

**NESNEYE DAYALI
PROGRAMLAMA
PROJE-1**

HAZIRLAYANLAR:

Kutay AVCI, 05180000091

Emre BALKAYA, 05180000056

Erdal Eren BAYAR, 05170000089

İÇİNDEKİLER

- 1-) ANALİZ**
- 2-) PROGRAMCI KATALOĞU**
- 3-) UML SINIF DİYAGRAMI**
- 4-) KAYNAK KODU**
- 5-) KULLANICI KATALOĞU**

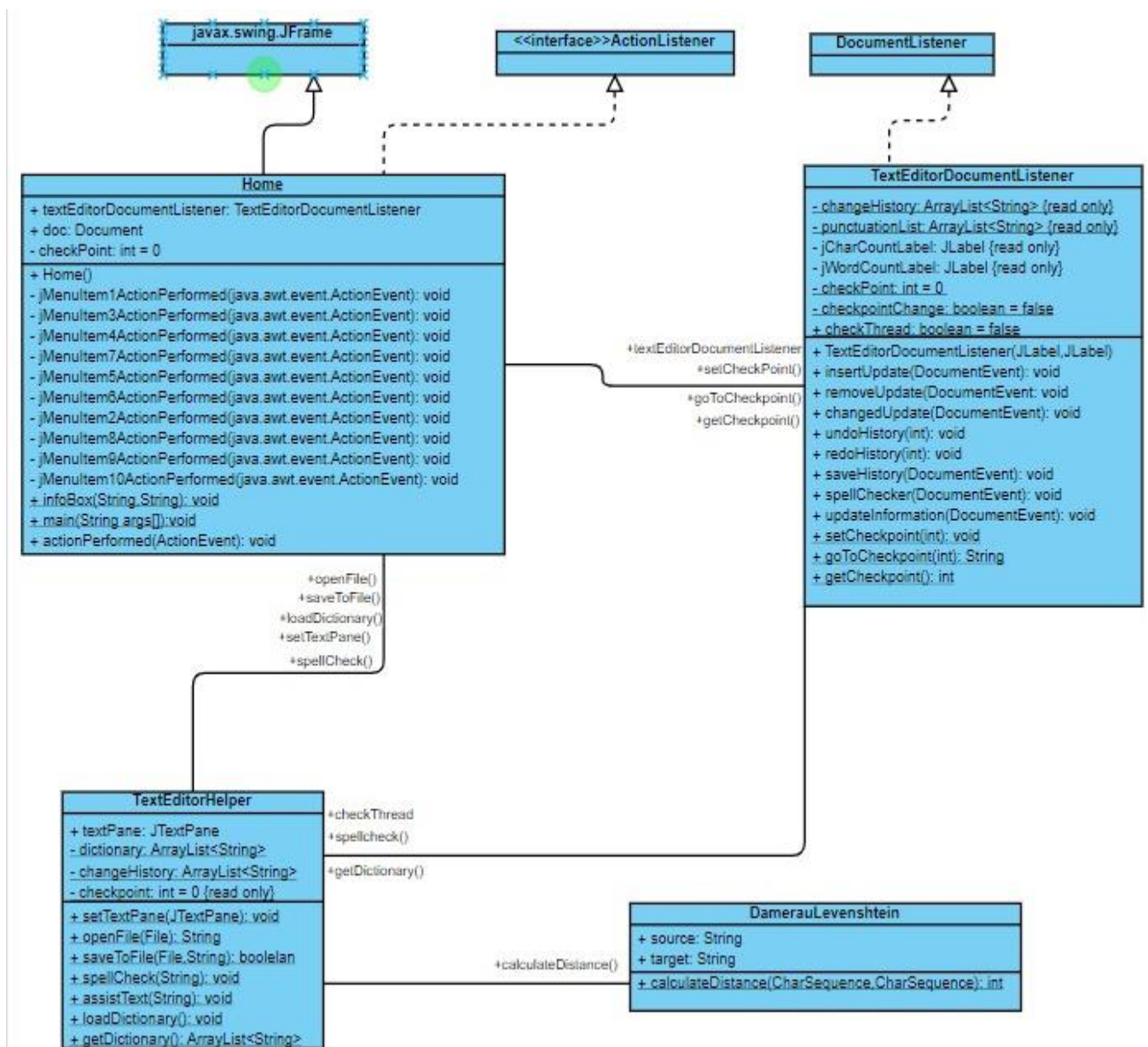
ANALİZ

- Home sınıfı javax.swing.JFrame sınıfından kalıtım yoluyla elde edilmiştir ve Action Listener arayüzü uygulanmıştır. Fonksiyonel işlemlerin gerçekleştirimi sırasında kullanmak için integer tipinde, checkPoint adında bir değişken ve TextEditorDocumentListener sınıfından aynı adlı değişken ile Document containerından doc isimli değişken tanımlanmıştır.
- Arayüz bileşenleri için değişkenler tanımlanmıştır.
- DamareuLevensthein sınıfı iki string arasındaki uzaklık değerini döndüren bir algoritmadır. Bu sınıf Yazım Hatası Kontrolü fonksiyonun gerçekleştirimi için kullanılmıştır. CalculateDistance metodu altında bitişik harflerin yer değiştirmesi sonucu oluşan yazım hataları tespit edilmiştir.
- TextEditorDocumentListener sınıfında DocumentListener arayüzü uygulanmıştır. Bu sınıfta ArrayList tipinde iki değişken bulunur. ChangeHistory değişkeni gerçekleşen son değişiklikleri tutan listedir. PunctuationList ise noktalama işaretlerini barındıran listedir. JLabel tipindeki iki değişken kelime ve harf sayacı için oluşturulmuştur. Kontrol noktaları içinse integer tipinde checkPoint sayacı oluşturulmuştur. Kontrol noktasındaki değişimlerin takibi için iki farklı Boolean tipi değişken oluşturulmuştur.
- Kelime ve harf sayacı için constructor metodu oluşturulmuştur.
- Kaydetme, silme ve değiştirme olayları için constructor metodlar oluşturulmuştur.
- SpellChecker metodu içinde Yazım Hatası Kontrolü ve değiştirilmesi işlemleri gerçekleştirilmiştir.
- TextEditorHelper sınıfında proje kapsamında istenilen fonksiyonel işlemlerin metodlarının gerçekleştirimi bu sınıf altında oluşturulmuştur.

PROGRAMCI KATALOĐU

Bu projenin tasarım süreci 2-3 günlük bir süreç iken gerekleřtirim süreci 8-10 gn srmřtr. Gerekli testler 1 gn ierisinde gerekleřtirilmiřtir.

