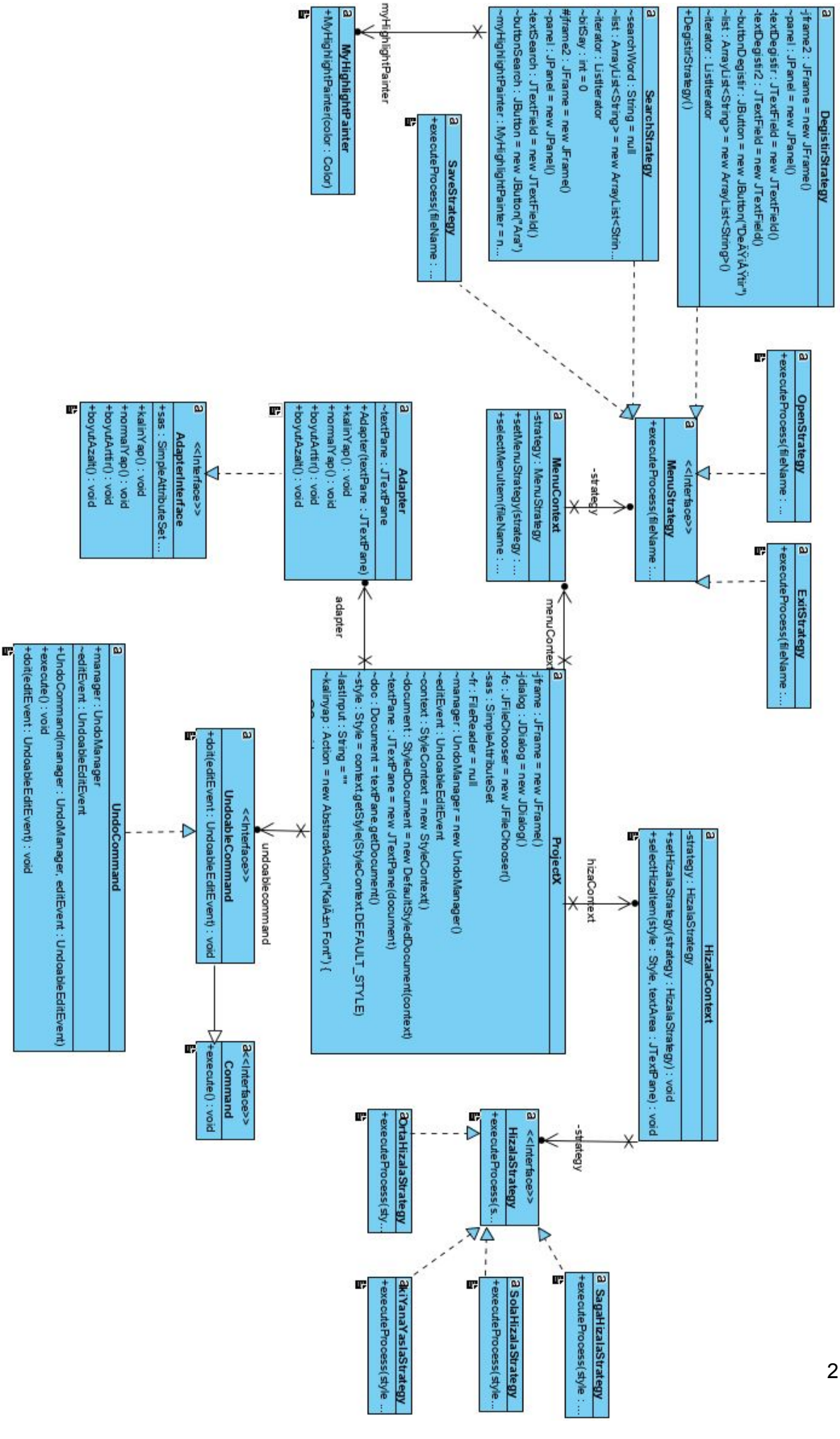
2019-2020 GÜZ DÖNEMİ

EGE ÜNİVERSİTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ
NESNEYE DAYALI PROGRAMLAMA 2019-2020 DÖNEM PROJESİ

Aytuğ Sevği 05160000539

Ceren Erdoğan 05160000581


DegistirStrategy
SearchStrategy
SaveStrategy
MyHighlightPainter
myHighlightPainter
OpenStrategy
ExitStrategy
<<Interface>>
MenuStrategy
MenuContext
-strategy
Adapter
<<Interface>>
AdapterInterface
adapter
menuContext
ProjectX
undoablecommand
<<Interface>>
UndoableCommand
UndoCommand
<<Interface>>
Command
HizalaContext
hizaContext
<<Interface>>
HizalaStrategy
OrtaHizalaStrategy
SolaHizalaStrategy
SagaHizalaStrategy
SkiYanaYaslaStrategy

Tasarım Desenleri

➢ **Command Tasarım Deseni :** Bir metin editörü (text editor) uygulamasında yazılan harfi geri almak için kullanılan tasarım desenidir.

