

# **A TEXT PROCESSING AND ANALYSIS TOOL FOR TURKISH**

by

Melek OKTAY

June 2007

# **A TEXT PROCESSING AND ANALYSIS TOOL FOR TURKISH**

by

Melek OKTAY

A thesis submitted to

the Graduate Institute of Sciences and Engineering

of

Fatih University

in partial fulfillment of the requirements for the degree of

Master of Science

in

Computer Engineering

June 2007  
Istanbul, Turkey

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

\_\_\_\_\_  
Prof. Dr. Bekir KARLIK  
Head of Department

This is to certify that I have read this thesis and that in my opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

\_\_\_\_\_  
Assist. Prof. Dr. Atakan KURT  
Supervisor

Examining Committee Members

Assist. Prof. Dr. Atakan KURT

\_\_\_\_\_

Assist. Prof. Dr. Veli HAKKOYMAZ

\_\_\_\_\_

Assoc. Prof. Dr. Mehmet KARA

\_\_\_\_\_

It is approved that this thesis has been written in compliance with the formatting rules laid down by the Graduate Institute of Sciences and Engineering.

\_\_\_\_\_  
Assist. Prof. Dr. Nurullah ARSLAN  
Director

Date  
June 2007

# **A TEXT PROCESSING AND ANALYSIS TOOL FOR TURKISH**

**Melek OKTAY**

M. S. Thesis - Computer Engineering  
July 2007

Supervisor: Assist. Prof. Atakan KURT

## **ABSTRACT**

The analysis of Turkish texts is significant in Turkish language, literature and a wide spectrum of areas. It is a complicated task to count language structures in the texts manually. By the way, a computer application that processes and analyzes Turkish text documents or document sets (corpus) is beneficial. In this thesis, the text processing and analyzing tool is developed to analyze the texts and computes various phonetic, syllable, affix, stem, word, sentence frequencies.

The text processing and analyzing tool developed can analyze Turkish texts using the frequency distributions of various language elements such as phonemes, syllables, affixes, words etc. The tool is developed with Java programming language and it is implemented according to PCMEF architecture. The program developed provides facilities for adding new languages and it is not difficult to extend to do the same for some Turkic dialects.

**Keywords:** Text processing and analyzing, Java, Turkish, internalization, PCMEF

# TÜRKÇE İÇİN BİR YAZI İŞLEME VE ANALİZ PROGRAMI

**Melek OKTAY**

Yüksek Lisan Tezi – Bilgisayar Mühendisliği  
Temmuz 2007

Tez Yöneticisi: Yrd. Doç. Dr. Atakan KURT

## ÖZ

Türkçe metinlerin analiz edilmesi, Türk dilinde, edebiyatında ve çok geniş bir spectrumda önemlidir. Metinlerdeki dil yapılarını elle saymak çok karmaşık bir iştir. Bu nedenle, Türkçe dökümanları ve sözlükleri işleyen ve analiz eden bir bilgisayar uygulaması gereklidir. Bu tezde, yazı işleme ve analiz etme aracı metinleri analiz etmek ve ses, hece, ek, kelime, kelime grubu, cümle, paragraf sıklıklarını hesaplamak için geliştirilmiştir.

Geliştirilen metin işleme ve analiz etme aracı, Türkçe metinleri ses, hece, ek, kök gibi çeşitli dil elemanlarının sıklık dağılımlarını kullanarak analiz eder. Araç Java programlama dili ile geliştirilmiştir ve PCMEF yapısına göre tasarlanmıştır. Geliştirilen program yeni dillerin eklenmesi için kolaylıklar sağlar ve çeşitli Türkçe lehçelerini de aynı şekilde eklemek zor değildir.

**Anahtar Kelimeler:** Metin işleme ve analizi, Java, Türkçe, öğrenme, PCMEF

## **ACKNOWLEDGEMENT**

I express sincere appreciation to Assist. Prof. Atakan KURT and Assoc. Prof. Mehmet KARA for their guidance and insight throughout the research.

Thanks go to the committee member Assist. Prof. Veli Hakkoymaz for his valuable suggestions and comments.

I express my thanks and appreciation to my wife Ayşe Betül OKTAY, my mother Beliğa OKTAY, my family and my friends for their understanding, motivation and patience.

## TABLE OF CONTENTS

ABSTRACT.....	iv
ÖZ .....	v
ACKNOWLEDGEMENT .....	vi
LIST OF TABLES.....	ix
LIST OF FIGURES .....	x
CHAPTER 1 INTRODUCTION.....	1
CHAPTER 2 BACKGROUND.....	4
2.1 The Turkish Phonetics .....	4
2.1.1 The Letter.....	4
2.1.2 The Syllable.....	5
2.1.3 The Word.....	6
2.2 Unicode.....	6
2.3 Internalization .....	7
2.3.1 Locales.....	8
2.3.1.1 Language Codes.....	9
2.3.1.2 Country (Region) Codes .....	9
2.3.1.3 Variant Code .....	10
2.3.2 Number Formats .....	10
2.3.3 Date And Time.....	11
2.3.4 Collation (Sorting).....	11
2.3.5 Message Formatting.....	12
2.3.6 Text Files And Character Sets .....	12

2.3.7	Resource Bundles .....	13
2.4	Pcmef .....	14
CHAPTER 3 TEXT PROCESSING AND ANALYSIS TOOL .....		17
3.1	The Text Editor .....	18
3.2	The Frequency Analysis Of Characters .....	20
3.3	The Frequency Analysis Of Syllables .....	27
3.4	The Frequency Analysis Of Words .....	32
3.5	The Frequency Analysis Of Sentences .....	39
3.6	The Frequency Analysis Of Paragraphs .....	42
CHAPTER 4 TOOL OBJECT MODEL.....		46
4.1	Use Case Diagram .....	46
4.2	Packages And Class Diagrams .....	47
4.3	Sequence Diagrams .....	54
CHAPTER 5 CONCLUSIONS .....		56
REFERENCES .....		58



## LIST OF TABLES

Table 2.1 Turkish Vowels.....	5
Table 2.2 Turkish consonants .....	5
Table 2.3 Turkish syllable types .....	6
Table 2.4 Language codes examples .....	9
Table 2.5 Country codes examples .....	9

## LIST OF FIGURES

Figure 3.1 General GUI of developed tool .....	17
Figure 3.2 Editor GUI.....	19
Figure 3.3 Character window.....	21
Figure 3.4 Frequency of letters in words in “Fatih Üniversitesi” text.....	23
Figure 3.5 Frequency of letters in syllables in “Fatih Üniversitesi” text.....	24
Figure 3.6 Letter frequency output of the text “Fatih Üniversitesi” .....	25
Figure 3.7 Character types output of the text “Fatih Üniversitesi” .....	26
Figure 3.8 Output window of letter analyzing.....	27
Figure 3.9 The syllable menu .....	28
Figure 3.10 The syllable output for “Fatih üniversitesi” .....	30
Figure 3.11 Ratio of syllables in “Fatih üniversitesi” .....	31
Figure 3.12 Colored ratio of syllables in “Fatih üniversitesi” text .....	32
Figure 3.13 Word Analyzing menu .....	33
Figure 3.14 Word frequency output.....	35
Figure 3.15 Word & syllable count frequency .....	36
Figure 3.16 Suffix frequency.....	37
Figure 3.17 Word root frequency .....	37
Figure 3.18– Letter count frequency.....	38
Figure 3.19 Words according to language frequency .....	38
Figure 3.20 Sentence window.....	40
Figure 3.21 Word count frequency .....	41
Figure 3.22 Syllable count frequency .....	42
Figure 3.23 Paragraph analyzing window .....	43
Figure 3.24 Sample output of paragraph .....	45
Figure 4.1 Use case diagram of the text processing tool .....	47

Figure 4.2 Package diagram of text processing tool .....	49
Figure 4.3 Class diagrams of presentation layer .....	50
Figure 4.4 Class diagrams of editor .....	51
Figure 4.5 Class diagram of control layer.....	52
Figure 4.6 Class diagram of domain-mediator layer .....	53
Figure 4.7 Class diagram of entity layer.....	54
Figure 4.8 Sequence diagram 1.....	55
Figure 4.9 Sequence diagram 2.....	55

## LIST OF SYMBOLS AND ABBREVIATIONS

### SYMBOL/ABBREVIATION

SQL	Structured query language
XML	Extensible markup language
CSV	Coma separated value
XSLT	Extensible style sheet language transformations
HTML	Hyper text markup language
PCMEF	Presentation control mediator entity foundation
POJO	Plain old java object
GUI	Graphical user interface
OOP	Object oriented programming
JDK	Java development kit
UTF	Unicode transformation format
ISO	International organization for standardization
XPath	XML path language
NLP	Natural Language Processing

# CHAPTER 1

## INTRODUCTION

The analysis of Turkish texts is a beneficial technique that can be used not only in Turkish language and literature itself but also in a wide spectrum of areas encompassing education, psychology, sociology, politics, business management, criminology, law and medicine (Adalı, 2004; Aksa, 1978; Toklu, 2005). Thus an application that can analyze Turkish texts using the frequency distributions of various language elements such as phonemes, syllables, affixes, words etc. will be quite valuable in all these areas (Tantuğ et al, 2006; Eryiğit and Oflazer, 2006).

As an example, we can study the transformation of the Turkish language from phonetic, morphologic, semantic points of view within a specific time frame by inspecting a corpus containing several documents from that time frame. The main thrust of such study should be based on the frequency analysis of letters, syllables, words, affixes, and sentences etc. Computing these frequencies for even a small document set is an enormous task for people. Nowadays, most main texts of Turkish language can be found in digital form. New materials that are produced already come into existence in digital form or can easily be transformed into such forms. Using computers in such study will reduce the time required dramatically and can eliminate errors drastically. As an added advantage the results can readily be used by other computer applications or can directly be used by people in their research.

Even though these types of applications have been developed for English and some other languages, there are certain constraints when using these applications for Turkish

texts. Turkish as a completely different language from English and others has a different alphabet, different rules for syllable generation, different morphologic structures, a different grammar and syntax, and different semantic and conceptual patterns (Tekcan and Göz, 2005). For these reasons, applications developed for English and other languages can't be used for Turkish texts. Even if they were used, the result will be erroneous and incomplete, thus unreliable.

An application that can analyze the frequencies of various language elements in Turkish texts should not be too difficult to extend to do the same for some Turkic dialects such as Azari, Turkmen because there are resemblance and commonality to a great degree between Turkish and some of these dialects. As a future work, the application to process texts in at least one dialect of Turkish can be extended. We are thinking of Turkmen as the first choice because Turkmen is one of the closest dialects to Turkish and there seems to be enough language research on Turkmen to support such an endeavor. Naturally, we expect that the extension and the success of the application on the dialects will be more limited compared to Turkish since more research and data is available for Turkish than most of these dialects.

In this study, a computer application that can help analyze Turkish text documents or document sets (corpus) by computing various phonetic, syllable, affix, stem, word, sentence frequencies is developed with Java programming language. First of all, the program has the basic editor capabilities. Opening a document / documents, saving, cutting, copying and pasting actions are allowed. By the way, the analyzed documents are arranged according to the user's preferences. Also text editing like changing font, size, text color, making text bold, italic, underlined are possible.

The most important part of the project is text analyzing. The analysis is done for five items: Letters, syllabus, words, sentences and paragraphs. The frequencies, ratios and statistics are found by the program in the document. User can choose the constraints for processing and format the output file. For example output can be in SQL, XML and CSV format. Also the output can be sorted according to the user's preferences.

This thesis is organized as follows: Chapter 2 gives information about the language items containing letters, syllabus, words, sentences and paragraphs and internalization. Chapter 3 presents the Turkish analyzing and processing tool developed. Chapter 4 presents the object model of the developed tool. Chapter 5 discusses future work and concludes the study.

## CHAPTER 2

### BACKGROUND

This chapter includes general background information about Turkish language, Unicode, internalization and PCMEF framework which are basis of the text processing and analyzing tool. This chapter is divided into several sections. In section 2.1, the Turkish phonetics including the letters, syllables, words and their characteristics are presented. In section 2.2, the Unicode is discussed. Topics about locales, number formats, date and time, collation, message formatting, text files and character sets and resource bundles are presented in section 2.3. The PCMEF architecture is presented in section 2.4.

#### 2.1 THE TURKISH PHONETICS

##### 2.1.1 The Letter

Turkish is a member of the proposed Altaic language family. The distinctive characteristics of Turkish are vowel harmony and extensive agglutination (International Phonetic Association, 1999). The Turkish alphabet is a variant of the Latin alphabet used for writing the Turkish language, consisting of 29 letters.

These letters are, in the upper case:

*A, B, C, Ç, D, E, F, G, Ğ, H, I, İ, J, K, L, M, N, O, Ö, P, R, S, Ş, T, U, Ü, V, Y, Z*

In the lower case:

*a, b, c, ç, d, e, f, g, ğ, h, ı, i, j, k, l, m, n, o, ö, p, r, s, ş, t, u, ü, v, y, z.*



The vowels in Turkish are a, e, ɪ, i, o, ö, u, ü and rest 21 characters are consonants. Vowels are divided into 2 categories because of their two features: front/back and rounded/unrounded. Vowel harmony is the principle by which a native Turkish word incorporates either exclusively back vowels (*a, ɪ, o, u*) or exclusively front vowels (*e, i, ö, ü*). The pattern of vowels is shown in the Table 2.1. (Oflazer, 1993)

**Table 2.1** Turkish Vowels

	Front		Back	
	Unrounded	Rounded	Unrounded	Rounded
<b>High</b>	i	ü	ɪ	u
<b>Low</b>	e	ö	a	o

The phonetic features of consonants in Turkish are shown in Table 2.2.

**Table 2.2** Turkish consonants

	Labial	Labio-dental	Dental	Palato-alveolar	Palatal	Velar	Glottal
<b>Voiceless Stop</b>	p		t	ç		k	
<b>Voiced Stop</b>	b		d	c		g	
<b>Voiceless Fricative</b>		f	s	ş			
<b>Voiced Fricative</b>		v	z	j			
<b>Nasal</b>	m		n				
<b>Liquid</b>			l, r				
<b>Approximant</b>					y		h

### 2.1.2 The Syllable

The syllables in Turkish are formed with the combination of consonants and vowels in many ways. There are six types of syllables in Turkish shown in Table 2.3. V shows

vowel and C shows consonant. V represents the syllables formed with a vowel. Also the one syllable words must obey the syllable rules.

**Table 2.3** Turkish syllable types

Syllable type	Example
V	a, e, ı, i, o, ö, u, ü
VC	ab.aç,iş
CV	ba, be, bı
CVC	bel, gel, köy, tır, ...
VCC	alt, üst, ırk, ...
CVCC	kurt, yurt, renk, türk

VC shows the syllables that contain a vowel and a consonant like ab, aç, iş. CV is for the syllables and includes first a consonant then a vowel like ba, be. The syllables formed with a consonant, a vowel and again a consonant are shown with CVC. VCC represents the syllables begin with a vowel and continue with two consonants. CVCC is the syllable type begins with consonant, then a vowel and end with two consonants. This type of syllable is rarely formed like kurt, renk and yurt.

The syllables that end with consonants are called closed syllables like köy, ab etc. The syllables ends with vowels are called open syllables.

### 2.1.3 The Word

Words are formed with the combination of syllables. Turkish extensively uses agglutination that is adding affixes to the base of the word to form new words from nouns and verbal stems. The majority of Turkish words originate from the application of derivative suffixes to a relatively small set of core vocabulary.