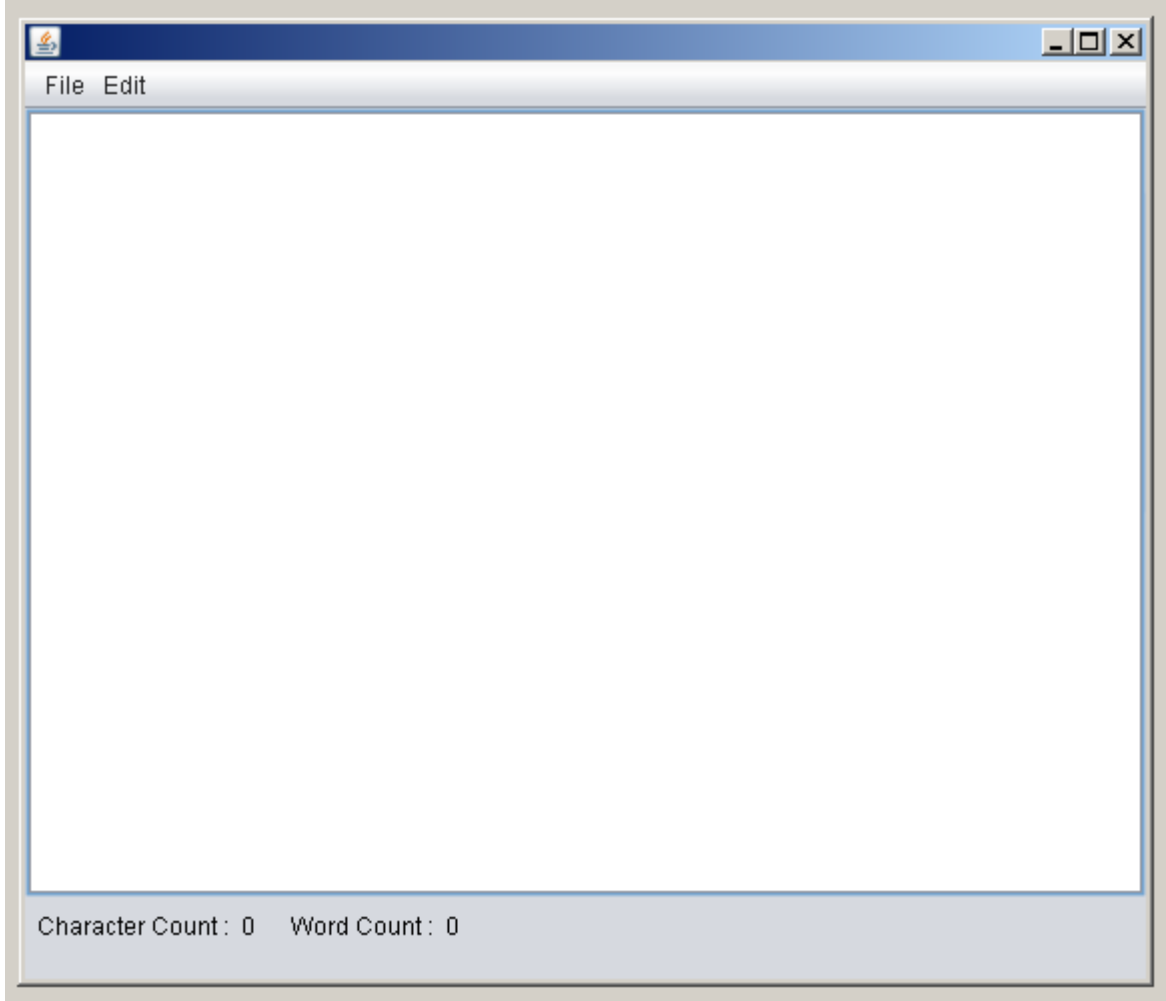
UML SINIF DİYAGRAMI



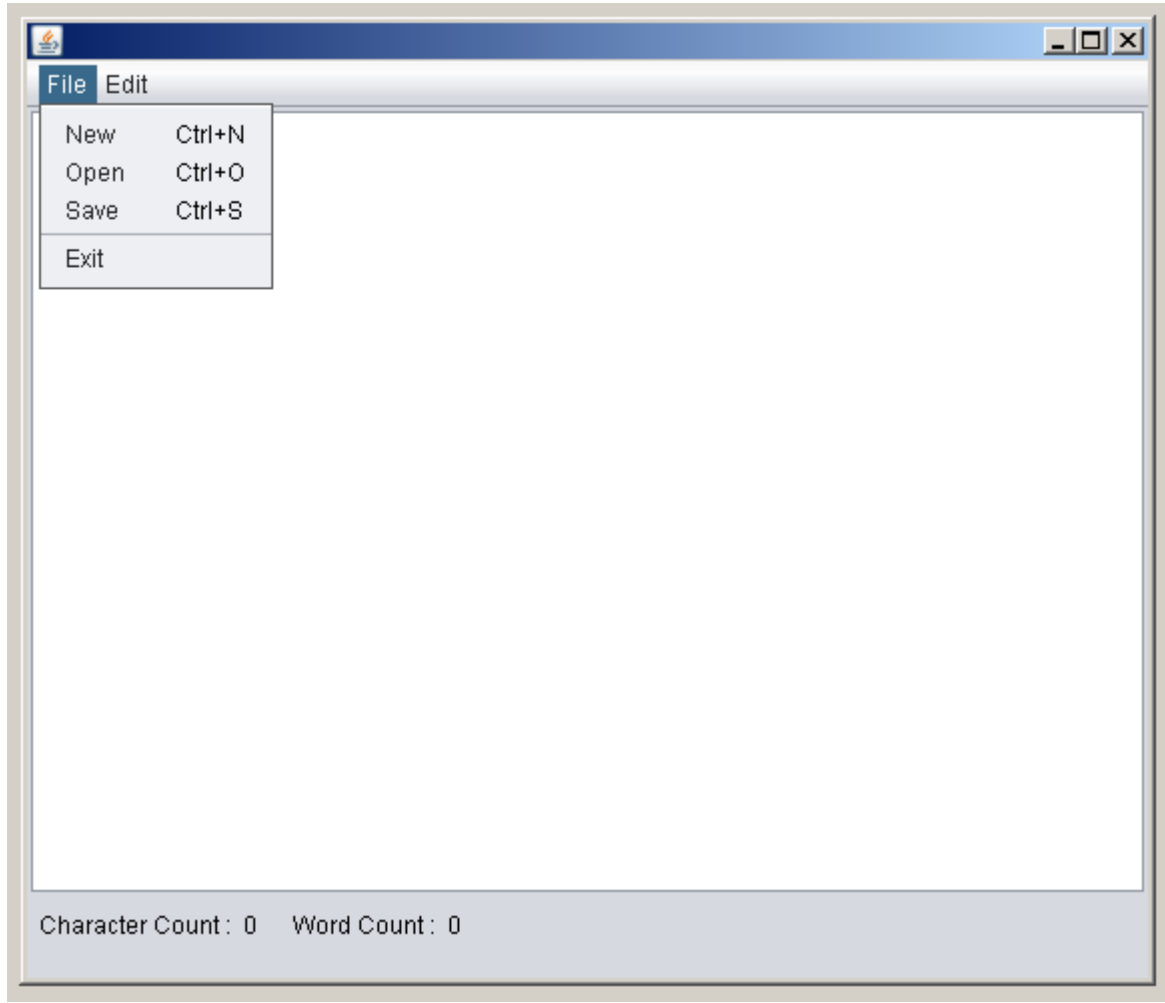
KAYNAK KOD

Proje kapsamında yazılan kaynak kodun teslimi EgeDers sistemi üzerinden oluşturulan Kaynak Kod Yükleme Klasörü'ne eklenmiştir.

