# NESNEYE DAYALI PROGRAMLAMA

Proje

Çeşitli işlevselliklere sahip metin editörünün (text editor) gerçekleştirimi

Ad-soyad, Numara

Mehmet Anıl Taysi, 05170000022

Emel Kayacı, 05180000087

# İÇİNDEKİLER

# 1. PROGRAMCI KATALOĞU

## 1.1 Sınıf ve Metot Açıklamaları

## MainFrom Sınıfı

GUI içerisindeki menü çubuğunda yer alan ve kullanıcının iletişime geçtiği seçeneklerin gereken işlevlerinin gerçekleştirdiği metotları içeren sınıftır.

- **yanlislariBul metodu:** Sözlükte yer almayan yani yanlış sayılan kelimeleri kırmızı renk ile vurgular.

- **yanlislariDuzelt metodu:** SingleTransposition olan kelimeleri doğruları ile değiştiren metottur.

- **bul metodu:** Düzen kısmında yer alan bul ve değiştir seçeneklerinin her ikisinde de bulma işlemi yapılmaktadır. Bul seçeneğinde sadece istenen kelimeler bulunurken değiştir de bulunan kelime ayrıca kullanıcının istediği kelime ile değiştiriliyordur.

  Modülerlik açısından bul bir metot haline getirilip bul ve değiştir seçeneklerine ait ActionPerformed metotlarında çağırımı yapılmaktadır.

- **kaydet metodu:** TextArea içerisindeki metindeki değişiklikleri kaydetmeye yarar. Eğer dosya Adsız ise yani bilgisayarda herhangi bir yerde oluşturulmamış ise farkliKaydet metodu çağırılır. Böylece kaydetme işleminin yanında oluşturma da sağlanmış olur.

- **farkliKaydet metodu:** JFileChooser ekranı sayesinde kullanıcı bilgisayar içerisinde dosyayı kaydetmek istediği konumu seçer. Proje içerisinde sadece text tipindeki dosyalar kullanıldığından kullanıcının dosya ismine txt takısı olmadan yalnızca dosya ismini yazması yeterli olur.

- **kaydetMessageBox metodu:** Kullanıcının karşısına dosyayı kaydetme, kaydetmeme ya da iptal seçeneklerinin çıktığı mesaj kutusunun bulunduğu metottur. Şu durumlarda kullanılır:

  1. Kullanıcı yeni seçeneğiyle daha önce bilgisayarda yaratılmamış bir text dosyası yarattıktan sonra içerisine bir metin yazdığında tekrar yeni tuşuna basıp yeni dosya açmak istediğinde yazdığı değişiklikleri kaydedip kaybetmediğini sormak için kullanılır.
  2. TextArea içerisinde metin varken kullanıcı uygulamayı kapat seçeneğinden kapatmak istediğinde kullanılır.

## SingleTransposition Sınıfı

SingleTransposition olarak adlandırılan ve iki harfin yer değiştirmesinden kaynaklı yazım hatalarını düzelten metotların bulunduğu sınıftır.

- **sozluguAktar metodu:** Bize verilmiş olan "words.txt" dosyasında yer alan sözlüğü program içerisinde kullanılabilir hale getiren metottur. Sözlüğün ArrayList veri yapısına buffered reader yardımıyla aktarır.

- **yanlislariBul metodu:** TextArea içerisinde yer alan metindeki yazım yanlışlarını bulur. Kelimenin yanlış sayılması sözlükte yer alıp almamasına bağlıdır. TextArea içerisindeki metin split metodu yardımıyla kelimelere ayrılır.

  Split metodu içerisinde yer alan **("[^a-zA-Z]+")** ifadesi sayesinde metindeki kelimeler nokta, virgül, noktalı virgül, boşluk vb. ayraçlarla ayrılmışsa sadece kelimeleri ayıracak şekilde ayarlanmış olur.

  Metin içerisinde kelimeler sözlükte tek tek aranarak yer almayanlar yanlisKelimeler listesiyle döndürülür.

- **yanlislariDuzelt metodu:** Single transposition burada gerçekleştirilir. yanlisKelimeler listesindeki her kelime öncelikle sözlükte yer alan kelimelerle uzunluk açısından kıyaslanır. Eğer uzunluklar aynıysa bu sefer de harf kıyaslaması yapılır.