1) undoablecommand nesnesi oluşturulur.

```
UndoManager manager = new UndoManager();
UndoableEditEvent editEvent;
UndoableCommand undoablecommand = new UndoCommand(manager,editEvent);
```

2) Text de yapılan her eventi ilgili fonksiyon aracılığıyla kaydediyoruz.

```
doc.addUndoableEditListener(new UndoableEditListener() {
public void undoableEditHappened(UndoableEditEvent evt) {
    undoablecommand.doit(evt);
     //UndoRedoManager command request classına giderek eventi ekler.
}
});
```

```
@Override
public void doit(UndoableEditEvent editEvent) {

    manager.addEdit(editEvent.getEdit());
}
```

3) "Geri Al" butonuna basınca metinde yapılan son değişiklik geri alınır.

```
Action Undo = new AbstractAction("Geri Al") {
    @Override
    public void actionPerformed(ActionEvent e) {
        undoablecommand.execute();
```

execute Fonksiyonu geri alma işlemini sağlar.

```
@Override
public void execute() {
    if(manager.canUndo()){
        manager.undo();
     }
     else{
         System.out.println("Geri alınacak öğe yok..");
```

➢ **Iterator Tasarım Deseni :** Bir metin editörü (text editor) uygulamasında istenilen kelimeyi bulmak veya değiştirmek için kullanıldı. Java'da bulunan hazır Iterator kullanıldı.

1) **DegistirStrategy sınıfında ListIterator tipinde iterator tanımlandı.**

```
ListIterator iterator;
```

2) **Iterator nesnesi oluşturuluyor ve text içinde istenen kelimeyi arıyor.**

```
iterator = list.listIterator();

while(iterator.hasNext()){
    siradakiKelime = (String) iterator.next();
    if(textDegistir.getText().equalsIgnoreCase(siradakiKelime)){

        allText = bul(allText,degistirilenWord,yeniWord);
        textPane.setText(allText);
    }
```

3) **Aynı şekilde SearchStrategy sınıfında da iterator'ı aynı amaç için kullanıyoruz.**

```
iterator = list.listIterator();
int searchWordLenght=0;
while(iterator.hasNext()){
    siradakiKelime = (String) iterator.next();
    if(searchWord.equalsIgnoreCase(siradakiKelime)){
        searchWordLenght = searchWord.length();
        bitSay = allText.indexOf(searchWord,bitSay);
```

➢ **Adapter Tasarım Deseni :** Bir metin editörü (text editor) uygulamasında metnin kalınlığını ve font boyutunu değiştirmek için kullanıldı.

1) **AdapterInterface aşağıdaki gibidir. SimpleAttributeSet javanın kendi kütüphanesidir. Bu kütüphane aracılığıyla kendi methodlarımızı oluşturacağız.**

```
public interface AdapterInterface {

    SimpleAttributeSet sas = new SimpleAttributeSet();

    public void kalinYap();
    public void normalYap();
    public void boyutArttir();
    public void boyutAzalt();
}
```

2) **Javanın kendi methodlarını kendi methodlarımıza dönüştürdük.**

```
public class Adapter implements AdapterInterface {
    JTextPane textPane;
    public Adapter(JTextPane textPane){
        this.textPane = textPane;
    }
    @Override
    public void kalinYap(){
        StyleConstants.setBold(sas, true);
        String text = textPane.getText();
        textPane.getStyledDocument().setCharacterAttributes(0, text.length(), sas, false);
    }
    public void normalYap(){
        StyleConstants.setBold(sas, false);
        String text = textPane.getText();
        textPane.getStyledDocument().setCharacterAttributes(0, text.length(), sas, false);
    }
    public void boyutArttir(){
        int size = StyleConstants.getFontSize(sas);
        StyleConstants.setFontSize(sas, size+4);
        String text = textPane.getText();
        textPane.getStyledDocument().setCharacterAttributes(0,text.length(),sas,false);
    }
    public void boyutAzalt(){
        int size = StyleConstants.getFontSize(sas);
        StyleConstants.setFontSize(sas, size-4);
        String text = textPane.getText();
        textPane.getStyledDocument().setCharacterAttributes(0,text.length(),sas,false);
```