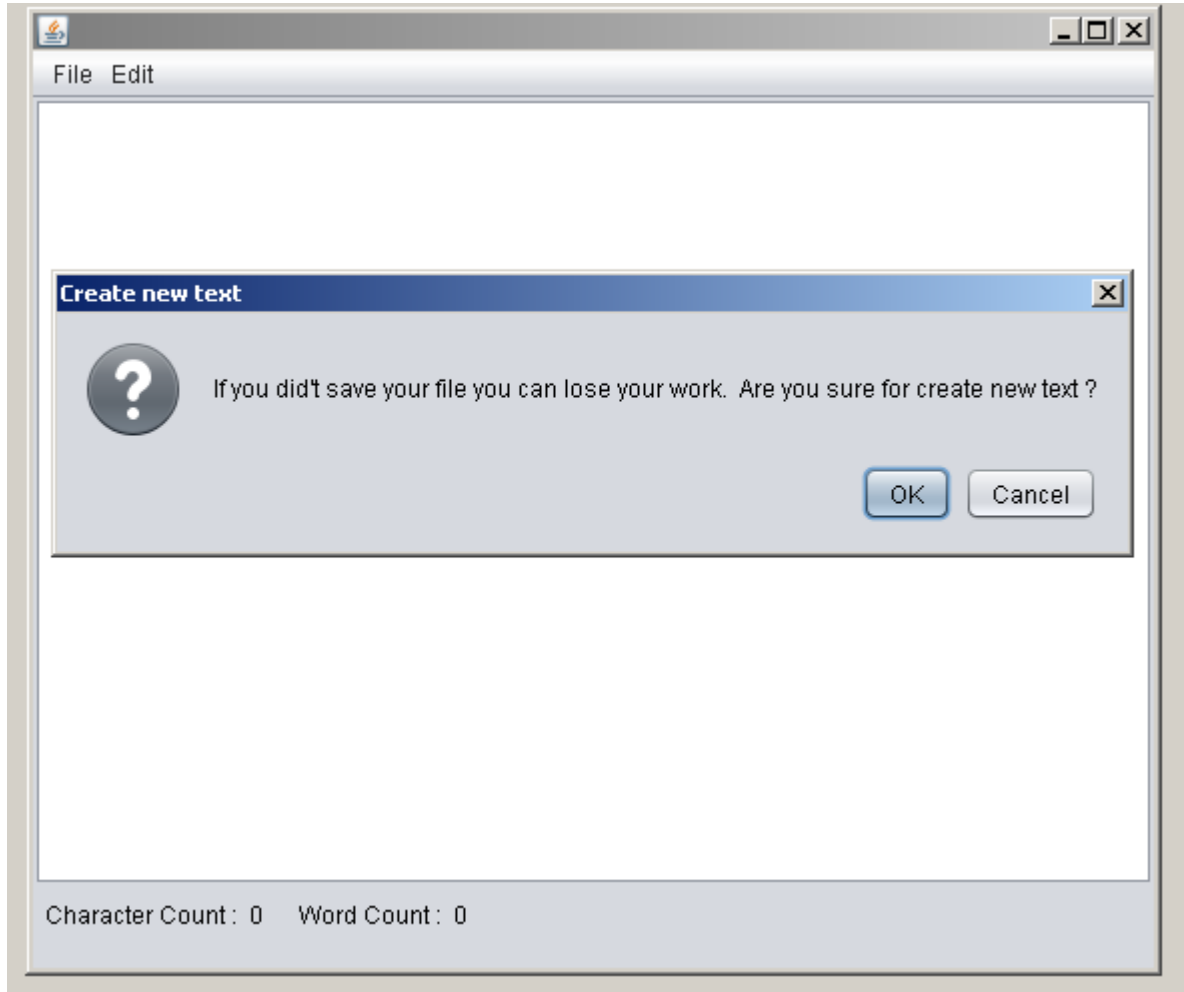
KULLANICI KATALOĐU



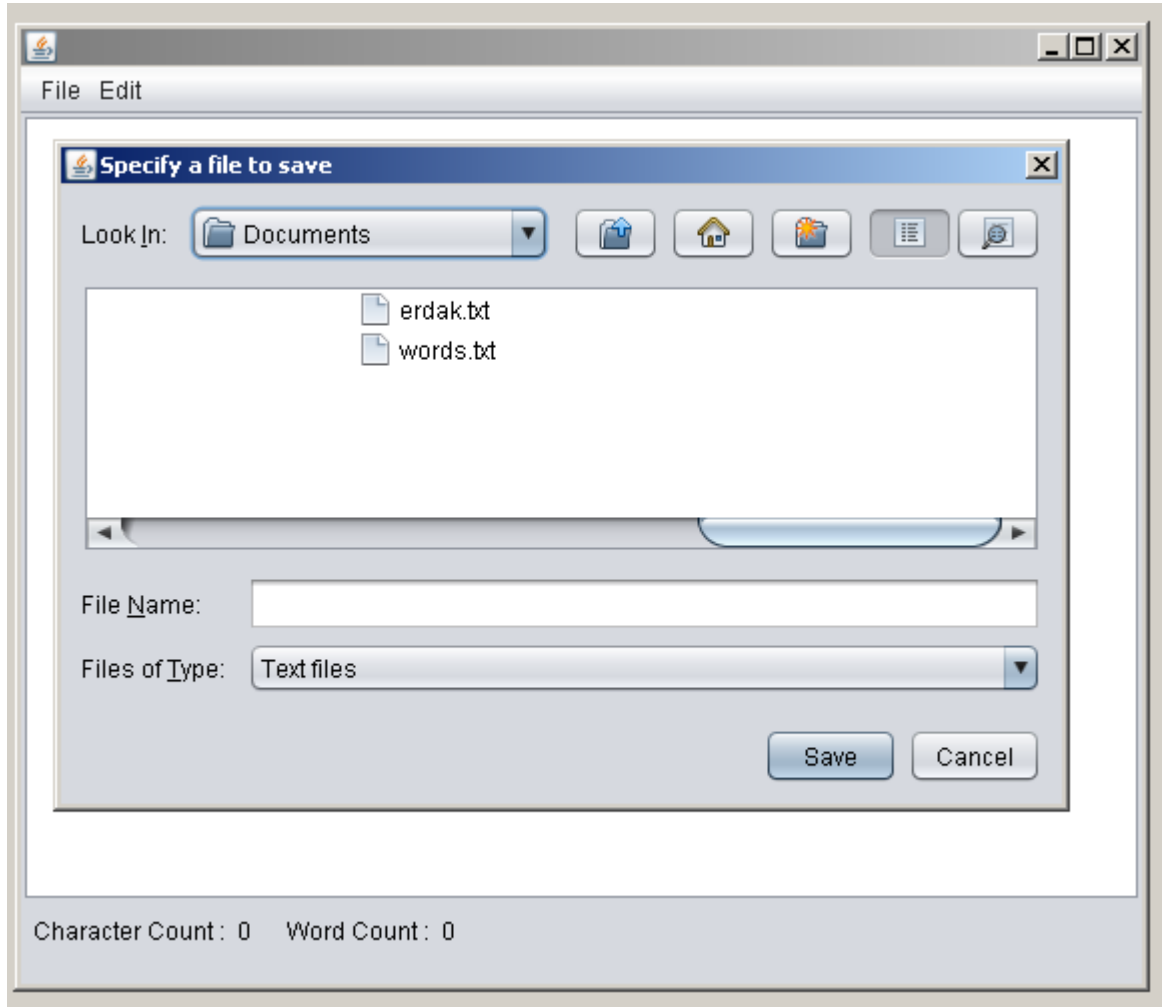
Program alıřtırıldığında kullanıcının karřılařacađı arayüz řekildeki gibidir.



