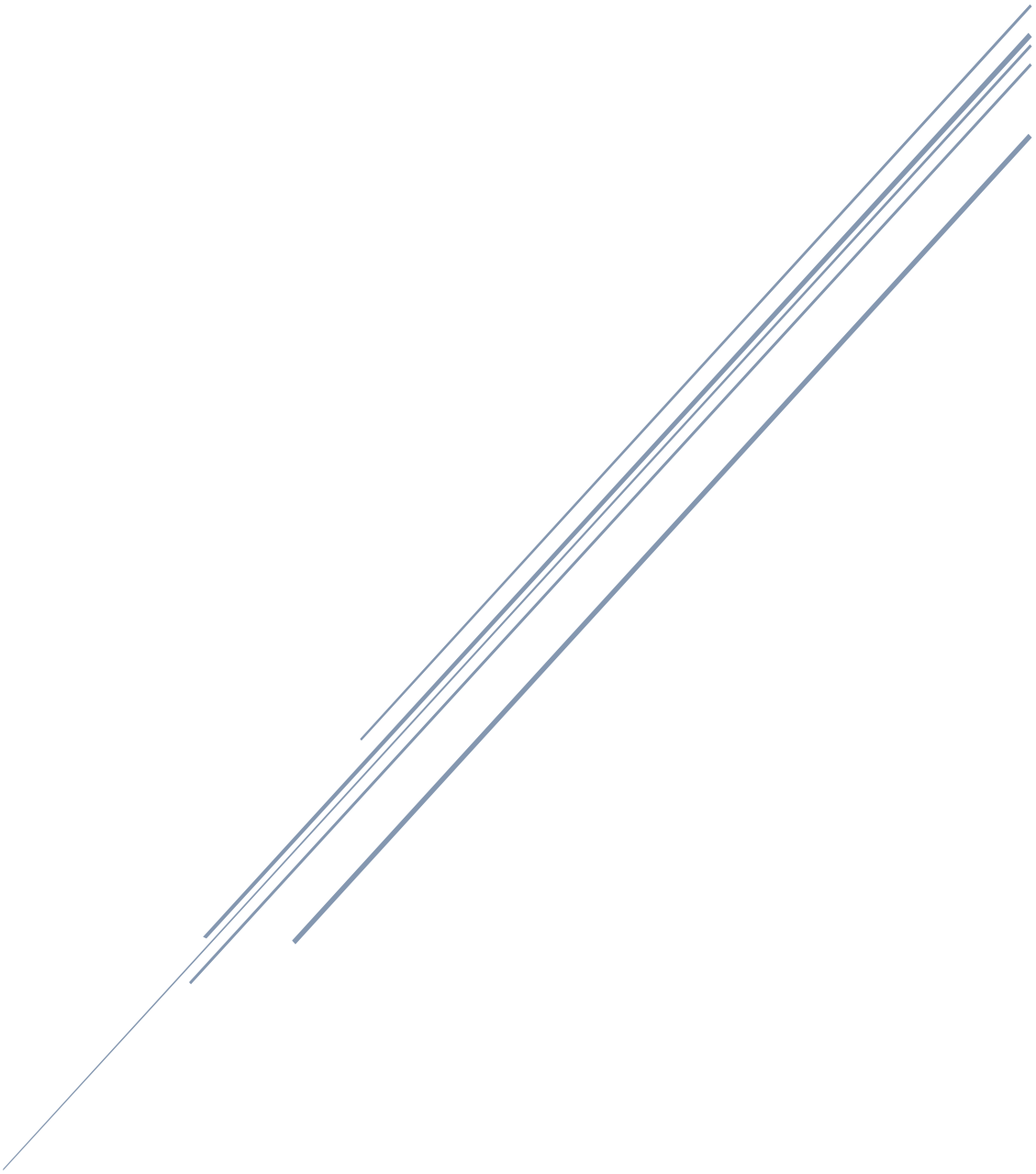
NESNEYE DAYALI PROGRAMLAMA

PROJE-2



05170000073-05170000049
HASAN ORAL-KEREM ALP ÖZECİK

NESNEYE DAYALI PROGRAMLAMA PROJE2 RAPORU

PROJENİN OLUŞTURULDUĞU DİL VE PLATFORM

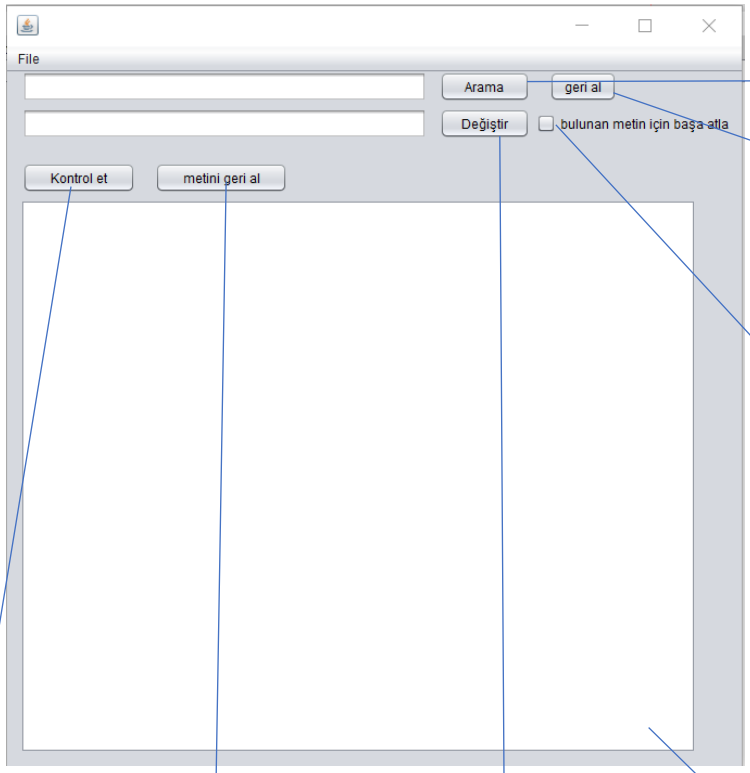
Projemizi Java dilinde yazdık ve gerçekleştirimini Netbeans 8.2 IDE'sinde yaptık.