  Harf kıyaslaması yapabilmek için hem sözlükteki kelimeler hem de yanlisKelime karakter dizisine aktarılır. Bu aktarımı yapmaktaki amaç karakterleri sort metodu ile sıralayabilmek içindir. Karakterler sıralandıktan sonra harf harf bir kıyaslama yapılır. Böylece sözlükteki kelime ile düzeltmek istediğimiz yanlış kelimenin aynı harflerden oluşlup oluşmadığına ulaşırız.

  Eğer harflerin hepsi aynı ise comp değişkeni artmamış olur ve bu sayede single transposition için bir sonraki adıma geçebiliriz. Yalnızca iki harfin yer değiştirmesinden kaynaklanan hatalar düzeltileceğinden dolayı maksYerDegistirme ile ne kadar harfin yerinin farklı olduğu belirlenir.

  Yeri farklı olan harf sayısı iki ise Map olarak isimlendirilen ve Java'da dictionary olarak kullanılan veri yapısına yanlisKelime ve doğrusu aktarılır.

## Birim Test Sınıfı

Projemizde birim test sınıfımız SingleTransposition sınıfı için SingleTranspositionTest.java sınıfıdır.

SingleTransposition sınıfında bulunan üç metodun gerçekleştirimlerinin doğru bir şekilde test edilebilmesi için dört farklı birim test metodu yazılmıştır. Birim test sınıfı Test Packages klasörü altındaki Test paketinde bulunabilir.

**compareTheFirstIndexOfVocabulary():** Bu metod adından da anlaşılacağı üzere, proje için sözlük olarak verilen 'words.txt' dosyasındaki ilk kelimeyi beklenen değer ile karşılaştırma üzerine kurulmuştur.

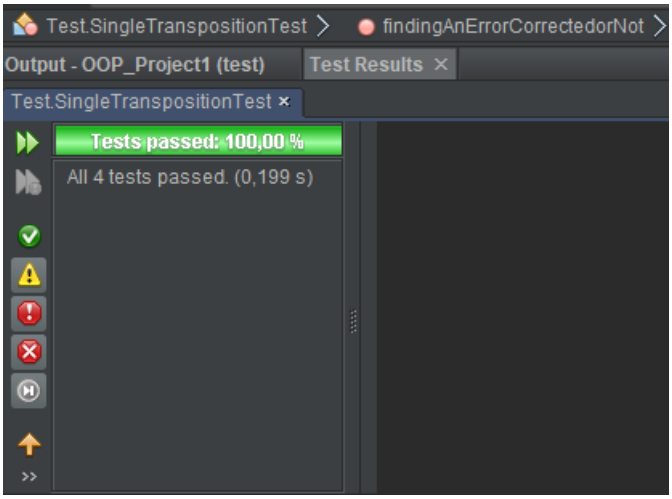
**findingExpectedErrorWord():** Bu test metodumuz ise, kullanıcı arayüzündeki TextArea'ya yazılan bir satır yazıda yanlışların bulunması üzerine bir simülasyondur. Metod içinde örnek bir metin tanımlanmış olup, yine proje için verilen Word dosyasına dayanarak bu kelimelerin doğru veya yanlış olup olmadıkları tespit edilmektedir. Fonksiyonun sonunda ise beklenen Array değeri ile SingleTransposition classındaki metodun ürettiği Array kıyaslanmatadır.

**findingAnErrorCorrectedorNot():** Bu test durumumuzda ise, düzeltilen adında bir map oluşturup, yanlış düzeltme metodumuzun doğru çalışıp çalışmadığına dair bir deneme gerçekleştiriyoruz. Yanlış olduğunu bildiğimiz kelimelerden oluşan array'i yanlislariDuzelt fonksiyonuna gönderiyoruz, ve doğru kelimeler ile düzelmiş olduğunu varsaydığımız kelimeleri kıyaslıyoruz. Böylece testimiz amacına ulaşmış oluyor.