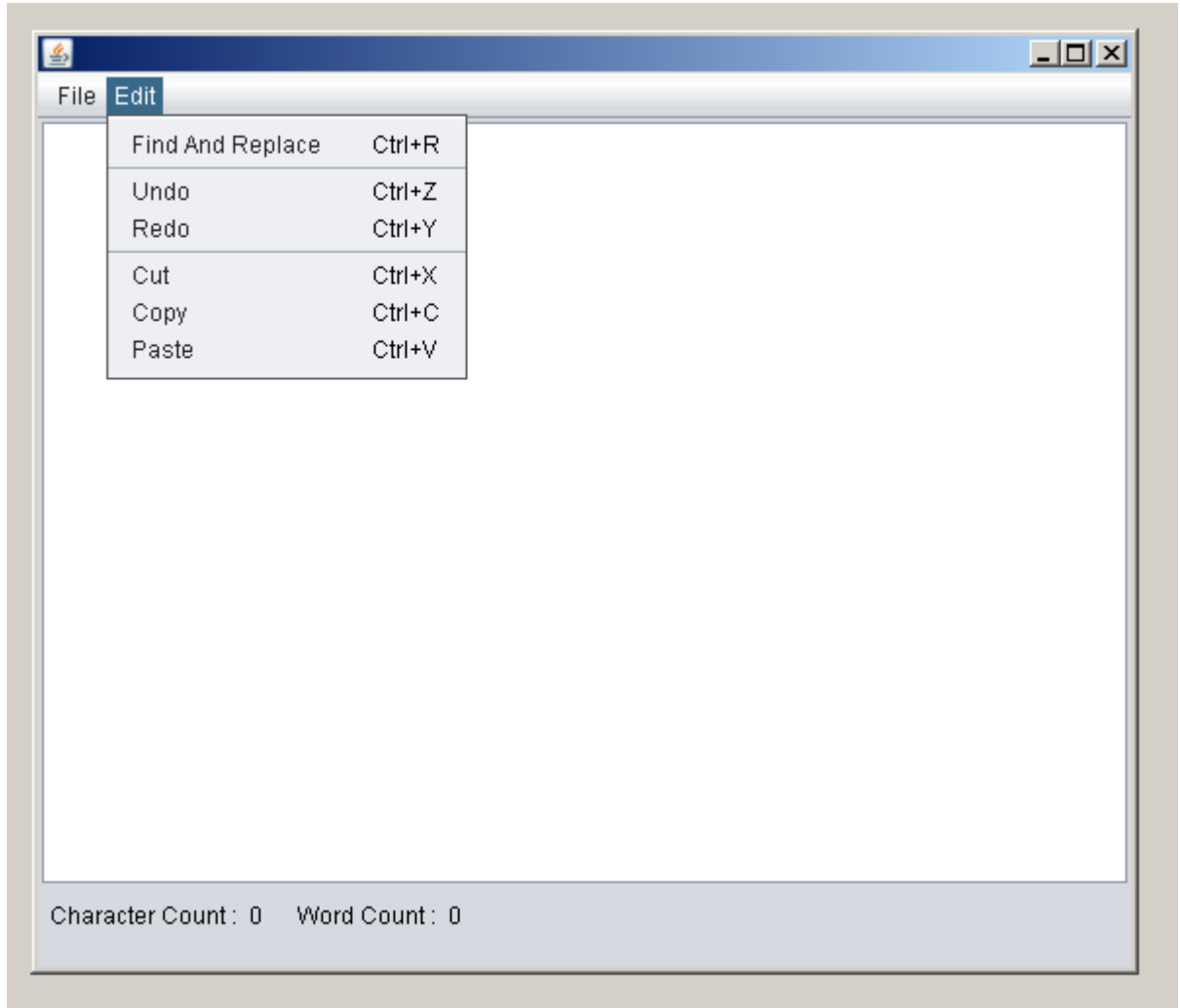
File sekmesi içerisinde New File, Open File, Save File ve Exit fonksiyonları ve kullanıcının bu fonksiyonlar için kullanabileceği kısayollar belirtilmiştir.