## 2.2 UNICODE

Unicode is a 16-bit character set standard and its encoding scheme represents and encodes all characters in the world such as Turkish, Japanese, Korean, and Western Union

etc. Unicode is designed and maintained by non-profit consortium Unicode Inc (Unicode, 2007).

Computers internally work with numbers; characters need be coded as numbers. Unicode gives every character of every language a unique number. Also, this number does not depend on language used in the text. For displaying characters successfully, suitable font (software for drawing characters) should be installed on computer.

Most of the operating systems such as Windows, Mac OS X, and Linux support Unicode. However, in order to use Unicode, all significant components must be Unicode enabled. For example, although Windows processes Unicode, an application program working on a Windows system might not process. Moreover, the display or printing of characters often fails since fonts still incomplete in covering the set of Unicode characters (Jukka, 2006).

Java supports Unicode and the concept of 16-bit Unicode character streams are introduced in JDK 1.1. *Java.io.Reader* and *java.io.Writer* are the abstract parent classes for character-stream based classes in the *java.io* package (Hemrajani, 1998). These classes support 16-bit streams, also subclasses of these automatically supports 16-bit character streams. When program reads text in files or saving documents to files, content of texts are not damaged.

### **2.3 INTERNALIZATION**

Software developers want lots of inhabitant interests program that they are developed. Software supports many language character sets in the world, but software developer does not change internal or external behaviors of program. But it should be considered in the beginning of the software development and software that is developed should develop in this manner.

The Java programming language was the first language designed from the ground up to support internationalization (Horstmann and Cornell, 2004). In addition to Character and

String used Unicode, there is a lot more to internationalizing programs in Java. For example; dates, times, and numbers formatted differently in different parts of the world.

### 2.3.1 Locales

Locales identify a specific language and geographic region. And they affect user interface, data formats (date, time, number) and collation (sorting). Locales are critical to many culturally and linguistically sensitive data operations (O'Conner, 2002). Java class libraries use *java.util.Locale* object to represent locales.

Software developer can use default Locale or s/he force use any Locale s/he want. Java environment uses default Locale if it is not specified which Locale is used. Method of getting default Locale in Java is below:

```
public static Locale getDefault ()
```

Also it can be used with any Locale like:

*Locale germanLocale = new Locale ("de");* “de” parameter represents German language.

*Locale germanGermanyLocale = new Locale ("de", "DE");* “de” parameter represents German language and “DE” represents German country.

*Locale germanSwitzerlandLocale = new Locale ("de", "CH");* “de” parameter represents German language and “CH” represents Switzerland country.

Also programmer can change default Locale with using below method in Java.

```
public static void setDefault(Locale locale)
```

Locales contain only a few important members: A language code, an optional country/region code, and an optional variant code. But these three members provide enough information for a specific linguistic or cultural purpose.

### 2.3.1.1 Language Codes

Language codes are defined by *ISO 639* which is an international standard. It assigns two and three letter codes to most languages of the world. Locale uses the two letter codes to identify their target language (O'Conner, 2002). Some examples of language codes are shown in Table 2.4.

**Table 2.4** Language codes examples

<b>Language</b>	<b>Code</b>
Arabic	<i>ar</i>
German	<i>de</i>
English	<i>en</i>
Spanish	<i>es</i>
Japanese	<i>ja</i>
Hebrew	<i>iw</i>

### 2.3.1.2 Country (Region) Codes

Country codes are defined by *ISO 3166* that is an international standard. It defines two and three letter abbreviations for each country or major region in the world. The country codes are uppercased unlike language codes. Locale uses the two letter codes instead of the three letter codes that are also defined by other versions of this standard. Table 2.5 shows some of the defined codes.

**Table 2.5** Country codes examples

<b>Country</b>	<b>Code</b>
United States	US
Canada	CA
France	FR
Japan	JP
Germany	DE

### 2.3.1.3 Variant Code

Operating system, browser, and some application vendors may use the variant to provide additional functionality or customization that isn't possible with just a language and country designation. For example, a software company may need to indicate a locale for a specific operating system, so they may create an `es_ES_MAC` or `es_ES_WIN` locale for the Macintosh or Windows platforms. One historical example from the Java 2 platform itself is the use of the EURO variant for European locales that use the Euro currency. During the transition period for those countries, the Java platform (version 1.3) used this variant. For example, although a `de_DE` (German-speaking Germany) locale existed, a `de_DE_EURO` (German-speaking German locale with a Euro variant) was added to the Java environment. The reason is that, the Euro currency is now the standard currency for the affected locales at this point, those variants have been removed since version 1.4 of the platform. Most application designs will probably not require variant locale definitions.

### 2.3.2 Number Formats

Numbers are dependent to locale. Java Platform supplies a collection of formatter that can format numbers in the `java.text` class libraries. Formatting numbers in Java requires some steps such as:

First of all, locale should be created:

```
Locale germanLocale = new Locale("de", "DE");
```

Next, format numbers with using `NumberFormat` class, but locale object should be given as a parameter.

```
NumberFormat nbrFrmtr = NumberFormat.getCurrencyInstance (germanLocale);
```

Then, number can be formatted with given locale type like below:

```
double number = 123456.78;
```

```
nbrFrmtr.format (number);
```

Result can be shown like 123.456,78 DM

### 2.3.3 Date and Time

Formatting date and time also highly locale dependent. *java.text.DateFormat* class handles many issues that they are required for formatting date and time such as; names of months and weekdays in the local language, time zone of the location and order of year, month, and day. Example usage of this class can be shown below:

```
DateFormat dateFormater =
```

```
DateFormat.getDateInstance(java.text.DateFormat.LONG, Locale.FRANCE);
```

Then, Date class object is created and when Date object created it takes default time of system. Date object is formatted like:

```
dateFormater.format(new Date());
```

and at last output is = *17 mai 2007*

### 2.3.4 Collation (Sorting)

Collation is the sorting of written information into a standard order. Sorting of strings in one language can be easy but when there are more than one language to sort it becomes challenging. For example, it is wanted to sort following five strings:

```
America  
Zulu  
Ant  
Zebra  
Ångström
```

English user wants to result is shown like below after sort string:

```
America  
Ångström
```

*Ant*  
*Zebra*  
*Zulu*

But, Swedish user wants to result is shown different order, because the letter Å is different from the letter A, and it is collated after the letter Z (Horstmann and Cornell, 2004). So result is listed by following:

*America*  
*Ant*  
*Zebra*  
*Zulu*  
*Ångström*

### 2.3.5 Message Formatting

MessageFormat provides producing concatenated messages in language-neutral way (the message format class). It is used for format the messages and display them to user. MessageFormat class takes a set of objects, and format them with given pattern. It is also depends on locale. Example for this class can be shown below:

```
String pattern = MessageFormat.format("On {2}, a {0} takes {1}" ,"Galatasaray",
"UEFA Cup" , new GregorianCalendar(2000, 5, 17).getTime());
```

```
MessageFormat messageFormat = new MessageFormat(pattern, locale);
String msg = messageFormat.format(new Object[] { values });
```

Output is can be shown like below;

*On 17/5/2000 09:45:00 PM, a Galatasaray takes UEFA Cup.*

### 2.3.6 Text Files and Character Sets

Although Java programming language is fully Unicode base, operating system uses its own character encoding such as ISO-8859-1. For this reason, when text document is



saved or opened, local character encoding should be given to desired objects. For example: text document is saved with using ISO-8859-1 encoding in FileWriter class.

```
fileWriter = new FileWriter(filename, "ISO-8859-1");
```

There is no link between locales and character encodings. For example, if someone selects the Taiwanese locale zh\_TW, no method in the Java programming language tells that the Big5 character encoding would be the most appropriate.

During the development of software in Java source files encoding is local encoding, interpreted class files is UTF-8 and in Java Virtual machine is UTF-16.

### **2.3.7 Resource Bundles**

Resource bundles contain language dependent text strings and when localizing software application. It must be produced a set of resource bundles. Resource bundles contain locale-specific objects, and when application is needed a locale specific resource, such as string, program can load it from resource bundle (The resource bundle class). In this way program is independent from static labels, buttons, messages names etc. All the message strings are defined in the external location which is called a resource. So, external resource files are edited without change source code of the program.

Application programmer use specific naming convention for these bundles, such as:

```
BundleName_language_country
```

For example, Turkish country and Turkish language bundle name format should be BundleName\_tr\_TR. However, bundle name should be same for the entire resource bundle that program supports. For example:

```
ResourceBundle.properties  
ResourceBundle_en.properties  
ResourceBundle_de_DE.properties  
ResourceBundle_tr_TR.Properties
```

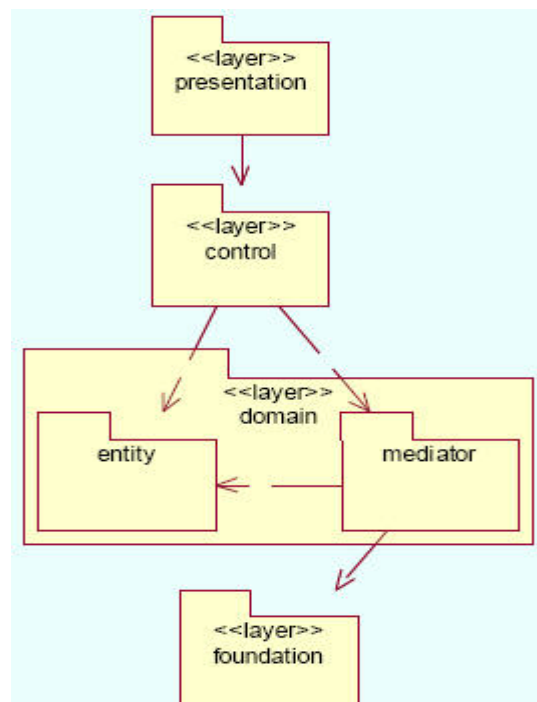
Application uses `desiredLocale` type resource bundle name and then, to be used resource bundle is determined by following code:

```
ResourceBundle resourcesBundle =  
ResourceBundle.getBundle(bundleName, desiredLocale);
```

## 2.4 PCMEF

PCMEF is layered paradigm that consists of presentation, control, domain and foundation layers (Maciaszek and Liong, 2004). Domain layer contained two packages: mediator and entity. PCMEF is based or enhanced form of Model – View – Controller (MVC) design pattern (Gamma et al, 1995).

The aim of PCMEF is minimizing package coupling, decreasing dependency and increasing stability with downward dependencies, higher layers depends on lower layers. Presentation depends on control, control depends on entity and mediator, and mediator depends on foundation as shown in Figure 2.2.



**Figure 2.2** PCMEF Framework

When upper layers are changed, lower layers are not affected; this provides loose coupling and allows programmer to build roundtrip architectural modeling (Maciaszek, 2005a; Maciaszek, 2005b).

The presentation layer contains classes used for Graphical User Interface (GUI). Examples of this layer in Java environment could be Swing, Applet, Java Server Faces classes. The classes in this layer are responsible for interacting users, showing results to the users and delivering actions received from the users to control layer.

The control layer receives requests from presentation layer and processes user interactions. This processing includes the application logic such as computations, algorithmic solution etc. During the processing, the layer can interact with the domain layer (entity and mediator) and the results will be returned to presentation layer.

Domain layer contains two layers, mediator and entity. Entity layer contains Plain Old Java Object (POJO); also it is known business objects. POJO is JavaBean that has a no-argument constructor, and also there are setter and getter methods for accessing its properties. The information stored in the POJO is the program data, and this data can be retrieved from databases, files or network.

The mediator layer sets up the communication channel between control, entity and foundation layers. In addition to isolating entity layer from the foundation layer, mediator also prevents the control layer classes from communicating directly with the foundation classes. This model is based on mediator design pattern (Gamma et al, 1995).

The communication between the program and the persistent data in database and files occurs in the foundation layer. The database transactions and queries from the program are handled in the foundation layer.

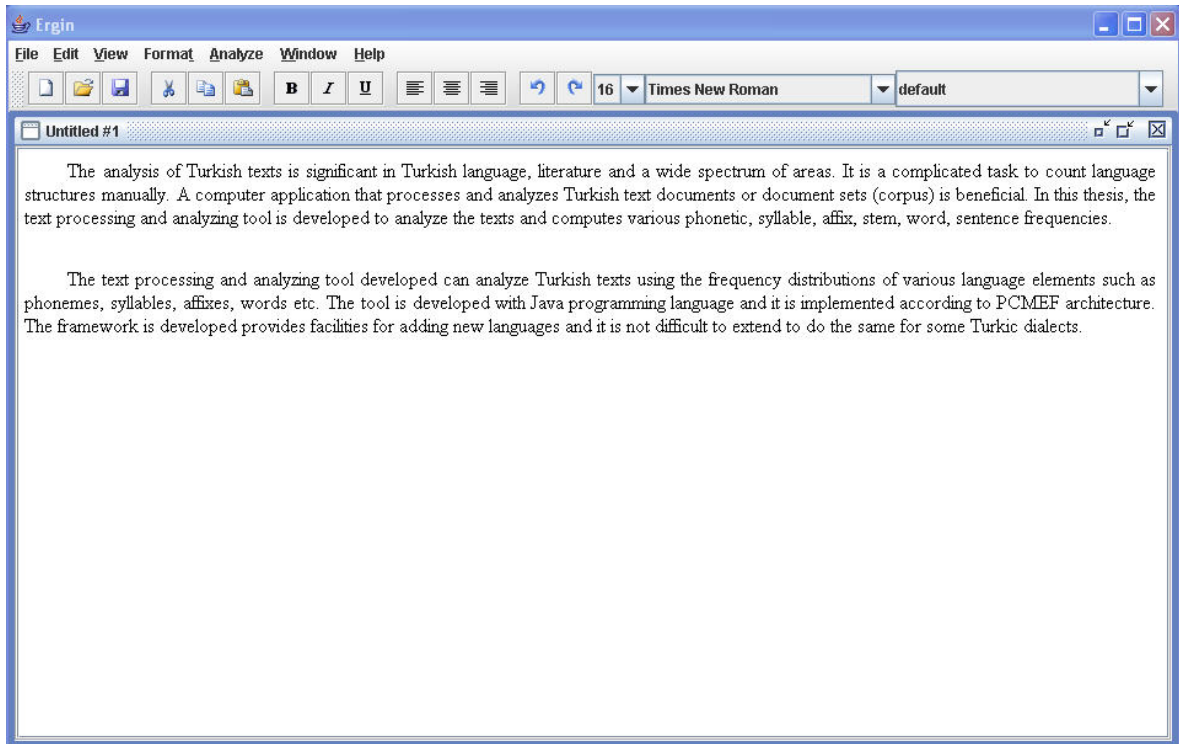
There are important factors in PCMEF framework that software developer must consider. The first factor is downward dependency of objects and upward notification. The objects are dependent to objects in the downward layers and the actions in down layer must call the upper layers by observers.

Another important factor is neighborhood communication. Each layer can communicate with its neighbor layer. Also naming must be ensured by adding the first character of each layer to the beginning of the names of the classes in that layer. In addition, all of the interfaces must be collected in acquaintance package.

## CHAPTER 3

### TEXT PROCESSING AND ANALYSIS TOOL

The text processing and analyzing tool for Turkish is an application that Turkish documents are processed easily. Besides including editor capabilities, it has also text analysis ability for characters, syllables, words, sentences and paragraphs in the given text. The graphical user interface (GUI) of the editor of the tool developed is shown in Figure 3.1.