Örneğin, 'pefrect' kelimesinin 'perfect' olarak doğrulanmasını bekleriz. (Single Transposition hatası içerdiğinden)

**withAssertTrue_findingAnErrorCorrectedorNot():** Yukarıdaki fonksiyonun örnek eşkil etmesi açısından boolean değerle birlikte 'assertTrue' test fonksiyonu ile tanımlanmış halidir. Üstteki fonksiyondan farklı bir örnekle gerçekletirilmiştir.

Sonuç olarak 4 test fonksiyonumuzda testi geçirmektedir. Bu da fonksiyonlarımızın istediğimiz doğrultuda çalıştığına işaret eder.
Test sınıfına girildiğinde görüleceği üzere, her birim test fonksiyonun nasıl gerçekleştirdiği yorum satırları ile desteklenmiştir.

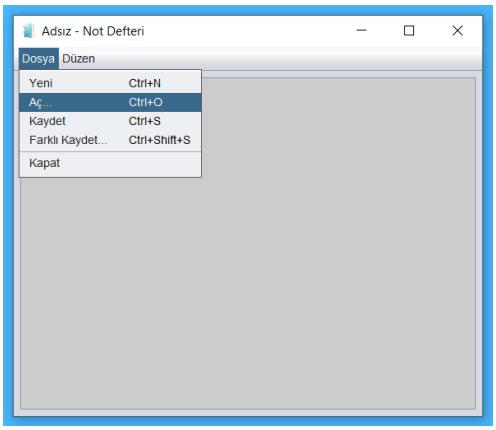



**Şekil.1** *Testin Gerçekleştirim Sonuçları*

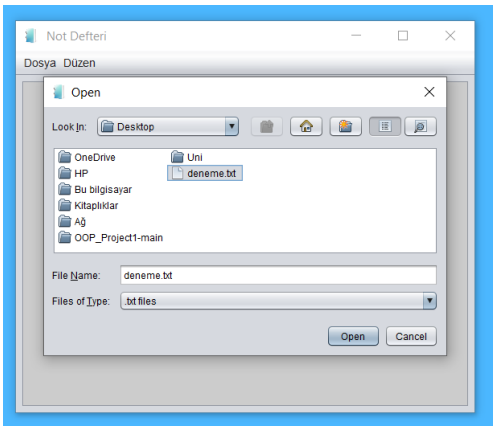
# 2. KULLANICI KATALOĞU

## 2.1 Dosya Açmak

Kullanıcı dosya aç seçeneğine tıkladığında (şekil 2) karşısında şekil 3 ile gösterilmiş ekran çıkar. Böylece bilgisayarın herhangi bir yerinde olan dosyaya erişimi olur.



