«ENDIF»(«FOREACH parameters AS par SEPARATOR ','»«par.name»«E
NDFOREACH»);//This method needs parametersand short field.«ELSEIF method.toString()==
"setExponent".toString()»«IF instanceVariable!=null»«instanceVariable.instanceVariabl
eName».«method»«ELSE»_«methodName»RSAPublicKey.«method»«ENDIF»(«FOREACH parameters AS
 par SEPARATOR',»«par.name»«ENDFOREACH»);//this method dont need any field.«ELSEIF me
thod.toString()=="setModulus".toString()»«IF instanceVariable!=null»«instanceVariable
.instanceVariableName».«method»«ELSE»_«methodName»RSAPublicKey.«method»«ENDIF»(«FOREA
CHparameters AS par SEPARATOR ','»«par.name»«ENDFOREACH»);//this method dont need any
field.«ELSEIF method.toString()=="clearKey".toString()»«IF instanceVariable!=null»«in
stanceVariable.instanceVariableName».«method»«ELSE»_«methodName»RSAPublicKey.«method»
«ENDIF»();//this method dont need any parameters or fields.«ELSEIF method.toString()=
="getSize".toString()»«IF instanceVariable!=null»«instanceVariable.instanceVariableNa
me».«method»«ELSE»_«methodName»RSAPublicKey.«method»«ENDIF»();//this methodneed short
 field.«ELSEIF method.toString()=="getType".toString()»«IF instanceVariable!=null»«in
stanceVariable.instanceVariableName».«method»«ELSE»_«methodName»RSAPublicKey.«method»
«ENDIF»();//this method need byte field.«ELSEIF method.toString()=="isInitialized".to
String()»«IF instanceVariable!=null»«instanceVariable.instanceVariableName».«method»«
ELSE»_«methodName»RSAPublicKey.«method»«ENDIF»();//this method need boolean field.«EN
DIF»
«ENDDEFINE»
«DEFINE secretKey(String methodName) FOR javaCardModel::SecretKey»
«IF fields!=null»«IF fields.dataType.toString()!="none".toString()»«fields.dataType»
«ELSE»«ENDIF»«IF fields.isArray»«fields.name»[]= «ELSE»«fields.name» = «ENDIF»«ELSE»«ENDIF
»«IF method.toString()=="clearKey".toString()»«IF instanceVariable!=null»«instanceVar
iable.instanceVariableName».«method»«ELSE»_«methodName»SecretKey.«method»«ENDIF»();//
this method dont need any parameters or fields.«ELSEIF method.toString()=="getSize".t
oString()»«IF instanceVariable!=null»«instanceVariable.instanceVariableName».«method»
«ELSE»_«methodName»SecretKey.«method»«ENDIF»();//this method need short field.«ELSEIF
 method.toString()=="getType".toString()»«IF instanceVariable!=null»«instanceVariable
.instanceVariableName».«method»«ELSE»_«methodName»SecretKey.«method»«ENDIF»();//this
method need byte field.«ELSEIF method.toString()=="isInitialized".toString()»«IF inst
anceVariable!=null»«instanceVariable.instanceVariableName».«method»«ELSE»_«methodName
»SecretKey.«method»«ENDIF»();//this method need boolean field.«ENDIF»
«ENDDEFINE»
«DEFINE aesKey(String methodName) FOR javaCardModel::AESKey»
«IF fields!=null»«IF fields.dataType.toString()!="none".toString()»«fields.dataType»
«ELSE»«ENDIF»«IF fields.isArray»«fields.name»[]= «ELSE»«fields.name» = «ENDIF»«ELSE»«ENDIF
»«IF method.toString()=="clearKey".toString()»«IF instanceVariable!=null»«instanceVar
iable.instanceVariableName».«method»«ELSE»_«methodName»AESKey.«method»«ENDIF»();//thi
s method dont need any parameters or fields.«ELSEIF method.toString()=="getSize".toSt
ring()»«IF instanceVariable!=null»«instanceVariable.instanceVariableName».«method»«EL
SE»_«methodName»AESKey.«method»«ENDIF»();//this method need short field.«ELSEIF metho
d.toString()=="getType".toString()»«IF instanceVariable!=null»«instanceVariable.insta
nceVariableName».«method»«ELSE»_«methodName»AESKey.«method»«ENDIF»();//this method ne
ed byte field.«ELSEIF method.toString()=="isInitialized".toString()»«IF instanceVaria
ble!=null»«instanceVariable.instanceVariableName».«method»«ELSE»_«methodName»AESKey.«
method»«ENDIF»();//this method need boolean field.«ELSEIF method.toString()=="getKey"
.toString()»«IF instanceVariable!=null»«instanceVariable.instanceVariableName».«metho
d»«ELSE»_«methodName»AESKey.«method»«ENDIF»(«FOREACH parameters AS parSEPARATOR ','»«
par.name»«ENDFOREACH»);//This method needs parameters and byte field.«ELSEIF method.to
String()=="setKey".toString()»«IF instanceVariable!=null»«instanceVariableName».«meth
od»«ELSE»_«methodName»AESKey.«method»«ENDIF»(«FOREACH parameters AS
par SEPARATOR ','»«par.name»«ENDFOREACH»);//This method needs parameters.«ENDIF»
«ENDDEFINE»
«DEFINE desKey(String methodName) FOR javaCardModel::DESKey»
«IF fields!=null»«IF fields.dataType.toString()!="none".toString()»«fields.dataType»
«ELSE»«ENDIF»«IF fields.isArray»«fields.name»[]= «ELSE»«fields.name» = «ENDIF»«ELSE»«ENDIF
»«IF method.toString()=="clearKey".toString()»«IF instanceVariable!=null»«instanceVar
iable.instanceVariableName».«method»«ELSE»_«methodName»DESKey.«method»«ENDIF»();//thi
s method dont need any parameters or fields.«ELSEIF method.toString()=="getSize".toSt
ring()»«IF instanceVariable!=null»«instanceVariable.instanceVariableName».«method»«EL
SE»_«methodName»DESKey.«method»«ENDIF»();//this method need short field.«ELSEIF metho
d.toString()=="getType".toString()»«IF instanceVariable!=null»«instanceVariable.insta
nceVariableName».«method»«ELSE»_«methodName»DESKey.«method»«ENDIF»();//this method ne
ed byte field.«ELSEIF method.toString()=="isInitialized".toString()»«IF instanceVaria
ble!=null»«instanceVariable.instanceVariableName».«method»«ELSE»_«methodName»DESKey.«
```

```
method»«ENDIF»();//this method need boolean field.«ELSEIF method.toString()=="getKey"
.toString()»«IF instanceVariable!=null»«instanceVariable.instanceVariableName».«metho
d»«ELSE»_«methodName»DESKey.«method»«ENDIF»(«FOREACH parameters AS parSEPARATOR ','»«
par.name»«ENDFOREACH»);//This method needs parameters and byte field.«ELSEIF method.to
String()=="setKey".toString()»«IF instanceVariable!=null»«instanceVariable.instanceVa
riableName».«method»«ELSE»_«methodName»DESKey.«method»«ENDIF»(«FOREACH parameters AS
par SEPARATOR ','»«par.name»«ENDFOREACH»);//This method needs parameters.«ENDIF»
«ENDDEFINE»
«DEFINE hmacKey(String methodName) FOR javaCardModel::HMACKey»
«IF fields!=null»«IF fields.dataType.toString()!="none".toString()»«fields.dataType»
«ELSE»«ENDIF»«IF fields.isArray»«fields.name»[]= «ELSE»«fields.name» = «ENDIF»«ELSE»«ENDIF
»«IF method.toString()=="clearKey".toString()»«IF instanceVariable!=null»«instanceVar
iable.instanceVariableName».«method»«ELSE»_«methodName»HMACKey.«method»«ENDIF»();//th
is method dont need any parameters or fields.«ELSEIF method.toString()=="getSize".toS
tring()»«IF instanceVariable!=null»«instanceVariable.instanceVariableName».«method»«E
LSE»_«methodName»HMACKey.«method»«ENDIF»();//this method need short field.«ELSEIF met
hod.toString()=="getType".toString()»«IF instanceVariable!=null»«instanceVariable.ins
tanceVariableName».«method»«ELSE»_«methodName»HMACKey.«method»«ENDIF»();//this method
 need byte field.«ELSEIF method.toString()=="isInitialized".toString()»«IF instanceVa