**Figure 3.1** General GUI of developed tool

This chapter introduces the text processing and analysis tool and its GUI. This chapter is divided into several sections. In section 3.1, the text editor is introduced. The character analyzing property of the tool is presented in section 3.2. The frequency analysis of the syllables is introduced in section 3.3. In section 3.4, the frequency analysis of the words and section 3.5 the frequency analysis of the sentences are presented. The frequency analysis of the paragraphs is explained in section 3.6.

### **3.1 THE TEXT EDITOR**

The text processing and analyzing tool has many editor features in order to manage the documents. Since, the user of the program deals with the documents, the needs of the users about managing the documents must be satisfied completely. The editor capabilities of the developed tool must be advanced and there is no need to use another text editor with the tool. The core abilities of the text editor are based on Mila project (Mila, 2001) and some of them are changed by us.

All of the editors have characteristic properties like opening, saving and editing a document. The user interface of the editor of our tool is shown in Figure 3.2.

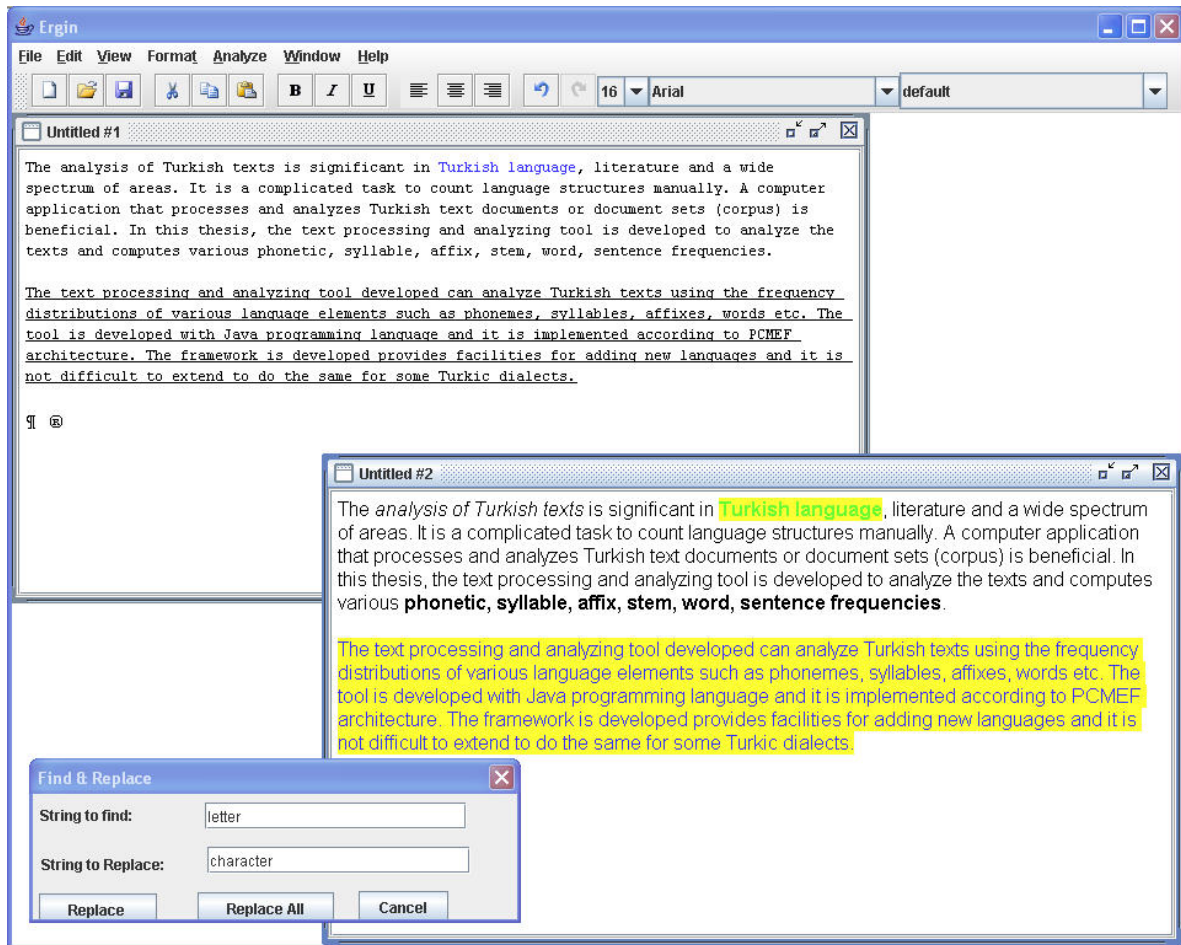


Figure 3.2 Editor GUI

The capabilities of the editor are:

- Open a new document
- Open an old document
- Close a document
- Save a document
- Print a document
- Exit from application
- Cut text
- Copy text
- Paste text

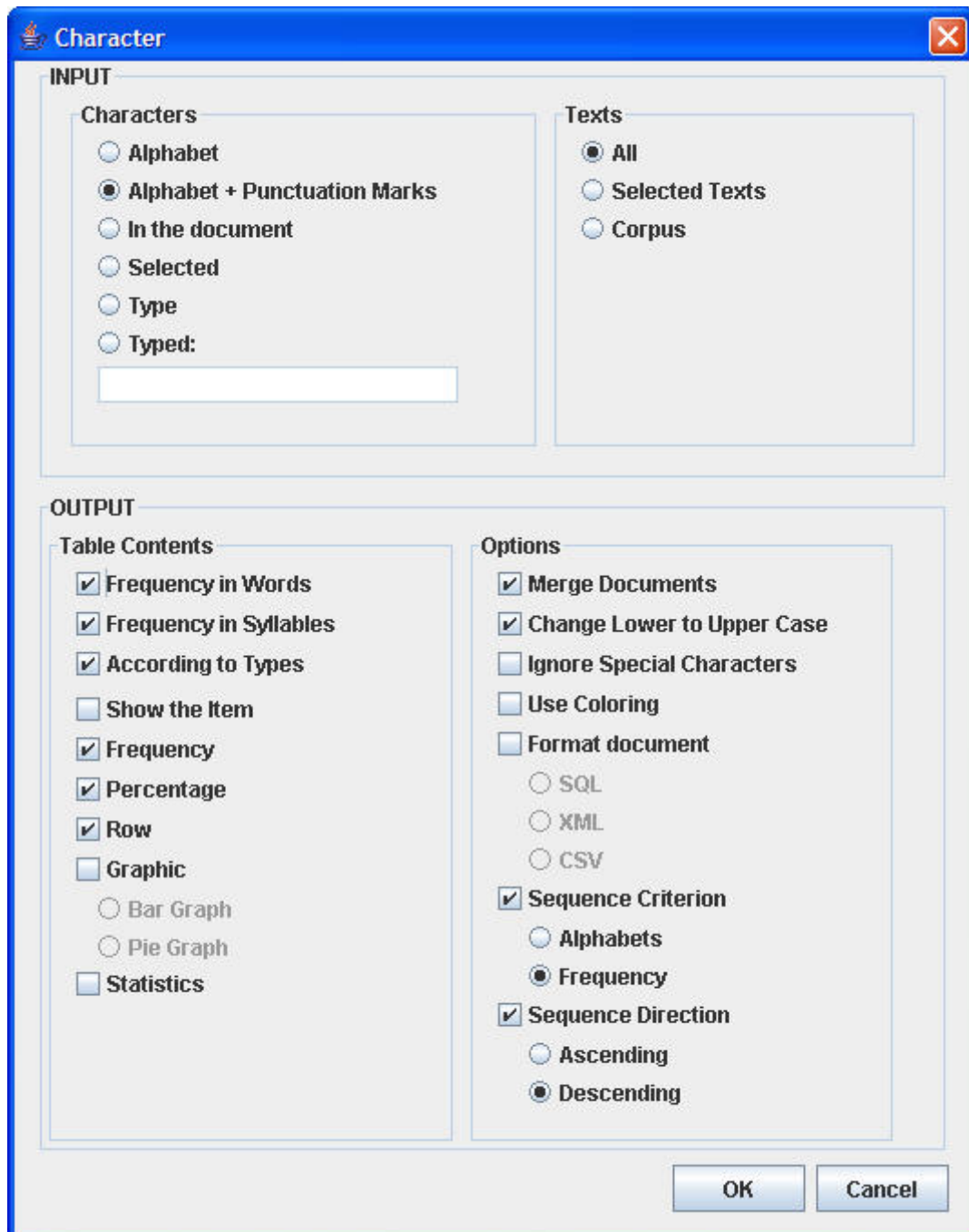
- Select all of the text
- Undo an action
- Redo an action
- Find text
- Find and replace text
- Make text bold
- Make text italic
- Make text underlined
- Align the text to right, left, center or justified
- Change the font of the selected text
- Change the size of the selected text
- Add symbols
- Bullets and numbering
- Change the foreground color of text
- Change the background color of text

### **3.2 THE FREQUENCY ANALYSIS OF CHARACTERS**

The text processing and analyzing tool can analyze the characters given in one or more documents. The information about frequency, ratio and percentage of letters in the input text are determined automatically.

The character analyzing performs functionalities according to the user's selection from the character in analyze menu. The GUI of the character window is shown in Figure 3.3.





**Figure 3.3** Character window

There is an input-output part in the character analyzing window that user determines the input text and output format. First of all, input part in the letter menu contains options for the characters and texts that are analyzed. The user can determine the characters that

will be analyzed. There are six choices for determining which letters to be processed. These choices are:

- Analyze all the letters in alphabet
- Analyze all the letters in the alphabet and punctuation marks
- Analyze the letters in the analyzed text document
- Analyze the selected letters
- Analyze letters according to their types
- Analyze the letters typed in the text box in the window

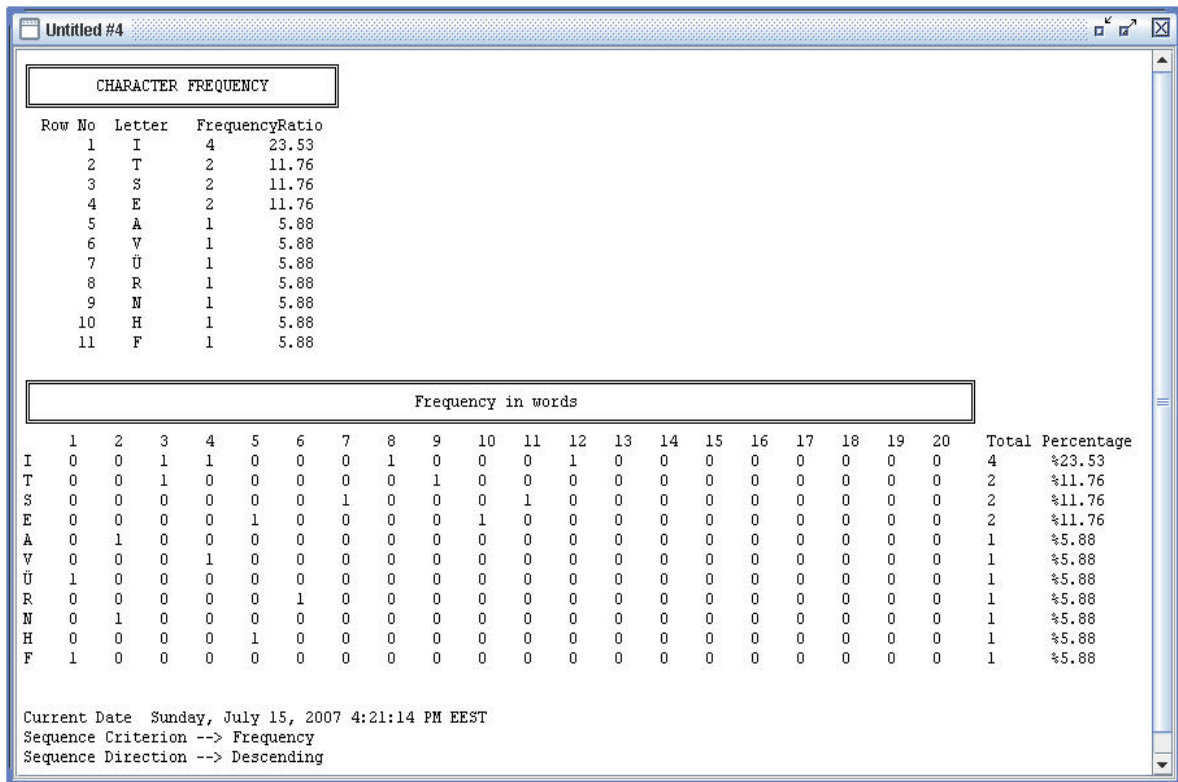
User can also select which text to analyze. There can be one or more than one text and at the same time user may want to analyze all of the texts. Also user may want to analyze only a small part of the text. For these reasons, there is a text part in the input part to determine which text to analyze. In the text part there are 2 options. The first option is “All” that processes all of the texts in the opened documents. The second one is “Selected texts” that processes only the selected text in a document.

Another important part in the letter menu is output. User can determine which characteristics of the text will be processed for output. The characteristics are the frequency, ratio, type and statistics.

The options in the output part for table contents are:

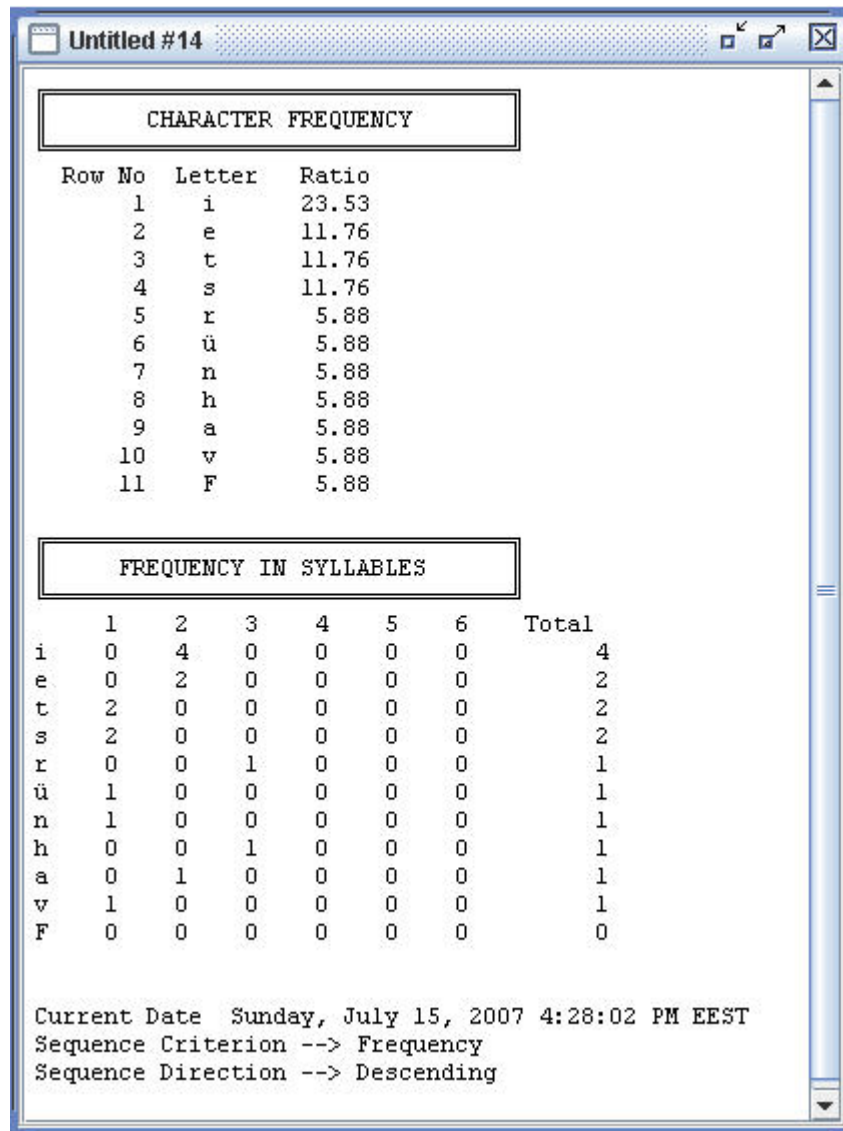
- Frequency of letters in words
- Frequency of letters in syllables
- According to types
- Show the item
- Percentage of letters
- Row
- Statistics

Frequency of a letter in the words is calculated by counting the number of occurrences of letters in text according to their order. For example, if we give the text “Fatih Üniversitesi” as input to the program it calculates frequency of each letter in the text and shows the results as in Figure 3.4. The frequency of letter “i” is one time as the third, fourth, eighth and twelfth character according to the word frequency. The frequency of letter “e” is one time as the fifth and tenth character according to the word frequency.