PROJENİN TANIMI

Bu projemizde ilk projede yaptığımız ödeve ek olarak metin editörü (text editor) uygulamasının tasarım desenleri (design patterns) kullanılarak yeniden yapılandırılması (yeni bir versiyon geliştirilmesi) beklenmektedir.

Bu açıklama doğrultusunda geri alma işlevselliğini "Command" tasarım deseni kullanarak gerçekleştirdik. Koleksiyon dolaşma gereken yerlerde Java ortamında hazır olarak bulunan "Iterator" deseni gerçekleştirimini kullandık ve bu iki pattern dışında text editörümüze eklediğimiz ekstra iki fonksiyonun gerçekleştirimini Abstract Factory Pattern ve Strategy Patternleri ile gerçekleştirdik. Abstract Factory Pattern ile tema değişikliği yapmamızı sağlayan fonksiyon ve Strategy Pattern ile kullanıcı tarafından girilen kelime sayısı, noktalama işareti sayısı ve karakter sayısını sayan fonksiyonlarımızın gerçekleştirimini yaptık.

KULLANIM KILAVUZU



Aranan kelime searchTextArea yazıldıktan sonra bu tuşa basıldığında JTextAreaya girilmiş kelimeleri words.txt'nin içinde arayan fonksiyonu kullanır.

searchTextArea'ya girilmiş kelimeleri harf harf siler.

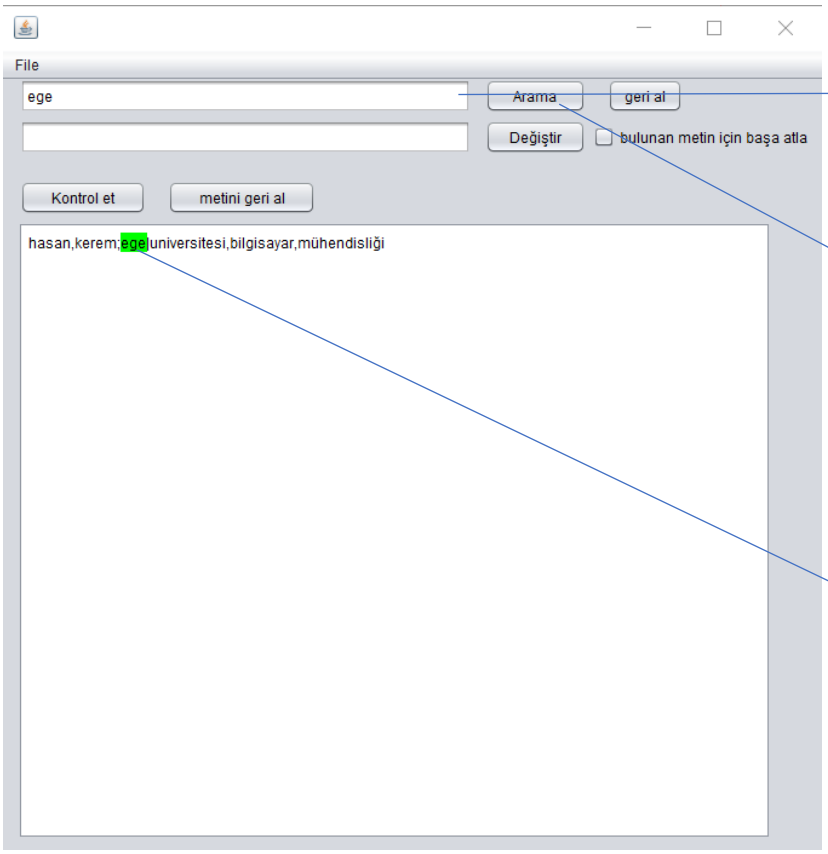
Aradığınız kelimeyi bulduktan sonra en başa almaya yarayan chechbox'tır. Eğer seçilmemişse en son bulduğu kelimede hata verir basılmışsa en son kelimeden sonra ilk bulduğu kelimeye döner.

JTextAreya girilen kelimelerin words.txt'de olup olmadığını kontrol eden butondur.

JTextArea'ya girilenleri harf harf silen butondur.

İstenilen kelimenin değiştirilmesini sağlayan butondur.

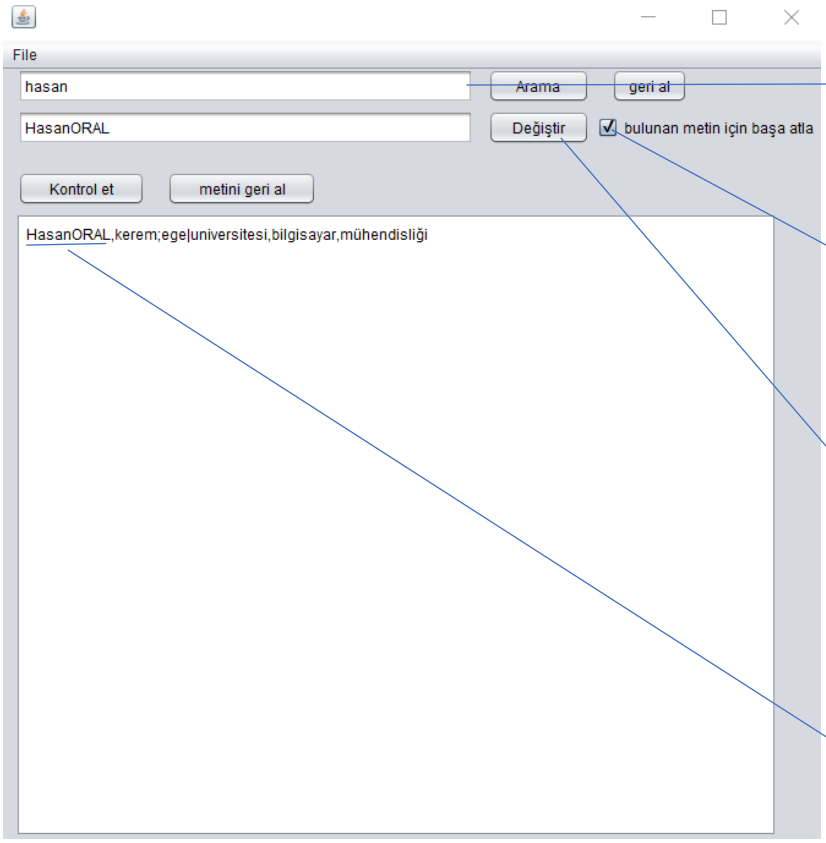
JTextArea'ya yani buraya girdiğimiz kelimeler words.txt'de var mı diye kontrol edebileceğimiz, arama, değiştirme işlemlerini yapabileceğimiz metinin girildiği yer.



