NESNEYE DAYALI PROGRAMLAMA PROJE – 2

05170000057 – Ramazan EYMİR

05170000021 – Enes Alper BALTA

05170000025 – Yağız BEŞİROĞLU

İçindekiler

Sorun Tanımı ve Çözüm Algoritması

Bizden daha önce oluşturduğumuz NotePad uygulamasını tasarım desenlerini kullanarak daha efektif bir versiyonu istenmiştir. İstenen bu program, daha önceki versiyon gibi bir txt belgesini okuyup içinde kullanıcının girdiği kelimeyi bulabilen, değiştirebilen, yazılan harfleri geri alabilen ve tek harf yanlışlığı sonucu oluşan yazım hatalarını düzeltebilen bir programdır.

Bu projede, tasarım desenleri kullanarak geliştirmemiz beklenmekte. Geri alma işlevselliğini Command, koleksiyon dolaşma gereken yerlerde hazır Iterator ayrıca bizden istenen ekstra 2 tasarım deseni olarak da Abstract Factory ile Strategy tasarım desenlerini kullandık.

Sınıfların Açıklanması

NotePad

Bu sınıf Abstract Factory tasarım deseninden gelen AbstractFactory'i temsil eden NotepadFactory nesnesinden bir referans tutar. Ayrıca Swing Framework'ünden kullandığımız GUI elementlerinin referanslarını da bulundurmaktadır. Command tasarım deseniyle oluşturduğumuz UndoableCommand objesi de burada oluşturulup GUI içerisinde kullanılmaktadır. Singleton tasarım desenini sınıf içerisinde uyguladık, böylece diğer sınıflarda kolaylıkla tek nesne üzerinden işlem yapabileceğiz. Ayrıca burada önemli bir işleve sahip olan NotepadCreator metoduna da sahibiz. Bu metot factoryleri parametre alıp buna göre GUI'yi oluşturmaktadır.

NotepadFactory

Frame, Panel, TextArea, MenuBar, Menu, Label, TextField, MenuItem ve Button için make metodları burada bulunuyor. Üsttekilerden farklı olarak ScrollPane için de bir make metodu bulunuyor. Abstract Factory Pattern şablonunda bulunan AbstractFactory yapısını burada oluşturduk. Default temanın gerçekleştirimi de burada oluyor. new işlemleri kullanıcıdan soyutlanıp makeX fonksiyonlarına donusturuldu. new işlemini gerçekleştirdiğimiz her bir swing elemanı bir Product yerine geçmektedir.

NotepadAbstractFactory

Main metodumuz burada ve metodumuz gelen argümana göre temamızı çağırıyor. Abstract Factory Pattern içerisindeki Client elemanı olan Notepad class'ının factory'i argüman olarak alan NotepadCreator metodunu burada kullanıyoruz. Ayrıca her temaya ait swing elemanları, temaya ait olan paket altında tanımlanmıştır.

blackModeFactory

Frame, Panel, TextArea, MenuBar, Menu, Label, TextField, MenuItem ve Button için make metodları burada bulunuyor. Abstract Factory Pattern sablonunda bulunan ConcreteFactory yapisini burada olusturduk. Black Mode temanin gerceklestirimi de burada oluyor. new islemleri kullanicidan soyutlanip makeX fonksiyonlarina donusturuldu. new islemini gerceklestirdigimiz her bir swing elemanı bir Product yerine gecmektedir.

blackModeButton

Sadece constructor'ı var. Product olan JButton elemanından ConcreteProduct olan blackModeButton elemanı extend ediliyor.

blackModeFrame

Sadece constructor var. Product olan JFrame elemanından ConcreteProduct olan blackModeFrame elemanı extend ediliyor.

blackModeLabel

Sadece constructor var. Product olan JLabel elemanından ConcreteProduct olan blackModeLabel elemanı extend ediliyor.

blackModeMenu

Sadece constructor var. Product olan JMenu elemanından ConcreteProduct olan blackModeMenu elemanı extend ediliyor.

blackModeMenuBar

Sadece constructor var. Product olan JMenuBar elemanından ConcreteProduct olan blackModeMenuBar elemanı extend ediliyor.

blackModeMenuItem

Sadece constructor var. Product olan JMenuItem elemanından ConcreteProduct olan blackModeMenuItem elemanı extend ediliyor.

blackModePanel

Sadece constructor var. Product olan JPanel elemanından ConcreteProduct olan blackModePanel elemanı extend ediliyor.

blackModeTextArea

Sadece constructor var. Product olan JTextArea elemanından ConcreteProduct olan blackModeTextArea elemanı extend ediliyor.

blackModeTextField

Sadece constructor var. Product olan JTextField elemanından ConcreteProduct olan blackModeTextField elemanı extend ediliyor.

darkModeFactory

Black moddaki gibi Frame, Panel, TextArea, MenuBar, Menu, Label, TextField, MenuItem ve Button için make metodları burada bulunuyor. Abstract Factory Pattern şablonunda bulunan ConcreteFactory yapısını burada olusturduk. Dark Mode temanın gerçekleştirimi de burada oluyor. new islemleri kullanicidan soyutlanıp makeX fonksiyonlarina dönüştürüldü. new islemini gerçeklestirdigimiz her bir swing elemanı bir Product yerine geçmektedir.

darkModeButton

Sadece constructor var. Product olan JButton elemanından ConcreteProduct olan darkModeButton elemanı extend ediliyor.

darkModeFrame

Sadece constructor var. Product olan JFrame elemanından ConcreteProduct olan darkModeFrame elemanı extend ediliyor.

darkModeLabel

Sadece constructor var. Product olan JLabel elemanından ConcreteProduct olan darkModeLabel elemanı extend ediliyor.

darkModeMenu

Sadece constructor var. Product olan JMenu elemanından ConcreteProduct olan darkModeMenu elemanı extend ediliyor.

darkModeMenuBar

Sadece constructor var. Product olan JMenuBar elemanından ConcreteProduct olan darkModeMenuBar elemanı extend ediliyor.

darkModeMenuItem

Sadece constructor var. Product olan JMenuItem elemanından ConcreteProduct olan darkModeMenuItem elemanı extend ediliyor.

darkModePanel

Sadece constructor var. Product olan JPanel elemanından ConcreteProduct olan darkModePanel elemanı extend ediliyor.

darkModeTextArea

Sadece constructor var. Product olan JTextArea elemanından ConcreteProduct olan darkModeTextArea elemanı extend ediliyor.

darkModeTextField

Sadece constructor var. Product olan JTextField elemanından ConcreteProduct olan darkModeTextField elemanı extend ediliyor.

Command

Command deseni için oluşturulan interface. İçinde execute() var.

UndoableCommand

Bu interface delete metodu içeriyoruz aynı zamanda kendi de Command interface'ini implement ediyor.

UndoCommand

UndoableCommand interface'ini impelent ediyor. Command'den implement ettiği execute metodunda Undo butonuna basıldığında ne olacağı yazılı, alamaz ise hata mesajı veriyor. UndoableCommand'den implement ettiği delete metodunda ise textarea'da yapılan değişiklikler UndoManager class'ından oluşturulan manager nesnesi içerisinde sıraya alınıyor.

DosyaInterface

İçinde sadece dosyaİslemi() var.

IslemYap

DosyaInterface işlemini gerçekleştiriyor.

DosyaAc

DosyaInterface implement eder. Oradaki dosyaİslemi() tanımlı. Dosya açma işlemine göre yazıldı.

DosyaKapat

DosyaInterface implement eder. Metodunun içinde çıkmak için gerekli satır var.

DosyaKaydet

DosyaInterface içerisindeki işlem dosya kaydetme işlemine göre yazıldı. İçindeki dosyaİslemi() metodunda dosyanın boş olup olmadığını kontrol eder ve kaydeder.

DosyaOlustur

DosyaInterface implement eder. Metodunda dosya oluşturur.

HashSetIterator

Iterator deseni burada yapılıyor. Iteratorumuzu HashSet için kullanacağımız için class ismimizi HashSetIterator yaptık.