**Figure 3.4** Frequency of letters in words in “Fatih Üniversitesi” text

Frequencies of letters in syllables are calculated by counting the number of occurrences of letters in each syllable. If again “Fatih Üniversitesi” text is given as input, it calculates and shows the frequencies of letters in the syllables (Figure 3.5). For example, the frequency of letter “i” in the text is 4 as the second letter in the syllables. This means that letter “i” is found 4 times as the second letter in the syllables of the given text.



**Figure 3.5** Frequency of letters in syllables in “Fatih Üniversitesi” text

Ratio of a letter is computed by dividing frequency of a letter to total of all frequencies in text. For the text “Fatih Üniversitesi”, the frequency of letter “i” is total 4 and the total frequencies of the letters (including space) is 18. The ratio of “i” is found by dividing 4 by 18 and it is equal to 23,53 (Figure 3.6).

CHARACTER FREQUENCY			
Row No	Letter	Frequency	Ratio
1	I	4	23.53
2	E	2	11.76
3	S	2	11.76
4	T	2	11.76
5	A	1	5.88
6	F	1	5.88
7	H	1	5.88
8	N	1	5.88
9	R	1	5.88
10	Ü	1	5.88
11	V	1	5.88
12	B	0	0.00
13	C	0	0.00
14	Ç	0	0.00
15	D	0	0.00
16	G	0	0.00
17	Ğ	0	0.00
18	İ	0	0.00
19	J	0	0.00
20	K	0	0.00
21	L	0	0.00
22	M	0	0.00
23	O	0	0.00
24	Ö	0	0.00
25	P	0	0.00
26	Ş	0	0.00
27	U	0	0.00
28	Y	0	0.00
29	Z	0	0.00

**Figure 3.6** Letter frequency output of the text “Fatih Üniversitesi”

Type of a letter is related to consonants and vowels’ type which is given to the Table 2.1 and Table 2.2. With this choice all of the consonants and vowels’ types are calculated. For example for the text “Faith Üniversitesi” the types of the letters are shown in Figure 3.7.

WORD FREQUENCY ACCORDING TO TYPES

Number of Vowel Characters	:	8
Number of Consonant Characters	:	9
Wovel --> Kalın	:	5
Wovel --> Ince	:	3
Wovel --> Duz	:	7
Wovel --> Yuvarlak	:	1
Wovel --> Genis	:	3
Wovel --> Dar	:	5
Consonant --> Sedalı	:	3
Consonant --> Sedasız	:	6
Consonant --> Surekli	:	7
Consonant --> Sureksiz	:	2
Consonant --> Akıcı	:	2
Consonant --> Sızıcı	:	5
Consonant --> Patlayıcı	:	2
Consonant --> Dudak	:	0
Consonant --> Diş Dudak	:	2
Consonant --> Diş	:	5
Consonant --> Diş Dudak	:	0
Consonant --> Ön Damak	:	1
Consonant --> Art Damak	:	0
Consonant --> Gırtlak	:	1

Current Date Sunday, July 15, 2007 4:30:27 PM EEST  
Sequence Criterion --> Frequency  
Sequence Direction --> Descending

**Figure 3.7** Character types output of the text “Fatih Üniversitesi”

In addition, it is required to remove some special characters in a text for example tab and space characters. In statistic choice, total number of characters is calculated and then; number of characters that are processed after removing special characters in text and etc. Use coloring option, changes the color of the output lines.

Constraints option is related to the outputs, results can be saved in different formats such as XML, CSV and SQL. Also the results can be listed according to frequency ratio or alphabet as increasing or decreasing order, and also these choices depend on the user’s

selection. A sample of result window can be shown in Figure 3.8, output are listed as frequency decreasing order.

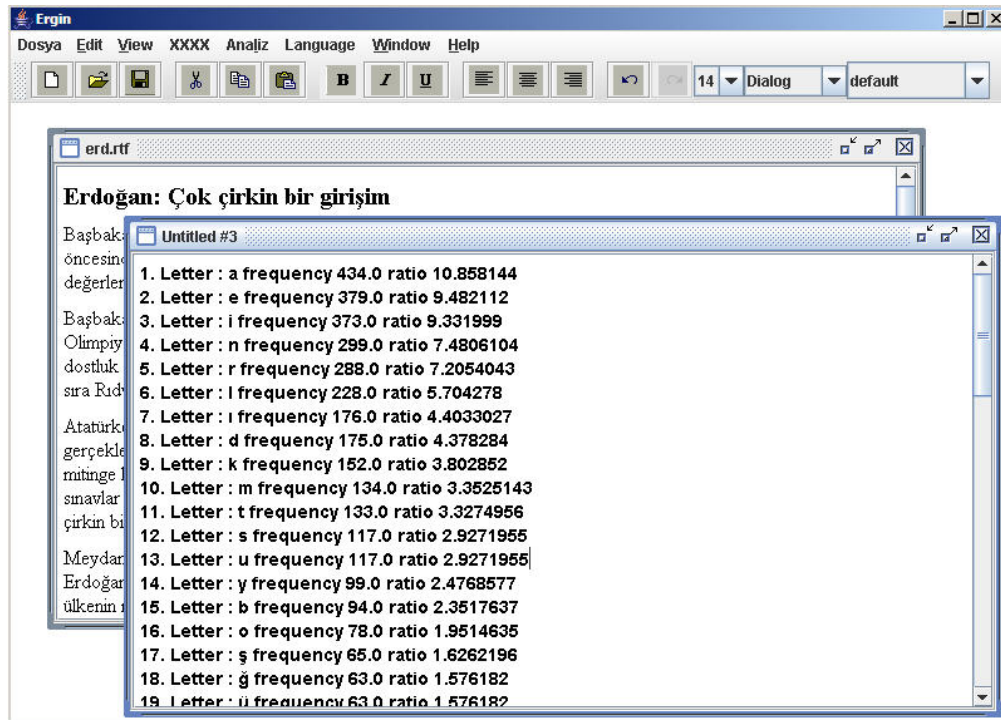
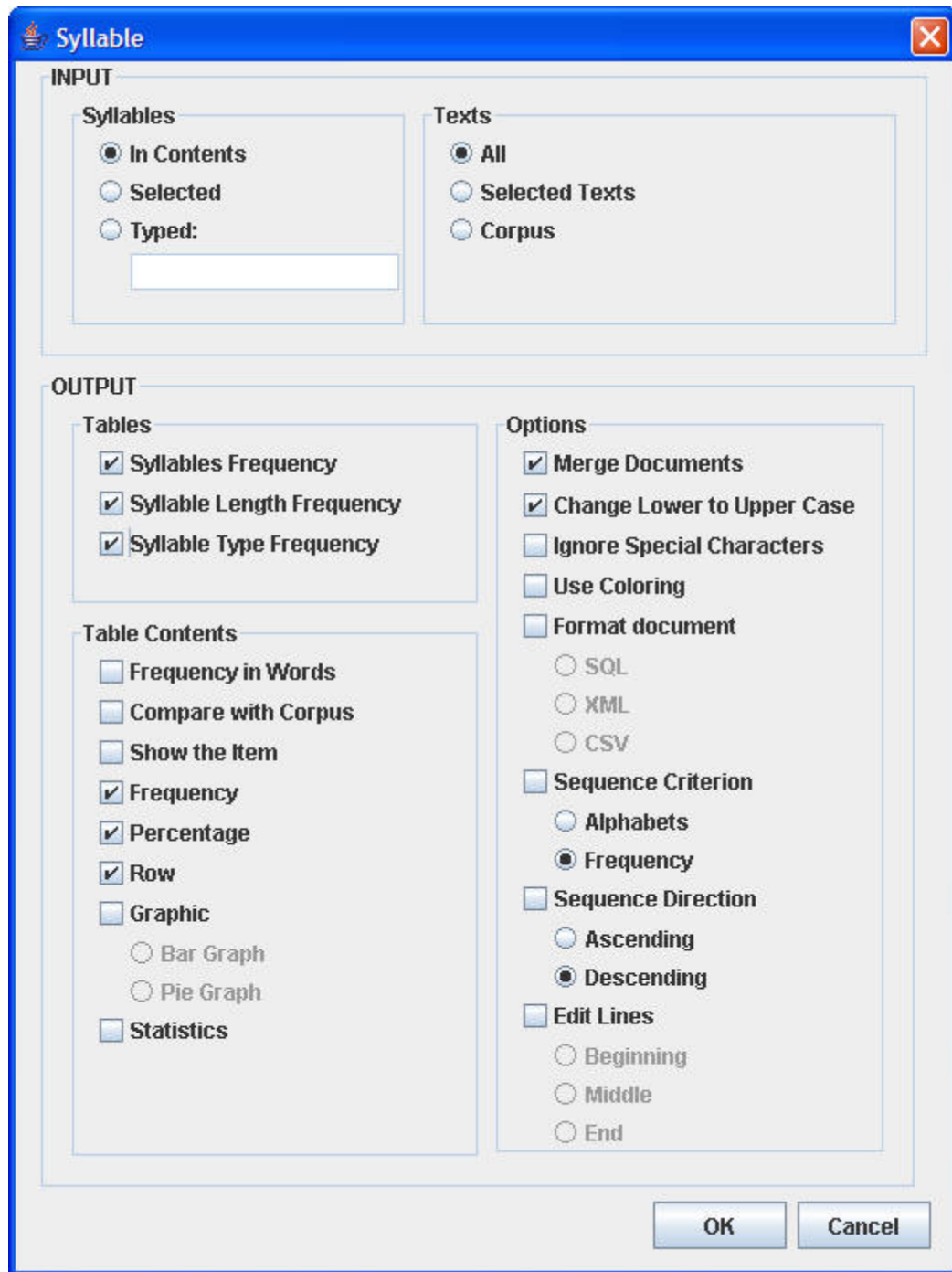


Figure 3.8 Output window of letter analyzing

### 3.3 THE FREQUENCY ANALYSIS OF SYLLABLES

The text processing and analyzing tool is capable of analyzing the syllables in the documents. The window of the syllable analyzing is similar to the character analyzing and it is opened from analyze menu. The GUI of the syllable menu is shown in Figure 3.9.



**Figure 3.9** The syllable menu

There is an input and output part in the syllable window. Input part contains options for which syllables and documents to analyze. There are 3 options to select which syllables. They are analyze the:



- Syllables in contents
- Only the selected syllables
- Syllables entered to the text box in the window

There are 2 options to select in which texts to analyze the syllables. They are:

- All of the texts
- Selected texts

Output part in the syllable menu allows user format the output. User can determine the analyzed features and define the constraints for the output.

Frequency of syllables, frequency of length of syllables and frequency of syllable types can be selected as output table. Frequency of syllables is calculated by counting the number of each syllable. For example in the text “Fatih üniversitesi” the frequency of the syllable “si” is 2. The frequency of length of syllables is calculated by counting the number of syllables according to their length and order in the words. For example there are 2 syllables in “Fatih üniversitesi” text that has 3 letters length. These syllables are “tih” and “ver”. The syllable “tih” is the second syllable and “ver” is the third syllable. The output of frequency of syllables, frequency of length of syllables and frequency of syllable types for the text “Fatih üniversitesi” is shown in Figure 3.10.

Untitled #17

SYLLABLE FREQUENCY

Row No	Syllable	Frequency	Ratio
1	si	2	25.00
2	ver	1	12.50
3	ü	1	12.50
4	ni	1	12.50
5	fa	1	12.50
6	tih	1	12.50
7	te	1	12.50

SYLLABLE TYPE FREQUENCY IN WORDS

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	Total
Syllable Type V	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
Syllable Type VC	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Syllable Type CV	1	1	0	1	1	1	0	0	0	0	0	0	0	0	0	5
Syllable Type CVC	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	2
Syllable Type VCC	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Syllable Type CVCC	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SYLLABLE LENGTH FREQUENCY IN WORDS

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	Total
1 letters	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
2 letters	1	1	0	1	1	1	0	0	0	0	0	0	0	0	0	5
3 letters	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	2
4 letters	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5 letters	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6 letters	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SYLLABLE TYPE FREQUENCY

Syllable Ty V	Frequency	1
Syllable Ty CVC	Frequency	2
Syllable Ty CV	Frequency	5

Current Date Sunday, July 15, 2007 5:23:34 PM EEST  
Sequence Criterion --> Frequency  
Sequence Direction --> Descending

**Figure 3.10** The syllable output for “Fatih üniversitesi”

The frequencies of syllable types are calculated according to types of syllables. As presented in section 2.1.2 there are 6 types of syllables in Turkish. Syllables are counted and classified according to their types as shown in Figure 3.6. There is only 1 syllable that has one vowel “ü” in “Fatih üniversitesi” text.

The ratios of the syllables are calculated by dividing the frequency of each syllable to the total syllable number. In “Fatih üniversitesi” text there are total 8 syllables and the

frequency of the syllable “si” is 2. The ratio of the syllable “si” is equal to 2/8 (% 25). The output of the syllable ratios are shown in Figure 3.11

SYLLABLE FREQUENCY	
Syllable	Ratio
si	25.00
ver	12.50
ü	12.50
ni	12.50
tih	12.50
fa	12.50
te	12.50

**Figure 3.11** Ratio of syllables in “Fatih üniversitesi”

Zemberek NLP library is used for separate words to their syllables (Zemberek, 2006). in the output part there are some constraints that are defined by user. They are:

- Merge documents
- Change lower case to uppercase letters
- Ignore special characters
- Use coloring
- Order according to alphabet or frequency
- Order ascending/descending

Analyzing all documents together (merge documents) provides processing the open documents together and getting only one output for all the texts. If this option is not selected, each document is analyzed individually and for each document output is displayed. Convert lowercases to uppercase letters ensures ignoring the case sensitivity in the syllables. Ignoring special characters provides not to analyze the special characters. Coloring option ensures lines in different colors. An example is shown in Figure 3.12. The syllables can be ordered ascending or descending from the ordering option.