1)JTextArea'da bulunmasını istediğimiz kelimeyi girdik.

Kullanılan buton.

Bulunan kelimemiz Highlighter kullanımı ile üstü boyalı şekilde işaretlenir.

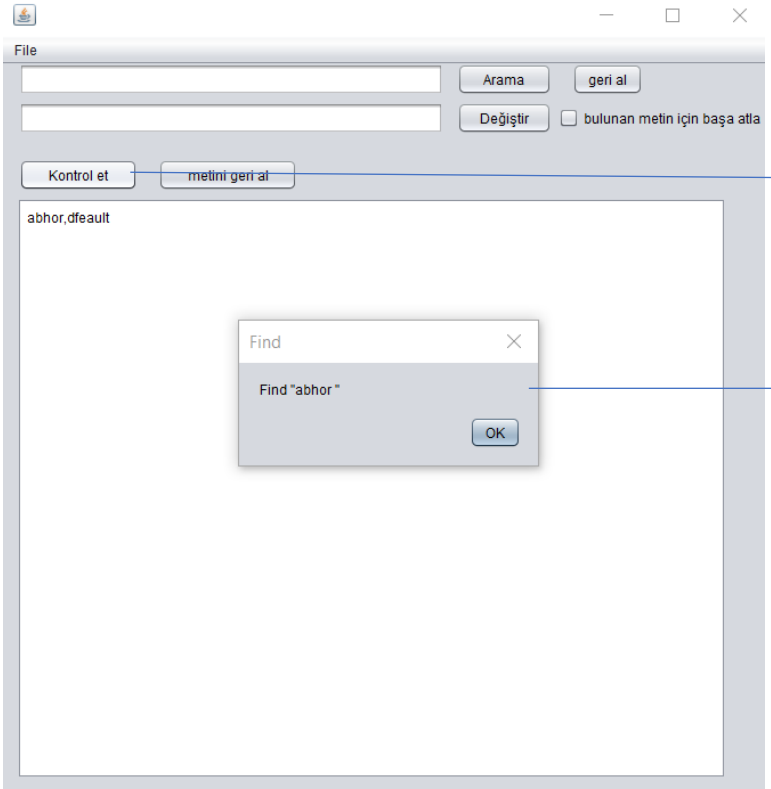


Değiştirilmesi istediğimiz kelimeyi searchTextArea'ya girdik.

İlk "ege" kelimesini aradığımız için ve "hasan" kelimesi önde olduğu için başa atla checkbox'nı işaretledik.

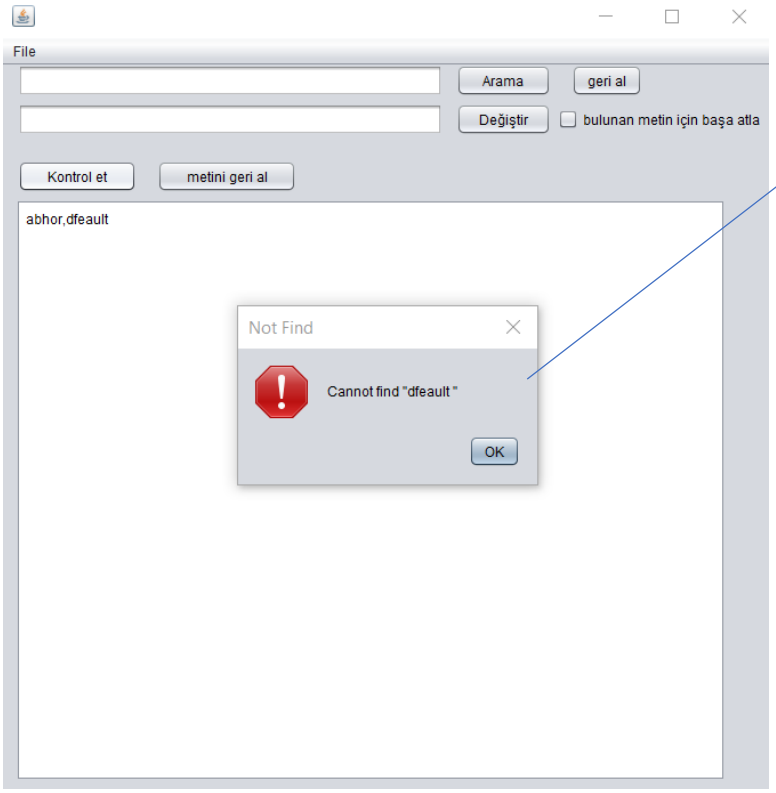
Kullanılan buton.

Görüldüğü üzere "hasan" kelimesi yerine artık "HasanORAL" yazıyor.