Search

Arama ve kelime değiştirme işlemleri burada gerçekleştiriliyor bunu yaparken de ArrayList'lerden faydalanıyoruz. Search butonuna kaç kez basıldığını tutuyoruz ki metin belgemizde tüm kelimelerde

gezinebilelim. Bu Class için Singleton yapısı oluşturuyoruz. searchAreaButton metodu Search butonunun işlevini tanımladığımız yer. searchTextArea metodunda highlight'lı kelimeleri temizleyip, daha önce kelime aratılıp aratılmadığına bakıyoruz. Burada Search butonuna kaç kez basıldığını tuttuğumuz değişkenimiz sayesinde kaç kelime olduğunu bulup, içerisinde dolaşıyoruz bu esnada da gereken zamanlarda bu değerimizi arttırıyoruz. isContainWord bu metodumuzda ise kelime var mı yok mu diye bakıyoruz, bunu yaparken de metodumuzda ragex kullanıyoruz. replaceWordButton Replace Word Button'ın işlevi tanımlı burada da. removeHighLight metodu da adından anlaşılacağı üzere highlight'ları kaldırmak için var.

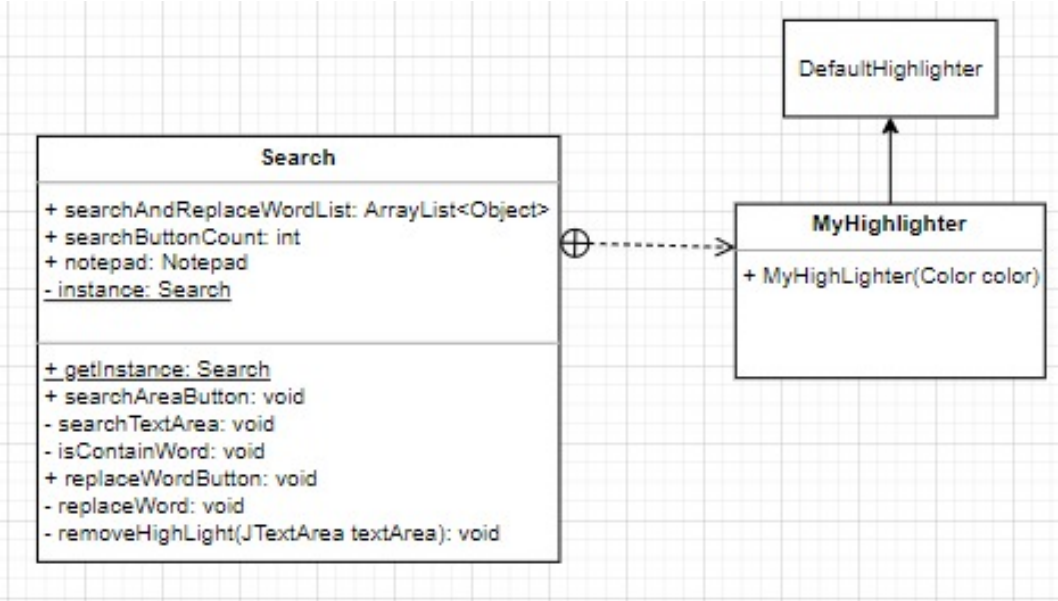
Words

words.txt, TextArea ve doğru olan kelimeleri tutmak için ayrı ayrı HashSet'lerimiz var. readWords metodu words.txt'yi okumak için var, eğer bulunamazsa hata mesajı veriyor. controlButton bu metotta kelimelerin doğruluğunu kontrol eden Control butonunun işlevi tanımlı. isNumeric ise karakterin sayı olup, olmadığını kontrol ediyor.