Untitled #22

SYLLABLE FREQUENCY		
Row No	Syllable	Frequency
1	si	2
2	ver	1
3	ü	1
4	ni	1
5	fa	1
6	tih	1
7	te	1

SYLLABLE TYPE FREQUENCY IN WORDS																
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	Total
Syllable T V	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
Syllable T VC	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Syllable T CV	1	1	0	1	1	1	0	0	0	0	0	0	0	0	0	5
Syllable T CVC	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	2
Syllable T VCC	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Syllable T CVCC	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

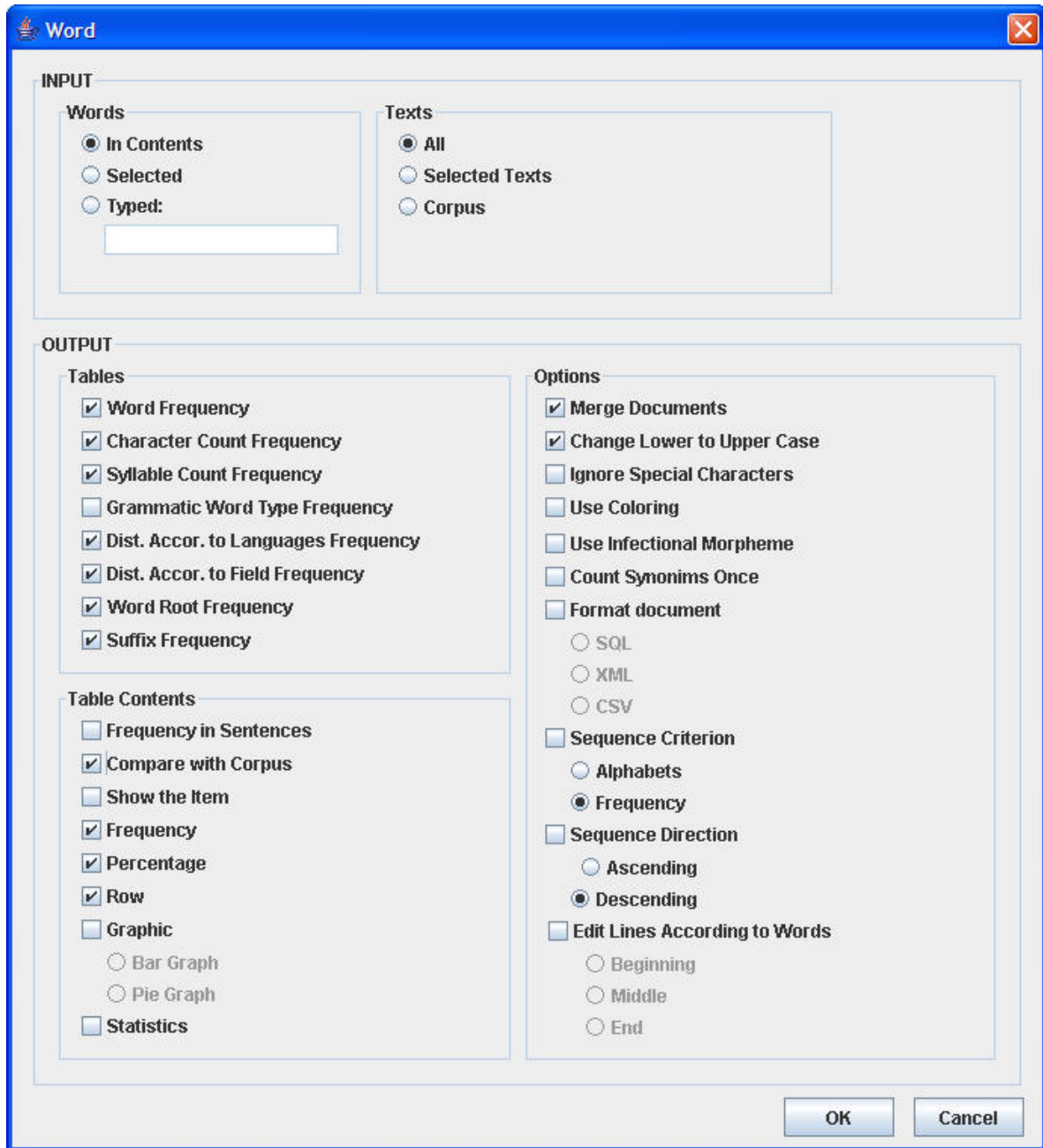
SYLLABLE LENGTH FREQUENCY IN WORDS																
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	Total
1 letters	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
2 letters	1	1	0	1	1	1	0	0	0	0	0	0	0	0	0	5
3 letters	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	2
4 letters	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5 letters	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6 letters	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Current Date Sunday, July 15, 2007 5:48:23 PM EEST  
Sequence Criterion --> Frequency  
Sequence Direction --> Descending

Figure 3.12 Colored ratio of syllables in “Fatih üniversitesi” text

### 3.4 THE FREQUENCY ANALYSIS OF WORDS

Analyzing the words is an important issue in text processing. The frequencies, ratios, statistics of the words can be calculated using word analyzing window by opening from analyze menu. The menu of the word analyzing is shown in Figure 3.13.



**Figure 3.13** Word Analyzing menu

Like letter and syllable menu, word analyzing menu consists of input and output parts. In the input part, words and texts that will be analyzed are chosen by the user. The all texts or the selected texts may be analyzed. The choices are:

- Analyze all of the words in the document
- Analyze the selected words

- Analyze the words entered into the text box

To choose the documents that will be analyzed there are 3 options. First one is analyzing the selected texts; other is for analyzing all of the texts. Third one is the corpus. The corpus is created by İlyas Göz (Göz, 2003). The analyzed words in the documents are also compared with the corpus. The corpuses for letters, syllables can be added as future work to the tool.

The tables in the output part are:

- Word frequency
- Character count frequency
- Syllable count frequency
- Word root frequency
- Suffix frequency
- Word maximal root frequency

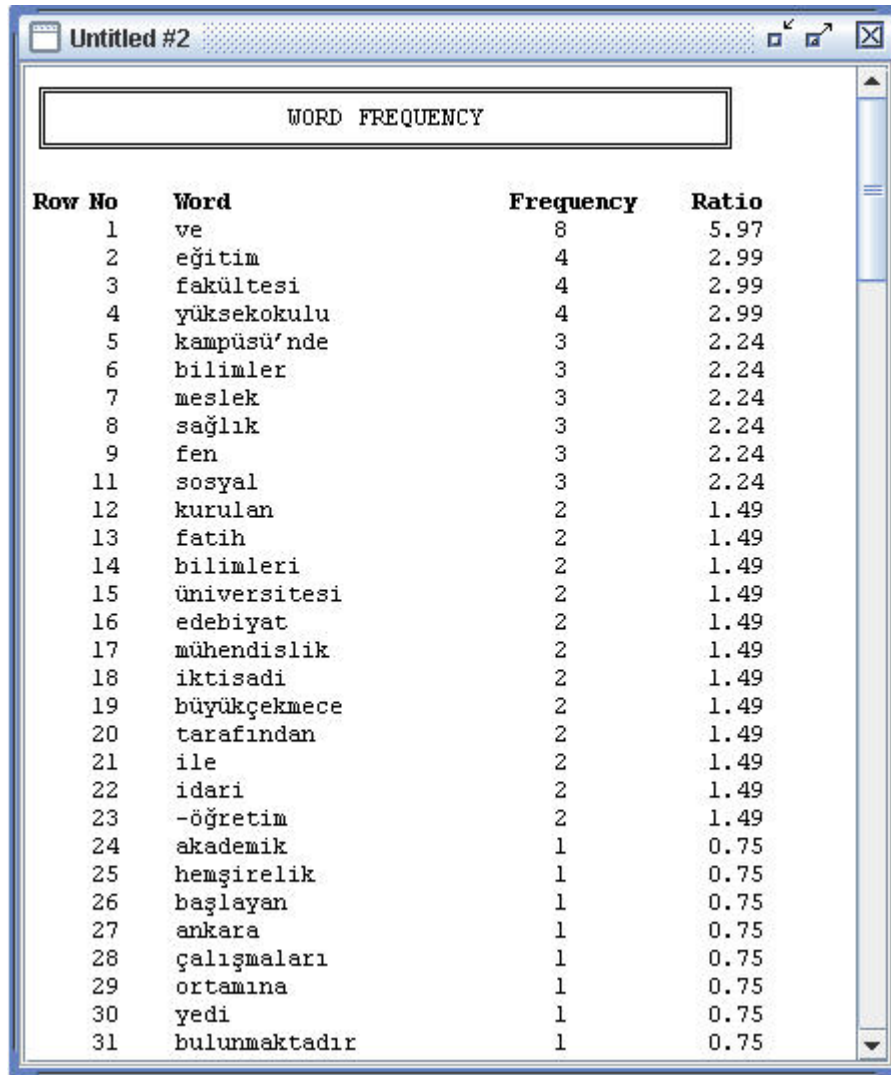
The text below is analyzed for word frequency, word count frequency and syllable count frequency. The text analyzed is:

“Türkiye Sağlık ve Tedavi Vakfı tarafından kurulan Fatih Üniversitesi, 18.11.1996 tarihinde Dokuzuncu Cumhurbaşkanımız Sayın Süleyman Demirel tarafından eğitim - öğretime açılmıştır. On yedi üyesi bulunan Mütevelli Heyeti ile yönetilmektedir. Üniversitemiz, Büyükçekmece Kampüsü’nde Fen - Edebiyat, İktisadi ve İdari Bilimler, Mühendislik Fakülteleri, Fen ve Sosyal Bilimler Enstitüleri ve İstanbul Meslek Yüksekokulu; Ostim Kampüsü’nde Tıp Fakültesi, Sağlık Bilimleri Enstitüsü, Hemşirelik Yüksekokulu, Sağlık Bilimleri Meslek Yüksekokulu ve Ankara Meslek Yüksekokulu ile eğitim – öğretim faaliyetlerini sürdürmektedir.

1997 – 1998 akademik yılında Büyükçekmece Kampüsü’nde eğitim-öğretime başlayan Fatih Üniversitesi; Fen - Edebiyat Fakültesi, İktisadi ve İdari Bilimler Fakültesi, Mühendislik Fakültesi sosyal tesisleri ve öğrenci yurtlarıyla modern bir eğitim ortamına sahiptir. Sosyal tesis binasında kütüphane, sinema salonu, kafeterya, yemekhane, kitabevi, kırtasiye, terzi, kuaför ve internet kafe bulunmaktadır. Fakültelerin bünyesinde kurulan laboratuvarlarda eğitim – öğretim faaliyetlerinin yanı sıra araştırma çalışmaları da sürdürülmektedir.“

Word frequency is the number of the words in the analyzed text. Ratio is calculated by dividing the frequency of the word to the total word count in the text. For example in the

given text the word “ve” is found 5 times. The word frequency output of the text above is shown in Figure 3.14.



Row No	Word	Frequency	Ratio
1	ve	8	5.97
2	eğitim	4	2.99
3	fakültesi	4	2.99
4	yüksekokulu	4	2.99
5	kampüsü'nde	3	2.24
6	bilimler	3	2.24
7	meslek	3	2.24
8	sağlık	3	2.24
9	fen	3	2.24
11	sosyal	3	2.24
12	kurulan	2	1.49
13	fatih	2	1.49
14	bilimleri	2	1.49
15	üniversitesi	2	1.49
16	edebiyat	2	1.49
17	mühendislik	2	1.49
18	iktisadi	2	1.49
19	büyükçekmece	2	1.49
20	tarafından	2	1.49
21	ile	2	1.49
22	idari	2	1.49
23	-öğretim	2	1.49
24	akademik	1	0.75
25	henşirelik	1	0.75
26	başlayan	1	0.75
27	ankara	1	0.75
28	çalışmaları	1	0.75
29	ortamina	1	0.75
30	yedi	1	0.75
31	bulunmaktadır	1	0.75

**Figure 3.14** Word frequency output

Word count frequency gives the number of characters in the words and syllable count frequency gives the count of syllables in the words. For example in the given text the words that have 16 letters are at 3 times. The output window of word count frequency and syllable count frequency is shown in Figure 3.15. Word root frequency and maximal root frequency are calculated according to another program (Shylov, 2007).

The image shows a screenshot of a software window titled "Untitled #2". Inside the window, there are two tables of frequency data. The first table is titled "WORD COUNT FREQUENCY" and the second is titled "SYLLABLE COUNT FREQUENCY".

WORD COUNT FREQUENCY			
16	letters	frequency	3
15	letters	frequency	3
14	letters	frequency	2
13	letters	frequency	2
12	letters	frequency	5
11	letters	frequency	13
10	letters	frequency	5
9	letters	frequency	17
8	letters	frequency	18
7	letters	frequency	7
6	letters	frequency	20
5	letters	frequency	11
4	letters	frequency	6
3	letters	frequency	7
2	letters	frequency	12

SYLLABLE COUNT FREQUENCY			
7	syllables	frequency	4
6	syllables	frequency	6
5	syllables	frequency	12
4	syllables	frequency	36
3	syllables	frequency	30
2	syllables	frequency	23
1	syllables	frequency	15

**Figure 3.15** Word & syllable count frequency



SUFFIX FREQUENCY	
Suffix	Frequency
yA	2
nHn	1
Hn	2
ndA	2
lA	1
yH	1
lHk	1
sH	3
yAn	1
lAr	2

**Figure 3.16** Suffix frequency

WORD ROOT FREQUENCY	
Word Root	Frequency
hemşire	1
bünye	1
fen	3
baş	1
faaliyet	2
idari	2
edebiyat	2
heyet	1
bina	1
il	2
bir	1

**Figure 3.17** Word root frequency

LETTER COUNT	
Letters	Frequency
16	3
15	3
14	2
13	2
12	5
11	13
10	5
9	17
8	18
7	7
6	20
5	11
4	6
3	7
2	12
1	3