riable!=null»«instanceVariable.instanceVariableName».«method»«ELSE»_«methodName»HMACK
ey.«method»«ENDIF»();//this method need boolean field.«ELSEIF method.toString()=="get
Key".toString()»«IF instanceVariable!=null»«instanceVariable.instanceVariableName».«m
ethod»«ELSE»_«methodName»HMACKey.«method»«ENDIF»(«FOREACH parameters AS parSEPARATOR
','»«par.name»«ENDFOREACH»);//This method needs parameters and byte field.«ELSEIF meth
od.toString()=="setKey".toString()»«IF instanceVariable!=null»«instanceVariable.insta
nceVariableName».«method»«ELSE»_«methodName»HMACKey.«method»«ENDIF»(«FOREACH paramete
rs AS par SEPARATOR ','»«par.name»«ENDFOREACH»);//This method needs parameters.«ENDIF»
«ENDDEFINE»
«DEFINE koreanSEEDKey(String methodName) FOR javaCardModel::KoreanSEEDKey»
«IF fields!=null»«IF fields.dataType.toString()!="none".toString()»«fields.dataType»
«ELSE»«ENDIF»«IF fields.isArray»«fields.name»[]= «ELSE»«fields.name» = «ENDIF»«ELSE»«ENDIF
»«IF method.toString()=="clearKey".toString()»«IF instanceVariable!=null»«instanceVar
iable.instanceVariableName».«method»«ELSE»_«methodName»KoreanSEEDKey.«method»«ENDIF»(
);//this method dont need any parameters or fields.«ELSEIF method.toString()=="getSiz
e".toString()»«IF instanceVariable!=null»«instanceVariable.instanceVariableName».«met
hod»«ELSE»_«methodName»KoreanSEEDKey.«method»«ENDIF»();//this method need short field
.«ELSEIF method.toString()=="getType".toString()»«IF instanceVariable!=null»«instance
Variable.instanceVariableName».«method»«ELSE»_«methodName»KoreanSEEDKey.«method»«ENDI
F»();//this method need byte field.«ELSEIF method.toString()=="isInitialized".toStrin
g()»«IF instanceVariable!=null»«instanceVariable.instanceVariableName».«method»«ELSE»
_«methodName»KoreanSEEDKey.«method»«ENDIF»();//this method need boolean field.«ELSEIF
 method.toString()=="getKey".toString()»«IF instanceVariable!=null»«instanceVariable.
instanceVariableName».«method»«ELSE»_«methodName»KoreanSEEDKey.«method»«ENDIF»(«FOREA
CH parameters ASpar SEPARATOR ','»«par.name»«ENDFOREACH»);//This method needs paramete
rs and byte field.«ELSEIF method.toString()=="setKey".toString()»«IF instanceVariable
!=null»«instanceVariable.instanceVariableName».«method»«ELSE»_«methodName»KoreanSEEDK
ey.«method»«ENDIF»(«FOREACH parameters AS par SEPARATOR ','»«par.name»«ENDFOREACH»);//
This method needs parameters.«ENDIF»
«ENDDEFINE»
«DEFINE iso7816(String methodName) FOR javaCardModel::ISO7816»
    «IF fields!=null»«IF fields.dataType.toString()!="none".toString()»«fields.dataTy
pe»
«ELSE»«ENDIF»«IF fields.isArray»«fields.name»[]= «ELSE»«fields.name» = «ENDIF»«ELSE»«ENDIF
»
«IF ISOExceptionthrowIt»ISOException.throwIt(ISO7816.«iso7816Field»);//This method is
 mainly used in this way.«ELSEIF spiralFieldName!=null»(«IF fields.dataType.toString(
)!="none".toString()»«fields.dataType»«ELSE»«ENDIF») («spiralFieldName»[ISO7816.«iso7
816Field» && 0xFF]); // You can change thisusage«ELSE»ISO7816.«iso7816Field»; //this
usage not true, be carefull!!«ENDIF»
«ENDDEFINE»
«DEFINE signatureMessageRecovery(String
methodName) FOR javaCardModel::SignatureMessageRecovery»
«IF fields!=null»«IF fields.dataType.toString()!="none".toString()»«fields.dataType»
«ELSE»«ENDIF»«IF fields.isArray»«fields.name»[]= «ELSE»«fields.name» = «ENDIF»«ELSE»«ENDIF
»«IF method.toString()=="beginVerify".toString()»«IF instanceVariable!=null»«instance
Variable.instanceVariableName».«method»«ELSE»_«methodName»SignatureMessageRecovery.«m
ethod»«ENDIF»(«FOREACH parameters AS par SEPARATOR ','»«par.name»«ENDFOREACH»);//This
method needs parameters and short field.«ELSEIFmethod.toString()=="getAlgorithm".toSt
ring()»«IF instanceVariable!=null»«instanceVariable.instanceVariableName».«method»«EL
```

```
SE»_«methodName»SignatureMessageRecovery.«method»«ENDIF»();//this method need byte fi
eld.«ELSEIF method.toString()=="getLength".toString()»«IF instanceVariable!=null»«ins
tanceVariable.instanceVariableName».«method»«ELSE»_«methodName»SignatureMessageRecove
ry.«method»«ENDIF»();//this method need short field.«ELSEIF method.toString()=="init"
.toString()»«IF instanceVariable!=null»«instanceVariable.instanceVariableName».«metho
d»«ELSE»_«methodName»SignatureMessageRecovery.«method»«ENDIF»(«FOREACH parameters AS
par SEPARATOR ','»«par.name»«ENDFOREACH»);//this method needs parameters.«ELSEIF metho
d.toString()=="sign".toString()»«IF instanceVariable!=null»«instanceVariable.instance
VariableName».«method»«ELSE»_«methodName»SignatureMessageRecovery.«method»«ENDIF»(«FO
REACH parameters AS par SEPARATOR ','»«par.name»«ENDFOREACH»);//This method needs para
metersand short field.«ELSEIF method.toString()=="update".toString()»«IF instanceVari
able!=null»«instanceVariable.instanceVariableName».«method»«ELSE»_«methodName»Signatu
reMessageRecovery.«method»«ENDIF»(«FOREACH parameters AS par SEPARATOR','»«par.name»«E
NDFOREACH»);//This method needs parameters.«ELSEIF method.toString()=="verify".toStri
ng()»«IF instanceVariable!=null»«instanceVariable.instanceVariableName».«method»«ELSE
»_«methodName»SignatureMessageRecovery.«method»«ENDIF»(«FOREACH parameters AS par SEP
ARATOR ','»«par.name»«ENDFOREACH»);//This method needs parameters and boolean field.«E
NDIF»
«ENDDEFINE»
«DEFINE keyEncryotion(String methodName) FOR javaCardModel::KeyEncryption»
«IF fields!=null»«IF fields.dataType.toString()!="none".toString()»«fields.dataType»
«ELSE»«ENDIF»«IF fields.isArray»«fields.name»[]= «ELSE»«fields.name» = «ENDIF»«ELSE»«ENDIF
»
«IF method.toString()=="getKeyCipher".toString()»«IFinstanceVariable!=null»«instanceV
ariable.instanceVariableName».«method»«ELSE»_«methodName»KeyEncryption.«method»«ENDIF
»();//This method needs Cipher field.«ELSEIF method.toString()=="setKeyCipher".toStri
ng()»«IF instanceVariable!=null»«instanceVariable.instanceVariableName».«method»«ELSE
»_«methodName»KeyEncryption.«method»«ENDIF»(«FOREACH parameters AS par SEPARATOR ','»
«par.name»«ENDFOREACH»);//this method need parameters.«ENDIF»
«ENDDEFINE»
«DEFINE util(String methodName) FOR javaCardModel::Util»
«IF fields!=null»«IF fields.dataType.toString()!="none".toString()»«fields.dataType»
«ELSE»«ENDIF»«IF fields.isArray»«fields.name»[]= «ELSE»«fields.name» = «ENDIF»«ELSE»«ENDIF
»«IF method.toString()=="arrayCompare".toString()»Util.«method»(«FOREACHparameters AS
 par SEPARATOR ','»«par.name»«ENDFOREACH»);//This method needs parameters and byte fie
ld.«ELSEIF method.toString()=="arrayCopy".toString()»Util.«method»(«FOREACH parameter
s AS par SEPARATOR ','»«par.name»«ENDFOREACH»);//This methodneeds parameters and short
 field.«ELSEIF method.toString()=="arrayCopyNonAtomic".toString()»Util.«method»(«FORE
ACH parameters AS par SEPARATOR ','»«par.name»«ENDFOREACH»);//This method needs parame
ters and short field.«ELSEIF method.toString()=="arrayFillNonAtomic".toString()»Util.