3) **Adapter adapter; Main classında tanımlanır ve bir butona tıklandığında adapter sınıfındaki method çağırılır.**

```
Action kalinyap = new AbstractAction("Kalın Font") {
    @Override
    public void actionPerformed(ActionEvent e) {
        adapter = new Adapter(textPane);
        adapter.kalinYap();
});
Action normalyap = new AbstractAction("Normal Font") {
    @Override
    public void actionPerformed(ActionEvent e) {
        adapter = new Adapter(textPane);
        adapter.normalYap();
});
Action boyutarttir = new AbstractAction("Yazı Boyutu Arttır") {
    @Override
    public void actionPerformed(ActionEvent e) {
        adapter = new Adapter(textPane);
        adapter.boyutArttir();
});
Action boyutazalt = new AbstractAction("Yazı Boyutu Azalt") {
    @Override
    public void actionPerformed(ActionEvent e) {
        adapter = new Adapter(textPane);
        adapter.boyutAzalt();
});
Action solahizala = new AbstractAction("Sola Hizala") {
    @Override
    public void actionPerformed(ActionEvent e) {
        //strategy pattern called
        hizaContext.setHizalaStrategy(new SolaHizalaStrategy());
        hizaContext.selectHizaItem(style, textPane);
```

➢ **Strategy Tasarım Deseni :** Bir metin editörü (text editor) uygulamasında open, save, exit, kelime bulma ve değiştirme menü işlemleri için strategy pattern kullanıldı.

1) **MenuStrategy adında bir interface oluşturuldu.**
2) **DegistirStrategy, ExitStrategy, OpenStrategy, SaveStrategy classları oluşturuldu ve ilgili kodlar içerisindeki @override eden fonksiyonda yazıldı.**
3) **MenuContext Sınıfı oluşturuldu.**

```
public class MenuContext {
    private MenuStrategy strategy;
    public void setMenuStrategy(MenuStrategy strategy){
        this.strategy = strategy;
    }
    public void selectMenuItem(String fileName,JFileChooser fc,JTextPane textArea,FileReader fr,JFrame jframe){
        strategy.executeProcess(fileName, fc, textArea, fr,jframe);
    }

}
```

4) **MenuContext menuContext = new MenuContext(); Main'de oluşturuldu.**
5) **Menuden seçilen item'a göre menuContext'ten strategy değiştirilip ilgili sınıfa ait method oluşturuldu.**

```
Action Search = new AbstractAction("Kelime Ara") {
    @Override
    public void actionPerformed(ActionEvent e) {
        menuContext.setMenuStrategy(new SearchStrategy());
        menuContext.selectMenuItem(null,null,textPane,null,jframe);}};
Action Degistir = new AbstractAction("Kelime Değiştir") {
    @Override
    public void actionPerformed(ActionEvent e) {
        menuContext.setMenuStrategy(new DegistirStrategy());
        menuContext.selectMenuItem(null,null,textPane,null,jframe);}};
Action Open = new AbstractAction("Dosya Aç") {
    @Override
    public void actionPerformed(ActionEvent e) {
        //menu context ile gerekli strateji nesnesini yaratma.
       if (fc.showOpenDialog(null)==JFileChooser.APPROVE_OPTION){
            menuContext.setMenuStrategy(new OpenStrategy());
            menuContext.selectMenuItem(fc.getSelectedFile().getAbsolutePath()
//saveaction
Action Save = new AbstractAction("Kaydet") {
    @Override
    public void actionPerformed(ActionEvent e) {
        //menu context ile gerekli strateji nesnesini yaratma.
        menuContext.setMenuStrategy(new SaveStrategy());
        menuContext.selectMenuItem(null, fc, textPane, fr,jframe);}};
Action Exit = new AbstractAction("Çıkış") {
    @Override
    public void actionPerformed(ActionEvent e) {
        //menu context ile gerekli strateji nesnesini yaratma.
        menuContext.setMenuStrategy(new ExitStrategy());
        menuContext.selectMenuItem(null, fc, textPane, fr,jframe);}};
```

6) **Hizalama işlemleri için ayrı bir ” HizalaInterface “oluşturduk.**

```
public interface HizalaStrategy {
     public void executeProcess(Style style,JTextPane textArea);
}
```

7) **SolaHizalaStrategy,SagaHizalaStrategy, İkiYanaYaslaStrategy, OrtaHizalaStrategy sınıfları oluşturuldu. Bu sınıflar HizalaInterface sınıfını implement etmektedir.**
8) **İlgili strategy’i seçmek için HizalaContext sınıfı oluşturuldu.**

```
public class HizalaContext {
    private HizalaStrategy strategy;
    public void setHizalaStrategy(HizalaStrategy strategy){
        this.strategy = strategy;
    }
    public void selectHizaItem(Style style, JTextPane textArea){
        strategy.executeProcess(style, textArea);
```