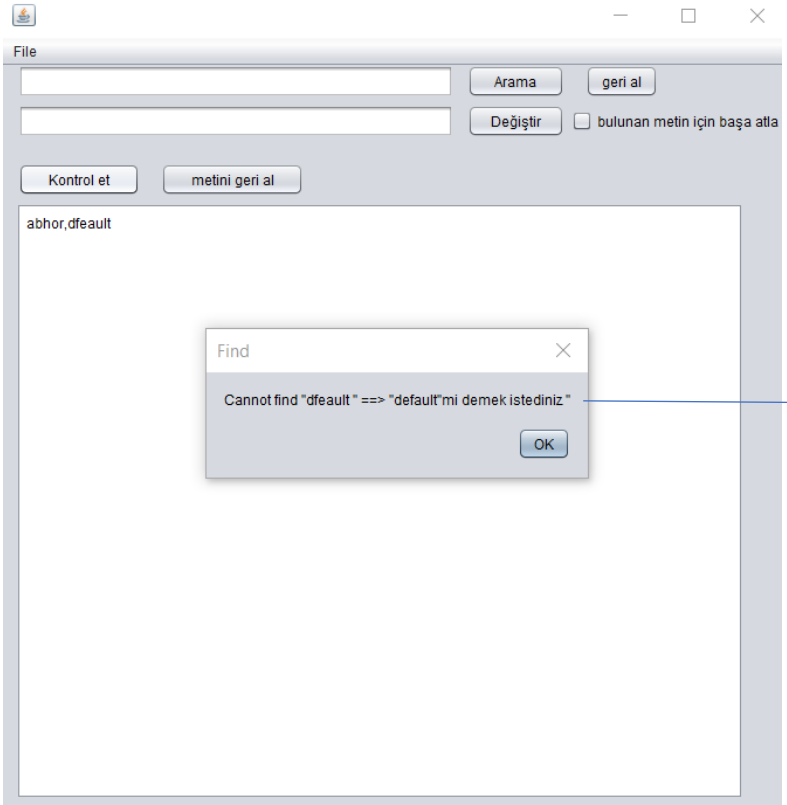


Words.txt dosyasında olan kelimenin varlığını ve yazım hatası olup olmadığını kontrol eden buton.

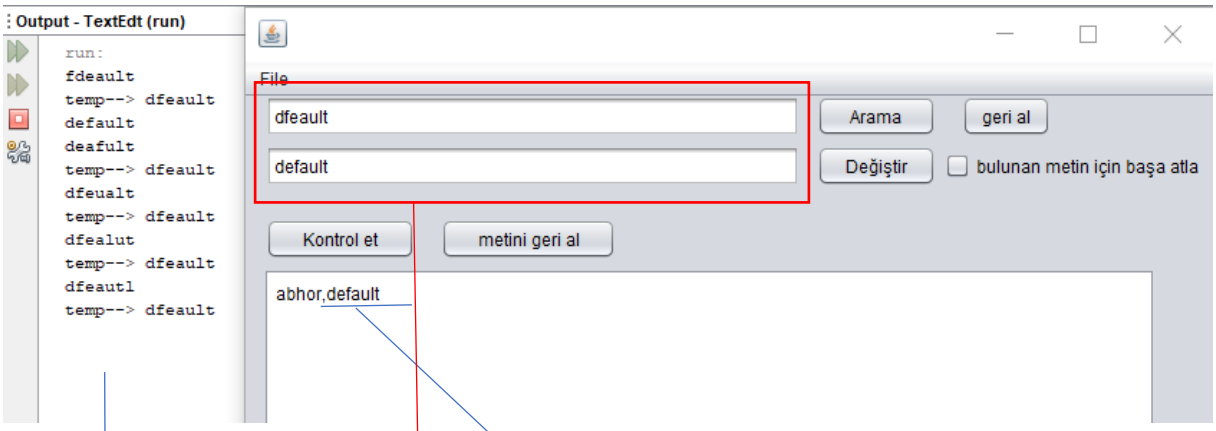
İlk olarak "abhor" kelimesinin varlığı kontrol edildi ve olduğu için bulundu bildirisi çıktı.



Gördüğümüz üzere word.txt'de "dfeault" kelimesi bulunamadı. Kelime bulunamadıktan sonra single transposition işlemi yapılır bu işlemden sonra eğer kelime words.txt'de varsa bulundu bildirisi, yoksa bulunamadı bildirisi ekrana gelir.



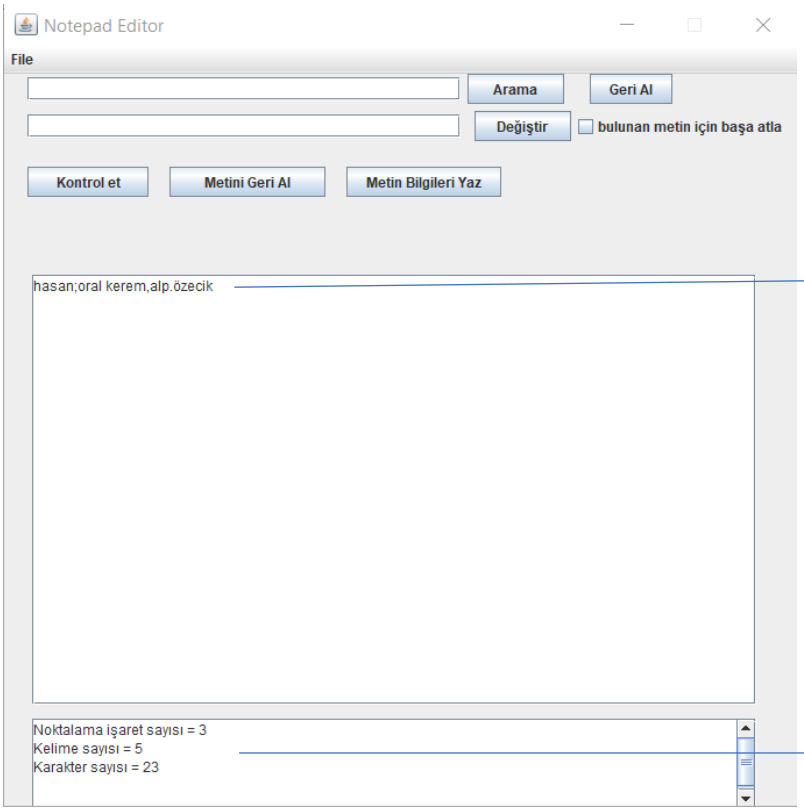
Single transposition yapıldıktan sonra eğer kelime bulunmuş ise önce bu şekilde uyarı mesajımız ekrana gelir.



Single transposition işlemimizin yapılırken konsoldaki görüntüsü.