«method»(«FOREACH parameters AS par SEPARATOR ','»«par.name»«ENDFOREACH»);//This metho
d needs parameters and short field.«ELSEIF method.toString()=="getShort".toString()»U
til.«method»(«FOREACH parameters AS parSEPARATOR ','»«par.name»«ENDFOREACH»);//This me
thod needs parameters and short field.«ELSEIF method.toString()=="makeShort".toString
()»Util.«method»(«FOREACH parameters AS par SEPARATOR ','»«par.name»«ENDFOREACH»);//Th
is method needs parametersand short field.«ELSEIF method.toString()=="setShort".toStr
ing()»Util.«method»(«FOREACH parameters AS par SEPARATOR ','»«par.name»«ENDFOREACH»);/
/This method needs parameters and short field.«ELSEIF method.toString()=="equals".toS
tring()»«IFinstanceVariable!=null»«instanceVariable.instanceVariableName».«method»«EL
SE»_«methodName»Util.«method»«ENDIF»(«FOREACH parameters AS par SEPARATOR ','»«par.nam
e»«ENDFOREACH»);//This method needs parameters and boolean field.«ENDIF»
«ENDDEFINE»
«DEFINE utilEx(String methodName) FOR javaCardModel::UtilException»
«IF exceptionUsageType.toString()
== "tryCatch".toString()»try{}catch(UtilException «fields.name»){} «ELSEIF exceptionUsa
geType.toString()
== "ThrowNew".toString()»throw new UtilException «fields.name»; «ELSEIF exceptionUsageT
ype.toString()
=="ifElse".toString()»if() {«IF fields!=null»«IF fields.dataType.toString()!="none".t
oString()»«fields.dataType»
«ELSE»«ENDIF»«IF fields.isArray»«fields.name»[]= «ELSE»«fields.name» = «ENDIF»«ELSE»«ENDIF
»«IF method.toString()=="UtilException".toString()»UtilException.throwIt(«IF utilExce
ptionField.toString()=="none".toString()»«FOREACH parameters AS par»«par.name»«ENDFORE
ACH»«ELSE» UtilException.«utilExceptionField»«ENDIF»);}else{}«ELSEIF method.toString(
)=="throwIt".toString()»UtilException.throwIt(«IFutilExceptionField.toString()=="none
".toString()»«FOREACH parameters AS par»«par.name»«ENDFOREACH»«ELSE» UtilException.«ut
ilExceptionField»«ENDIF»);}else{}«ELSEIF method.toString()=="getReason".toString()»«I
F instanceVariable!=null»«instanceVariable.instanceVariableName».«method»«ELSE»_«meth
```

```
odName»UtilException.«method»«ENDIF»(«IF utilExceptionField.toString()=="none".toStri
ng()»«FOREACH parameters AS par»«par.name»«ENDFOREACH»«ELSE» UtilException.«utilExcept
ionField»«ENDIF»);}else{}«ELSEIF method.toString()=="setReason".toString()»«IF instan
ceVariable!=null»«instanceVariable.instanceVariableName».«method»«ELSE»_«methodName»U
tilException.«method»«ENDIF»(«IF utilExceptionField.toString()=="none".toString()»«FO
REACH parameters ASpar»«par.name»«ENDFOREACH»«ELSE» UtilException.«utilExceptionField»
«ENDIF»);}else{}«ENDIF»«ELSE»«IF fields!=null»«IF fields.dataType.toString()!="none".
toString()»«fields.dataType»
«ELSE»«ENDIF»«IF fields.isArray»«fields.name»[]= «ELSE»«fields.name» =«ENDIF»«ELSE»«ENDIF»
«IF method.toString()=="UtilException".toString()»UtilException.throwIt(«IF utilExcep
tionField.toString()=="none".toString()»«FOREACH parameters AS par»«par.name»«ENDFOREA
CH»«ELSE» UtilException.«utilExceptionField»«ENDIF»);«ELSEIF method.toString()=="thro
wIt".toString()»UtilException.throwIt(«IF utilExceptionField.toString()=="none".toStr
ing()»«FOREACH parameters AS par»«par.name»«ENDFOREACH»«ELSE» UtilException.«utilExcep
tionField»«ENDIF»);«ELSEIFmethod.toString()=="getReason".toString()»«IF instanceVaria
ble!=null»«instanceVariable.instanceVariableName».«method»«ELSE»_«methodName»UtilExce
ption.«method»«ENDIF»(«IF utilExceptionField.toString()=="none".toString()»«FOREACH p
arameters ASpar»«par.name»«ENDFOREACH»«ELSE» UtilException.«utilExceptionField»«ENDIF»
);«ELSEIF method.toString()=="setReason".toString()»«IF instanceVariable!=null»«insta
nceVariable.instanceVariableName».«method»«ELSE»_«methodName»UtilException.«method»«E
NDIF»(«IF utilExceptionField.toString()=="none".toString()»«FOREACH parameters AS par
»«par.name»«ENDFOREACH»«ELSE» UtilException.«utilExceptionField»«ENDIF»);«ENDIF»
«ENDIF»
«ENDDEFINE»
«DEFINE crtEx(String methodName) FOR javaCardModel::CardException»
«IF exceptionUsageType.toString()
== "tryCatch".toString()»try{}catch(CardException «fields.name»){}«ELSEIF exceptionUsag
eType.toString()
== "ThrowNew".toString()»throw new CardException «fields.name»;«ELSEIF exceptionUsageTy
pe.toString()
=="ifElse".toString()»if() {«IF fields!=null»«IF fields.dataType.toString()!="none".t
oString()»«fields.dataType»
«ELSE»«ENDIF»«IF fields.isArray»«fields.name»[]= «ELSE»«fields.name» = «ENDIF»«ELSE»«ENDIF
»«IF method.toString()=="CardException".toString()»CardException.throwIt(«FOREACH par
ameters AS par»«par.name»«ENDFOREACH»);}else{}«ELSEIF method.toString()=="throwIt".toS
tring()»CardException.throwIt(«FOREACH parameters AS par»«par.name»«ENDFOREACH»);}else
{}«ELSEIF method.toString()=="getReason".toString()»«IF instanceVariable!=null»«insta
nceVariable.instanceVariableName».«method»«ELSE»_«methodName»CardException.«method»«E
NDIF»();}else{}«ELSEIF method.toString()=="setReason".toString()»«IF instanceVariable
!=null»«instanceVariable.instanceVariableName».«method»«ELSE»_«methodName»CardExcepti
on.«method»«ENDIF»(«FOREACH parameters AS par»«par.name»«ENDFOREACH»);}else{}«ENDIF»«E
LSE»«IF fields!=null»«IF fields.dataType.toString()!="none".toString()»«fields.dataTy
pe»
«ELSE»«ENDIF»«IF fields.isArray»«fields.name»[]= «ELSE»«fields.name» = «ENDIF»«ELSE»«ENDIF