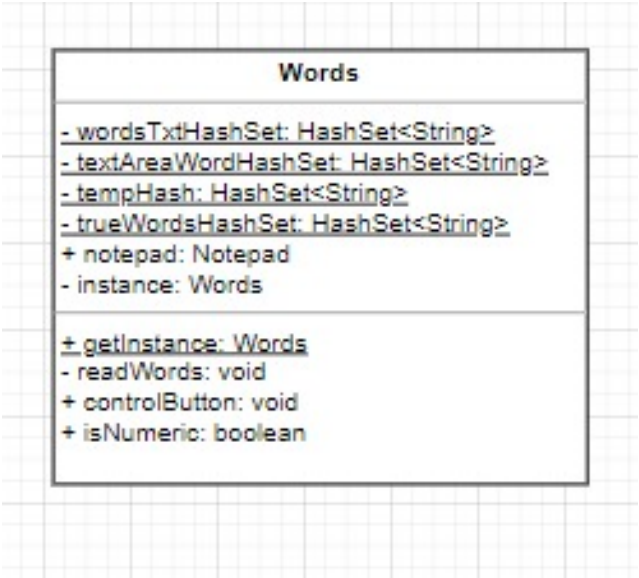
UML Diagramları

- Notepad.java

- Search.java

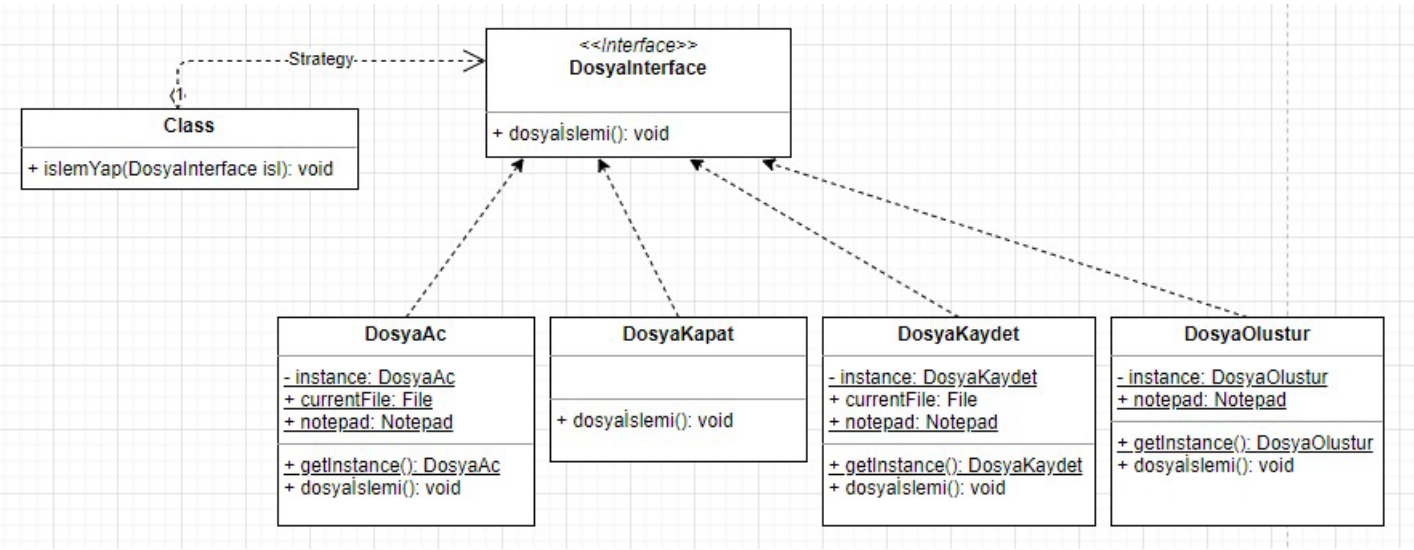


- Words.java



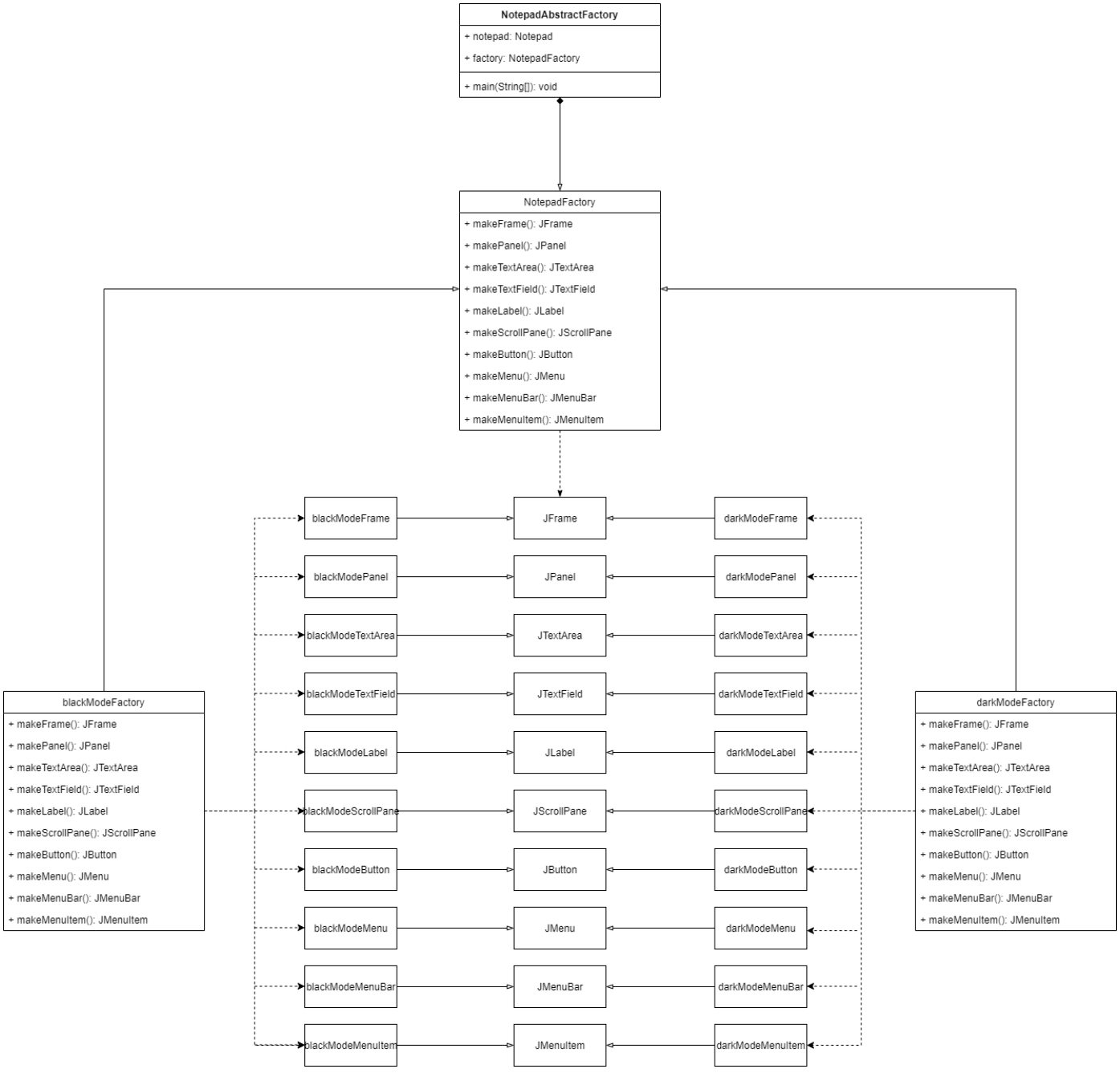
Tasarım Desenleri (Design Patterns)

- Strategy



Strategy tasarım desenini gerçekleştirdiğimiz dosya işlemleri üzerinde uyguladık. Bu amaçla DosyaInterface adında bir interface oluşturduk ve her bir işlemi ayrı bir sınıfa böldük. Sonra bu interface'i implemente ettik. Sınıflara böldüğümüz bu işlemleri, islemYap adlı sınıf içerisinde kullanarak tasarım desenimizi gerçekleştirmiş oluyoruz.

Abstract Factory



Abstracy Factory tasarım desenini gerçekleştirme amacımız, notepad uygulamamıza tema seçenekleri eklemek istememizdir. Bu amaçla tasarım deseninde Abstract Factory yerine geçen ve herhangi bir tema kullanılmadığında varsayılan temayı oluşturacak olan

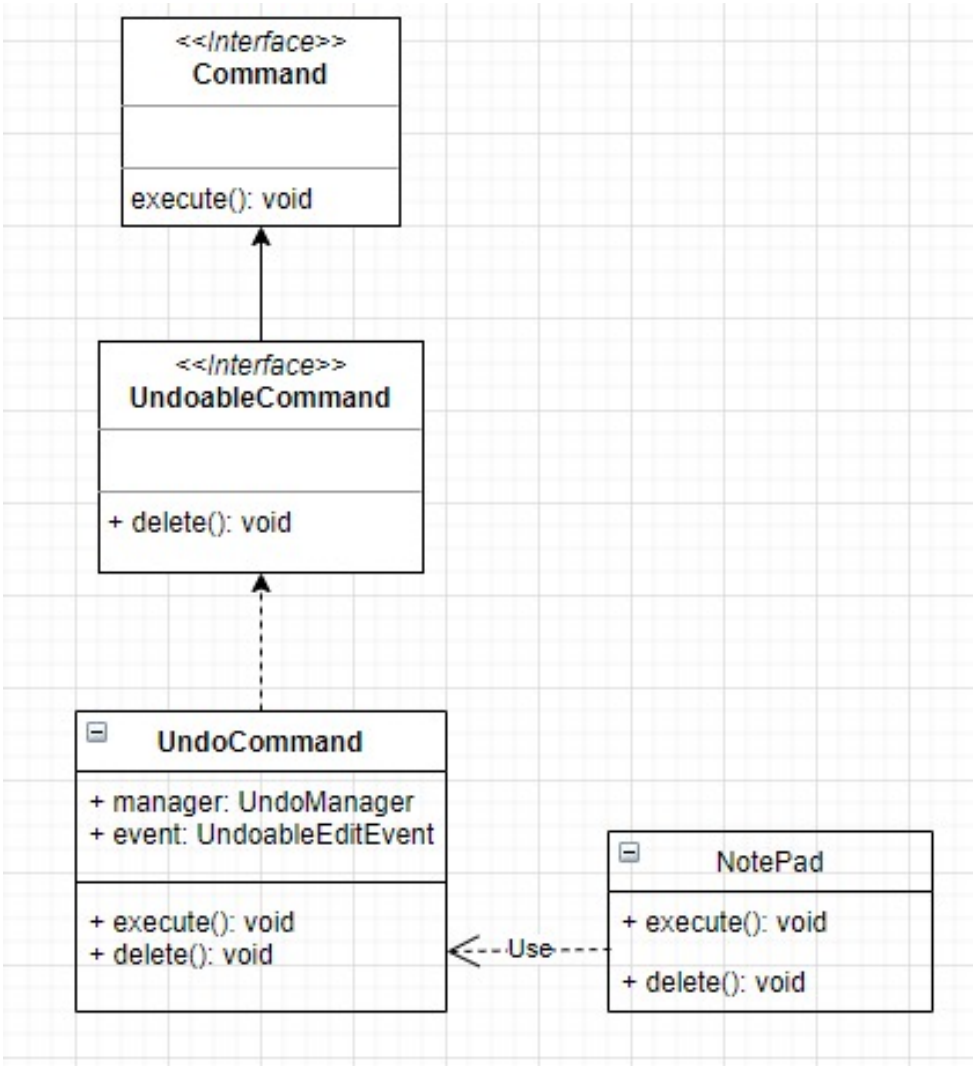
NotepadFactory isimli bir factory sınıfı oluşturduk. Eklemek istediğimiz blackMode ve darkMode isimli iki temaya ait factory sınıfları oluşturduk. Bunlar blackModeFactory ve darkModeFactory'dir. Bu sınıflar NotepadFactory'i extend etmektedir yani özelliklerini miras almaktadır. Bu factory sınıfları sayesinde nesne oluşturma işlemi kullanıcıdan soyutlanmış olur. Daha sonra temaya özgü olan GUI elementlerini oluşturduk. Bunları factoryler içerisinde tanımladık. Notepad sınıfı içerisinde bulunan NotepadCreator metoduna factory'leri parametre olarak vererek GUI'mizde temaların gerçekleşmesini sağlıyoruz. Ayrıca GUI içerisinde Menude yer alan 'Themes' kısmından tema değişikliğini gerçekleştirebiliyoruz. NotepadAbstractFactory içerisinde de hangi temanın çalışacağını belirleyen karar mekanizması bulunmaktadır.