**Figure 3.18**– Letter count frequency

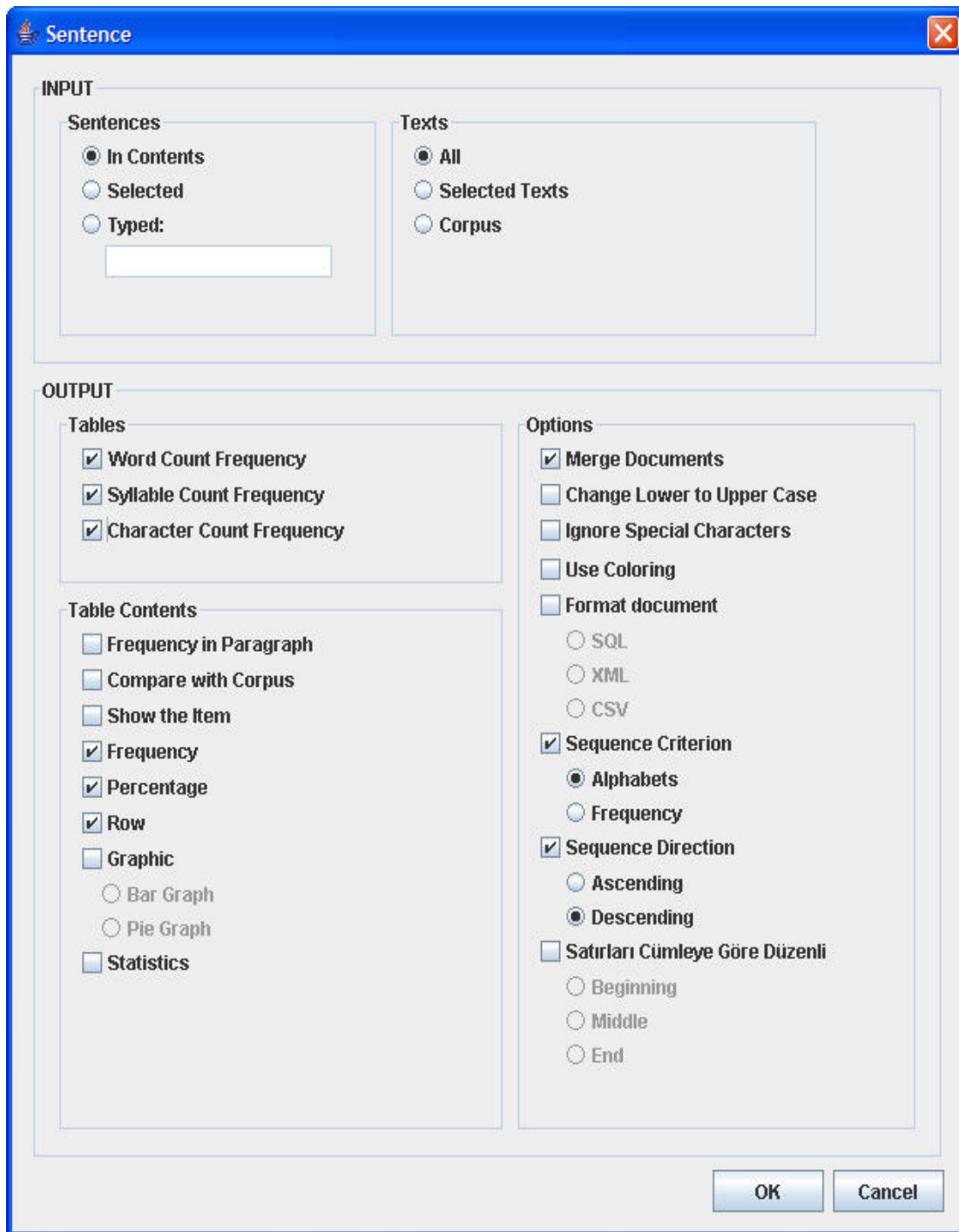
According to Language Frequency		
Word	Frequency	Language
ve	8	arapça
eğitim	4	
fakültesi	4	
yüksekokulu	4	
kampüsü'nde	3	
bilimler	3	
meslek	3	arapça
sağlık	3	
fen	3	arapça
sosyal	3	fransızca
kurulan	2	
fatih	2	arapça
bilimleri	2	
üniversitesi	2	
edebiyat	2	arapça
mühendislik	2	
iktisadi	2	arapça
büyükçekmece	2	
tarafından	2	
ile	2	
idari	2	arapça
-öğretim	2	
akademik	1	fransızca

**Figure 3.19** Words according to language frequency

### **3.5 THE FREQUENCY ANALYSIS OF SENTENCES**

Analyzing the sentences can give information about many things including words, syllables and letters. The program we developed processes the sentences and can analyze the frequencies, ratios and percentage.

The sentence window is shown in Figure 3.20. Sentence window has input and output parts like others. Input part has two parts sentences and texts. The sentences in the documents, selected sentences or the sentences entered in the text box can be chosen to be analyzed. The all texts, the selected texts or the texts in the corpus can be analyzed.



**Figure 3.20** Sentence window

The input part is very similar to the input parts in the syllables and word windows. The word count frequency, syllable count frequency and letter count frequency can be selected for analyzing. Also, the frequency in the paragraph, frequency and ratio can be calculated.

The tables in the output part are:

- Sentence frequency
- Word count frequency
- Syllable count frequency
- Character count frequency

For example, if we give same text in word section to the program as input, it produces output like Figure 3.21. Every sentence in text exists only once, and also word count frequency can be shown in bottom of the Figure 3.15. In the example, there are two sentences that they contain 21 words.

SENTENCE FREQUENCY	
Sentence	Frequency
Üniversitemiz, Büyükçekmece K	1
Türkiye Sağlık ve Tedavi Vakfı	1
Sosyal tesis binasında kütüph	1
Ostim Kampüsü'nde Tıp Fakülte	1
On yedi üyesi bulunan Mütewel	1
Fen - Edebiyat Fakültesi, İkt	1
Fakültelerin bünyesinde kurul	1
1997 -1998 akademik yılında	1
1996 tarihinde Dokuzuncu Cumhu	1

WORD COUNT FREQUENCY	
Word Frequency	Frequency
8	1
22	1
21	2
16	1
13	1
12	1
10	2
1	1

**Figure 3.21** Word count frequency

Syllable count frequency and character count frequency results are shown in Figure 3.16. Syllable count frequency calculates syllables count in the sentences. For example in

Figure 3.22, there are 9 sentences and all of them have different number of syllables. Also this situation is same for character count frequency option, there are 9 sentences and all of them have different number of characters.

SYLLABLE COUNT FREQUENCY	
Syllable Frequenc	Frequency
74	1
63	1
59	1
52	1
50	1
37	1
34	1
26	1
24	1

Letter Frequency	
Letter Frequency	Frequency
93	1
87	1
62	1
51	1
2	1
175	1
150	1
139	1
120	1
114	1

**Figure 3.22** Syllable count frequency

### 3.6 THE FREQUENCY ANALYSIS OF PARAGRAPHS

Paragraphs include many structures like sentences, words, syllables and letters. Analyzing a paragraph gives information about all of these structures. The program we developed processes the paragraphs and analyzes the frequencies, ratios and percentages of the paragraphs.

The paragraph window consists of input and output parts like the other windows (Figure 3.23). The frequencies of paragraphs and the number of sentences can be analyzed. Also sorting the output, finding the ratio and percentages are possible.

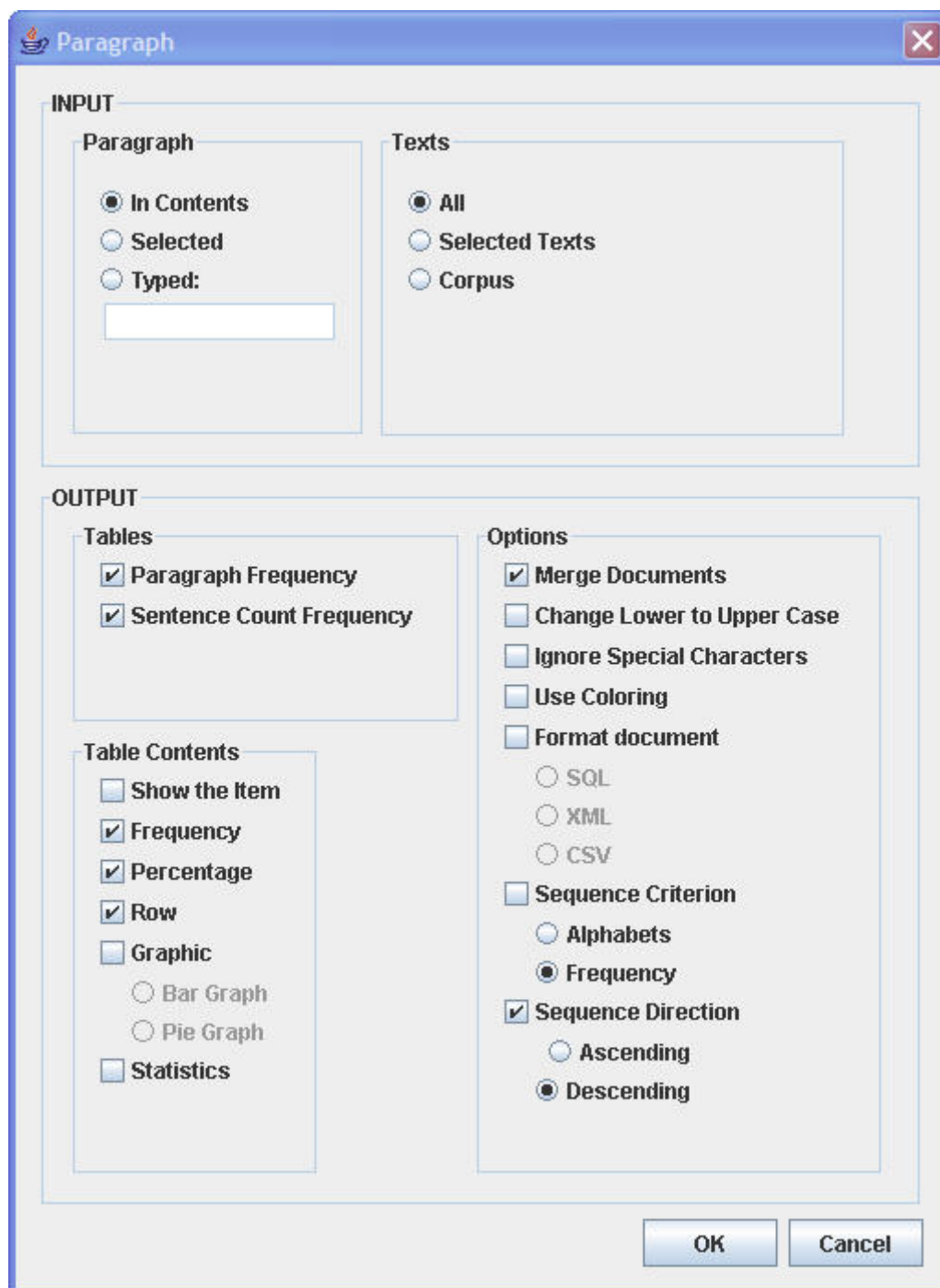


Figure 3.23 Paragraph analyzing window

The tables in the output part are:

- Paragraph frequency
- Sentence count frequency
- Word count frequency
- Syllable count frequency
- Character count frequency

The results for same input with word and sentence parts are shown on Figure 3.24. For example, paragraph count frequency result has two paragraphs, and one of them have 7 sentences and the other have 4 sentences. And also, word count, syllable count and character count are given in same table.



PARAGRAPH COUNT FREQUENCY
---------------------------

Paragraph	Frequency	Sentence C	Word Count	Syllable C	Letter Cou
1997 -1998 akademik	1	4	60	195	460
Türkiye Sağlık ve Te	1	7	74	224	533

SENTENCE COUNT FREQUENCY
--------------------------

Sentence Count Frequency	
7	1
4	1

WORD COUNT FREQUENCY
----------------------

Word Count Frequency Frequency	
74	1
60	1

SYLLABLE COUNT FREQUENCY
--------------------------

Syllable Count Frequency Frequency	
195	1
224	1

LETTER COUNT FREQUENCY
------------------------

Letter Count Frequency	Frequency
460	1
533	1

**Figure 3.24** Sample output of paragraph

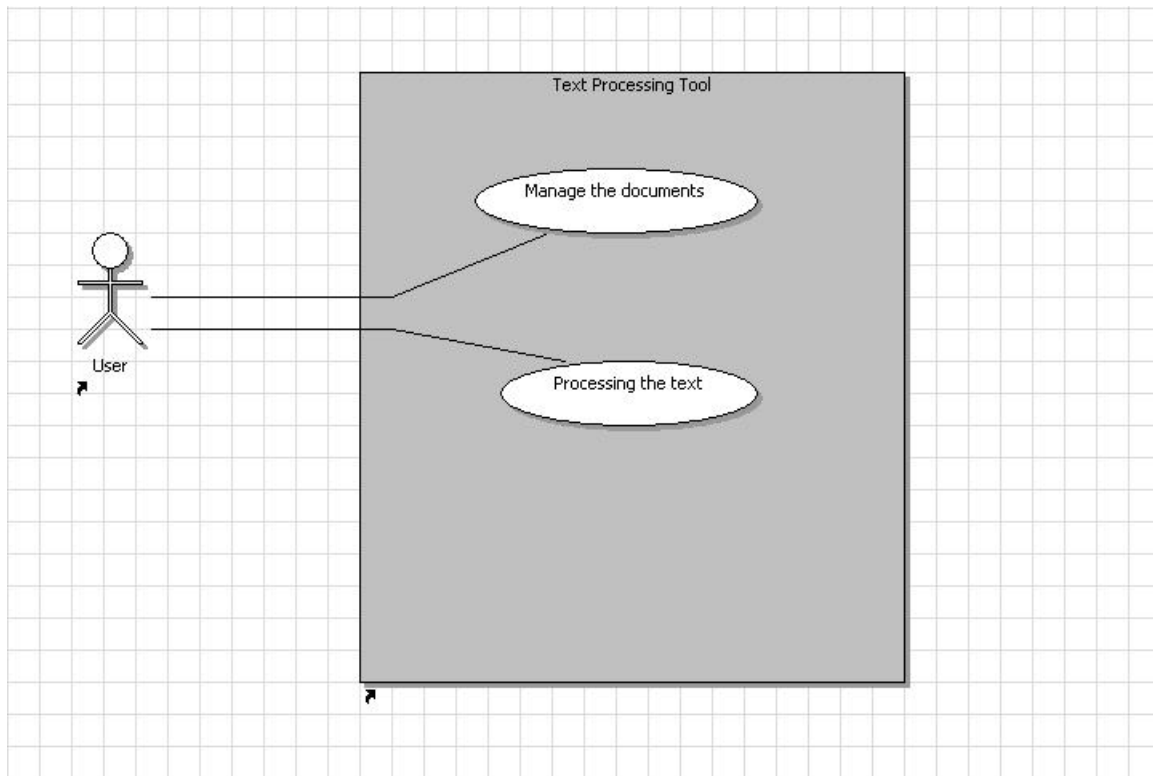
## **CHAPTER 4**

### **TOOL OBJECT MODEL**

The text processing and analyzing tool is developed with Java programming language. The object models of the tool are presented in this chapter. Section 4.1 presents the use case diagrams of the program. Section 4.2 presents packages and class diagrams and the sequence diagrams are presented in section 4.3.

#### **4.1 USE CASE DIAGRAM**

Use case diagram shows the general functional requirements of the programs. The use case diagram of the text processing and analyzing tool is shown in Figure 4.1. There are 2 basic functions in the use case. One is managing the documents and other is processing the text.



**Figure 4.1** Use case diagram of the text processing tool

Managing the document case: In managing the document case, user of the program can manage the documents by opening, changing or saving the documents to the hard disk of the computer. This use case is related with the text editor part of the program, so it can be used for editor abilities such as changing fonts, highlighting the text , resizing the text font size etc.