»     «IF method.toString()=="CardException".toString()»CardException.throwIt(«FOREAC
H parameters AS par»«par.name»«ENDFOREACH»);«ELSEIFmethod.toString()=="throwIt".toStri
ng()»CardException.throwIt(«FOREACH parameters AS par»«par.name»«ENDFOREACH»);«ELSEIF
method.toString()=="getReason".toString()»«IF instanceVariable!=null»«instanceVariabl
e.instanceVariableName».«method»«ELSE»_«methodName»CardException.«method»«ENDIF»();«E
LSEIF method.toString()=="setReason".toString()»«IF instanceVariable!=null»«instanceV
ariable.instanceVariableName».«method»«ELSE»_«methodName»CardException.«method»«ENDIF
»(«FOREACHparameters AS par»«par.name»«ENDFOREACH»);«ENDIF»«ENDIF»
«ENDDEFINE»
«DEFINE usrEx(String methodName) FOR javaCardModel::UserException»
«IF exceptionUsageType.toString()
== "tryCatch".toString()»try{}catch(UserException «fields.name»){}«ELSEIF exceptionUsag
eType.toString()
== "ThrowNew".toString()»throw new UserException «fields.name»;«ELSEIF exceptionUsageTy
pe.toString()
=="ifElse".toString()»if() {«IF fields!=null»«IF fields.dataType.toString()!="none".t
oString()»«fields.dataType»
«ELSE»«ENDIF»«IF fields.isArray»«fields.name»[]= «ELSE»«fields.name» = «ENDIF»«ELSE»«ENDIF
»«IF method.toString()=="UserException".toString()»UserException.throwIt(«FOREACH par
ameters AS par»«par.name»«ENDFOREACH»);}else{}«ELSEIF method.toString()=="throwIt".toS
tring()»UserException.throwIt(«FOREACH parameters AS par»«par.name»«ENDFOREACH»);}else
{}«ELSEIF method.toString()=="getReason".toString()»«IF instanceVariable!=null»«insta
nceVariable.instanceVariableName».«method»«ELSE»_«methodName»UserException.«method»«E
NDIF»();}else{}«ELSEIF method.toString()=="setReason".toString()»«IF instanceVariable
```

```
!=null»«instanceVariable.instanceVariableName».«method»«ELSE»_«methodName»UserExcepti
on.«method»«ENDIF»(«FOREACH parameters AS par»«par.name»«ENDFOREACH»);}else{}«ENDIF»«E
LSE»«IF fields!=null»«IF fields.dataType.toString()!="none".toString()»«fields.dataTy
pe»
«ELSE»«ENDIF»«IF fields.isArray»«fields.name»[]= «ELSE»«fields.name» = «ENDIF»«ELSE»«ENDIF
»«IF method.toString()=="UserException".toString()»UserException.throwIt(«FOREACH par
ameters AS par»«par.name»«ENDFOREACH»);«ELSEIFmethod.toString()=="throwIt".toString()»
UserException.throwIt(«FOREACH parameters AS par»«par.name»«ENDFOREACH»);«ELSEIF metho
d.toString()=="getReason".toString()»«IF instanceVariable!=null»«instanceVariable.ins
tanceVariableName».«method»«ELSE»_«methodName»UserException.«method»«ENDIF»();«ELSEIF
 method.toString()=="setReason".toString()»«IF instanceVariable!=null»«instanceVariab
le.instanceVariableName».«method»«ELSE»_«methodName»UserException.«method»«ENDIF»(«FO
REACHparameters AS par»«par.name»«ENDFOREACH»);«ENDIF»«ENDIF»
«ENDDEFINE»
«DEFINE cryEx(String methodName) FOR javaCardModel::CryptoException»
«IF exceptionUsageType.toString()
== "tryCatch".toString()»try{}catch(CryptoException «fields.name»){}«ELSEIF exceptionUs
ageType.toString()
== "ThrowNew".toString()»throw new CryptoException «fields.name»;«ELSEIF exceptionUsage
Type.toString()
=="ifElse".toString()»if() {«IF fields!=null»«IF fields.dataType.toString()!="none".t
oString()»«fields.dataType»
«ELSE»«ENDIF»«IF fields.isArray»«fields.name»[]= «ELSE»«fields.name» = «ENDIF»«ELSE»«ENDIF
»«IF method.toString()=="CryptoException".toString()»CryptoException.throwIt(«IF cryp
toExceptionField.toString()=="none".toString()»«FOREACH parameters AS par»«par.name»«E
NDFOREACH»«ELSE» CryptoException.«cryptoExceptionField»«ENDIF»);}else{}«ELSEIF method
.toString()=="throwIt".toString()»CryptoException.throwIt(«IF cryptoExceptionField.to
String()=="none".toString()»«FOREACH parameters AS par»«par.name»«ENDFOREACH»«ELSE» Cr
yptoException.«cryptoExceptionField»«ENDIF»);}else{}«ELSEIF method.toString()=="getRe
ason".toString()»«IF instanceVariable!=null»«instanceVariable.instanceVariableName».«
method»«ELSE»_«methodName»CryptoException.«method»«ENDIF»(«IF cryptoExceptionField.to
String()=="none".toString()»«FOREACH parameters AS par»«par.name»«ENDFOREACH»«ELSE» Cr
yptoException.«cryptoExceptionField»«ENDIF»);}else{}«ELSEIF method.toString()=="setRe
ason".toString()»«IF instanceVariable!=null»«instanceVariable.instanceVariableName».«
method»«ELSE»_«methodName»CryptoException.«method»«ENDIF»(«IF cryptoExceptionField.to
String()=="none".toString()»«FOREACH parameters AS par»«par.name»«ENDFOREACH»«ELSE» Cr
yptoException.«cryptoExceptionField»«ENDIF»);}else{}«ENDIF»«ELSE»«IF fields!=null»«IF
 fields.dataType.toString()!="none".toString()»«fields.dataType»
«ELSE»«ENDIF»«IFfields.isArray»«fields.name»[]= «ELSE»«fields.name» = «ENDIF»«ELSE»«ENDIF»
«IF method.toString()=="CryptoException".toString()»CryptoException.throwIt(«IF crypt
oExceptionField.toString()=="none".toString()»«FOREACH parameters ASpar»«par.name»«END
FOREACH»«ELSE» CryptoException.«cryptoExceptionField»«ENDIF»);«ELSEIF method.toString
()=="throwIt".toString()»CryptoException.throwIt(«IF cryptoExceptionField.toString()=
="none".toString()»«FOREACH parameters ASpar»«par.name»«ENDFOREACH»«ELSE» CryptoExcept
ion.«cryptoExceptionField»«ENDIF»);«ELSEIF method.toString()=="getReason".toString()»
«IF instanceVariable!=null»«instanceVariable.instanceVariableName».«method»«ELSE»_«me
thodName»CryptoException.«method»«ENDIF»(«IF cryptoExceptionField.toString()=="none".