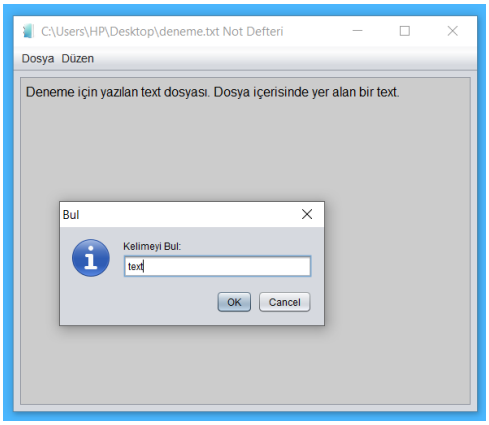

**Şekil 2.** *Aç seçeneği*



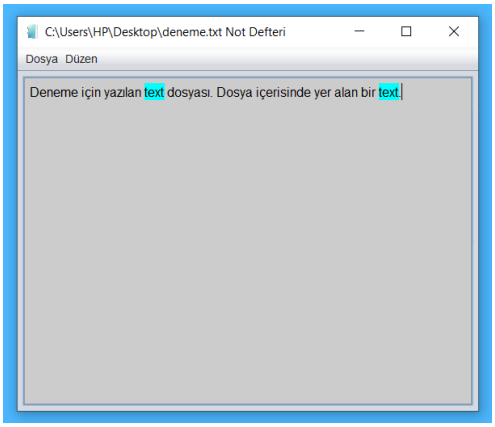

**Şekil 3.** *File Chooser ile istenen konuma ulaşma*

## 2.2 Aranan Kelimenin Bulunması

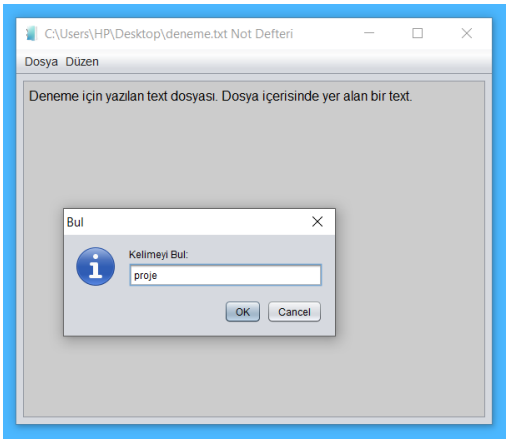
Giriş ekranı (şekil 4) ile kullanıcı metin içerisinde aramak istediği kelimeyi girer. Eğer kelime var ise açık mavi vurgu (şekil 5) ile belirtilmiş olur. Fakat aranan kelime (şekil 6) metinde bulunamadıysa kullanıcı uyarı mesajı (şekil 7) ile karşılaşır.



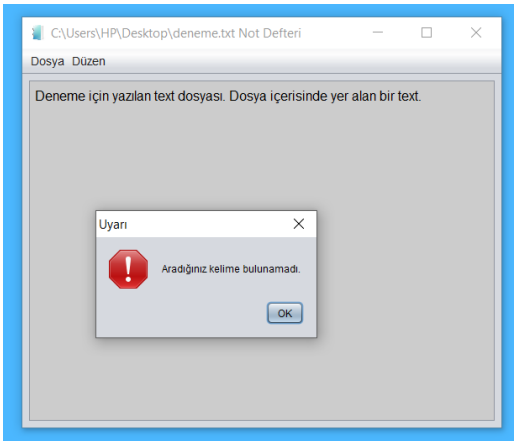

**Şekil 4.** *Kelime giriş ekranı*




**Şekil 5.** *Aranan kelimenin vurgulanması*



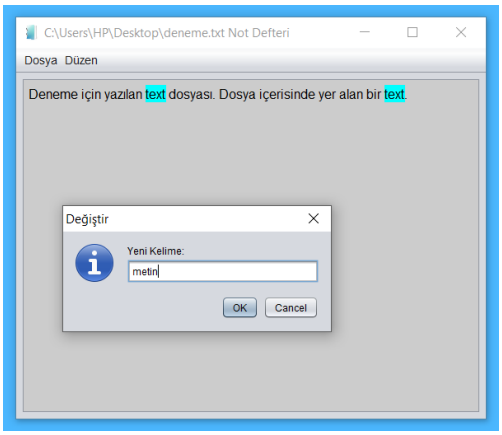

**Şekil 6.** *Metinde yer almayan kelimenin aranması*



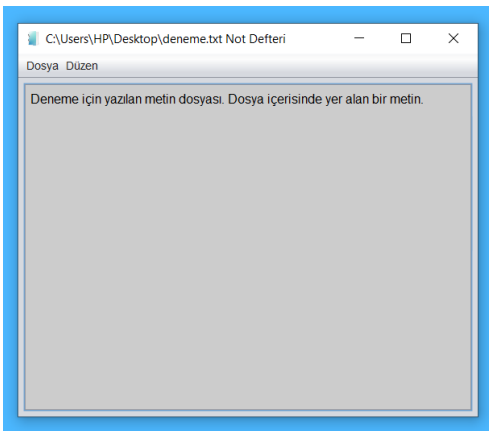

**Şekil 7.** *Uyarı mesajı*

## 2.3 Metindeki Kelimenin Değiştirilmesi

Giriş ekranı (şekil 8) ile kullanıcı metin içerisinde değiştirmek istediği kelimeyi girer. Değiştirmek için önce kelimenin bulunması gerektiğinden kelime bulunamadıysa kullanıcı üstte belirtilmiş olan (şekil7) uyarı mesajı ile karşılaşır. Değiştirilmiş metin şekil 9'de gösterilmiştir.