Iterator

HashSetIterator adında bir sınıf oluşturduk. Bu sınıf, varsayılan Iterator interface'ini implemente etmektedir. Gereken metotları uygulayarak Iterator tasarım desenini gerçekleştirmiş oluyoruz. Oluşturduğumuz bu iteratorü

Words.java dosyası altında 72.satırdan itibaren while döngüsünü gerçekleştirerek kullanıyoruz.

Command



Command tasarım desenini notepad içerisinde yazılan son harfi geri alma işlemini yaptırarak gerçekleştiriyoruz.

Kaynak Kodları

blackModeButton

```
package blackMode;

import javax.swing.*;
import java.awt.*;

// Product olan JButton elemanindan ConcreteProduct olan blackModeButton elemani
extend ediliyor.

public class blackModeButton extends JButton {
    public blackModeButton(String name){
        super(name);
        setBackground(Color.BLACK);
        setForeground(Color.white);
    }
}
```

blackModeFactory

```
package blackMode;
import notepad.NotepadFactory;

import javax.swing.*;

// Abstract Factory Pattern sablonunda bulunan ConcreteFactory yapisini burada
olusturduk.
// Black Mode temanin gerceklestirimi de burada oluyor.
// new islemleri kullanicidan soyutlanip makeX fonksiyonlarina donusturuldu.
// new islemini gerceklestirdigimiz her bir swing elemani bir Product yerine
gecmektedir.

public class blackModeFactory extends NotepadFactory{
    @Override
    public JFrame makeFrame(String name) {
        return new blackModeFrame(name);
    }

    @Override
    public JPanel makePanel() {
        return new blackModePanel();
    }

    @Override
    public JTextArea makeTextArea() {
        return new blackModeTextArea();
    }

    @Override
    public JMenuBar makeMenuBar() {
        return new blackModeMenuBar();
    }

    @Override
    public JMenu makeMenu(String name) {
        return new blackModeMenu(name);
    }

    @Override
    public JLabel makeLabel(String text) {
        return new blackModeLabel(text);
    }
```

```
    @Override
    public JTextField makeTextField(int column) {
        return new blackModeTextField(column);
    }

    @Override
    public JMenuItem makeMenuItem(String name) {
        return new blackModeMenuItem(name);
    }

    @Override
    public JButton makeButton(String name) {
        return new blackModeButton(name);
    }
}
```

blackModeFrame

```
package blackMode;

import javax.swing.*;
import java.awt.*;

// Product olan JFrame elemanindan ConcreteProduct olan blackModeFrame elemani
extend ediliyor.

public class blackModeFrame extends JFrame {

    public blackModeFrame(String name){
        super(name);
        setBackground(Color.BLACK);

    }
}
```

blackModeLabel

```
package blackMode;

import javax.swing.*;
import java.awt.*;

// Product olan JLabel elemanindan ConcreteProduct olan blackModeLabel elemani
extend ediliyor.


public class blackModeLabel extends JLabel {
    public blackModeLabel(String name){
        super(name);
        setBackground(Color.BLACK);
        setForeground(Color.white);
    }
}
```

blackModeMenu

```
package blackMode;

import javax.swing.*;
import java.awt.*;

// Product olan JMenu elemanindan ConcreteProduct olan blackModeMenu elemani extend
ediliyor.
```

```
public class blackModeMenu extends JMenu {
    public blackModeMenu(String name){
        super(name);
        setBackground(Color.BLACK);
        setForeground(Color.white);
    }
}
```

blackModeMenuBar

```
package blackMode;

import javax.swing.*;
import java.awt.*;

// Product olan JMenuBar elemanindan ConcreteProduct olan blackModeMenuBar elemani
extend ediliyor.


public class blackModeMenuBar extends JMenuBar {
    public blackModeMenuBar(){
        setBackground(Color.BLACK);
    }
}
```

blackModeMenuItem

```
package blackMode;

import javax.swing.*;
import java.awt.*;

// Product olan JMenuItem elemanindan ConcreteProduct olan blackModeMenuItem
elemani extend ediliyor.


public class blackModeMenuItem extends JMenuItem {
    public blackModeMenuItem(String name){
        super(name);
        setBackground(Color.BLACK);
        setForeground(Color.white);
    }
}
```

blackModePanel

```
package blackMode;

import javax.swing.*;
import java.awt.*;

// Product olan JPanel elemanindan ConcreteProduct olan blackModePanel elemani
extend ediliyor.


public class blackModePanel extends JPanel {
    public blackModePanel() {
        setBackground(Color.black);
    }
}
```

blackModeTextArea

```
package blackMode;

import javax.swing.*;
import java.awt.*;

// Product olan JTextArea elemanindan ConcreteProduct olan blackModeTextArea
elemani extend ediliyor.

public class blackModeTextArea extends JTextArea {
    public blackModeTextArea(){
        setBackground(Color.DARK_GRAY);
        setForeground(Color.white);
    }
}
```

blackModeTextField

```
package blackMode;

import javax.swing.*;
import java.awt.*;

// Product olan JTextField elemanindan ConcreteProduct olan blackModeTextField
elemani extend ediliyor.

public class blackModeTextField extends JTextField {

    public blackModeTextField(int column){
        super(column);
        setBackground(Color.BLACK);
        setForeground(Color.white);
    }
}
```

darkModeButton

```
package darkMode;

import javax.swing.*;
import java.awt.*;

// Product olan JButton elemanindan ConcreteProduct olan darkModeButton elemani
extend ediliyor.


public class darkModeButton extends JButton {
    public darkModeButton(String name){
        super(name);
        setBackground(Color.DARK_GRAY);
        setForeground(Color.white);
    }
}
```

darkModeFactory

```
package darkMode;
import notepad.NotepadFactory;

import javax.swing.*;
```

```java
// Abstract Factory Pattern sablonunda bulunan ConcreteFactory yapisini burada
olusturduk.
// Dark Mode temanin gerceklestirimi de burada oluyor.
// new islemleri kullanicidan soyutlanip makeX fonksiyonlarina donusturuldu.
// new islemini gerceklestirdigimiz her bir swing elemani bir Product yerine
gecmektedir.

public class darkModeFactory extends NotepadFactory{
    @Override
    public JFrame makeFrame(String name) {
        return new darkModeFrame(name);
    }

    @Override
    public JPanel makePanel() {
        return new darkModePanel();
    }

    @Override
    public JTextArea makeTextArea() {
        return new darkModeTextArea();
    }

    @Override
    public JMenuBar makeMenuBar() {
        return new darkModeMenuBar();
    }

    @Override
    public JMenu makeMenu(String name) {
        return new darkModeMenu(name);
    }

    @Override
    public JLabel makeLabel(String text) {
        return new darkModeLabel(text);
    }

    @Override
    public JTextField makeTextField(int column) {
        return new darkModeTextField(column);
    }

    @Override
    public JMenuItem makeMenuItem(String name) {
        return new darkModeMenuItem(name);
    }

    @Override
    public JButton makeButton(String name) {
        return new darkModeButton(name);
    }
}
```

darkModeFrame

```java
package darkMode;

import javax.swing.*;
import java.awt.*;

// Product olan JFrame elemanindan ConcreteProduct olan darkModeFrame elemani
extend ediliyor.

public class darkModeFrame extends JFrame {

    public darkModeFrame(String name){
```

```
        super(name);
        setBackground(Color.DARK_GRAY);

    }
}
```