toString()»«FOREACH parameters AS par»«par.name»«ENDFOREACH»«ELSE» CryptoException.«cr
yptoExceptionField»«ENDIF»);«ELSEIF method.toString()=="setReason".toString()»«IFinst
anceVariable!=null»«instanceVariable.instanceVariableName».«method»«ELSE»_«methodName
»CryptoException.«method»«ENDIF»(«IF cryptoExceptionField.toString()=="none".toString
()»«FOREACH parameters AS par»«par.name»«ENDFOREACH»«ELSE»CryptoException.«cryptoExcep
tionField»«ENDIF»);«ENDIF»«ENDIF»
«ENDDEFINE»
«DEFINE isoEx(String methodName) FOR javaCardModel::ISOException»
«IF exceptionUsageType.toString()
== "tryCatch".toString()»try{}catch(ISOException «fields.name»){}«ELSEIF exceptionUsage
Type.toString()
== "ThrowNew".toString()»throw new ISOException «fields.name»;«ELSEIF exceptionUsageTyp
e.toString()
=="ifElse".toString()»if() {«IF fields!=null»«IF fields.dataType.toString()!="none".t
oString()»«fields.dataType»
«ELSE»«ENDIF»«IF fields.isArray»«fields.name»[]= «ELSE»«fields.name» = «ENDIF»«ELSE»«ENDIF
»«IF method.toString()=="ISOException".toString()»ISOException.throwIt(«FOREACH param
eters AS par»«par.name»«ENDFOREACH»);}else{}«ELSEIF method.toString()=="throwIt".toStr
ing()»ISOException.throwIt(«FOREACH parameters AS par»«par.name»«ENDFOREACH»);}else{}«
ELSEIF method.toString()=="getReason".toString()»«IF instanceVariable!=null»«instance
Variable.instanceVariableName».«method»«ELSE»_«methodName»ISOException.«method»«ENDIF
```

```
»(«FOREACH parameters AS par»«par.name»«ENDFOREACH»);}else{}«ELSEIF method.toString()=
="setReason".toString()»«IFinstanceVariable!=null»«instanceVariable.instanceVariableN
ame».«method»«ELSE»_«methodName»ISOException.«method»«ENDIF»(«FOREACH parameters AS p
ar»«par.name»«ENDFOREACH»);}else{}«ENDIF»«ELSE»«IF fields!=null»«IF fields.dataType.to
String()!="none".toString()»«fields.dataType»
«ELSE»«ENDIF»«IF fields.isArray»«fields.name»[]= «ELSE»«fields.name» = «ENDIF»«ELSE»«ENDIF
»«IF method.toString()=="ISOException".toString()»ISOException.throwIt(«FOREACH param
eters ASpar»«par.name»«ENDFOREACH»);«ELSEIF method.toString()=="throwIt".toString()»IS
OException.throwIt(«FOREACH parameters AS par»«par.name»«ENDFOREACH»);«ELSEIF method.t
oString()=="getReason".toString()»«IF instanceVariable!=null»«instanceVariable.instan
ceVariableName».«method»«ELSE»_«methodName»ISOException.«method»«ENDIF»(«FOREACH para
meters AS par»«par.name»«ENDFOREACH»);«ELSEIF method.toString()=="setReason".toString(
)»«IF instanceVariable!=null»«instanceVariable.instanceVariableName».«method»«ELSE»_«
methodName»ISOException.«method»«ENDIF»(«FOREACH parameters AS par»«par.name»«ENDFOREA
CH»);«ENDIF»«ENDIF»
«ENDDEFINE»
«DEFINE srvEx(String methodName) FOR javaCardModel::ServiceException»
«IF exceptionUsageType.toString()
== "tryCatch".toString()»try{}catch(ServiceException «fields.name»){}«ELSEIF exceptionU
sageType.toString()
== "ThrowNew".toString()»throw new ServiceException «fields.name»;«ELSEIF exceptionUsag
eType.toString()
=="ifElse".toString()»if() {«IF fields!=null»«IF fields.dataType.toString()!="none".t
oString()»«fields.dataType»
«ELSE»«ENDIF»«IF fields.isArray»«fields.name»[]= «ELSE»«fields.name» = «ENDIF»«ELSE»«ENDIF
»«IF method.toString()=="ServiceException".toString()»ServiceException.throwIt(«IF se
rviceExceptionField.toString()=="none".toString()»«FOREACH parameters AS par»«par.name
»«ENDFOREACH»«ELSE» ServiceException.«serviceExceptionField»«ENDIF»);}else{}«ELSEIF m
ethod.toString()=="throwIt".toString()»ServiceException.throwIt(«IF serviceExceptionF
ield.toString()=="none".toString()»«FOREACH parameters AS par»«par.name»«ENDFOREACH»«E
LSE» ServiceException.«serviceExceptionField»«ENDIF»);}else{}«ELSEIF method.toString(
)=="getReason".toString()»«IF instanceVariable!=null»«instanceVariable.instanceVariab
leName».«method»«ELSE»_«methodName»ServiceException.«method»«ENDIF»(«IF serviceExcept
ionField.toString()=="none".toString()»«FOREACH parameters ASpar»«par.name»«ENDFOREACH
»«ELSE» ServiceException.«serviceExceptionField»«ENDIF»);}else{}«ELSEIF method.toStri
ng()=="setReason".toString()»«IF instanceVariable!=null»«instanceVariable.instanceVar
iableName».«method»«ELSE»_«methodName»ServiceException.«method»«ENDIF»(«IF serviceExc
eptionField.toString()=="none".toString()»«FOREACH parameters AS par»«par.name»«ENDFOR
EACH»«ELSE» ServiceException.«serviceExceptionField»«ENDIF»);}else{}«ENDIF»«ELSE»«IF
fields!=null»«IFfields.dataType.toString()!="none".toString()»«fields.dataType»
«ELSE»«ENDIF»«IF fields.isArray»«fields.name»[]= «ELSE»«fields.name» = «ENDIF»«ELSE»«ENDIF
»«IF method.toString()=="ServiceException".toString()»ServiceException.throwIt(«IF se
rviceExceptionField.toString()=="none".toString()»«FOREACH parameters AS par»«par.name
»«ENDFOREACH»«ELSE» ServiceException.«serviceExceptionField»«ENDIF»);«ELSEIF method.t
oString()=="throwIt".toString()»ServiceException.throwIt(«IF serviceExceptionField.to
String()=="none".toString()»«FOREACH parameters AS par»«par.name»«ENDFOREACH»«ELSE» Se
rviceException.«serviceExceptionField»«ENDIF»);«ELSEIF method.toString()=="getReason"
.toString()»«IF instanceVariable!=null»«instanceVariable.instanceVariableName».«metho
d»«ELSE»_«methodName»ServiceException.«method»«ENDIF»(«IF serviceExceptionField.toStr
ing()=="none".toString()»«FOREACH parameters AS par»«par.name»«ENDFOREACH»«ELSE» Servi
ceException.«serviceExceptionField»«ENDIF»);«ELSEIF method.toString()=="setReason".to
String()»«IF instanceVariable!=null»«instanceVariable.instanceVariableName».«method»«
ELSE»_«methodName»ServiceException.«method»«ENDIF»(«IF serviceExceptionField.toString
()=="none".toString()»«FOREACH parametersAS par»«par.name»«ENDFOREACH»«ELSE» ServiceEx
ception.«serviceExceptionField»«ENDIF»);«ENDIF»«ENDIF»
«ENDDEFINE»
«DEFINE sysEx(String methodName) FOR javaCardModel::SystemException»
«IF exceptionUsageType.toString()
== "tryCatch".toString()»try{}catch(SystemException «fields.name»){}«ELSEIF exceptionUs
ageType.toString()
== "ThrowNew".toString()»throw new SystemException «fields.name»;«ELSEIF exceptionUsage
Type.toString()
=="ifElse".toString()»if() {«IF fields!=null»«IF fields.dataType.toString()!="none".t
oString()»«fields.dataType»
«ELSE»«ENDIF»«IF fields.isArray»«fields.name»[]= «ELSE»«fields.name» = «ENDIF»«ELSE»«ENDIF
»«IF method.toString()=="SystemException".toString()»SystemException.throwIt(«IF syst