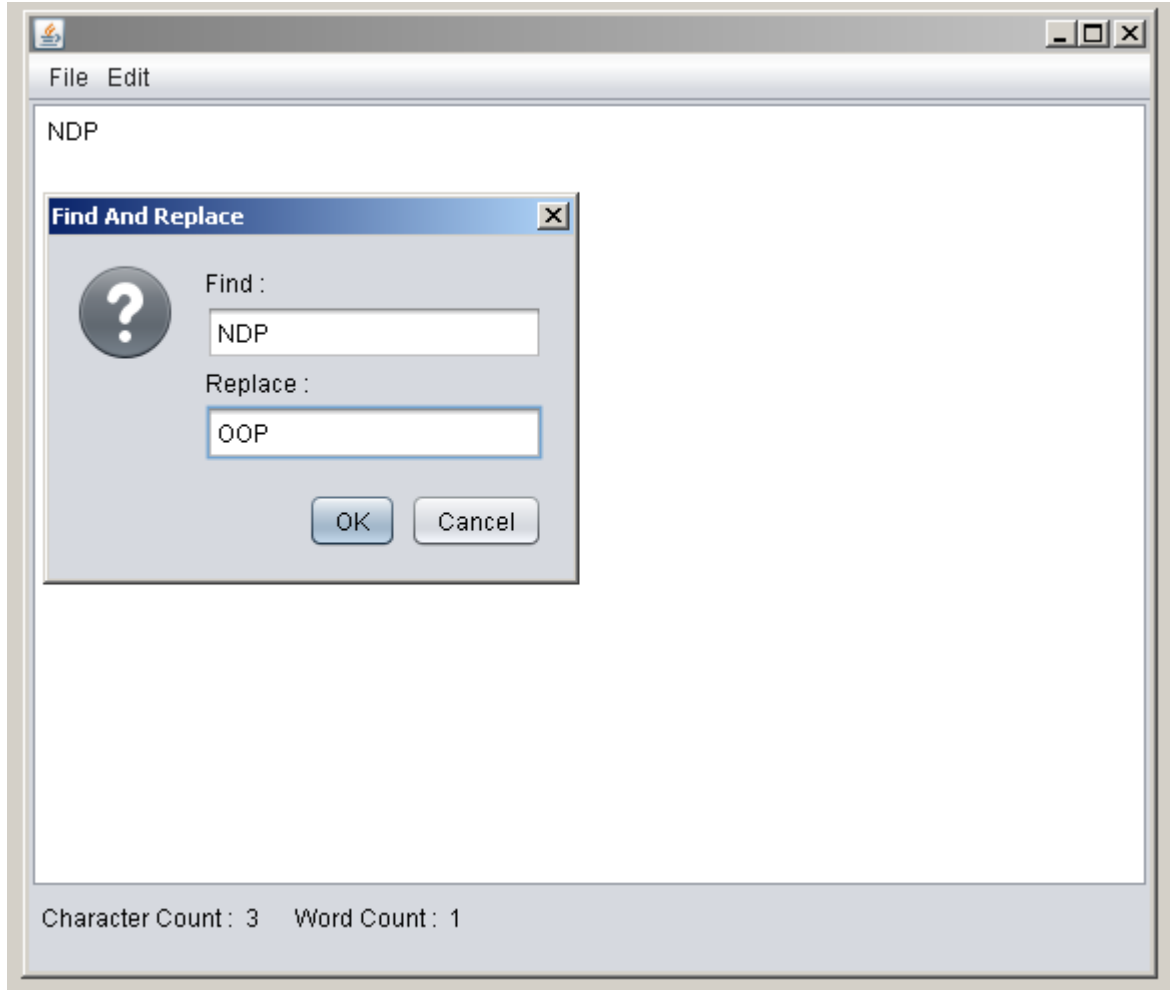
Kullanıcı New File komutunu kullandığında mevcut çalışmasını kaydetmediyse çalışmasının kaybolacağı hakkında bir uyarı verilip, yeni dosya açmak istediğine emin misin sorusu sorulmuştur. Kullanıcı mevcut çalışmasını kaydetmediyse Cancel butonuyla vazgeçebilir. Ya da OK butonuyla mevcut çalışmasını bırakarak yeni bir dosya oluşturabilir.