darkModeLabel

```
package darkMode;

import javax.swing.*;
import java.awt.*;

// Product olan JLabel elemanindan ConcreteProduct olan darkModeLabel elemani
extend ediliyor.

public class darkModeLabel extends JLabel {
    public darkModeLabel(String name){
        super(name);
        setBackground(Color.DARK_GRAY);
        setForeground(Color.white);
    }
}
```

darkModeMenu

```
package darkMode;

import javax.swing.*;
import java.awt.*;

// Product olan JMenu elemanindan ConcreteProduct olan darkModeMenu elemani extend
ediliyor.

public class darkModeMenu extends JMenu {
    public darkModeMenu(String name){
        super(name);
        setBackground(Color.DARK_GRAY);
        setForeground(Color.white);
    }
}
```

darkModeMenuBar

```
package darkMode;

import javax.swing.*;
import java.awt.*;

// Product olan JMenuBar elemanindan ConcreteProduct olan darkModeMenuBar elemani
extend ediliyor.

public class darkModeMenuBar extends JMenuBar {
    public darkModeMenuBar(){
        setBackground(Color.DARK_GRAY);
    }
}
```

darkModeMenuItem

```
package darkMode;

import javax.swing.*;
```

```java
import java.awt.*;

// Product olan JMenuItem elemanindan ConcreteProduct olan darkModeMenuItem elemani
extend ediliyor.

public class darkModeMenuItem extends JMenuItem {
    public darkModeMenuItem(String name){
        super(name);
        setBackground(Color.DARK_GRAY);
        setForeground(Color.white);
    }
}
```

darkModePanel

```java
package darkMode;

import javax.swing.*;
import java.awt.*;

// Product olan JPanel elemanindan ConcreteProduct olan darkModePanel elemani
extend ediliyor.

public class darkModePanel extends JPanel {
    public darkModePanel() {
        setBackground(Color.DARK_GRAY);
    }
}
```

darkModeTextArea

```java
package darkMode;

import javax.swing.*;
import java.awt.*;

// Product olan JTextArea elemanindan ConcreteProduct olan darkModeTextArea elemani
extend ediliyor.

public class darkModeTextArea extends JTextArea {
    public darkModeTextArea(){
        setBackground(Color.GRAY);
    }
}
```

darkModeTextField

```java
package darkMode;

import javax.swing.*;
import java.awt.*;

// Product olan JTextField elemanindan ConcreteProduct olan darkModeTextField
elemani extend ediliyor.

public class darkModeTextField extends JTextField {

    public darkModeTextField(int column){
        super(column);
        setBackground(Color.DARK_GRAY);
        setForeground(Color.white);
    }
}
```

Command

```
package notepad;

//command patterni için interface
public interface Command {
    public void execute();
}
```

DosyaAc

```
package notepad;


import javax.swing.*;
import java.io.File;
import java.util.Scanner;

//DosyaInterface içerisindeki işlem
//Dosya acma işlemine göre yazıldı.
public class DosyaAc implements DosyaInterface {
    private static DosyaAc instance;
    public static File currentFile; //static olarak tanımlandı cunku dosya kaydetme
işleminde dosya daha onceden acıldıysa kaydedilecek yer sormadan
                                    //direkt olarak o dosyanın uzerine kaydetme
saglandı

    public static Notepad notepad = Notepad.getInstance();

    public static DosyaAc getInstance(){
        if (instance == null){
            instance = new DosyaAc();
        }
        return instance;
    }

    @Override
    public void dosyaİslemi() {
        JFileChooser fileChooser = new JFileChooser();
        int r = fileChooser.showOpenDialog(null);
        if(r == JFileChooser.APPROVE_OPTION){
            String fileName = fileChooser.getSelectedFile().getName();
            notepad.getFrame().setTitle(fileName);
            File file = new File(fileChooser.getSelectedFile().getAbsolutePath());
            currentFile = file;
            try{
                String fileString = "";
                Scanner textFile = new Scanner(file);
                while(textFile.hasNextLine()){
                    fileString = fileString + textFile.nextLine() + "\n";
                }
                notepad.getTextArea().setText(fileString);
            }catch (Exception evt){
                JOptionPane.showMessageDialog(null,evt.getMessage());
            }
        }
        else{
            JOptionPane.showMessageDialog(null,"Dosya açma iptal edildi.");
        }
    }
}
```

DosyaInterface

```
package notepad;

//Strategy metodu için interface
//sadece bir dosya işlemi tanımlandı
public interface  DosyaInterface {
    void dosyaİslemi();
}
```

DosyaKapat

```
package notepad;

//DosyaInterface içerisindeki işlem
//Dosya kapatma işlemine göre yazıldı.
public class DosyaKapat implements DosyaInterface{
    @Override
    public void dosyaİslemi() {
        System.exit(1);
    }
}
```

DosyaKaydet

```
package notepad;


import javax.swing.*;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;

//DosyaInterface içerisindeki işlem
//Dosya kaydetme işlemine göre yazıldı.
public class DosyaKaydet implements DosyaInterface {
    private static DosyaKaydet instance;
    File currentFile = DosyaAc.currentFile;
    public static Notepad notepad = Notepad.getInstance();

    public static DosyaKaydet getInstance(){
        if (instance == null){
            instance = new DosyaKaydet();
        }
        return instance;
    }

    @Override
    public void dosyaİslemi() {
        if(DosyaAc.currentFile != null){
            try {
                File file = new File(currentFile.getAbsolutePath());
                DosyaAc.currentFile = file;
                FileWriter fileWriter = new FileWriter(file);
                BufferedWriter w = new BufferedWriter(fileWriter);
                w.write(notepad.getTextArea().getText());
                w.flush();
                w.close();
            } catch (IOException e) {
                JOptionPane.showMessageDialog(null, "File Not Found!");
            }
        }
        else {
```

```
            JFileChooser j = new JFileChooser();
            int r = j.showSaveDialog(null);
            if (r == JFileChooser.APPROVE_OPTION) {
                File file = new File(j.getSelectedFile().getAbsolutePath() +
".txt");
                try {
                    FileWriter fileWriter = new FileWriter(file);
                    BufferedWriter w = new BufferedWriter(fileWriter);
                    w.write(notepad.getTextArea().getText());
                    w.flush();
                    w.close();
                    DosyaAc.currentFile = file;
                    notepad.getFrame().setTitle(j.getSelectedFile().getName());
                } catch (IOException e) {
                    JOptionPane.showMessageDialog(null, "File Not Found!");
                }
            }
        }
    }
}
```

DosyaOlustur

```
package notepad;


import java.io.File;

//DosyaInterface içerisindeki işlem
//Dosya olusturma işlemine göre yazıldı.
public class DosyaOlustur implements DosyaInterface {

    private static DosyaOlustur instance;
    //File currentFile = DosyaAc.currentFile;
    public static Notepad notepad = Notepad.getInstance();

    public static DosyaOlustur getInstance(){
        if (instance == null){
            instance = new DosyaOlustur();
        }
        return instance;
    }
    @Override
    public void dosyaİslemi() {
        notepad.getFrame().setTitle("Yeni Metin Belgesi");
        notepad.getTextArea().setText(" ");
        DosyaAc.currentFile = null;
    }
}
```

HashSetIterator

```
package notepad;

import java.util.HashSet;
import java.util.Iterator;

// Iterator Pattern Burada Gerceklestirildi.

// Iteratorumuzu HashSet icin kullanacagimiz icin class ismimize HashSetIterator
verdik.

public class HashSetIterator implements Iterator {

    // Kullanilacak olan HashSetin referansini olusturduk.
    HashSet<String> hs;
```

```
    // Indexi tutabilmek icin bir int deger olusturduk.
    int current_index = 0;

    public HashSetIterator(HashSet<String> hs) {
        this.hs = hs;
    }

    @Override
    public boolean hasNext() {
        return (current_index<hs.size()?true:false);
    }

    @Override
    public Object next() {
        Object[] temp = hs.toArray();
        return temp[current_index++];
    }
}
```

islemYap

```
package notepad;

//DosyaInterface islemini gerceklestiriyor
public class IslemYap {
    public void islemYap(DosyaInterface isl){
        isl.dosyaİslemi();
    }
}
```