emExceptionField.toString()=="none".toString()»«FOREACH parameters AS par»«par.name»«E
NDFOREACH»«ELSE» SystemException.«systemExceptionField»«ENDIF»);}else{}«ELSEIF method
```

```
.toString()=="throwIt".toString()»SystemException.throwIt(«IF systemExceptionField.to
String()=="none".toString()»«FOREACH parameters AS par»«par.name»«ENDFOREACH»«ELSE» Sy
stemException.«systemExceptionField»«ENDIF»);}else{}«ELSEIF method.toString()=="getRe
ason".toString()»«IF instanceVariable!=null»«instanceVariable.instanceVariableName».«
method»«ELSE»_«methodName»SystemException.«method»«ENDIF»(«IF systemExceptionField.to
String()=="none".toString()»«FOREACH parameters ASpar»«par.name»«ENDFOREACH»«ELSE» Sys
temException.«systemExceptionField»«ENDIF»);}else{}«ELSEIF method.toString()=="setRea
son".toString()»«IF instanceVariable!=null»«instanceVariable.instanceVariableName».«m
ethod»«ELSE»_«methodName»SystemException.«method»«ENDIF»(«IF systemExceptionField.toS
tring()=="none".toString()»«FOREACH parameters AS par»«par.name»«ENDFOREACH»«ELSE» Sys
temException.«systemExceptionField»«ENDIF»);}else{}«ENDIF»«ELSE»«IF fields!=null»«IFf
ields.dataType.toString()!="none".toString()»«fields.dataType»
«ELSE»«ENDIF»«IF fields.isArray»«fields.name»[]= «ELSE»«fields.name» = «ENDIF»«ELSE»«ENDIF
»«IF method.toString()=="SystemException".toString()»SystemException.throwIt(«IFsyste
mExceptionField.toString()=="none".toString()»«FOREACH parameters AS par»«par.name»«EN
DFOREACH»«ELSE» SystemException.«systemExceptionField»«ENDIF»);«ELSEIF method.toStrin
g()=="throwIt".toString()»SystemException.throwIt(«IFsystemExceptionField.toString()=
="none".toString()»«FOREACH parameters AS par»«par.name»«ENDFOREACH»«ELSE» SystemExcep
tion.«systemExceptionField»«ENDIF»);«ELSEIF method.toString()=="getReason".toString()
»«IF instanceVariable!=null»«instanceVariable.instanceVariableName».«method»«ELSE»_«m
ethodName»SystemException.«method»«ENDIF»(«IF systemExceptionField.toString()=="none"
.toString()»«FOREACH parameters AS par»«par.name»«ENDFOREACH»«ELSE» SystemException.«s
ystemExceptionField»«ENDIF»);«ELSEIF method.toString()=="setReason".toString()»«IF in
stanceVariable!=null»«instanceVariable.instanceVariableName».«method»«ELSE»_«methodNa
me»SystemException.«method»«ENDIF»(«IF systemExceptionField.toString()=="none".toStri
ng()»«FOREACH parametersAS par»«par.name»«ENDFOREACH»«ELSE» SystemException.«systemExc
eptionField»«ENDIF»);«ENDIF»«ENDIF»
«ENDDEFINE»
«DEFINE tranEx(String methodName) FOR javaCardModel::TransactionException»
«IF exceptionUsageType.toString()
== "tryCatch".toString()»try{}catch(TransactionException «fields.name»){}«ELSEIF except
ionUsageType.toString()
== "ThrowNew".toString()»throw new TransactionException «fields.name»;«ELSEIF exception
UsageType.toString()
=="ifElse".toString()»if() {«IF fields!=null»«IF fields.dataType.toString()!="none".t
oString()»«fields.dataType»
«ELSE»«ENDIF»«IF fields.isArray»«fields.name»[]= «ELSE»«fields.name» = «ENDIF»«ELSE»«ENDIF
»«IF method.toString()=="TransactionException".toString()»TransactionException.throwI
t(«IF transactionExceptionField.toString()=="none".toString()»«FOREACH parameters AS
par»«par.name»«ENDFOREACH»«ELSE» TransactionException.«transactionExceptionField»«ENDI
F»);}else{}«ELSEIFmethod.toString()=="throwIt".toString()»TransactionException.throwI
t(«IF transactionExceptionField.toString()=="none".toString()»«FOREACH parameters AS
par»«par.name»«ENDFOREACH»«ELSE» TransactionException.«transactionExceptionField»«ENDI
F»);}else{}«ELSEIF method.toString()=="getReason".toString()»«IF instanceVariable!=nu
ll»«instanceVariable.instanceVariableName».«method»«ELSE»_«methodName»TransactionExce
ption.«method»«ENDIF»(«IF transactionExceptionField.toString()=="none".toString()»«FO
REACH parameters AS par»«par.name»«ENDFOREACH»«ELSE» TransactionException.«transaction
ExceptionField»«ENDIF»);}else{}«ELSEIF method.toString()=="setReason".toString()»«IF
instanceVariable!=null»«instanceVariable.instanceVariableName».«method»«ELSE»_«method
Name»TransactionException.«method»«ENDIF»(«IF transactionExceptionField.toString()==
none".toString()»«FOREACH parameters AS par»«par.name»«ENDFOREACH»«ELSE» TransactionEx
ception.«transactionExceptionField»«ENDIF»);}else{}«ENDIF»«ELSE»«IF fields!=null»«IF
fields.dataType.toString()!="none".toString()»«fields.dataType»
«ELSE»«ENDIF»«IF fields.isArray»«fields.name»[]= «ELSE»«fields.name» = «ENDIF»«ELSE»«ENDIF
»«IF method.toString()=="TransactionException".toString()»TransactionException.throwI
t(«IF transactionExceptionField.toString()=="none".toString()»«FOREACH parameters AS
par»«par.name»«ENDFOREACH»«ELSE» TransactionException.«transactionExceptionField»«ENDI
F»);«ELSEIF method.toString()=="throwIt".toString()»TransactionException.throwIt(«IF
transactionExceptionField.toString()=="none".toString()»«FOREACH parameters AS par»«p
ar.name»«ENDFOREACH»«ELSE» TransactionException.«transactionExceptionField»«ENDIF»);«E
LSEIF method.toString()=="getReason".toString()»«IF instanceVariable!=null»«instanceV
ariable.instanceVariableName».«method»«ELSE»_«methodName»TransactionException.«method
»«ENDIF»(«IF transactionExceptionField.toString()=="none".toString()»«FOREACH paramet
ers ASpar»«par.name»«ENDFOREACH»«ELSE» TransactionException.«transactionExceptionField
»«ENDIF»);«ELSEIF method.toString()=="setReason".toString()»«IF instanceVariable!=nul
l»«instanceVariable.instanceVariableName».«method»«ELSE»_«methodName»TransactionExcep
tion.«method»«ENDIF»(«IF transactionExceptionField.toString()=="none".toString()»«FOR
EACH parameters AS par»«par.name»«ENDFOREACH»«ELSE» TransactionException.«transactionE
```

```
xceptionField»«ENDIF»);«ENDIF»«ENDIF»
«ENDDEFINE»
«DEFINE arEx(String methodName) FOR javaCardModel::ArithmeticException»
«IF exceptionUsageType.toString()
== "tryCatch".toString()»try{}catch(ArithmeticException «fields.name»){}«ELSEIF excepti
onUsageType.toString()
== "ThrowNew".toString()»throw new ArithmeticException «fields.name»; «ELSEIF exception
UsageType.toString()
=="ifElse".toString()»if() {«IF fields!=null»«IF fields.dataType.toString()!="none".t
oString()»«fields.dataType»
«ELSE»«ENDIF»«IF fields.isArray»«fields.name»[]= «ELSE»«fields.name» = «ENDIF»«ELSE»«ENDIF
»«IF method.toString()=="ArithmeticException".toString()»ArithmeticException.throwIt(
«FOREACH parameters AS par»«par.name»«ENDFOREACH»);}else{}«ELSE »«IF instanceVariable!