9) **HizalaContext hizaContext = new HizalaContext(); Main’de oluşturuldu.**

10) **Menuden seçilen item’a göre HizalaContext’ten strategy değiştirilip ilgili sınıfa ait method oluşturuldu.**

```
Action solahizala = new AbstractAction("Sola Hizala") {
    @Override
    public void actionPerformed(ActionEvent e) {
        //strategy pattern called
         hizaContext.setHizalaStrategy(new SolaHizalaStrategy());
         hizaContext.selectHizaItem(style, textPane);}};
Action ortahizala = new AbstractAction("Ortala") {
    @Override
    public void actionPerformed(ActionEvent e) {
        //strategy pattern called
        hizaContext.setHizalaStrategy(new OrtaHizalaStrategy());
        hizaContext.selectHizaItem(style, textPane);;}};
Action sagahizala = new AbstractAction("Sağa Hizala") {
    @Override
    public void actionPerformed(ActionEvent e) {
        //strategy pattern called
        hizaContext.setHizalaStrategy(new SagaHizalaStrategy());
        hizaContext.selectHizaItem(style, textPane);}};
Action ikiyanahizala = new AbstractAction("İki Yana Yasla") {
    @Override

    public void actionPerformed(ActionEvent e) {
      hizaContext.setHizalaStrategy(new IkiYanaYaslaStrategy());
      hizaContext.selectHizaItem(style, textPane);
```

Birim Test

DegistirStrategy sınıfında bulunan “bul( )” fonksiyonu için birim test oluşturduk.

```
@Test
public void testBulDegistir() {
    DegistirStrategy degistir = new DegistirStrategy();
    String allText = "merhaba bu benim eski cümlem";
    String degistirilenWord = "eski"; String yeniWord = "yeni";
    allText = degistir.bul(allText, degistirilenWord, yeniWord);

    Assert.assertEquals("merhaba bu benim yeni cümlem",allText);
}
```

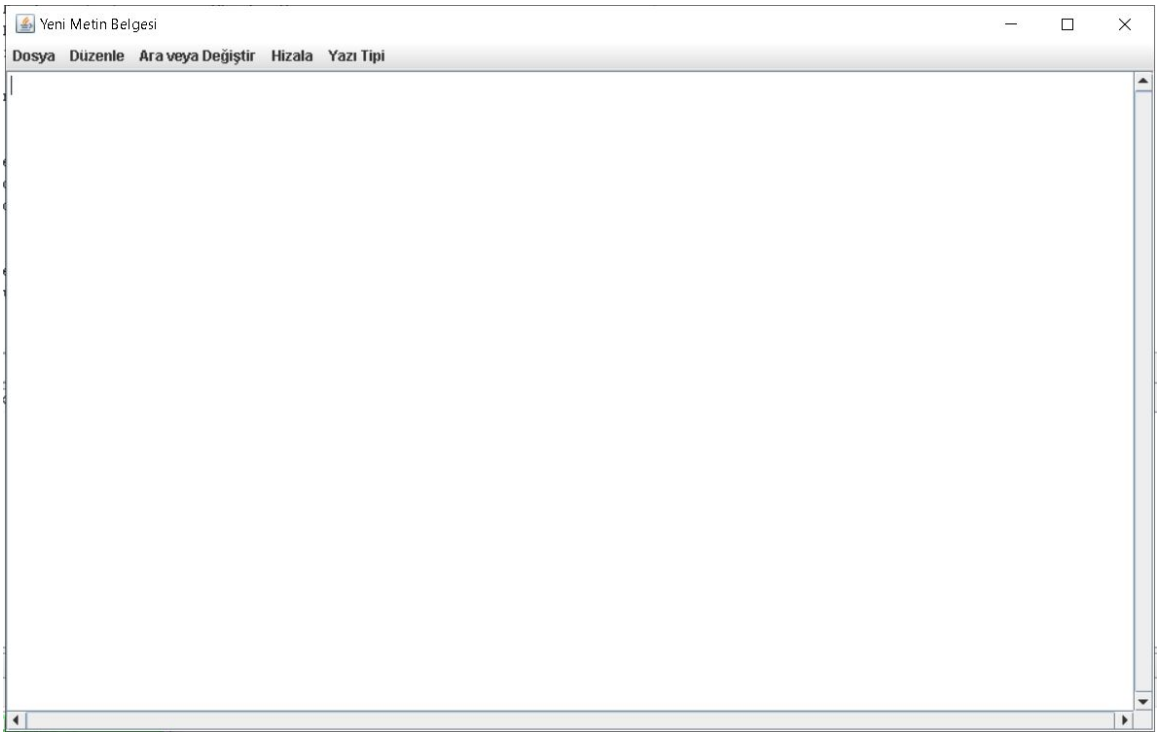
Bu birim test ile methodumuzun doğru sonuç verip vermediğini test ettik.Fonksiyon 3 parametre almaktadır.Tüm metni, değiştirilecek kelimeyi ve yeni kelimeyi almaktadır. Sonucun beklenildiği gibi olup olmadığını assertEquals( ) yardımıyla test ettik.

```
@Test
public void tearDown() {
    hizaContext.setHizalaStrategy(new SolaHizalaStrategy());
    hizaContext.selectHizaItem(style, textPane);
    Object sonuc = StyleConstants.getAlignment(style);
    assertEquals(sonuc.toString(), "0");

    hizaContext.setHizalaStrategy(new OrtaHizalaStrategy());
    hizaContext.selectHizaItem(style, textPane);
    sonuc = StyleConstants.getAlignment(style);
    assertEquals(sonuc.toString(), "1");

    hizaContext.setHizalaStrategy(new SagaHizalaStrategy());
    hizaContext.selectHizaItem(style, textPane);
    sonuc = StyleConstants.getAlignment(style);
    assertEquals(sonuc.toString(), "2");

    hizaContext.setHizalaStrategy(new IkiYanaYaslaStrategy());
    hizaContext.selectHizaItem(style, textPane);
    sonuc = StyleConstants.getAlignment(style);
    assertEquals(sonuc.toString(), "3");
```