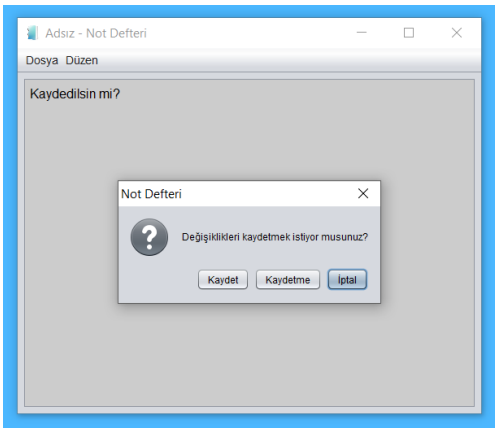



**Şekil 8.** *Değiştirilmek istenen kelimenin bulunması*




**Şekil 9.** *Kelimenin değiştirilmesi*

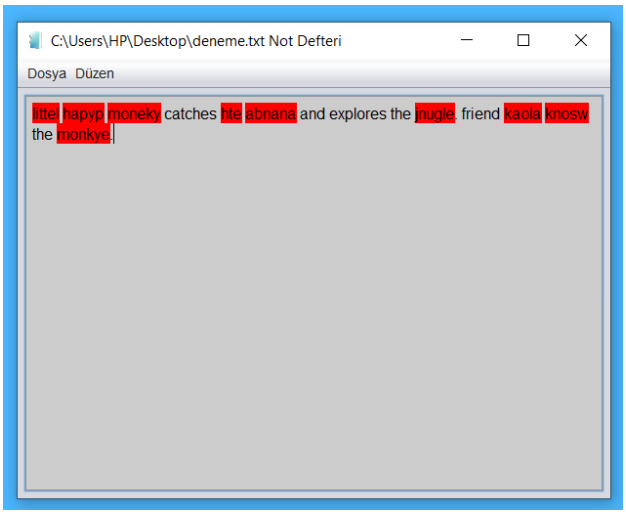
## 2.4 Kaydetme Mesaj Kutusunun Gösterimi



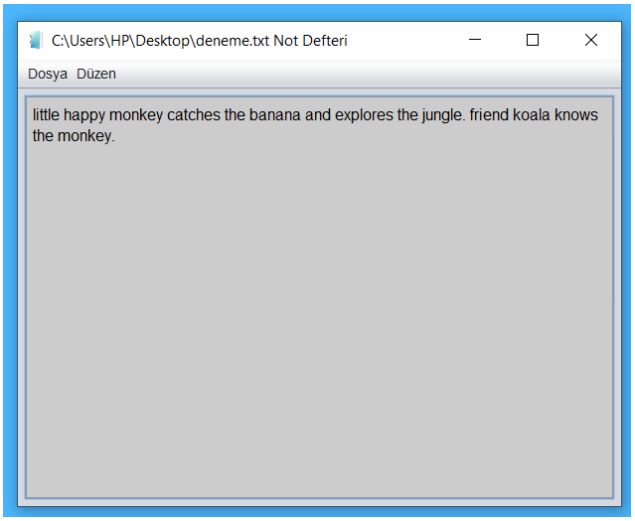

**Şekil 10.** *kaydetMessageBox metodunda belirtilmiş durumlarda gelen mesaj kutusu*

### 2.5 Sözlükte Olmayan (yanlış) Kelimelerin Düzeltilmesi

Sözlükte olmayan kelimeleri yanlışları bul seçeneği ile kullanıcı görüntüleyebilmektedir. Yanlışları düzelt seçeneğinde ise kırmızı ile vurgulanmış (şekil 11) bu yanlışların düzeltilmiş hali (şekil 12) görülmektedir.

