NESNEYE DAYALI PROGRAMLAMA 2020/2021 GÜZ DÖNEMİ

PROJECT 2 (DESIGN PATTERNS)

İZZET OĞUZ BOZAK
05170000043

FATİH RAMAZAN BATMAN
05170000786

01 ŞUBAT 2021

İZMİR

1)Command tasarım deseni için *geriAlCommand* ve *yonlendiriciCommandDesignPattern* classları oluştrulmuştur.Geri al butonuna tıklanıldığında gerekli işlemler yapılmaktadır.Geri al butonu code kısmında açıklama olarak belirtilmiştir.

geriAlCommand Classı:

```
public class geriAlCommand implements Islem {

      public String geriAlSonHarf(String kelime){

      kelime=kelime.substring(0, kelime.length()-1);

      return kelime;

                                  }

}
```

YonlendiriciCommandDesignPattern Classı

```
public class yonlendiriciCommandDesignPattern {

   public static String yonlendir(Islem islem, String kelime){

      String sonHarfSilinen = islem.geriAlSonHarf(kelime);

      return sonHarfSilinen;

   }

}
```

2)Iterator tasarım deseni için hazır kullanım sağlanmış olup kelimeHataliMi metodu bu pattern vasıtasıyla gerçekleştirilmiştir.

```
int i = 0;    // Iterator design pattern burada kullanıldı.
   for (Iterator iter = metinNesnesi.getIteratorSozcukler(); iter.hasNext(); i++) {
      String nextString=(String) iter.next();
      if( metinNesnesi.kelimeHataliMi(nextString,sozlukArrayList) )
        metinNesnesi.getSozcukler().set(i, metinNesnesi.singleTransposition
      (nextString,sozlukArrayList) );
   }
     metin.setText(metinNesnesi.metinBirlestir(metinNesnesi.getSozcukler()));
```

```
}
```

3)Memento tasarım deseni command tasarım desenine benzer olarak geri al işlevi için kullanılmıştır.Bu arada ayrım yapılabilmesi için iki desenden biri program açıldığında yorum satırı içinde durmaktadır.Gerçekleştirim yapılabilmesi için MetinMemento classı oluşturulmuştur.Değişiklik yapıldıkça her olayı hafızaya alıp geri dönebilmektedir.

MetinMemento Classı

```
public class MetinMemento {
   private String metinAdi;
   private LinkedList<String> sozcukler;

   public MetinMemento(String metinAdi,String metin){
      this.metinAdi=metinAdi;
      sozcukler=metinParcala(metin);
   }
   public MetinMemento(){};

   public MetinMemento(String metinAdi,LinkedList<String> sozcukler){
      this.metinAdi=metinAdi;
      this.sozcukler=sozcukler;
   }
   public LinkedList<String> metinParcala(String metin){
      String[] parts = metin.split("(?!^)\\b");
      LinkedList<String> liste = new LinkedList<String>();
      for(String kelime:parts) liste.add(kelime);
      return liste;
   }

   /**
    * @return the metinAdi
    */
```

```
    public String getMetinAdi() {
        return metinAdi;
    }

    /**
     * @param metinAdi the metinAdi to set
     */
    public void setMetinAdi(String metinAdi) {
        this.metinAdi = metinAdi;
    }

    /**
     * @return the sozcukler
     */
    public LinkedList<String> getSozcukler() {
        return sozcukler;
    }

    /**
     * @param sozcukler the sozcukler to set
     */
    public void setSozcukler(LinkedList<String> sozcukler) {
        this.sozcukler = sozcukler;
    }

}
```

hafizaMemento Classı

```
public class hafizaMemento {
    private LinkedList<MetinMemento> listeMetinMemento;
    private MetinMemento[] sonCikarilanListeMetinMemento;
```

```
public hafizaMemento(){
   listeMetinMemento= new LinkedList<MetinMemento>();
   sonCikarilanListeMetinMemento= new MetinMemento[1];
}

public MetinMemento getSonCikarilanListeMetinMemento(){
   return sonCikarilanListeMetinMemento[0];
}

// En fazla 50 geri dön kullanılabilir.
public void hafizaMementoEkle(MetinMemento memento){
   listeMetinMemento.add(memento);
   if (listeMetinMemento.size() > 50) listeMetinMemento.removeFirst();
}

public LinkedList<MetinMemento> getListeMetinMemento(){
   return listeMetinMemento;
}

public MetinMemento sonVersiyonuGetir(){
   if(!listeMetinMemento.isEmpty()){
      MetinMemento sonEleman =
  listeMetinMemento.get(listeMetinMemento.size()-1);
      listeMetinMemento.remove(sonEleman);
      sonCikarilanListeMetinMemento[0]=sonEleman;
      return sonEleman;
   } else{
      throw new ArrayIndexOutOfBoundsException("Geri al kullanılamaz.");

   }
}
```

```
}
```

4)Observer tasarım deseni dosyanın kaydedilip kaydedilmemesini kontrol etmek için kullanıldı.Kullanıcı kaydetmeden çıkmak isterse uyarı mesajı vererek durum kontrolü sağlanıyor olmaktadır.Observer ve Observable isimli 2 adet class oluşturulmuştur.

Observer Classı

```
public interface Observer {
   void update(Observable o);
}
```

Observable Classı

```
public abstract class Observable {
   private List<Observer> observerList;
   public Observable() {
      observerList=new ArrayList<>();
   }

   public void ekle(Observer observer){
      observerList.add(observer);
   }

   public void cikar(Observer observer){
      observerList.remove(observer);
   }

   public void haberVer(){
      for(Observer o:observerList) o.update(this);
   }

}
```

Aşağıda verilen Text sınıfı ise Observable Classından kalıtım alarak kaydedilme işlemi için yazılmıştır.

```
public class Text extends Observable{
   private String metin;
   public Text(String m){
      metin=m;
   }

   /**
    * @return the metin
    */
   public String getMetin() {
      return metin;
   }
   /**
    * @param metin the metin to set
    */
   public void setMetin(String metin) {
      this.metin = metin;
   }
   /**
    * @return the kayitliOlanMetin
    */
   public boolean kaydedildiMiKontrolEt(String butunSatirlar){  // Nesnede bulunan değişiklikler
      if(!metin.equals(butunSatirlar)  ){ // dosya olarak kaydedildimi sorgular.
         haberVer();
         return false;
                                          }
      else return true;
   }
}
```

UML SINIF DİYAGRAMI