HizalaStrategy’lerimizin doğru sonuç verip verilmediği test edildi.(ALIGN_LEFT alignment’ın değerini 0, ALIGN_RIGHT aligment’ın değerini 2 yapmaktadır.)

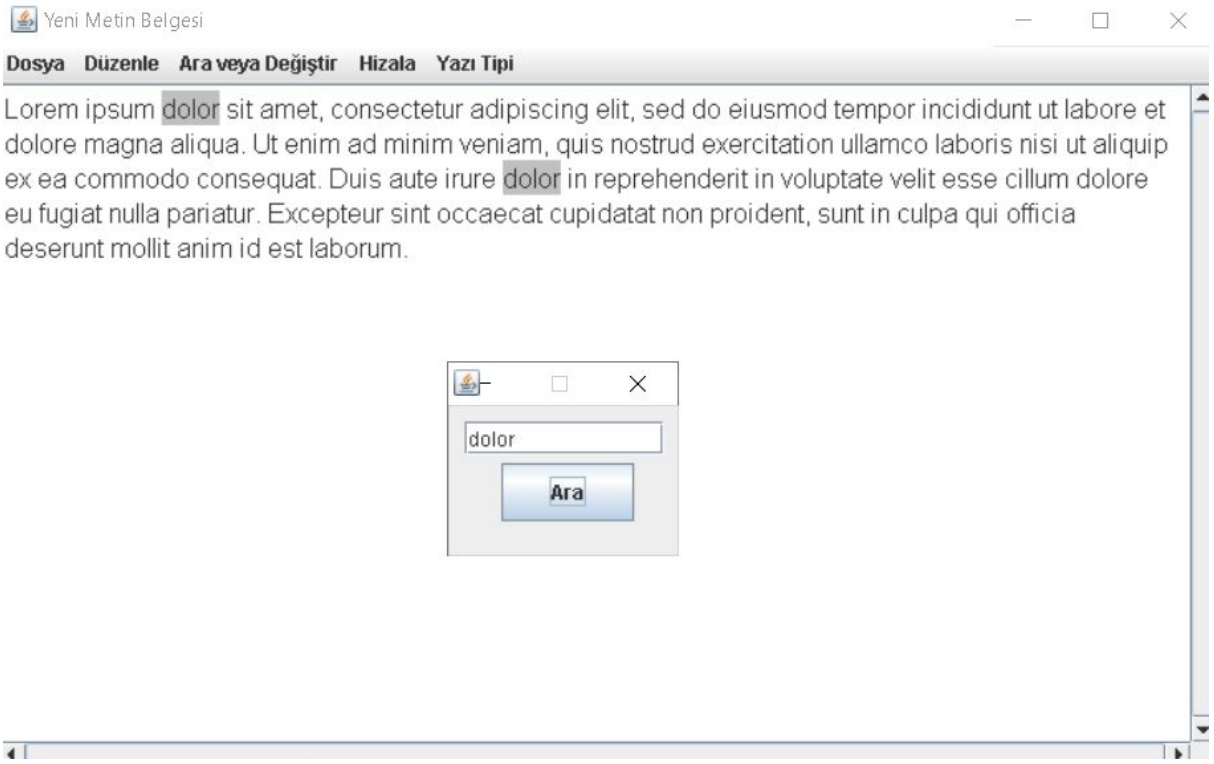
KULLANICI ARAYÜZÜ

Projemizi çalıştırdığımızda karşımıza bir text editör açılacaktır.Şekilde görüldüğü gibi:

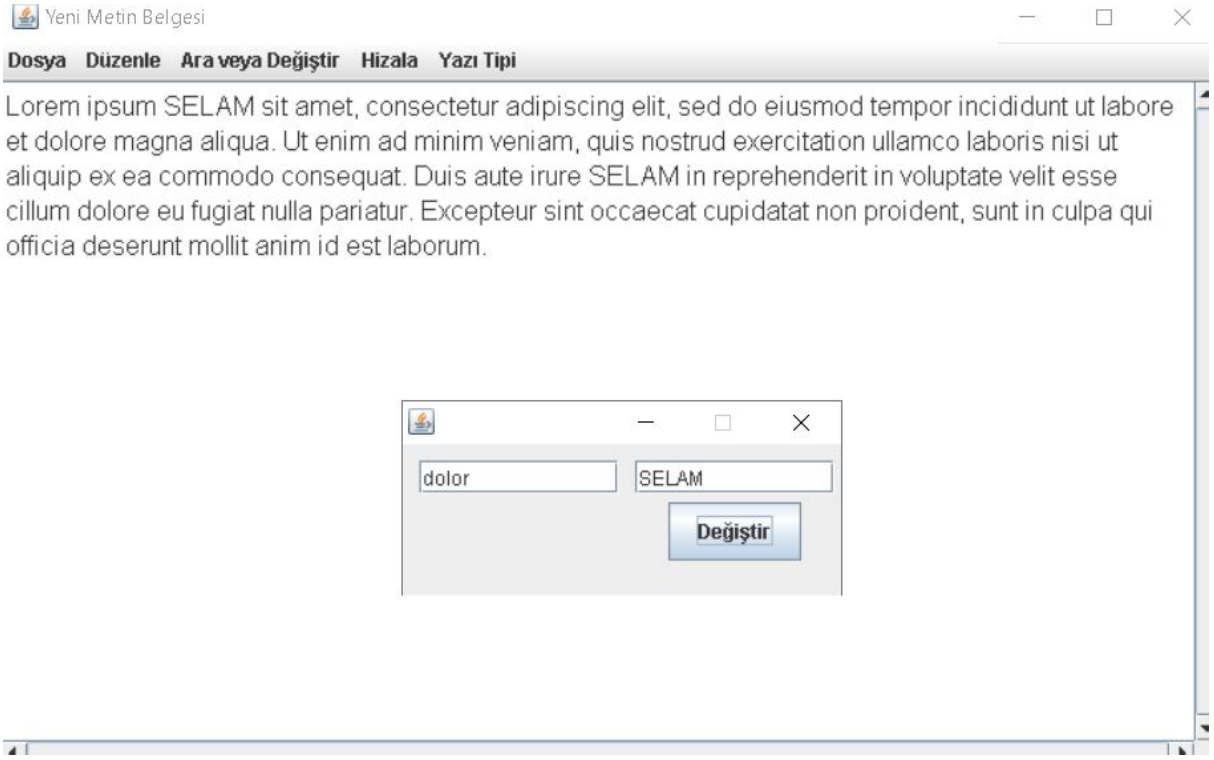


Klavyenin tuşlarına bastığınız takdirde uygulamada bir takım harfler göreceksiniz. Yazdığınız yazı dışında görüntü şu şekilde olacaktır :

Menü kısmıdan dosya işlemlerinizi yapabilirsiniz.Bu dosya menüsünde ; dosya açma, dosya kaydetme ve çıkış seçenekleri bulunmaktadır. Düzenle menüsünde "geri al" a basarsanız metin belgesinde yaptığınız son değişikliği geri alırsınız. Ara veya Değiştir menüsünde aradığınız bir kelimeyi metin belgesinde bulabillir ve dilediğiniz kelime ile değiştirebilirsiniz.

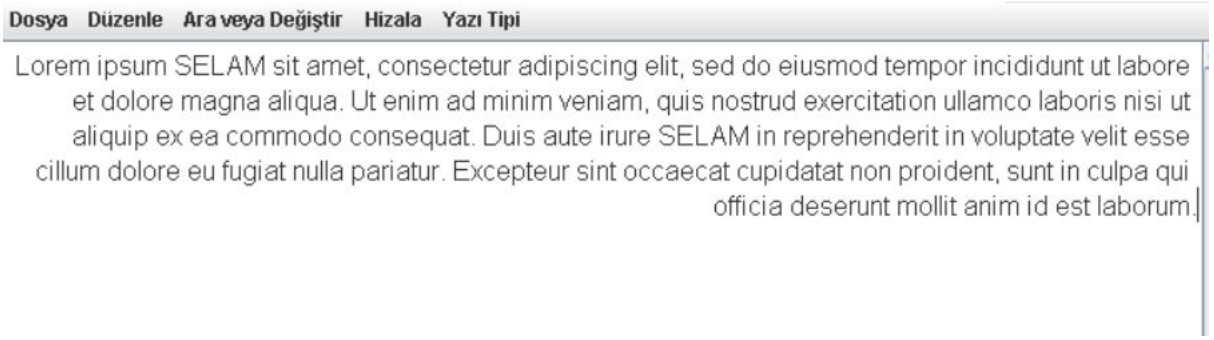


Değiştirmek istersek metinin yeni hali şu şekilde olacaktır.