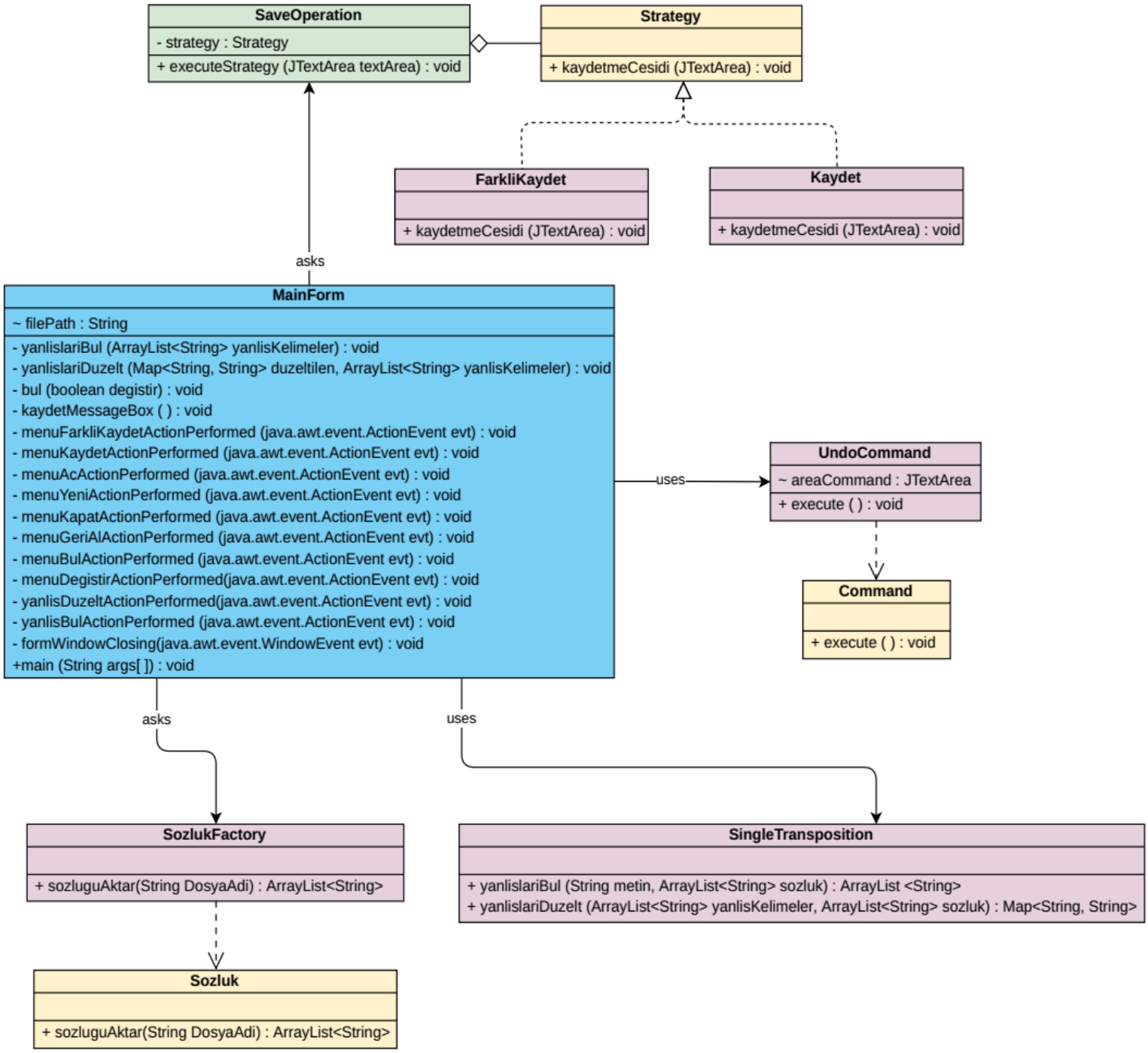



**Şekil 11.** ***Yanlış kelimelerin vurgulanması***




**Şekil 12.** ***Yanlış kelimelerin düzeltilmesi***

# 3.UML ŞEMASI


SaveOperation
- strategy : Strategy
+ executeStrategy (JTextArea textArea) : void
Strategy
+ kaydetmeCesidi (JTextArea) : void
FarkliKaydet
+ kaydetmeCesidi (JTextArea) : void
Kaydet
+ kaydetmeCesidi (JTextArea) : void
asks
MainForm
~ filePath : String
- yanlislariBul (ArrayList<String> yanlisKelimeler) : void
- yanlislariDuzelt (Map<String, String> duzeltilen, ArrayList<String> yanlisKelimeler) : void
- bul (boolean degistir) : void
- kaydetMessageBox ( ) : void
- menuFarkliKaydetActionPerformed (java.awt.event.ActionEvent evt) : void
- menuKaydetActionPerformed (java.awt.event.ActionEvent evt) : void
- menuAcActionPerformed (java.awt.event.ActionEvent evt) : void
- menuYeniActionPerformed (java.awt.event.ActionEvent evt) : void
- menuKapatActionPerformed (java.awt.event.ActionEvent evt) : void
- menuGeriAlActionPerformed (java.awt.event.ActionEvent evt) : void
- menuBulActionPerformed (java.awt.event.ActionEvent evt) : void
- menuDegistirActionPerformed(java.awt.event.ActionEvent evt) : void
- yanlisDuzeltActionPerformed(java.awt.event.ActionEvent evt) : void
- yanlisBulActionPerformed (java.awt.event.ActionEvent evt) : void
- formWindowClosing(java.awt.event.WindowEvent evt) : void
+main (String args[ ]) : void
uses
UndoCommand
~ areaCommand : JTextArea
+ execute ( ) : void
Command
+ execute ( ) : void
asks
uses
SozlukFactory
+ sozluguAktar(String DosyaAdi) : ArrayList<String>
SingleTransposition
+ yanlislariBul (String metin, ArrayList<String> sozluk) : ArrayList <String>
+ yanlislariDuzelt (ArrayList<String> yanlisKelimeler, ArrayList<String> sozluk) : Map<String, String>
Sozluk
+ sozluguAktar(String DosyaAdi) : ArrayList<String>

# 4. Tasarım Desenlerinin Kullanımı

Bu projemizde, ‘geri al’ metodu için istenen command tasarım kalıbını, dolaşma gerektiren durumlarda generic veri tipleri üzerinde dolaşma sağlamak için iterator tasarım kalıbını, ayrıca 2 adet tasarım kalıbı kullanım gereksimini karşılamak üzere factory method ve strategy kalıplarını kullandık.