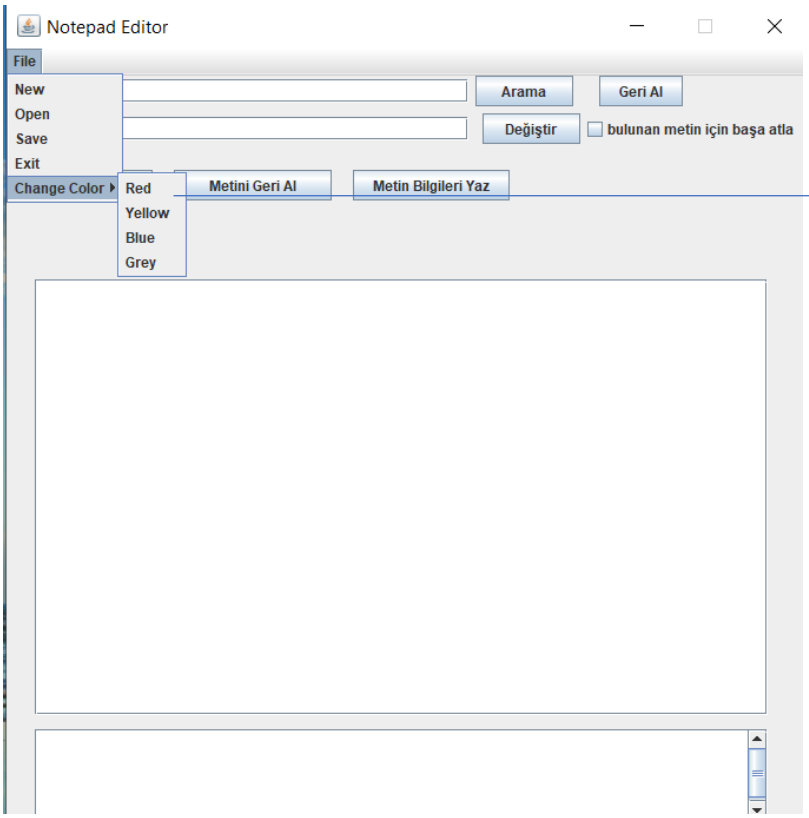
Görüldüğü üzere kelime bulunduğu için JTextArea'da düzeltildi.

Single Transposition ile kelimenin doğrusu olduğu zaman kelimenin hatalı hali ile doğru hali gerekli yerlere atanarak kelime doğru hali ile değiştirilir.

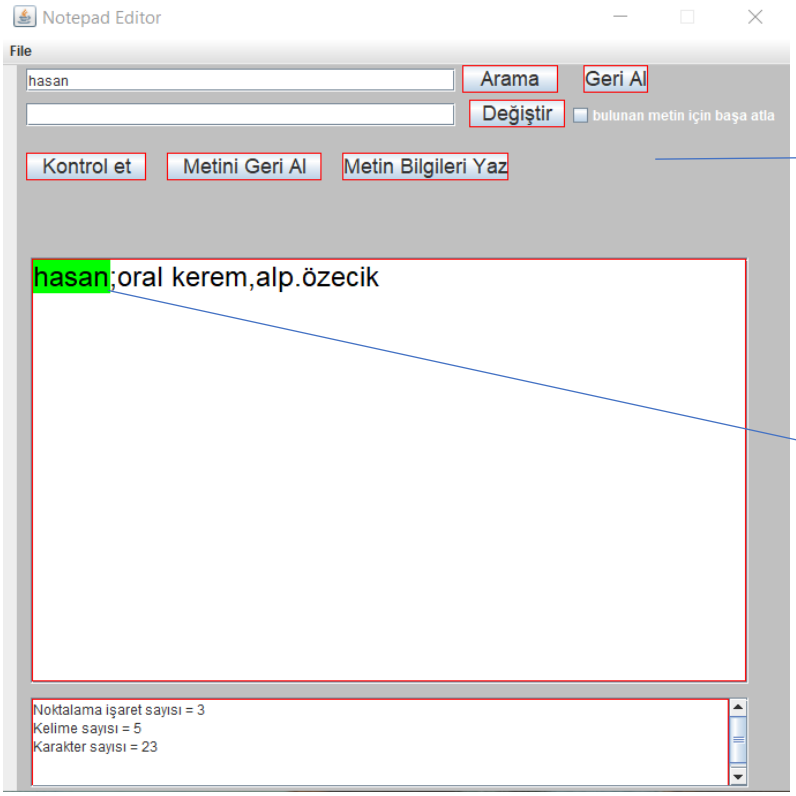


İlk projede olduğu gibi aranmasını, kontrol edilmesini istediğimiz ya da değiştirmek isteğimiz kelimeler JTextArea'ya yazılır.

Strategy pattern yapısı ile yazdığımız yeni fonksiyonumuz JTextArea'ya girilen kelime sayısı, karakter sayısı ve noktalama işareti sayısını bize verir.