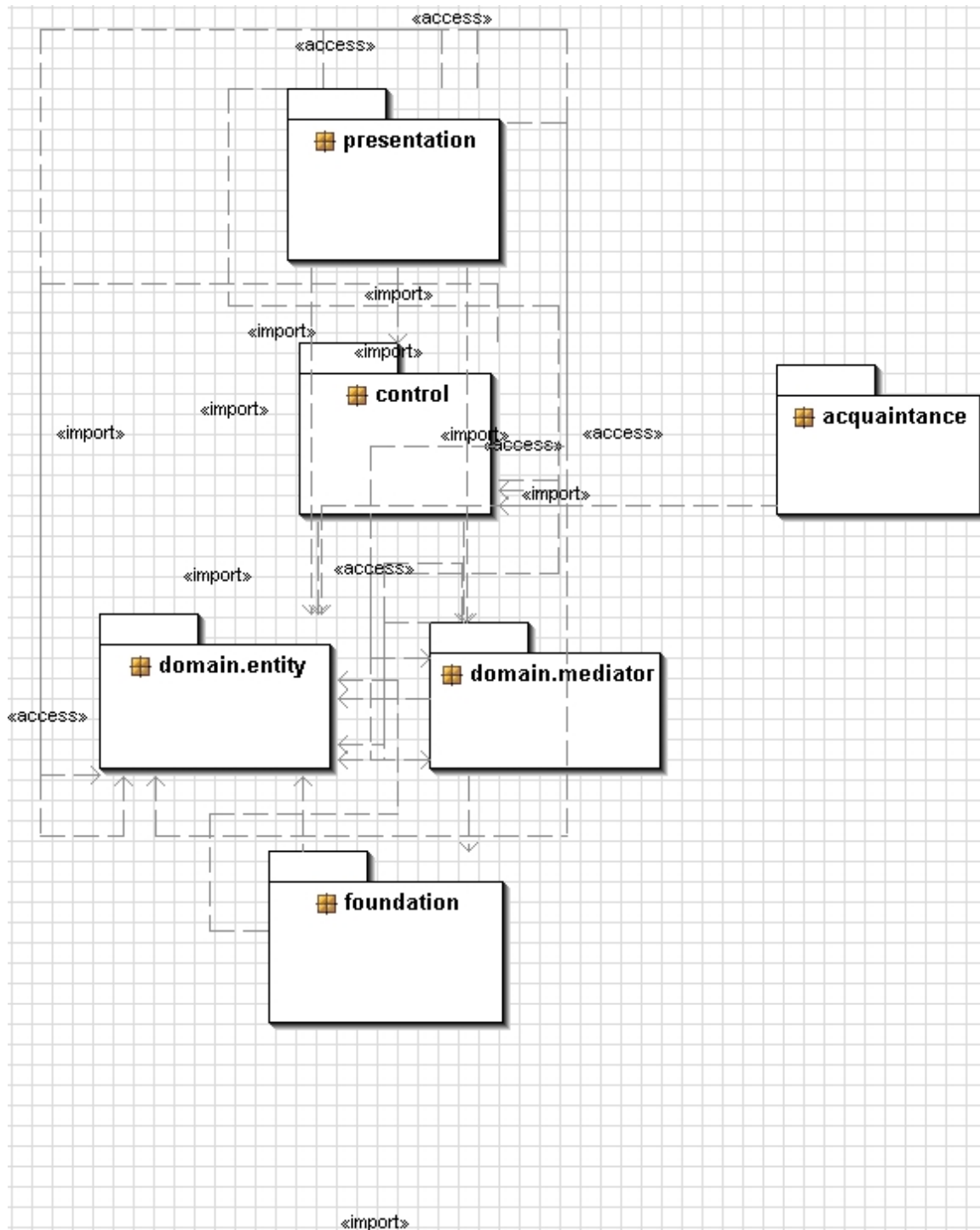
Processing the text case: Processing the text case is the most important part of the program. Frequency analysis, ratio analysis and the important statistics about characters, syllables, words, sentences and paragraphs are determined by program for text processing. Multi documents can be processed by user, and results can be saved.

## 4.2 PACKAGES AND CLASS DIAGRAMS

Packages diagram and dependencies between the packages are be shown in Figure 4.2. This software project are developed according to PCMEF (Presentation, Control, Mediator, Entity and Foundation) meta framework principles. Presentation package

depends on control layer, and control layer depends on domain layer, and domain layer depends on foundation layer.

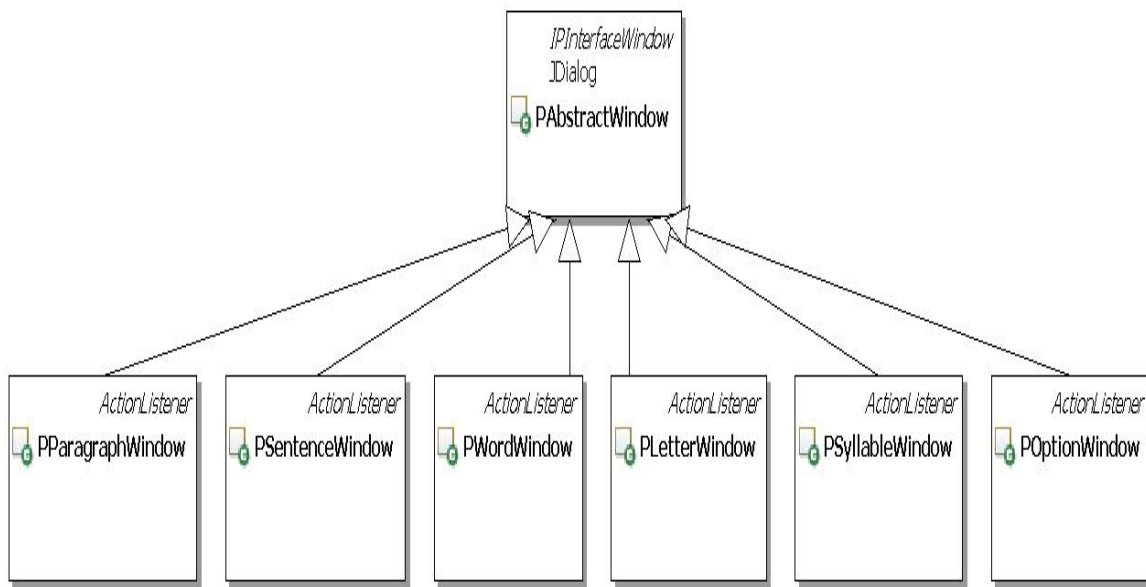
In addition to these dependencies, all interfaces are built on acquaintance package. Text processing software running on local computer and it is developed with Java Swing technology for creating GUI. In the future, this software can be run on internet with using on of the Servlet, JSP or JSF technologies. The advantage of running on internet is the usage of the application with simple internet explorer such as Microsoft Internet Explorer, Mozilla Firefox etc. By the way, changing GUI from Swing to Servlet, JSP or JSF is enough to run the application.



**Figure 4.2** Package diagram of text processing tool

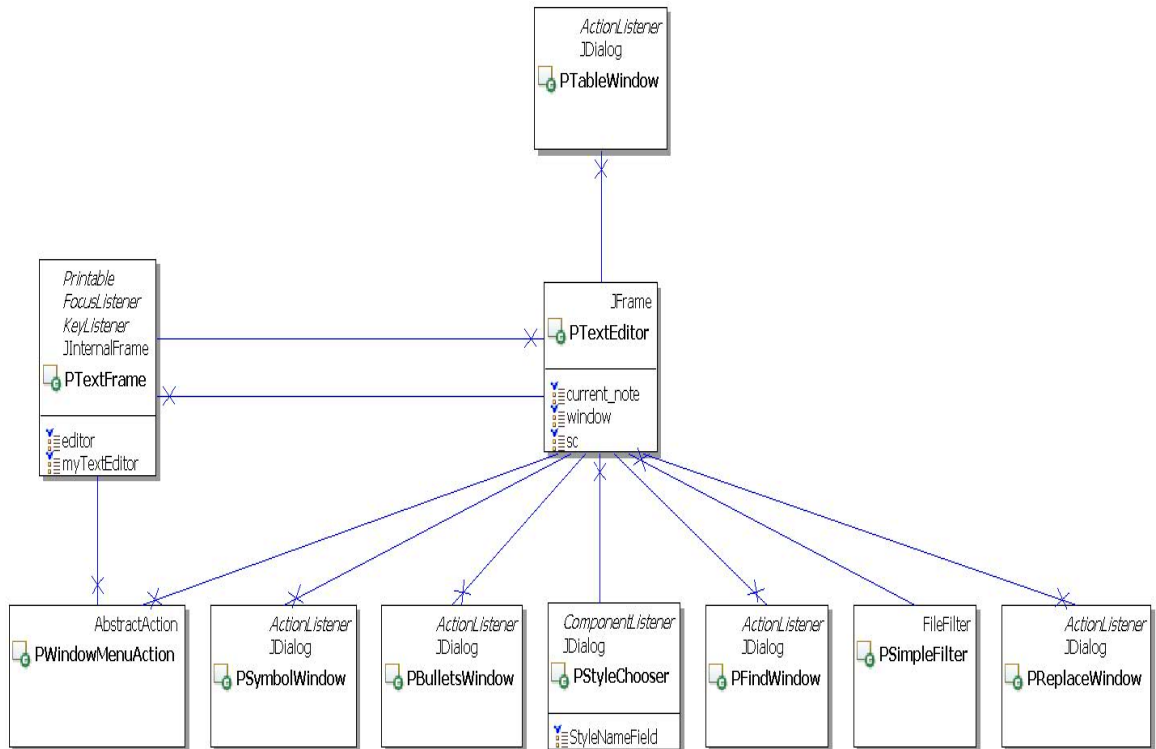
Presentation tier shows output to the users and interacts with user in order to receive actions. For this reason, Java Swing library (Graphical User Interface library) is used to showing data to user and taking requirement actions. PAbstractWindow extends from

javax.swing.JDialog java class and it is a base class of PLetterWindow, PSyllableWindow, PWordWindow, PSentenceWindow, PParagraphWindow, and POptionWindow. It has important methods for subclasses such as addResultToView, getStringFromTextEditor etc. These subclasses of PAbstractWindow are used to present letter, syllable, word, sentence and paragraph windows to the user. Also it takes actions and delivers them to the required actions. The class diagram of the presentation layer of the text processing tool is shown in Figure 4.3.



**Figure 4.3** Class diagrams of presentation layer

The class diagram of text processing tool about text editor is shown in Figure 4.4. In the program most important class is PTextEditor class. It is the container of all GUI classes and it is the main class that program starts from. Also, it has many internal classes that get user actions and interact with all presentation layer classes. This class is also a subclass of javax.swing.JFrame and it is used for creating new window frames. It is the main Editor class. This editor is capable of supporting any number of open documents. It supports edit actions like cut, copy, find, inset, symbols etc.

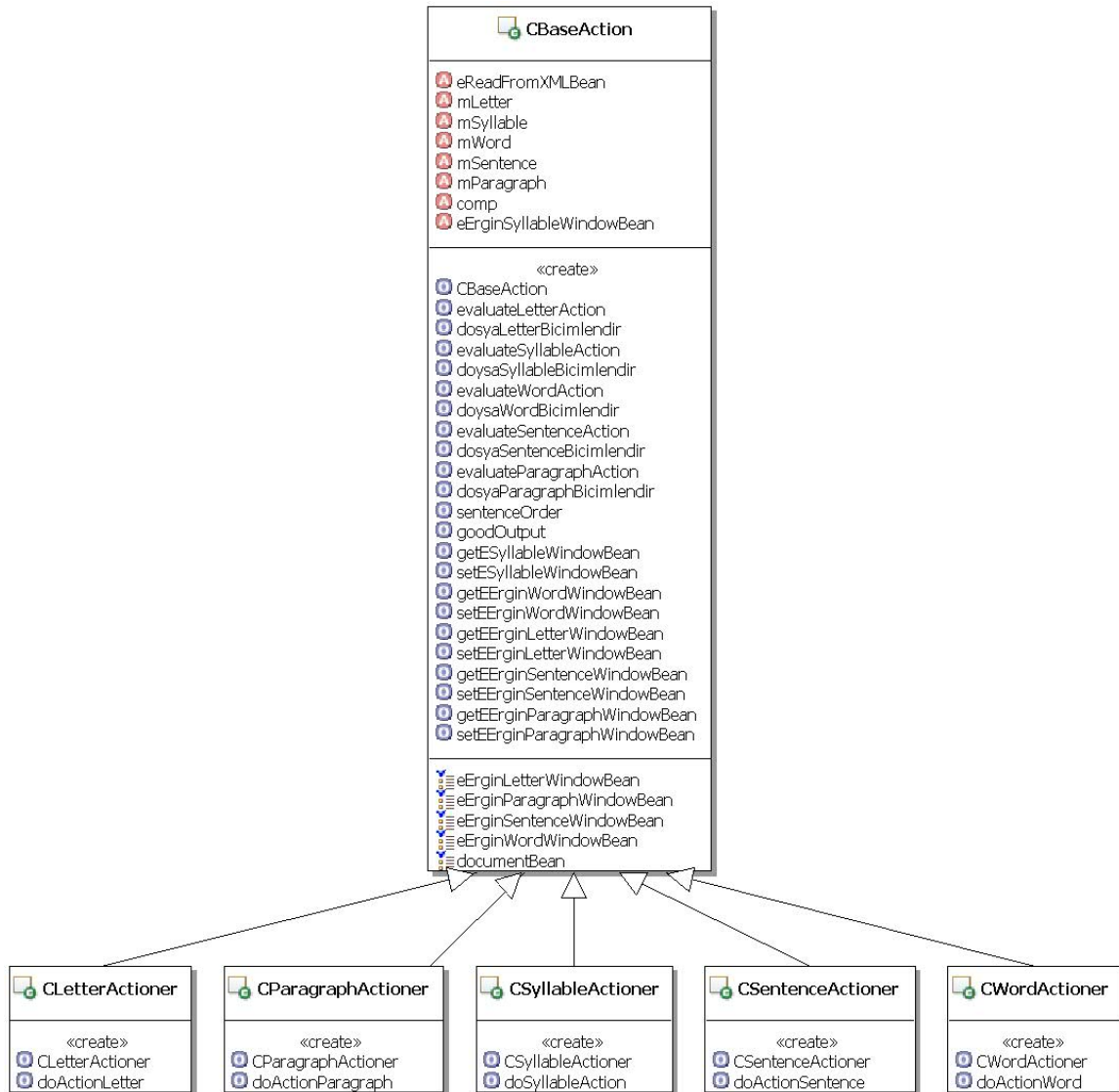


**Figure 4.4** Class diagrams of editor

Another important class in the presentation layer is `PTextFrame` class. This is the main document class. Each document is capable of cut, copy, paste, print actions, as well as find, find & replace, undo, redo and more. And also, this class is subclass of `javax.swing.JInternalFrame` to achieve work on multi documents at the same time. `PFindWindow` is used for find window frame.

Control layer classes takes actions from presentation layer classes and processes these actions using another layers. Application logic and algorithmic solutions are implemented in this layer. `CBaseAction` class is the base class of all module classes such as `CLetterActioner`, `CSyllableActioner`, and `CWordActioner` etc. These classes take required actions from presentation classes and forward actions to the `CBaseAction` class. This class has a very significant role in this project. It holds several objects in Mediator layer and `MLetter`, `MSyllable`, `MWord`, `MSentence` and `MParagraph` are examples of this layer. Outputs and format of outputs of the program are determined in this class, and then the result object is sent to required presentation class. During the processing,

EReadFromXMLBean class object is hold and this object contains target language properties such as alphabet (case sensitive), vowel letters, and consonant letters. The class diagram of the control layer is shown in Figure 4.5.