=null»«instanceVariable.instanceVariableName».«method»«ELSE»_«methodName»ArithmeticEx
ception.«method»«ENDIF»(«FOREACHparameters AS par»«par.name»«ENDFOREACH»);}else{}«ENDI
F»«ELSE»«IF fields!=null»«IF fields.dataType.toString()!="none".toString()»«fields.da
taType»
«ELSE»«ENDIF»«IF fields.isArray»«fields.name»[]= «ELSE»«fields.name» = «ENDIF»«ELSE»«ENDIF
»«IFmethod.toString()=="ArithmeticException".toString()»ArithmeticException.throwIt(«
FOREACH parameters AS par»«par.name»«ENDFOREACH»);«ELSE»«IF instanceVariable!=null»«in
stanceVariable.instanceVariableName».«method»«ELSE»_«methodName»ArithmeticException.«
method»«ENDIF»(«FOREACH parameters AS par»«par.name»«ENDFOREACH»);«ENDIF»«ENDIF»
«ENDDEFINE»
«DEFINE arrStoEx(String methodName) FOR javaCardModel::ArrayStoreException»
«IF exceptionUsageType.toString()
== "tryCatch".toString()»try{}catch(ArrayStoreException «fields.name»){}«ELSEIF excepti
onUsageType.toString()
== "ThrowNew".toString()»throw new ArrayStoreException «fields.name»; «ELSEIF exception
UsageType.toString()
=="ifElse".toString()»if() {«IF fields!=null»«IF fields.dataType.toString()!="none".t
oString()»«fields.dataType»
«ELSE»«ENDIF»«IF fields.isArray»«fields.name»[]= «ELSE»«fields.name» = «ENDIF»«ELSE»«ENDIF
»«IF method.toString()=="ArrayStoreException".toString()»ArrayStoreException.throwIt(
«FOREACH parameters AS par»«par.name»«ENDFOREACH»);}else{}«ELSE »«IF instanceVariable!
=null»«instanceVariable.instanceVariableName».«method»«ELSE»_«methodName»ArrayStoreEx
ception.«method»«ENDIF»(«FOREACHparameters AS par»«par.name»«ENDFOREACH»);}else{}«ENDI
F»«ELSE»«IF fields!=null»«IF fields.dataType.toString()!="none".toString()»«fields.da
taType»
«ELSE»«ENDIF»«IF fields.isArray»«fields.name»[]= «ELSE»«fields.name» = «ENDIF»«ELSE»«ENDIF
»«IFmethod.toString()=="ArrayStoreException".toString()»ArrayStoreException.throwIt(«
FOREACH parameters AS par»«par.name»«ENDFOREACH»);«ELSE»«IF instanceVariable!=null»«in
stanceVariable.instanceVariableName».«method»«ELSE»_«methodName»ArrayStoreException.«
method»«ENDIF»(«FOREACH parameters AS par»«par.name»«ENDFOREACH»);«ENDIF»«ENDIF»
«ENDDEFINE»
«DEFINE cllCastEx(String methodName) FOR javaCardModel::ClassCastException»
«IF exceptionUsageType.toString()
== "tryCatch".toString()»try{}catch(ClassCastException «fields.name»){}«ELSEIF exceptio
nUsageType.toString()
== "ThrowNew".toString()»throw new ClassCastException «fields.name»; «ELSEIF exceptionU
sageType.toString()
=="ifElse".toString()»if() {«IF fields!=null»«IF fields.dataType.toString()!="none".t
oString()»«fields.dataType»
«ELSE»«ENDIF»«IF fields.isArray»«fields.name»[]= «ELSE»«fields.name» = «ENDIF»«ELSE»«ENDIF
»«IF method.toString()=="ClassCastException".toString()»ClassCastException.throwIt(«F
OREACH parameters AS par»«par.name»«ENDFOREACH»);}else{}«ELSE »«IF instanceVariable!=n
ull»«instanceVariable.instanceVariableName».«method»«ELSE»_«methodName»ClassCastExcep
tion.«method»«ENDIF»(«FOREACH parametersAS par»«par.name»«ENDFOREACH»);}else{}«ENDIF»«
ELSE»«IF fields!=null»«IF fields.dataType.toString()!="none".toString()»«fields.dataT
ype»
«ELSE»«ENDIF»«IF fields.isArray»«fields.name»[]= «ELSE»«fields.name» = «ENDIF»«ELSE»«ENDIF
»«IFmethod.toString()=="ClassCastException".toString()»ClassCastException.throwIt(«FO
REACH parameters AS par»«par.name»«ENDFOREACH»);«ELSE»«IF instanceVariable!=null»«inst
anceVariable.instanceVariableName».«method»«ELSE»_«methodName»ClassCastException.«met
hod»«ENDIF»(«FOREACH parameters AS par»«par.name»«ENDFOREACH»);«ENDIF»«ENDIF»
«ENDDEFINE»
«DEFINE inBEx(String methodName) FOR javaCardModel::IndexOutOfBoundsException»
«IF exceptionUsageType.toString()
== "tryCatch".toString()»try{}catch(IndexOutOfBoundsException «fields.name»){}«ELSEIF e
```

```
xceptionUsageType.toString()
== "ThrowNew".toString()»throw new IndexOutOfBoundsException «fields.name»; «ELSEIFexce
ptionUsageType.toString()
== "ifElse".toString()»if() {«IF fields!=null»«IF fields.dataType.toString()!="none".
toString()»«fields.dataType»
«ELSE»«ENDIF»«IF fields.isArray»«fields.name»[]= «ELSE»«fields.name» = «ENDIF»«ELSE»«ENDIF
»«IF method.toString()=="IndexOutOfBoundsException".toString()»IndexOutOfBoundsExcept
ion.throwIt(«FOREACH parameters AS par»«par.name»«ENDFOREACH»);}else{}«ELSE »«IF insta
nceVariable!=null»«instanceVariable.instanceVariableName».«method»«ELSE»_«methodName»
IndexOutOfBoundsException.«method»«ENDIF»(«FOREACH parameters AS par»«par.name»«ENDFOR
EACH»);}else{}«ENDIF»«ELSE»«IF fields!=null»«IF fields.dataType.toString()!="none".to
String()»«fields.dataType»
«ELSE»«ENDIF»«IF fields.isArray»«fields.name»[]=«ELSE»«fields.name» = «ENDIF»«ELSE»«ENDIF»
«IF method.toString()=="IndexOutOfBoundsException".toString()»IndexOutOfBoundsExcepti
on.throwIt(«FOREACH parameters AS par»«par.name»«ENDFOREACH»);«ELSE»«IF instanceVariab
le!=null»«instanceVariable.instanceVariableName».«method»«ELSE»_«methodName»IndexOutO
fBoundsException.«method»«ENDIF»(«FOREACH parameters AS par»«par.name»«ENDFOREACH»);«E
NDIF»«ENDIF»
«ENDDEFINE»
«DEFINE arBoundsEx(String
methodName) FOR javaCardModel::ArrayIndexOutOfBoundsException»
«IF exceptionUsageType.toString()
== "tryCatch".toString()»try{}catch(ArrayIndexOutOfBoundsException «fields.name»){}«ELS
EIF exceptionUsageType.toString()
== "ThrowNew".toString()»throw new ArrayIndexOutOfBoundsException «fields.name»; «ELSEI
FexceptionUsageType.toString()
== "ifElse".toString()»if() {«IF fields!=null»«IF fields.dataType.toString()!="none".