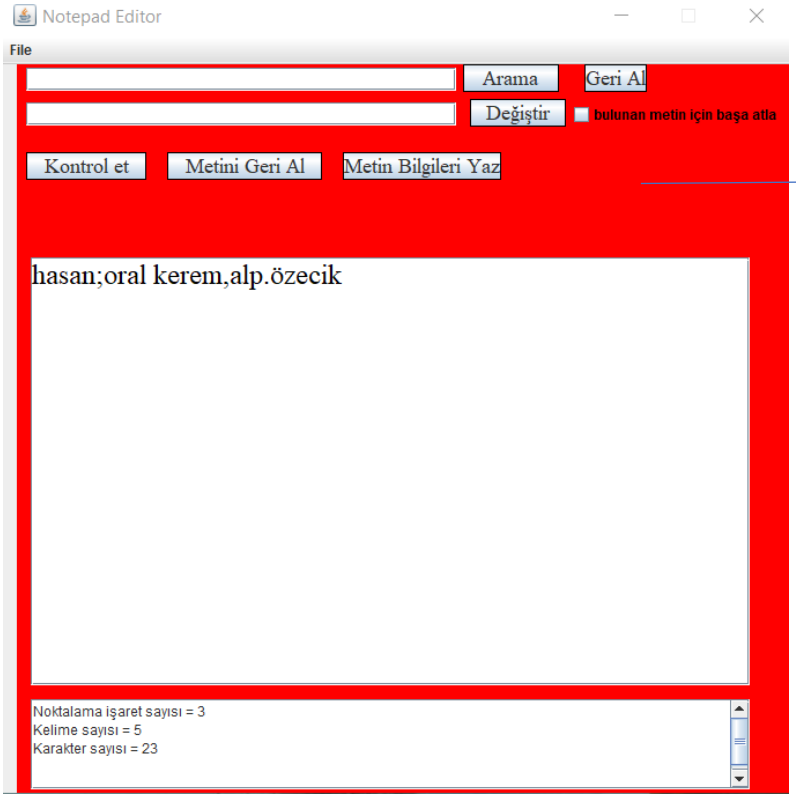


Abstract Factory Pattern yapısı ile eklediğimiz fonksiyon ile text editörümüzün temasını değiştirebiliyoruz.



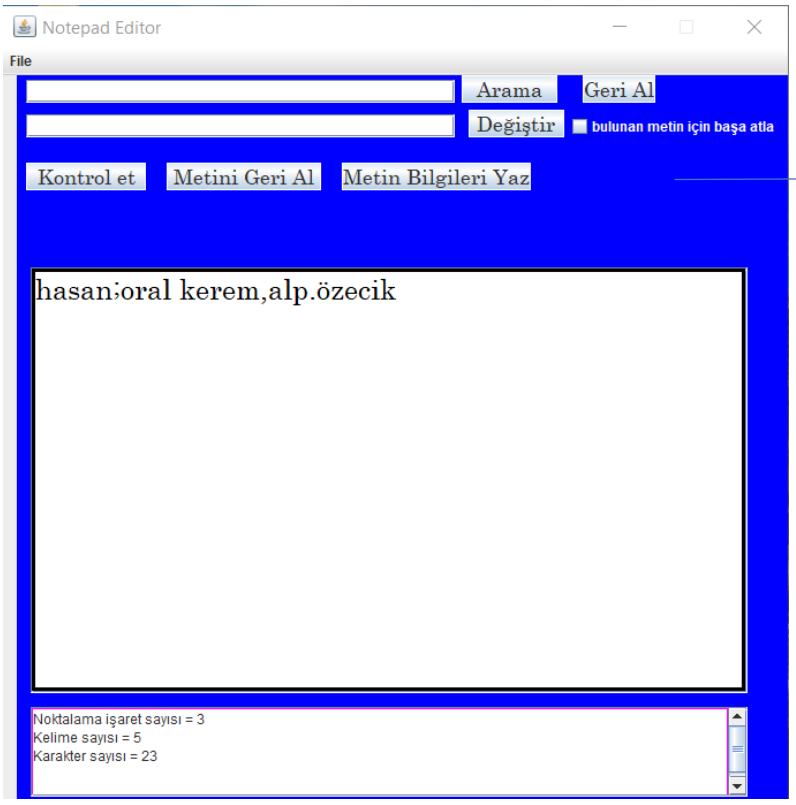
Grey seçeneğini seçtiğimizde arka plan fon rengi ve kullandığımız yazı fontu değişiyor.

Görüldüğü üzere tema değişimi önceki projede oluşturduğumuz fonksiyonların çalışmasına engel arz etmiyor.



Red seçeneğini seçtiğimizde arka plan fon rengi ve kullandığımız yazı fontu değişiyor.

Yellow seçeneğini seçtiğimizde arka plan fon rengi ve kullandığımız yazı fontu değişiyor.



Blue seçeneğini seçtiğimizde arka plan fon rengi ve kullandığımız yazı fontu değişiyor.

KULLANILAN PATTERNLER

Abstract Factory Pattern: Factory tasarım deseni, benzer özellikleri olan alt sınıflara ortak bir arayüz uygular ve nesne yaratma sorumluluğunu, bu arayüz üzerinden gerçekleştiren bir factory sınıfına verir.

Command Pattern: Davranışsal (ing. Behavioral) grupta yer alan bir tasarım kalıbıdır. Bir isteği nesneye dönüştürerek, isteğin kullanıcı sınıfları tarafından rahatça erişilebilmesi sağlar. Nesne üzerinde bir işleminin nasıl yapıldığını bilmediğimiz ya da kullanılmak istenen nesneyi tanımadığımız durumlarda kullanılır. Yapılmak istenen işlemi bir nesneye dönüştürerek, alıcı nesne tarafından işlemin yerine getirilmesi sağlar.

Iterator Pattern: Birleşik bir nesnenin bileşenlerine, nesnenin esas ifadesinin gösterilimini açığa çıkarmadan sırayla erişebilmeyi sağlar. Tekrarlayıcı tasarım kalıbı, bir listenin yapısının ve çalışma tarzının uygulamanın diğer kısımları ile olan bağlantılarını en aza indirmek için; listede yer alan nesnelerin, sırasıyla uygulamadan soyutlanması amacıyla kullanılır.

Strategy Pattern: Bir işlemi yapabilmek için birden fazla yöntem (algoritma) mevcut olabilir. Yerine göre bir yöntem seçip, uygulamak için Strategy tasarım şablonu kullanılır. Her yöntem (algoritma) bir sınıf içinde implemente edilir.

KULLANDIĞIMIZ SINIFLAR VE PATTERN AÇIKLAMALARI

AbstractFactory.java: Abstract Factory Pattern'i için abstract class açılır. getTheme fonksiyonumuz burada bulunur ve bağlantılı sınıflarımıza override edilir.

ArrayIterator.java: Iterator Design Pattern için oluşturduğumuz sınıfımız, içinde statik diziler için hasNext ve next fonksiyonları bulunuyor.(NOT: Hazır iteratör classından extend ediliyor.).

BackSpace.java: Geri alma işlemi için yapılan Command pattern'in BackSpace classıdır. İçinde command pattern için en son kullandığımız execute fonksiyonunun temeli olan delete fonksiyonu bu classta yazılmıştır.

BackSpaceCommand.java: Geri alma işlemi için yapılan Command pattern'in BackSpaceCommand classıdır. Execute fonksiyonu bu classta bulunuyor.