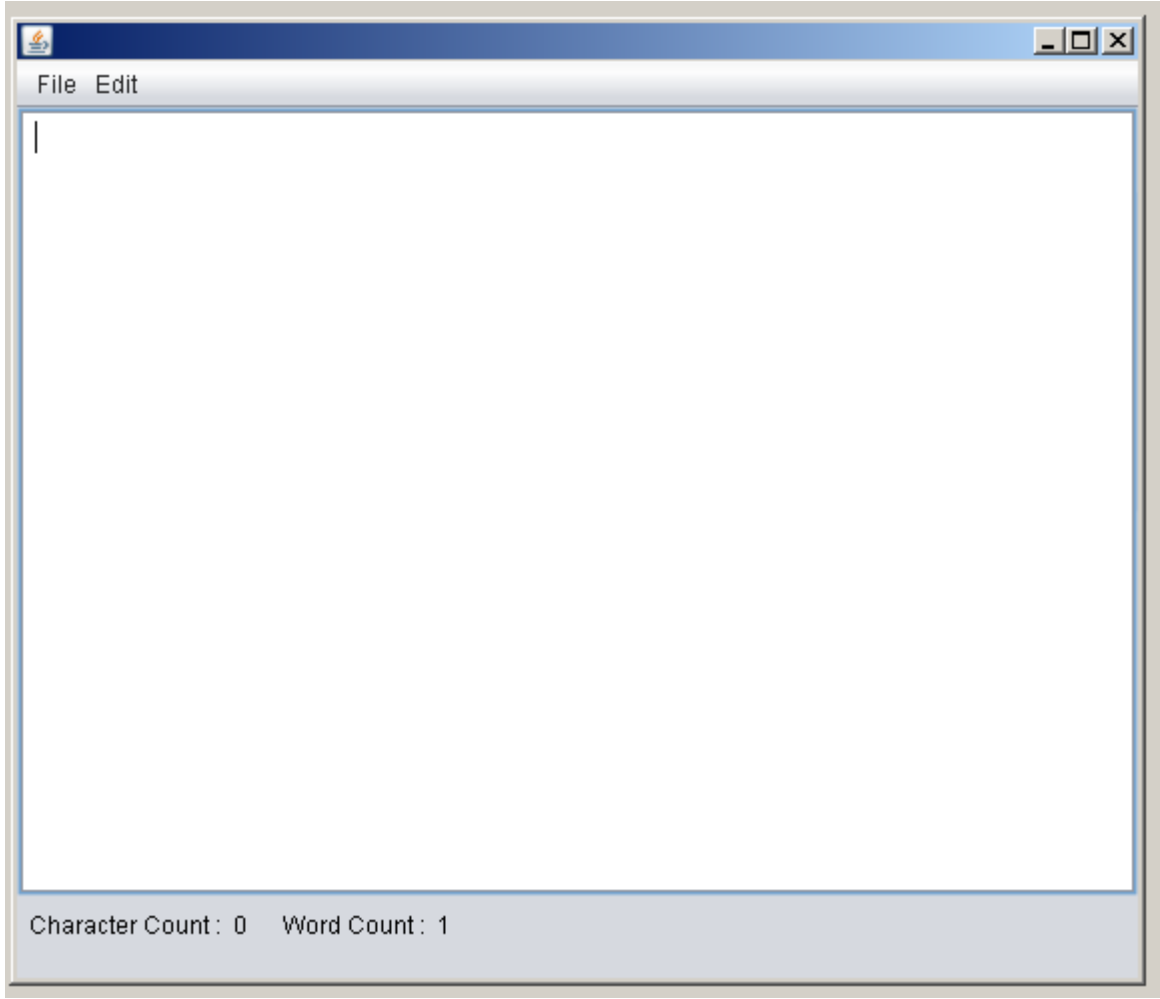
Kullanıcı Save File butonunu kullandığında çalışmasını hangi konuma kaydedeceğini belirlemesi için seçenek sunulur.



Edit sekmesi altında Find and Replace, Undo, Redo, Cut, Copy ve Paste fonksiyonları için butonlar bulunur ayrıca kullanıcının bu fonksiyonlar için kullanabilecekleri kısayollar belirtilmiştir.



Find and Replace butonu kullanıldığında kullanıcının karşısına bulmasını ve değiştirmesini istediği kelimeyi belirtmesi istenmiştir. Kullanıcı bunun dışındaki fonksiyonları ister buton kullanarak isterse kısayolları kullanarak editör üzerinde gerçekleştirebilir.



Kullanıcı sađ üstteki butonlar yardımıyla sırasıyla editörü simge durumuna küçültebilir, ekran boyutuna kaplayabilir ya da editörü kapatabilir.

```

@Test
public void testCalculateDistance() {
    assertEquals(0, DamerauLevenshtein.calculateDistance("", ""));
    assertEquals(0, DamerauLevenshtein.calculateDistance(" ", " "));
    assertEquals(1, DamerauLevenshtein.calculateDistance("", " "));
    assertEquals(1, DamerauLevenshtein.calculateDistance(" ", ""));
    assertEquals(0, DamerauLevenshtein.calculateDistance("test", "test"));
    assertEquals(1, DamerauLevenshtein.calculateDistance("Test", "test"));
    assertEquals(1, DamerauLevenshtein.calculateDistance("test", "testy"));
    assertEquals(1, DamerauLevenshtein.calculateDistance("testy", "test"));
    assertEquals(1, DamerauLevenshtein.calculateDistance("test", "tets"));
    assertEquals(1, DamerauLevenshtein.calculateDistance("test", "test "));
}

@Test(expected = IllegalArgumentException.class)
public void testSourceNull() {
    DamerauLevenshtein.calculateDistance(null, "");
}

@Test(expected = IllegalArgumentException.class)
public void testTargetNull() {
    DamerauLevenshtein.calculateDistance("", null);
}

@Test(expected = IllegalArgumentException.class)
public void testSourceAndTargetNull() {
    DamerauLevenshtein.calculateDistance(null, null);
}

```

Yazım hatası kontrolü ve düzeltilmesi için kullanılan Damerau-Levensthein uzaklık algoritmasının JUnit Testinin gerçekleştirimi şekildeki gibi yapılmıştır.