toString()»«fields.dataType»
«ELSE»«ENDIF»«IF fields.isArray»«fields.name»[]= «ELSE»«fields.name» = «ENDIF»«ELSE»«ENDIF
»«IF method.toString()=="ArrayIndexOutOfBoundsException".toString()»ArrayIndexOutOfBo
undsException.throwIt(«FOREACH parameters AS par»«par.name»«ENDFOREACH»);}else{}«ELSE
»«IF instanceVariable!=null»«instanceVariable.instanceVariableName».«method»«ELSE»_«m
ethodName»ArrayIndexOutOfBoundsException.«method»«ENDIF»(«FOREACH parameters AS par»«
par.name»«ENDFOREACH»);}else{}«ENDIF»«ELSE»«IF fields!=null»«IF fields.dataType.toStri
ng()!="none".toString()»«fields.dataType»
«ELSE»«ENDIF»«IFfields.isArray»«fields.name»[]= «ELSE»«fields.name» = «ENDIF»«ELSE»«ENDIF»
«IF method.toString()=="ArrayIndexOutOfBoundsException".toString()»ArrayIndexOutOfBou
ndsException.throwIt(«FOREACH parameters AS par»«par.name»«ENDFOREACH»);«ELSE»«IFinsta
nceVariable!=null»«instanceVariable.instanceVariableName».«method»«ELSE»_«methodName»
ArrayIndexOutOfBoundsException.«method»«ENDIF»(«FOREACH parameters AS par»«par.name»«E
NDFOREACH»);«ENDIF»«ENDIF»
«ENDDEFINE»
«DEFINE negativeArraySizeEx(String
methodName) FOR javaCardModel::NegativeArraySizeException»
«IF exceptionUsageType.toString()
== "tryCatch".toString()»try{}catch(NegativeArraySizeException «fields.name»){}«ELSEIF
exceptionUsageType.toString()
== "ThrowNew".toString()»throw new NegativeArraySizeException «fields.name»; «ELSEIFexc
eptionUsageType.toString()
== "ifElse".toString()»if() {«IF fields!=null»«IF fields.dataType.toString()!="none".
toString()»«fields.dataType»
«ELSE»«ENDIF»«IF fields.isArray»«fields.name»[]= «ELSE»«fields.name» = «ENDIF»«ELSE»«ENDIF
»«IF method.toString()=="NegativeArraySizeException".toString()»NegativeArraySizeExce
ption.throwIt(«FOREACH parameters AS par»«par.name»«ENDFOREACH»);}else{}«ELSE »«IF ins
tanceVariable!=null»«instanceVariable.instanceVariableName».«method»«ELSE»_«methodNam
e»NegativeArraySizeException.«method»«ENDIF»(«FOREACH parameters AS par»«par.name»«END
FOREACH»);}else{}«ENDIF»«ELSE»«IF fields!=null»«IF fields.dataType.toString()!="none"
.toString()»«fields.dataType»
«ELSE»«ENDIF»«IF fields.isArray»«fields.name»[]=«ELSE»«fields.name» = «ENDIF»«ELSE»«ENDIF»
«IF method.toString()=="NegativeArraySizeException".toString()»NegativeArraySizeExcep
tion.throwIt(«FOREACH parameters AS par»«par.name»«ENDFOREACH»);«ELSE»«IF instanceVari
able!=null»«instanceVariable.instanceVariableName».«method»«ELSE»_«methodName»Negativ
eArraySizeException.«method»«ENDIF»(«FOREACH parameters AS par»«par.name»«ENDFOREACH»)
;«ENDIF»«ENDIF»
«ENDDEFINE»
«DEFINE nullPointerEx(String methodName) FOR javaCardModel::NullPointerException»
«IF exceptionUsageType.toString()
== "tryCatch".toString()»try{}catch(NullPointerException «fields.name»){}«ELSEIF except
```

```
ionUsageType.toString()
== "ThrowNew".toString()»throw new NullPointerException «fields.name»; «ELSEIF exceptio
nUsageType.toString()
=="ifElse".toString()»if() {«IF fields!=null»«IF fields.dataType.toString()!="none".t
oString()»«fields.dataType»
«ELSE»«ENDIF»«IF fields.isArray»«fields.name»[]= «ELSE»«fields.name» = «ENDIF»«ELSE»«ENDIF
»«IF method.toString()=="NullPointerException".toString()»NullPointerException.throwI
t(«FOREACH parameters AS par»«par.name»«ENDFOREACH»);}else{}«ELSE »«IF instanceVariabl
e!=null»«instanceVariable.instanceVariableName».«method»«ELSE»_«methodName»NullPointe
rException.«method»«ENDIF»(«FOREACH parameters AS par»«par.name»«ENDFOREACH»);}else{}«
ENDIF»«ELSE»«IF fields!=null»«IF fields.dataType.toString()!="none".toString()»«field
s.dataType»
«ELSE»«ENDIF»«IF fields.isArray»«fields.name»[]= «ELSE»«fields.name» = «ENDIF»«ELSE»«ENDIF
»«IFmethod.toString()=="NullPointerException".toString()»NullPointerException.throwIt
(«FOREACH parameters AS par»«par.name»«ENDFOREACH»);«ELSE»«IF instanceVariable!=null»«
instanceVariable.instanceVariableName».«method»«ELSE»_«methodName»NullPointerExceptio
n.«method»«ENDIF»(«FOREACH parameters AS par»«par.name»«ENDFOREACH»);«ENDIF»«ENDIF»
«ENDDEFINE»


«FOREACH m.ecPrivateKeys AS ecPriKey ITERATOR ecPriKeyIt»«IF ecPriKeyIt.counter1 ==
1»«LET "ECPrivateKey _" + m.methodName
+ "ECPrivateKey" + ";" AS ECPriKeyVar»«ECPriKeyVar»//You may delete or rename to use
this variable in yourmethods«ENDLET»«ENDIF»«ENDFOREACH»
    «FOREACH m.ecPublicKeys AS ecPubKey ITERATOR ecPubKeyIt»«IF ecPubKeyIt.counter1
== 1»«LET "ECPublicKey _" + m.methodName
+ "ECPublicKey" + ";" AS ECPubKeyVar»«ECPubKeyVar»//You may delete or rename to use t
his variable in yourmethods«ENDLET»«ENDIF»«ENDFOREACH»
    «FOREACH m.keys AS key ITERATOR keyIt»«IF keyIt.counter1 == 1»«LET "Key _" +
m.methodName
+ "Key" + ";" AS KeyVar»«KeyVar»//You may delete or rename to use this variable in yo
ur methods«ENDLET»«ENDIF»«ENDFOREACH»
        «FOREACH m.privateKeys AS privateKey ITERATOR privateKeysIt»«IF privateKeysIt
.counter1 == 1»«LET "PrivateKey _" + m.methodName
+ "PrivateKey" + ";" AS PrKeyVar»«PrKeyVar»//You may delete or rename to use this var
iable in yourmethods«ENDLET»«ENDIF»«ENDFOREACH»
        «FOREACH m.publicKeys AS publicKey ITERATOR publicKeyIt»«IF publicKeyIt.count
er1 == 1»«LET "PublicKey _" + m.methodName
+ "PublicKey" + ";" AS PubKeyVar»«PubKeyVar»//You may delete or rename to use this va
riable in yourmethods«ENDLET»«ENDIF»«ENDFOREACH»
        «FOREACH m.rsaPrivateKeys AS rsaPrivateKey ITERATOR iterator»«IF iterator.cou
nter1 == 1»«LET "RSAPrivateKey _" + m.methodName
+ "RSAPrivateKey" + ";" AS Var»«Var»//You may delete or rename to use this variable i
n yourmethods«ENDLET»«ENDIF»«ENDFOREACH»
        «FOREACH m.rsaPrivateCrtKeys AS rsaPrivateCrtKey ITERATOR iterator»«IF iterat
or.counter1 == 1»«LET "RSAPrivateCrtKey _" + m.methodName
+ "RSAPrivateCrtKey" + ";" AS Var»«Var»//You may delete or rename to use this variabl
e in yourmethods«ENDLET»«ENDIF»«ENDFOREACH»
    «FOREACH m.rsaPublicKeys AS rsaPublicKey ITERATOR iterator»«IF iterator.counter1
== 1»«LET "RSAPublicKey _" + m.methodName
+ "RSAPublicKey" + ";" AS Var»«Var»//You may delete or rename to use this variable in
 your methods«ENDLET»«ENDIF»«ENDFOREACH»
```