## 4.1 Command Design Pattern Kullanımı

Öncelikle Command.java interface’i oluşturup **void execute()** adında bir method tanımladık, bunun amacı, Command interface’ini implement eden her bir sınıfın execute() metodunu gerçekleştirmesini bekliyoruz.

Bu proje için, UndoCommand.java sınıfımızdan bu execute() işlemini gerçekleştirmesini bekliyoruz. Ayrıca, ‘geri alma’ işlemi için textArea değişkenimizi de bu sınıfta tanımlamamız gerekiyor, dolayısıyla bir JtextArea değişkeni oluşturup sınıfın constructor’ına da textArea’yı parametre olarak verip, bu ikisini eşitliyoruz, böylece UndoCommand sınıfının içinde, textArea üzerinde işlem yapabileceğiz.

GeriAl butonuna basıldığında,

```
UndoCommand geriAl = new UndoCommand(textArea);

geriAl.execute();
```

Yukarıda belirttiğimiz gibi UndoCommand sınıfından bir obje oluşuyor ve textArea’yı parametre olarak alıyor. Sonrasında Command pattern’ının da amacı olan execute işlemimizi gerçekleştirip gerekliliği sağlıyoruz.

## 4.2 Iterator Kullanımı

Iterator, generic tipler üzerinde gezinti yapma (sadece ileri yönlü) ve gerekirse remove() metodu ile bileşen silme özelliğine sahip bir çeşit dolaşma gerecidir. Biz, projemizdeki her bir generic dolaşma yapısı için (ArrayList’ler için oluşturulan ForEach döngüleri) yapıyı yeniden tasarlayıp iterator olarak değiştirdik.

Örneğin, for (yanlisKelime : yanlisKelimeler) şeklinde dolaştığımız yapıyı,

-yanlisKelime = String, yanlisKelimeler = ArrayList-

```
Iterator itr = new yanlisKelimeler.iterator();

While(itr.hasNext()) {

YanlisKelime = yanlisKelimeler.next();

…

… }
```

Şeklinde yeniden oluşturabildik. Burada while döngüsü yanlisKelimeler arraylist'inin son elemanına ulaşıncaya kadar dönüyor, ileri iterasyon işlemini de yenlisKelimeler.next() ile yapıyoruz. Fakat Iterator yapısının bazı sıkıntıları da var. Örneğin bu tanımı normal bir String array'i üzerinde gerçekleştiremedik, ayrıca Iterator ile yaptığımız yerlerde NetBeans IDE'si bunun yerine ForEach döngüsü kullanmamızı öneriyor.