ChangeBlueTheme.java: Blue teması için yapılan değişiklikler(Abstract Factory Pattern) bu classta yer alır.

ChangeRedTheme.java: Red teması için yapılan değişiklikler(Abstract Factory Pattern) bu classta yer alır.

ChangeYellowTheme.java: Yellow teması için yapılan değişiklikler(Abstract Factory Pattern) bu classta yer alır.

ChangeGreyTheme.java: Grey teması için yapılan değişiklikler(Abstract Factory Pattern) bu classta yer alır.

Command.java: Command Pattern'ı için oluşturduğumuz interfacedir. Execute fonksiyonu burada da tanımlıdır.

Context.java: Strategy Pattern'i için oluşturduğumuz classtır. Bu classımızda executeStrategy fonksiyonumuz bulunur, bu fonksiyonumuzu mainde kullanıyoruz. Böylece kelime, noktalama işareti ve karakter sayısını bulma işlemlerini gerçekleştiriyoruz.

FactoryProducer.java: Abstract Factory Pattern'i için FactoryProducer classını açtık. Bu classın işlevi "color"un aldığı değere göre tema değişir. Color parametresi mainde getTheme ile hangi temayı kullanacağına karar verir. getTheme fonksiyonu bu classımız içinde yer alıyor.

Fonksiyonlar.java: İlk projemizde bu sınıfta çeşitli fonksiyonlarımız bulunuyordu fakat patternlarla düzenlemeler yapıldıktan sonra bu classımızda sadece openFile fonksiyonu yer alıyor.

Invoker.java: Command Pattern için oluşturduğumuz Invoker classıdır. “execute” fonksiyonu(geri alma-silme işlemleri için)yer alır.

OperationCharacterCounter.java: Strategy pattern için kullanılacak olan karakteri sayma sınıfıdır. “doCount” fonksiyonu ile karakterleri sayan fonksiyonumuzdur.

OperationWordCounter.java: Strategy pattern için kullanılacak olan Kelime sayma sınıfıdır. “doCount” fonksiyonu ile kelimeleri sayan fonksiyonumuzdur.

OperationNoktalamaCounter.java: Strategy pattern için kullanılacak olan noktalama işaretlerini sayma sınıfıdır. “doCount” fonksiyonu ile noktalama işaretlerini sayan fonksiyonumuzdur.

Strategy.java: Strategy Patterni için interfacedir. Diğer fonksiyonlarda kullanılacak olan sayma fonksiyonumuz doCount fonksiyonu burada tanımlıdır.

TextAreaChanger.java: Abstract Factory Pattern için interfacedir. Buradaki change fonksiyonu diğer classlarda kullanılacak tema değiştirme işlevini görür.

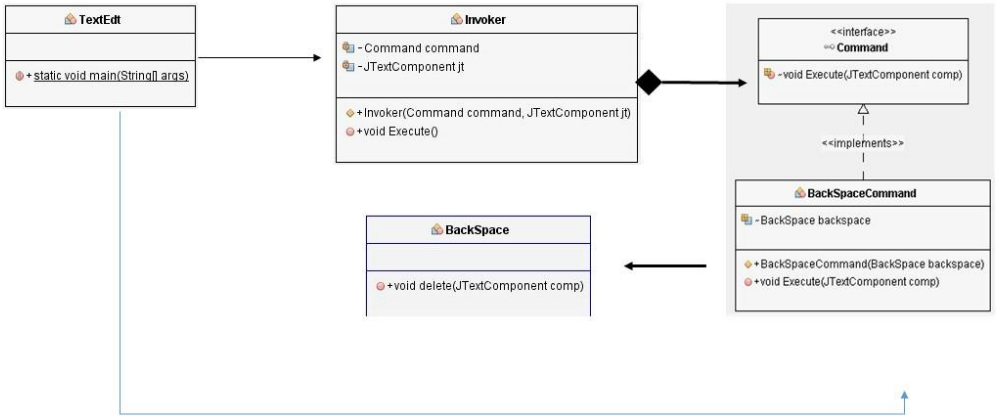
ThemeFactory.java: Abstract Factory Pattern'i için ThemeFactory classının abstract classa extend edildiği classımızdır. AbstractFactory.java’daki getTheme fonksiyonumuz burada override edilir.

TextEdt.java: Main classımızdır.

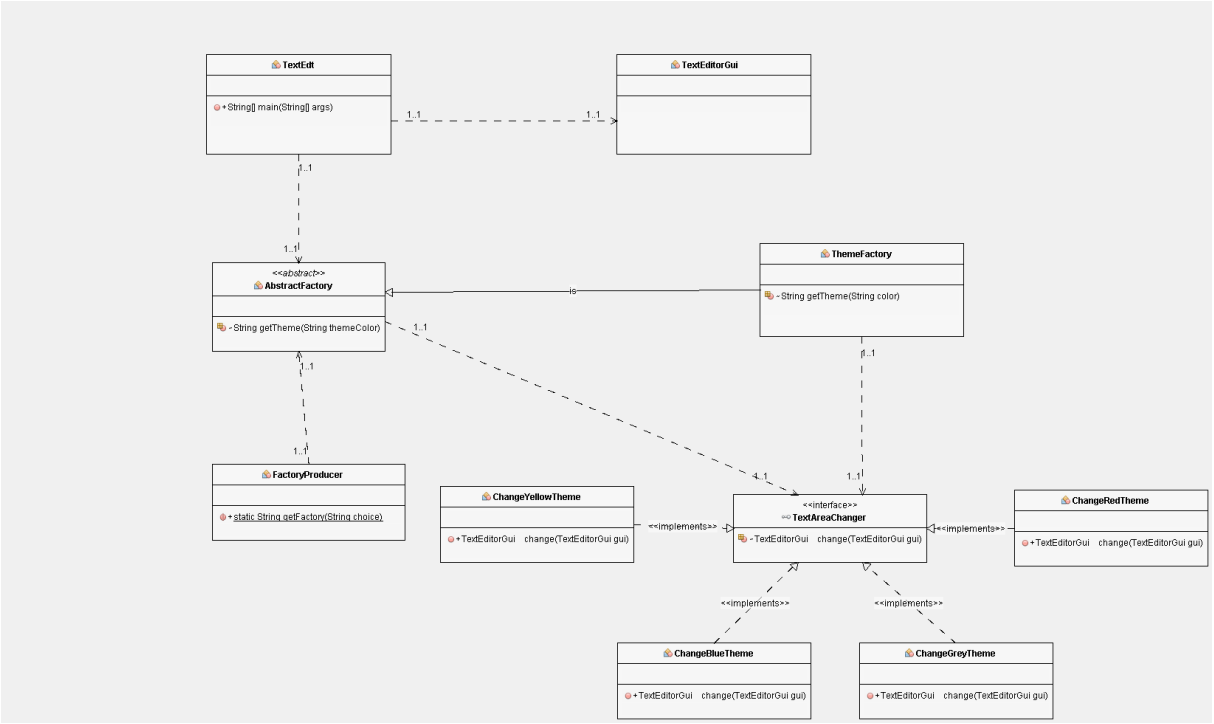
TextEdtGui.java: Arayüzümüzün bulunduğu ve çoğunlukla buton işlevlerini yerine getiren fonksiyonların olduğu classımızdır.