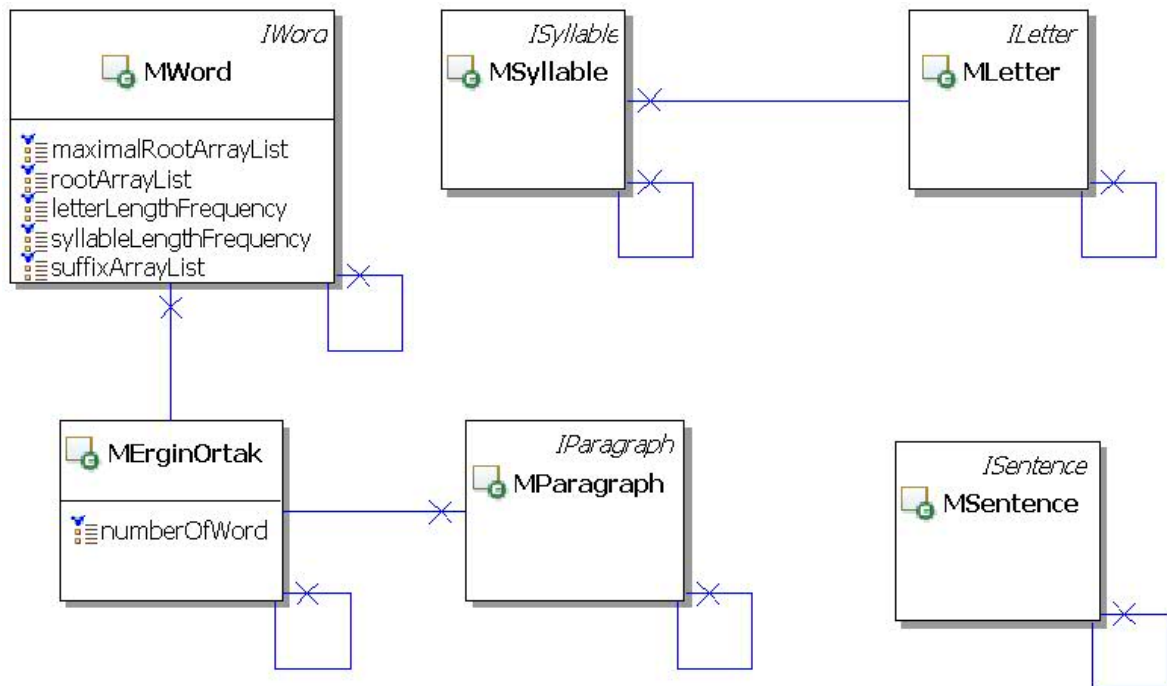


**Figure 4.5** Class diagram of control layer

The Mediator layer classes handle interactions between the entity layer and the foundation layer. Also it handles object-relational mapping. The Mediator objects then have dependencies on the foundation and the entity layer objects. Classes of this layer are shown

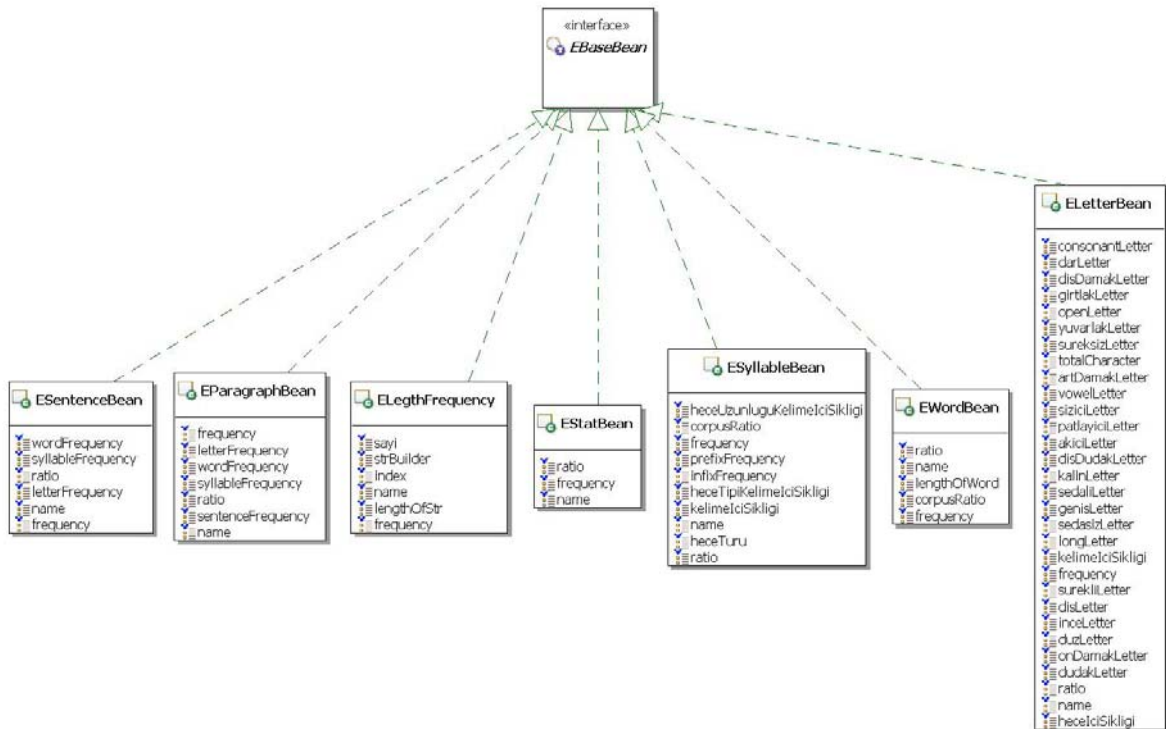


in Figure 4.6. All objects in this layer are applied Singleton pattern (Gamma et al, 1995), so during the life cycle of the project only one object can be created it specific class.



**Figure 4.6** Class diagram of domain-mediator layer

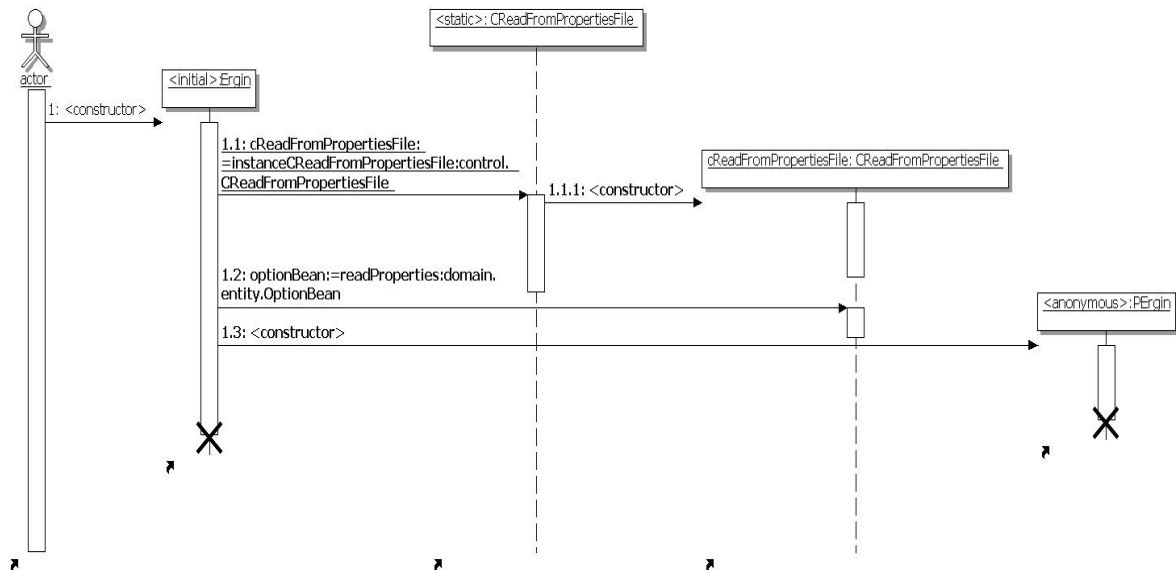
Entity layer handles the business objects. This layer classes are the subclasses of the EBaseBean class (Figure 4.7). In this project every letter, syllable, word, sentence and paragraph are represented by POJO object. Items name, frequency and ratio are important attributes for these objects.



**Figure 4.7** Class diagram of entity layer

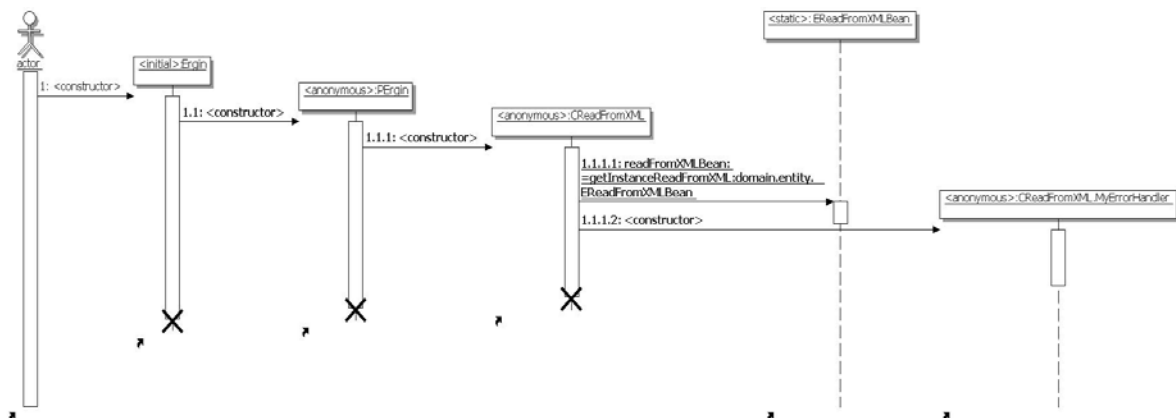
### 4.3 SEQUENCE DIAGRAMS

Some important objects and their behaviors should be described in order to understand program capabilities well. After the program starts the properties file that contains program configuration is read. Objects' collaborations and interactions are shown in Figure 4.8. First of all, Ergin object creates CReadFromProperties objects and sends it file name will be read. Next, CReadFromProperties object takes file name and reads this properties file. Then, data read from properties file are stored in OptionBean POJO object.



**Figure 4.8** Sequence diagram 1

The XML file which will be parsed by the program is read from properties file. Then, the file name is sent to the CReadFromXML object by Ergin object. After that, CReadFromXML file parsing document with XPath (Apache Xalan) queries and results are stored in EReadFromXMLBean POJO objects. Later on, this information will be used by program to determine consonant, vowel characters etc. Flow of this sequence diagram is shown in Figure 4.9.



**Figure 4.9** Sequence diagram 2

## **CHAPTER 5**

### **CONCLUSIONS**

Analyzing Turkish texts is very important for not only in Turkish language and literature itself but also in a wide spectrum of areas including education, psychology, sociology, politics, business management, criminology, law and medicine. Therefore an application that can analyze Turkish texts using the frequency distributions of various language elements such as letters, syllables, words, sentences and paragraphs is quite significant in all these areas.

In this thesis, text analyzing and processing tool for Turkish is developed. The system is developed with Java programming language because of the advantages of Java like supporting Unicode. Unicode represents all of the letters in all of the alphabets in the world. The program developed is designed as to analyze all the languages in the future; but now it only works for Turkish.

The text processing and analyzing tool is designed according to the PCMEF framework. The presentation package depends on control package, control depends on entity and mediator packages, and mediator depends on foundation package. PCMEF framework minimizes package coupling, decreases dependency and increases stability with using downward dependencies, higher layers depends on lower layers. The architectural design of the PCMEF framework will provide the easy construction of web based form of the analyzing tool in the future.

The developed tool has text processing features like copy, paste, change color etc. The important part of the developed tool is analyzing. There are 5 basic structures in

Turkish language that are analyzed. They are letter, syllable, word, sentence and paragraph. The frequencies, ratios and percentage of the structures are analyzed in the selected documents and displayed in the output screen.

In the future, the text processing and analyzing tool developed can be implemented as web based system. Also, support for Turkmen language can be implemented. The data mining like document classification and clustering (Karanikas, 2002) can be added to the tool.

## REFERENCES

- Adalı, O., *Türkiye Türkçesinde Biçimbirimler*, Papatya Yayıncılık, 2004.
- Aksan, D., *Anlambilimi ve Türk Anlambilimi*, Ankara Üniversitesi Dil ve Tarih-Coğrafya Fakültesi Yayınları, Ankara, 1978.
- Banguoglu, T., *Türkçenin Grameri*, Türk Dil Kurumu, Ankara, 2000.
- Ergin, M., *Türk Dil Bilgisi*, Boğaziçi Yayınları, İstanbul, 1998.
- Eryiğit, G., Oflazer, K., Statistical dependency parsing of Turkish. *Proceedings of EACL 2006 11th Conference of the European Chapter of the Association for Computational Linguistics*, Trento, Italy, April 2006.
- Gamma, E., Helm. R., Johnson, R., Vlissides, J., *Design Patterns, Elements of Reusable Software*, Addison Wesley, 1995.
- Göz, İ., *Yazılı Türkçenin Kelime Sıklığı Sözlüğü*, Türk Dil Kurumu Yayınları, Ankara, 2003.
- Hemrajani, A., "Java I/O Streams", 1998,  
<http://java.sun.com/developer/technicalArticles/Streams/ProgIOStreams/>
- Horstmann, C., *Object-Oriented Design & Patterns*, Wiley, San Jose State University, 2006.
- International Phonetic Association, *Handbook of the International Phonetic Association: A guide to the use of the International Phonetic Alphabet*, Cambridge University Press, 1999.
- Jukka, K. K., *Unicode Explained*, O'Reilly, New York, 2006
- Karahan, L., *Türkçede Söz Dizimi*, Akçağ Yayınları, Ankara, 1997.
- Karanikas, H., Theodoulidis. B., *Knowledge Discovery in Text and Text Mining*, Software Publication of CRIM., 2002.

- Maciaszek, L. A., “Developing Supportable Enterprise Information Systems – Architectural, Managerial and Engineering Imperatives,” Proceedings of the 21st IEEE Int. Conf. on Software Maintenance, pp.721-722, 2005.
- Maciaszek, L. A., “Managing complexity of enterprise information systems”, Springer Netherlands, 2006.
- Maciaszek, L. A., Liong B. L., *Practical Software Engineering*, Addison Wesley, 2004.
- Maciaszek, L. A., “Roundtrip Architectural Modeling”, Conf. in Research and Practice in Information Technology Series, Vol. 107, 2005.
- Mila Text Editor,  
<http://softlab-pro-web.technion.ac.il/projects/Mila/html/webPageDir/Description.htm>
- O'Conner, J., “Understanding Locale in the Java 2 Platform”, 2002  
<http://www.joconner.com/wp-content/uploads/2007/04/locale.html>
- Oflazer, K., Two-level Description of Turkish Morphology,  
 Department of Computer Engineering and Information Science, 1993
- Richardson, C., *POJOs in Action: Developing Enterprise Applications with Lightweight Frameworks*, Manning, 2006.
- Shylov M., Dilmac, Ms Thesis, Fatih University, in progress.
- Tantuğ A.C., Adalı, E., Oflazer, K., “A Prototype Machine Translation System Between Turkmen and Turkish”, *Proceedings of the Turkish Artificial Intelligence and Neural Networks*, TAINN 2006, Muğla, Turkey, 2006.
- Tekcan, A., Göz, İ., *Türkçe Kelime Normları*, Boğaziçi Üniversitesi Yayınevi, İstanbul, 2005.
- The message format class,  
<http://java.sun.com/j2se/1.4.2/docs/api/java/text/MessageFormat.html>
- The official Unicode web site, <http://unicode.org/>
- The resource bundle class  
<http://java.sun.com/j2se/1.4.2/docs/api/java/util/ResourceBundle.html>
- Toklu, M. O., *Dilbilime Giriş*, Akçağ Yayınları, 2005.
- Uzun, N. E., *Biçimbilim Temel Kavramları*, Papatya Yayıncılık, Ankara, 2006.
- Zemberek official web site,  
<http://code.google.com/p/zemberek/> , <https://zemberek.dev.java.net/>