Notepad

```
package notepad;

import javax.swing.*;
import javax.swing.event.UndoableEditEvent;
import javax.swing.event.UndoableEditListener;
import javax.swing.undo.UndoManager;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;


// Abstract Factory pattern icerisindeki Client elemani bu classtir.
// NotepadCreator adli metod, icerisine bir factory alarak make islemleri
gerceklestirir.


public class Notepad  implements  ActionListener{
    private NotepadFactory factory;
    private JFrame frame;
    private JPanel panel;
    private JPanel panel1;
    private JPanel panel2;
    private JMenuBar mb;
    private JMenu m1;
    private JMenuItem newMenuItem;
    private JMenuItem openMenuItem;
    private JMenuItem saveMenuItem;
    private JMenuItem exitMenuItem;
    private JLabel infoTextField1;
    private JLabel infoTextField2;
    private JTextField textField1;
    private JTextField textField2;
    private JButton searchButton;
    private JButton delButton;
```

```
private JButton replaceButton;
private JButton controlButton;
private JTextArea textArea;
private JScrollPane jScrollPane;


private JMenu themeMenu;
private JMenuItem stockMenuItem;
private JMenuItem darkModeMenuItem;
private JMenuItem blackModeMenuItem;

private JPanel panel3;

UndoManager manager = new UndoManager();
UndoableEditEvent event;
UndoableCommand undoablecommand = new UndoCommand(manager,event);
private static Notepad instance;

public static Notepad getInstance() {
    if (instance == null) {
        instance = new Notepad();
    }
    return instance;
}

private Notepad(){}

public NotepadFactory getFactory() {
    return factory;
}

public void setFactory(NotepadFactory factory) {
    this.factory = factory;
}

public JFrame getFrame() {
    return frame;
}

public void setFrame(JFrame frame) {
    this.frame = frame;
}

public JPanel getPanel() {
    return panel;
}

public void setPanel(JPanel panel) {
    this.panel = panel;
}

public JTextField getTextField1() {
    return textField1;
}

public JTextField getTextField2() {
    return textField2;
}

public JButton getDelButton() {
    return delButton;
}

public void setDelButton(JButton delButton) {
    this.delButton = delButton;
}

public JButton getReplaceButton() {
```

```
        return replaceButton;
    }

    public JTextArea getTextArea() {
        return textArea;
    }

    public void setTextArea(JTextArea textArea) {
        this.textArea = textArea;
    }

    public JMenu getThemeMenu() {
        return themeMenu;
    }

    public void setThemeMenu(JMenu themeMenu) {
        this.themeMenu = themeMenu;
    }


    public static void setInstance(Notepad instance) {
        Notepad.instance = instance;
    }

    public void NotepadCreator(NotepadFactory x) {
        getInstance();
        factory = x;
        frame = factory.makeFrame("Notepad");
        panel = factory.makePanel();
        panel1 = factory.makePanel();
        panel2 = factory.makePanel();
        mb = factory.makeMenuBar();
        m1 = factory.makeMenu("File");
        newMenuItem = factory.makeMenuItem("New");
        openMenuItem = factory.makeMenuItem("Open");
        saveMenuItem = factory.makeMenuItem("Save");
        exitMenuItem = factory.makeMenuItem("Exit");

        infoTextField1 = factory.makeLabel("Aranacak Kelime:");
        infoTextField2 = factory.makeLabel("Degisecek Kelime:");
        textField1 = factory.makeTextField(25);
        textField2 = factory.makeTextField(25);
        searchButton = factory.makeButton("Search");
        delButton = factory.makeButton("Undo");
        replaceButton = factory.makeButton("Replace");
        controlButton = factory.makeButton("Control");

        textArea = factory.makeTextArea();
        textArea.setLineWrap (true);
        textArea.setWrapStyleWord (false);
        textArea.getDocument().addUndoableEditListener(new UndoableEditListener() {
            @Override
            public void undoableEditHappened(UndoableEditEvent event) {
                undoablecommand.delete(event);
            }
        });
        jScrollPane = factory.makeScrollPane(textArea);

jScrollPane.setHorizontalScrollBarPolicy(ScrollPaneConstants.HORIZONTAL_SCROLLBAR_N
EVER);

jScrollPane.setVerticalScrollBarPolicy(ScrollPaneConstants.VERTICAL_SCROLLBAR_AS_NE
EDED);

        newMenuItem.addActionListener(this);
        openMenuItem.addActionListener(this);
        saveMenuItem.addActionListener(this);
        exitMenuItem.addActionListener(this);
```

```
        textField1.addActionListener(this);
        textField2.addActionListener(this);

        m1.add(newMenuItem);
        m1.add(openMenuItem);
        m1.add(saveMenuItem);
        m1.add(exitMenuItem);
        mb.add(m1);

        searchButton.addActionListener(this);
        delButton.addActionListener(this);
        replaceButton.addActionListener(this);
        controlButton.addActionListener(this);

        panel.setLayout(new FlowLayout(FlowLayout.CENTER,25,5));
        panel.add(infoTextField1);
        panel.add(textField1);
        panel.add(searchButton);
        panel.add(delButton);

        panel1.setLayout(new FlowLayout(FlowLayout.CENTER,20,5));
        panel1.add(infoTextField2);
        panel1.add(textField2);
        panel1.add(replaceButton);
        panel1.add(controlButton);

        panel3 = factory.makePanel();
        panel3.setLayout(new CardLayout(10,20));
        panel3.add(jScrollPane);


        panel2.setLayout(new BoxLayout(panel2, BoxLayout.Y_AXIS));
        panel2.add(panel);
        panel2.add(panel1);


        themeMenu = factory.makeMenu("Themes");
        stockMenuItem = factory.makeMenuItem("Stock Theme");
        darkModeMenuItem = factory.makeMenuItem("Dark Mode");
        blackModeMenuItem = factory.makeMenuItem("Black Mode");

        stockMenuItem.addActionListener(this);
        darkModeMenuItem.addActionListener(this);
        blackModeMenuItem.addActionListener(this);

        themeMenu.add(stockMenuItem);
        themeMenu.add(darkModeMenuItem);
        themeMenu.add(blackModeMenuItem);

        mb.add(themeMenu);
        frame.setJMenuBar(mb);
        frame.setLayout(new BorderLayout());
        frame.add(panel2,BorderLayout.NORTH);
        frame.add(panel3,BorderLayout.CENTER);

        frame.setSize(700, 700);
        frame.setLocationRelativeTo(null);
        textField2.setEnabled(false);
        replaceButton.setEnabled(false);

        frame.setVisible(true);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    }

    @Override
    public void actionPerformed(ActionEvent e) {
```

```
        String s = e.getActionCommand();  // yukarıda ekledigimiz
actionlistener'lar burada alınıyor
        Search search = Search.getInstance();  // Search,replace işlemleri için
Search nesnesi tanımlandı
        Words words = Words.getInstance();      //Kontrol için nesne tanımlandı.
        IslemYap islem = new IslemYap();        //Dosya işlemleri strategy
paternine göre yapıldı. Hepsi benzer işlemler yaptığı için DosyaInterface
arayüzünden türeyen bi IslemYap nesnesi olusturuldu
                                                //Sadece içerisinde islem yap olan
ve bu fonksiyonda DosyaInterface tipinden islem alan bi fonksiyon tanımlandı
                                                //Bu işlemler için IslemYap
turunden bi nesne oluşturuldu

        if (s.equals("New")) {
            islem.islemYap(new DosyaOlustur());   //islemYap metodunun icerisinde
DosyaOlustur turunden DosyaInterface işlemi gönderildi.
        } else if (s.equals("Open")) {
            islem.islemYap(new DosyaAc());          //islemYap metodunun icerisinde
DosyaAc turunden DosyaInterface işlemi gönderildi.
        } else if (s.equals("Exit")) {
            islem.islemYap(new DosyaKapat());       //islemYap metodunun icerisinde
DosyaKapat turunden DosyaInterface işlemi gönderildi.
        } else if (s.equals("Save")) {
            islem.islemYap(new DosyaKaydet());      //islemYap metodunun icerisinde
DosyaKaydet turunden DosyaInterface işlemi gönderildi.
        } else if (s.equals("Search")) {
            search.searchAreaButton();
        } else if (s.equals("Replace")) {
            search.replaceWordButton();
            replaceButton.setEnabled(false);
            textField2.setEnabled(false);
        } else if (s.equals("Undo")) {
            undoablecommand.execute();
        } else if (s.equals("Control")) {
            words.controlButton();
        } else if (s.equals("Stock Theme")) {
            frame.setVisible(false);
            frame.dispose();
            NotepadAbstractFactory.main(new String[] {"Stock Theme"});
        } else if (s.equals("Dark Mode")) {
            frame.setVisible(false);
            frame.dispose();
            NotepadAbstractFactory.main(new String[] {"Dark Mode"});
        } else if (s.equals("Black Mode")) {
            frame.setVisible(false);
            frame.dispose();
            NotepadAbstractFactory.main(new String[] {"Black Mode"});
        }

    }
}
```