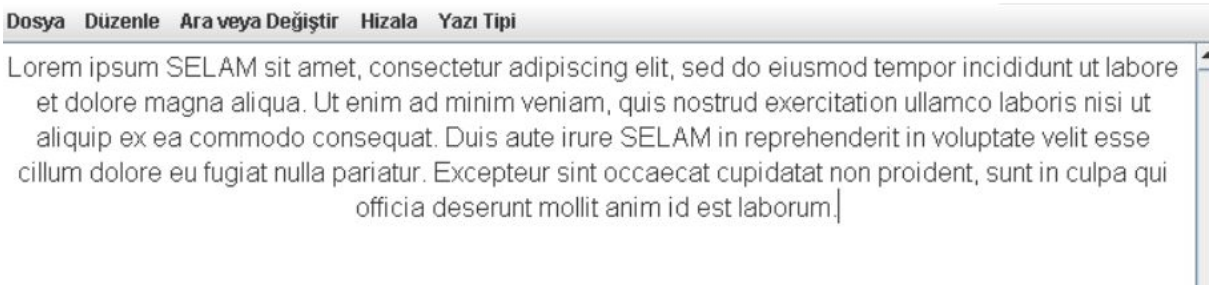


Hizala Menüsünden Sola, sağa, iki yana hizalayabiliriz veya metni ortalayabiliriz.

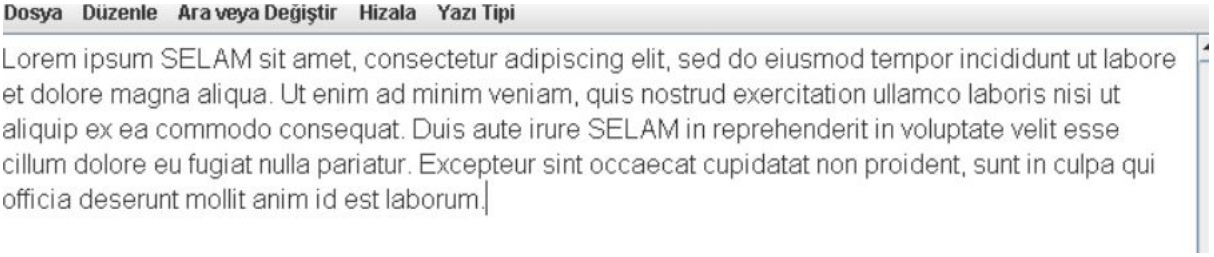
Sağa Hizala :



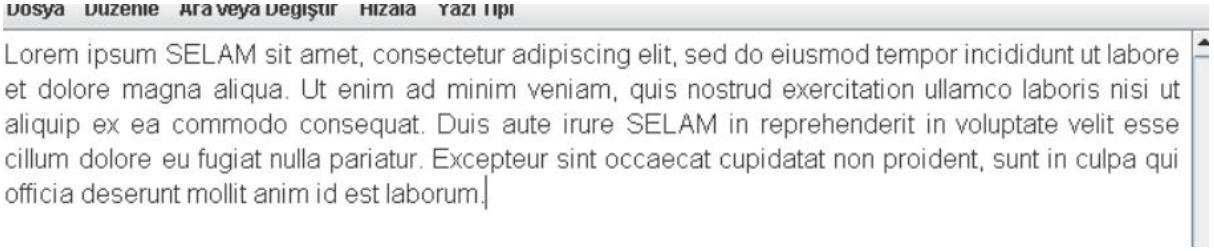
Ortala :



Sola Hizala :

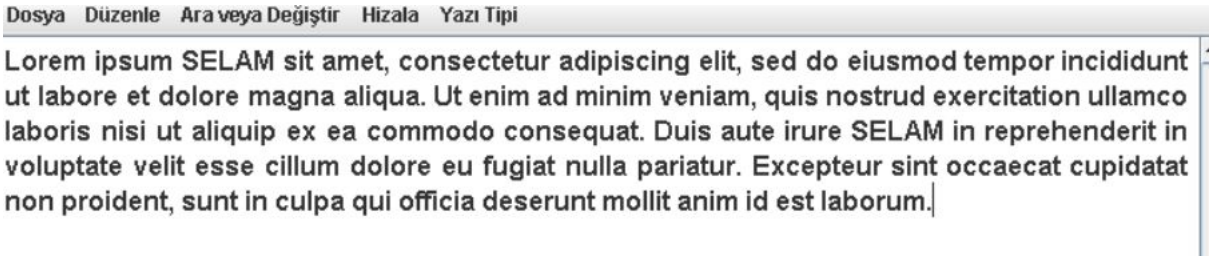


İki Yana Yasla:

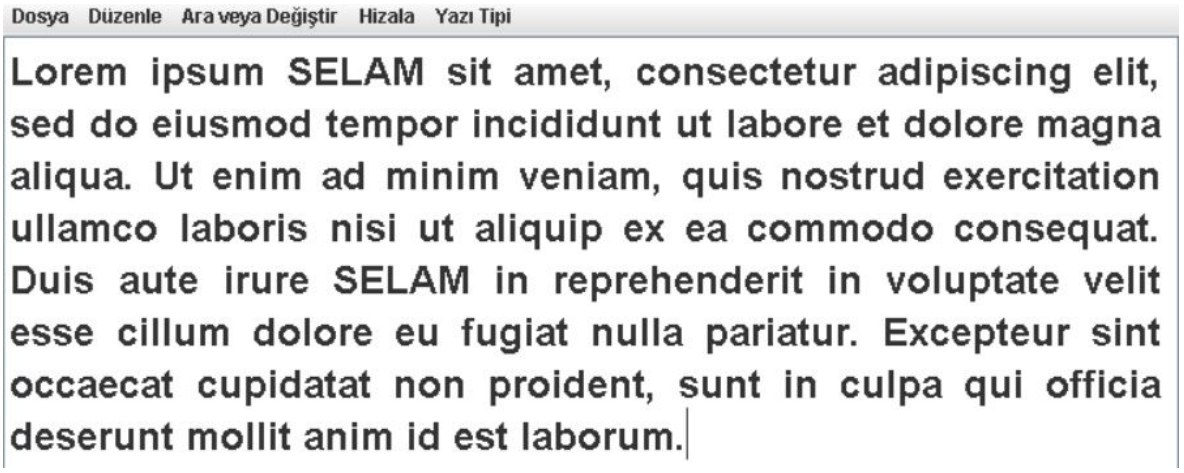


Yazı Tipi menüsünde metninizin fontunu kalınlaştırıp normal hale çevirebiirsiniz.Ayrıca font büyüklüğünü arttırıp azaltabilirsiniz.

Kalın :



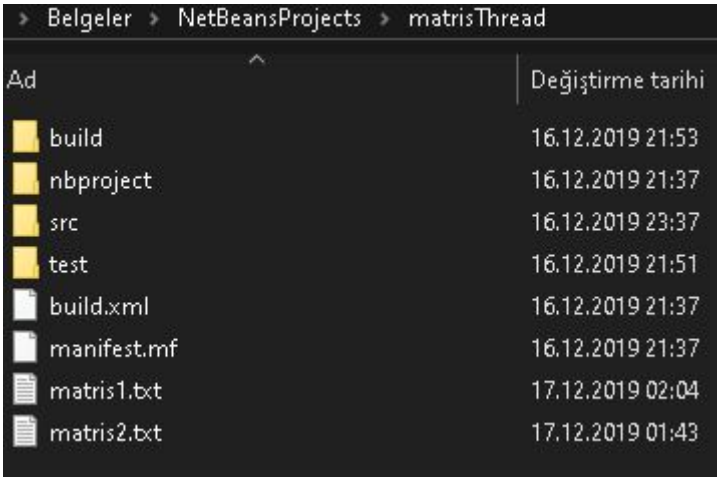
Yazı Boyutunu Arttır(+4) :



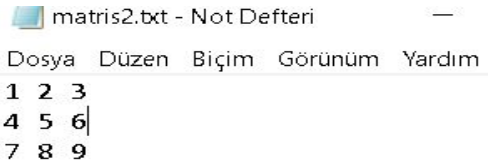
PROJE-2

KULLANICI ARAYÜZÜ

Matris dosyalarımız şu dizinde bulunmaktadır.



Girdiğin matris değerlerinin arasında birer boşluk bulunmalıdır ve şu şekilde olmalıdır.



Girdiğiniz 2 matris çarpıma uygun olmalıdır.İlk matris matris1.txt ikinci matris matris2.txt dosyası kabul edilmiştir. Buna göre ilk matrisin sütunuyla ikinci matrisin satır sayısı eşit olmalıdır.
Sonrasında kodu çalıştırdığınızda çarpım matrisi ekrana yazılacaktır.