## 4.3 Factory Method ve Strategy Design Pattern'larının Kullanımı

### 4.3.1 Factory Method Pattern

Sozluk.java Interface'inde sozluguAktar(String DosyaAdi) şeklinde bir ArrayList<String> değeri döndüren metodumuzu tanımlıyoruz, sonrasında SozlukFactory.Java sınıfında, bu metodu oluşturuyoruz (daha önce SingleTransposition.java sınıfı içindeydi). Çağırım işlemi için,

SozlukFactory sozlukFac = new SozlukFactory();

ArrayList<String> sozluk = sozlukFac.sozluguAktar("words.txt");

Seklinde sözlüğümüzü programa aktarıyoruz.

Bu Design pattern'ın kullanım amacı, özellikle farklı dosya tipleri için programa veri aktarımı yapılacak olduğu durumlarda, tekrar Sozluk interface'ini implement edecek şekilde farklı sınıfların aynı fonksiyon ismi üzerinden aynı amaca hzimet eden farklı metodlar geliştirebilmesini sağlamaktır. Örneğin eğer bir .csv dosyasından Sözlüğümüzü alıyor olsaydık, SozlukCsvFactory.java adında bir sınıf tanımlayıp, Sozlugu implement edip, override edilen metodu gerektiği şekilde gerçekleştirebilirdik. Bu yapı modülerliği arttırmış oluyor, ayrıca aynı java sınıfını farklı projelerde de rahatlıkla kullanbilme esnekliği sağlıyor.

### 4.3.2 Strategy Pattern

Strategy.java adında bir interface oluşturup, kaydetmeCesidi(JtextArea textArea); seklinde void return tipine sahip metodumuzu tanımlıyoruz. Sonrasında Kaydet.java ve FarkliKaydet.java olarak iki sınıfta, Strategy interface'ini implement ediyoruz. Böylece iki sınıf için de kaydetmCesidi metodumuzu override etmeye zorluyoruz. Bu metodun içinde, daha önce MainForm.java sınıfında tanımlı olan kaydetme metodlarını gerçekleştiriyoruz. Bu design pattern'ın gerçekleştirimi için, SaveOperation.java adında bir operasyon sınıfı oluşturuyoruz. Bu metodun içinde tanımlı private Strategy nesnesi tanımlı olmakla beraber, sınıfın constructor metoduna parametre olarak yine bir Strategy nesnesi veriyoruz (Design pattern'ın bizce en kritik noktası), sonrasında executeStrategy adında bir metod tanımlayıp buna parametre olarak da JtextArea'yı veriyoruz.

Şimdi kaydet ve farklikaydet özelliklerini MainForm'da nasıl gerçekleştirdiğimize bakalım,

SaveOperation operation = new SaveOperation(new FarkliKaydet());

operation.executeStrategy(textArea);

şeklinde bir çağırım şekli var. SaveOperation sınıfından bir nesne türetip, bu nesneye parametre olarak Strategy interface'inden türemiş istediğimiz herhangi bir sınıfı verebiliriz. Operation üzerinden executeStrategy metodumuzu, çağırım sırasında hangi sınıftan olduğuna dikkat etmeksizin kullanabiliriz. Bu da programa yine daha statik ve modüler bir yapı kazandırıyor.

Örneğin Kaydet özelliğğini de gerçekleştirelim,

SaveOperation operation = new SaveOperation(new Kaydet());

operation.executeStrategy(textArea);

Göründüğü üzere tek fark nesne oluşturulurken parametre oalrak hangi sınıf objesinin oluşturulduğudur. Ekstra olarak, SaveOperation operation; oepration objesini programın giriş kısmında oluşup, gerektiği zaman constructor çağırımını yapıp kullanabiliyoruz.