NotepadAbstractFactory

```
package notepad;

import blackMode.blackModeFactory;
import darkMode.darkModeFactory;

// Abstract Factory Pattern icerisindeki Client elemani olan Notepad classinin
factoryi arguman olarak alan NotepadCreator metodunu burada kullaniyoruz.
// main metoduna gelen argumana gore temamiz cagiriliyor.

// Ayrica her temaya adit swing elemanlari, temaya ait olan paket altinda
tanimlanmistir.

public class NotepadAbstractFactory {
```

```
    public static void main(String[] args){
        Notepad notepad = Notepad.getInstance();
        NotepadFactory factory = null;

        if (args.length > 0) {
            if ("Dark Mode".equals(args[0])) {
                factory = new darkModeFactory();
            } else if ("Black Mode".equals(args[0])) {
                factory = new blackModeFactory();
            } else if ("Stock Theme".equals(args[0])) {
                factory = new NotepadFactory();
            }
        }
        if (factory == null) {
            factory = new NotepadFactory();
        }
        notepad.NotepadCreator(factory);
    }
}
```

NotepadFactory

```
package notepad;

import javax.swing.*;

// Abstract Factory Pattern sablonunda bulunan AbstractFactory yapisini burada
olusturduk.
// Default temanin gerceklestirimi de burada oluyor.
// new islemleri kullanicidan soyutlanip makeX fonksiyonlarina donusturuldu.
// new islemini gerceklestirdigimiz her bir swing elemani bir Product yerine
gecmektedir.

public class NotepadFactory {

    public JFrame makeFrame(String name){
        return new JFrame(name);
    }

    public JPanel makePanel(){
        return new JPanel();
    }

    public JTextArea makeTextArea(){
        return new JTextArea();
    }

    public JMenuBar makeMenuBar(){
        return new JMenuBar();
    }

    public JMenu makeMenu(String name){
        return new JMenu(name);
    }

    public JLabel makeLabel(String text){
        return new JLabel(text);
    }

    public JTextField makeTextField(int column){
        return new JTextField(column);
    }

    public JMenuItem makeMenuItem(String name){
        return new JMenuItem(name);
    }
```

```
    public JButton makeButton(String name){
        return new JButton(name);
    }

    public  JScrollPane makeScrollPane(JTextArea textArea){ return  new
JScrollPane(textArea);}
}
```

Search

```
package notepad;


import javax.swing.*;
import javax.swing.text.BadLocationException;
import javax.swing.text.DefaultHighlighter;
import javax.swing.text.Document;
import javax.swing.text.Highlighter;
import java.awt.*;
import java.util.ArrayList;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class Search {
    // Arama yapmak ve kelimeyi değiştirmek için bu arrayListi kullanıyoruz.
    // 0.indexte kelimenin kendisi, 1.indexten itibaren de kelimenin text
içerisinde hangi indexlerde olduğunu tutuyor.
    ArrayList<Object> searchAndReplaceWordList = new ArrayList<>();
    // Search butonuna kaç kere basıldığını tutuyor. Böylece text içerisindeki tüm
kelimelerde gezinebiliyoruz.
    int searchButtonCount;
    // Notepad classintan notepad adinda bir instance olusturmak icin Notepad
classi icerisinde bulunan getInstance metodunu kullaniyoruz.
    Notepad notepad = Notepad.getInstance();

    // Search Classi icin Singleton yapisini olusturduk.
    private static Search instance;
    public static Search getInstance(){
        if(instance==null){
            instance = new Search();
        }
        return instance;
    }


    public void searchAreaButton(){
        // Search butonunun işlevini burada tanımladık.
        // ArrayListimiz boş değilse içeride kelime vardır anlamına gelir.
        notepad.getTextField2().setEnabled(true);
        notepad.getReplaceButton().setEnabled(true);
        if(!searchAndReplaceWordList.isEmpty()){

if(!searchAndReplaceWordList.get(0).equals(notepad.getTextField1().getText())){
                searchAndReplaceWordList = new ArrayList<>();
                searchButtonCount = 0;
            }
        }
        searchTextArea();
    }

    private void searchTextArea(){
        // Highlightlı kelimeleri temizliyoruz.
        removeHighLight(notepad.getTextArea());
        Highlighter highlight = notepad.getTextArea().getHighlighter();
        // Eğer hiç aratılmamışsa kelime var mı diye kontrol ediyoruz.
        if (searchButtonCount == 0){
            isContainWord();
```

```
            searchButtonCount++;
        }
        // Eğer textarea içerisindeki son kelimedeysek count değerini 1 yaparak en
başa dönüyoruz.
        // 1 yapmamızın nedeni searchAndReplaceWordList listesinde 0.indexte
kelimenin kendisi 1.indexten itibaren konumları yer alıyor.
        if(searchButtonCount == searchAndReplaceWordList.size()){
            searchButtonCount = 1;
        }
        // Eğer kelime birden fazla varsa ve sonuna gelinmediyse işaretlemeye devam
ediyoruz.
        if(searchButtonCount != 0 && searchButtonCount <
searchAndReplaceWordList.size()){
            try {
                highlight.addHighlight((int)
searchAndReplaceWordList.get(searchButtonCount), ((int)
searchAndReplaceWordList.get(searchButtonCount) + ((String)
searchAndReplaceWordList.get(0)).length()), highLighter);
            } catch (BadLocationException e) {
                e.printStackTrace();
            }
            searchButtonCount++;
        }

    }

    private void isContainWord(){
        // Kelimenin olup olmadığını bulmak için bu metodu kullanıyoruz.
        // Bu metotta regex kullandık.
        String fullString = notepad.getTextArea().getText();
        String partWord = notepad.getTextField1().getText();
        String pattern = "\\b" + partWord + "\\b";
        Pattern p = Pattern.compile(pattern);
        Matcher m = p.matcher(fullString);
        // listenin 0. indexine kelimenin kendisini ekliyoruz.
        searchAndReplaceWordList.add(partWord);
        int position;
        while(m.find()){
            position = m.start();
            // listenin 1. indexinden itibaren kelimenin konumlarını ekliyoruz.
            searchAndReplaceWordList.add(position);
        }
    }

    public void replaceWordButton(){
        // Replace butonunun işlevini burada tanımladık.
        replaceWord();
        searchAndReplaceWordList = new ArrayList<>();
        isContainWord();
        searchButtonCount--;
        notepad.getTextField2().setText("");
    }

    private void replaceWord() {
        // Kelimeleri değiştirme işlemini burada yapıyoruz.
        StringBuilder finalText = new StringBuilder("");
        Document document = notepad.getTextArea().getDocument();
        StringBuilder text = null;
        try {
            text = new StringBuilder(document.getText(0, document.getLength()));
        } catch (BadLocationException e) {
            e.printStackTrace();
        }
        String takeReplaceWord = notepad.getTextField2().getText();
        finalText = text.replace((int)
searchAndReplaceWordList.get(searchButtonCount - 1), (int)
searchAndReplaceWordList.get(searchButtonCount - 1) + ((String)
searchAndReplaceWordList.get(0)).length(), takeReplaceWord);
```

```
        notepad.getTextArea().setText(finalText.toString());
    }

    // Highlighter
    static class MyHighLighter extends DefaultHighlighter.DefaultHighlightPainter {

        public MyHighLighter(Color color) {
            super(color);
        }
    }

    DefaultHighlighter.DefaultHighlightPainter highLighter = new
MyHighLighter(Color.yellow);

    private void removeHighLight(JTextArea textArea) {
        Highlighter removeHighlighter = textArea.getHighlighter();
        Highlighter.Highlight[] remove = removeHighlighter.getHighlights();

        for (Highlighter.Highlight highlight : remove) {
            if (highlight.getPainter() instanceof MyHighLighter) {
                removeHighlighter.removeHighlight(highlight);
            }
        }
    }

}
```