UML DİAGRAMI

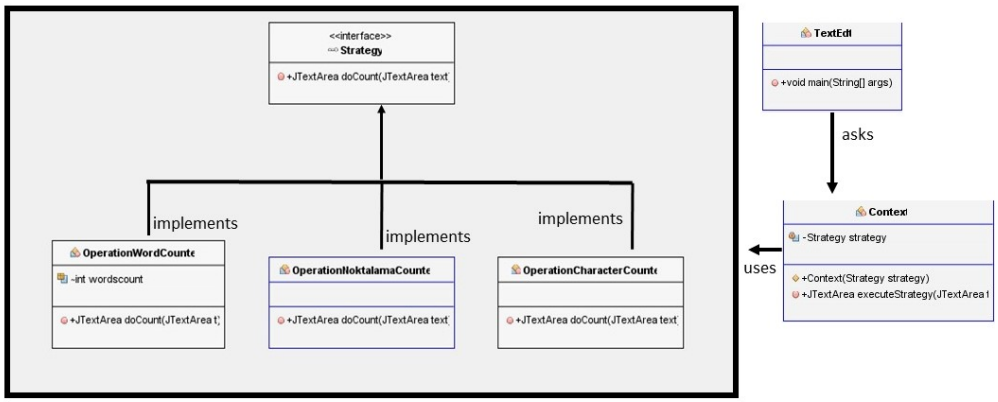
-Command Pattern



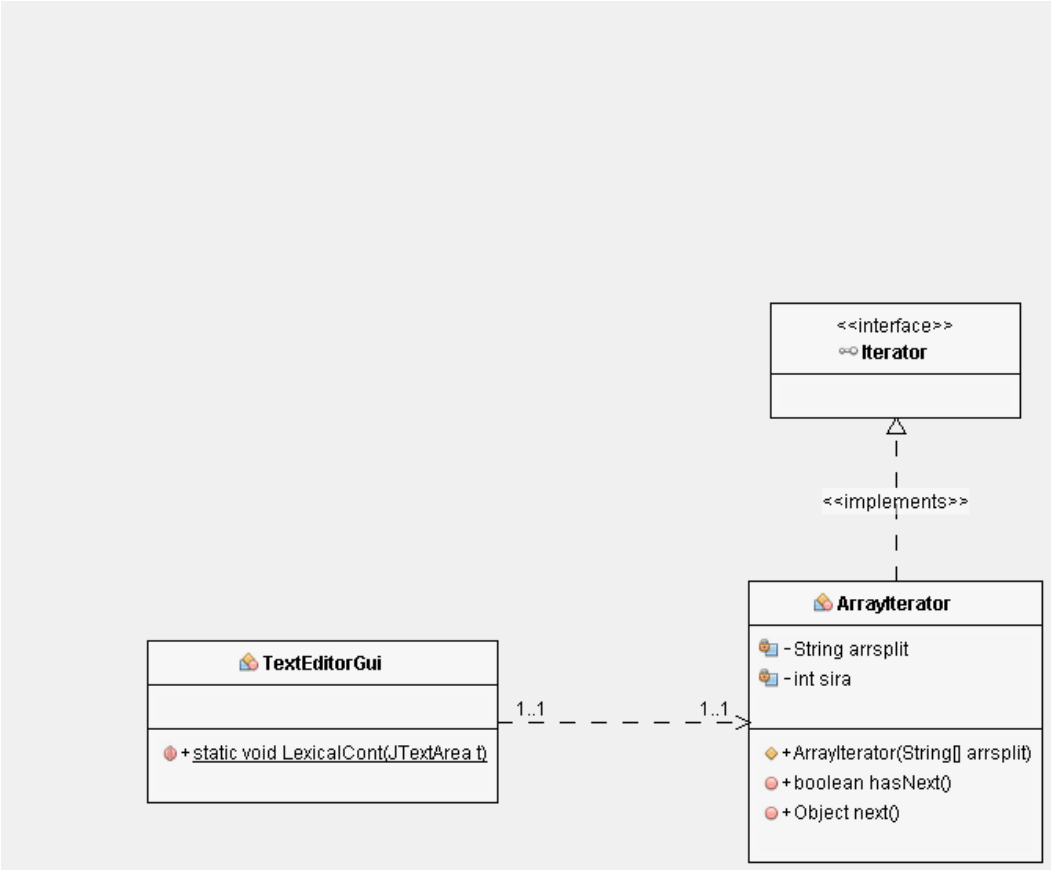
-Abstract Factory Pattern



-Strategy Pattern



-Iterator Pattern



PROJE İÇİN HARCANAN ZAMAN

Bu proje için grup olarak (2 kişi) toplam 8 saat ayırdık. Bu 7 saat içinde 1 saati planlama, 4.5 saati program tasarımı ve yapımı, kalan 2.5 saati de raporlamak için harcadık.