UndoCommand

```
package notepad;

import javax.swing.*;
import javax.swing.event.UndoableEditEvent;
import javax.swing.undo.UndoManager;

public class UndoCommand implements UndoableCommand {

    //swing textarea'da geri alma isleminin yapılması için undomanager tanımlandı
    public UndoManager manager;
    //geri alma tusuna basılma eventi
    UndoableEditEvent event;
    //Constructor
    public UndoCommand(UndoManager manager, UndoableEditEvent event){
        this.manager = manager;
        this.event = event;
    }

    //Undo butonuna basıldığında bu fonksiyon çalışıyor
    //Fonksiyon UndoManager ile geri alınabilecek ögeleri her buton tıklamasında
geri almaya başlıyor.
    //Alamazsa bir hata mesajı ekrana bastırılıyor.
    @Override
    public void execute() {
        if(manager.canUndo()){
            manager.undo();
        }
        else{
            JOptionPane.showMessageDialog(null, "Geri alınacak öğe yok..");
        }
    }

    //textarea'da yapılan değişiklikler UndoManager classından oluşturulan manager
nesnesi içerisinde sıraya alınıyor.
    @Override
    public void delete(UndoableEditEvent editEvent) {
        manager.addEdit(editEvent.getEdit());
    }
```

```
}
```

UndoableCommand

```
package notepad;

import javax.swing.event.UndoableEditEvent;

//text area'da yapılan son islemleri kendi içerisinde sıraya alır
//undocommand içerisinde execute edildiginde bu sırayla geri almayı saglar
public interface UndoableCommand extends Command {
    public void delete(UndoableEditEvent editEvent);
}
```

Words

```
package notepad;


import javax.swing.*;
import java.io.File;
import java.io.FileNotFoundException;
import java.util.ArrayList;
import java.util.HashSet;
import java.util.Scanner;
import java.util.StringTokenizer;

public class Words {
    // words.txt içerisinde Kelimeleri bu hashset içerisinde tutuyoruz.
    private static HashSet<String> wordsTxtHashSet;
    // TextArea içerisindeki kelimeleri bu HashSet içerisinde tutuyoruz.
    private static HashSet<String> textAreaWordHashSet;
    // control metodunda hata almamak için gecici tempHash oluşturduk.
    private static HashSet<String> tempHash;
    // Doğru olan kelimeleri bu hashset içerisinde tutuyoruz.
    private static HashSet<String> trueWordsHashSet;

    // Notepad classindan bir instance olusturuyoruz.
    Notepad notepad = Notepad.getInstance();

    private Words(){
        wordsTxtHashSet = new HashSet<>();
        readWords();
    }

    // Words classi icin Singleton patternini kullaniyoruz.
    private static Words instance;
    public static Words getInstance(){
        if(instance==null){
            instance = new Words();
        }
        return instance;
    }

    private void readWords(){
        try {
            Scanner textFile = new Scanner(new File("words.txt"));
            while (textFile.hasNext()) {
                wordsTxtHashSet.add(textFile.next().trim());
            }
                textFile.close();
        } catch (FileNotFoundException e) {
                System.out.println("Words File Not Found!");
            }

    }
```

```
    public void controlButton(){
        // Kelimelerin doğruluğunu kontrol eden Control butonunun işlevini burada
tanımladık.
        // StringTokenizer sayesinde aşağıdaki işaretleri metinden temizlenerek
sadece kelimeleri  textAreaWordHashSet içerisine ekliyoruz.
        String regex = "[!._,;-':@?\n ]";
        StringTokenizer str = new StringTokenizer(notepad.getTextArea().getText(),
regex);
        textAreaWordHashSet = new HashSet<String>();
        tempHash = new HashSet<>();
        while(str.hasMoreTokens()){
            textAreaWordHashSet.add(str.nextToken().toLowerCase());
        }

        // textArea içindeki kelimelerden oluşan trueWordsHashSet adında bir hs
oluşturuyoruz.
        // trueWords içerisinde sadece doğru kelimeleri bırakıyoruz.
        // trueWords içerisindeki doğru kelimeleri textArea içerisinden çıkartarak
yanlış olabilecek kelimeler ile başbaşa kalıyoruz.
        trueWordsHashSet = new HashSet<>(textAreaWordHashSet);
        trueWordsHashSet.retainAll(wordsTxtHashSet);
        textAreaWordHashSet.removeAll(trueWordsHashSet);

        // Iterator Pattern'ini burada kullaniyoruz.
        HashSetIterator iterator = new HashSetIterator(textAreaWordHashSet);
            // Burada sayısal değerleri de doğru kelime olarak alıyoruz.
        while(iterator.hasNext()){
            String current = (String) iterator.next();
            if(isNumeric(current)){
                trueWordsHashSet.add(current);
                textAreaWordHashSet.remove(current);
            }
        }

        // Foreach içerisinde remove metodunu gerçekleştirirken hata almamak için
geçici tempHash oluşturduk.
        // İçerisine kalan yanlış olabilecek kelimeleri attık.
        tempHash.addAll(textAreaWordHashSet);

        // Single Transposition olan kelimeleri depolamak için gecici olarak
oluşturduk.
        ArrayList singleTranspositionTexts = new ArrayList();
        ArrayList singleTransTrueVersion = new ArrayList();

        for(String falseText : textAreaWordHashSet){
            StringBuilder temp = new StringBuilder(falseText);
            for(int i=0; i < falseText.length()-1; i++){
                char tempChar = temp.charAt(i);
                temp.setCharAt(i,temp.charAt(i+1));
                temp.setCharAt(i+1,tempChar);

                // Ayarlanan kelime var mı yok mu diye kontrol ediyoruz.
                if(wordsTxtHashSet.contains(temp.toString())) {
                    trueWordsHashSet.add(temp.toString());
                    singleTransTrueVersion.add(temp.toString());

                    // Yerini bulduktan sonra kelimeyi düzeltme işlemini burada
yapıyoruz.
                    int index =
notepad.getTextArea().getText().toLowerCase().indexOf(falseText);
                    notepad.getTextArea().replaceRange(temp.toString(), index,
index + falseText.length());
                    // Single Transposition olan kelimeyi arrayliste ekliyoruz.
                    singleTranspositionTexts.add(falseText);
                    tempHash.remove(falseText);
                }
                temp = new StringBuilder(falseText);
```

```
            }
        }
        JOptionPane.showMessageDialog(null,"Single Transposition Olan Kelimeler :
\n" + singleTranspositionTexts + "\nDoğru Versiyonları : \n" +
singleTransTrueVersion);
        textAreaWordHashSet.removeAll(singleTranspositionTexts);
        JOptionPane.showMessageDialog(null, "True def.Words : \n" +
trueWordsHashSet+"\nFalse words: \n" + textAreaWordHashSet);
    }

    // karakterin numerik olup olmadığını bu metotla kontrol ediyoruz.
    public boolean isNumeric(String strNum) {
        if (strNum == null) {
            return false;
        }
        try {
            Double.parseDouble(strNum);
        } catch (NumberFormatException nfe) {
            return false;
        }
        return true;
    }

}
```