Okan AKYOL
05180000094
Mühendislik Departmanı, BİLGİSAYAR MÜHENDİSLİĞİ
EGE Üniversitesi

Proje-2

Genel Bakış

Ders kapsamında istenilen gereksinimleri sağlayan basit bir arayüze sahip yazı editörü gerçekleştirimi.

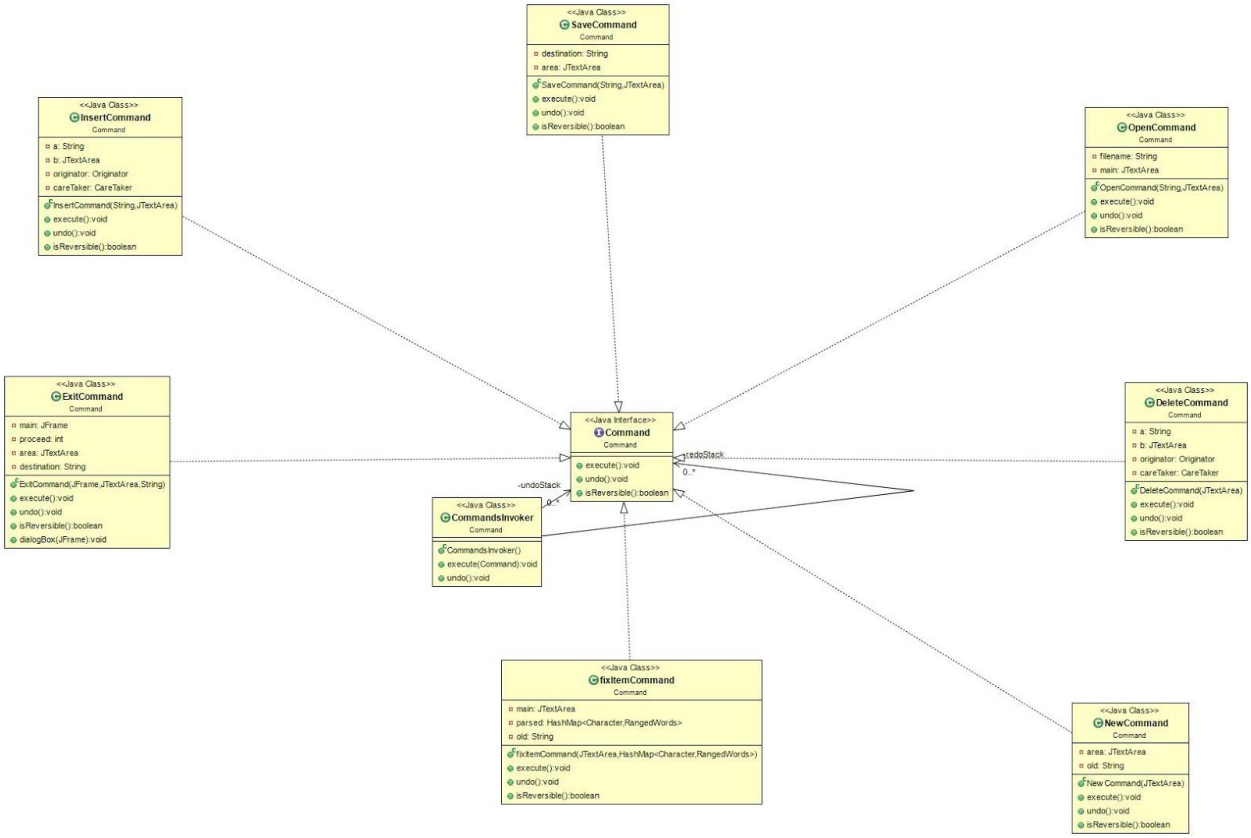
İÇİNDEKİLER

PLATFORM

JAVA 8,JAVA SE 15.0.1,NetBeans

1. “Command” Tasarım Deseni ve Gerçekleştirimi

“Command Paketinin UML Diyagrami”



“Command.java”

```
public interface Command {
	void execute();
	void undo();
	boolean isReversible();
}
```

Arayüz tanımlaması yapıldı ve “isReversible()” ile “CommandsInvoker.java” çalıştırıldığı geri döndürülebilirliğine göre yığıta ekleme gerçekleştirildi.

“CommandsInvoker.java”

```
public CommandsInvoker() {
     undoStack = new Stack<Command>();
     redoStack = new Stack<Command>();
}
public void execute(Command any) {

     any.execute();
     if (any.isReversible()){
          undoStack.push(any);
           }
}
public void undo() {

     if (undoStack.isEmpty()==false) {
          Command a = undoStack.pop();
          redoStack.push(a);
                    a.undo();
           }
}
```

Not:Memento tasarım desenini ve “ChainofResponsibility” tasarım desenini command ile birlikte kullanan kısımlara aşağıda ek olarak anlatıldı.

1.a “DeleteCommand”

Silinen yazıyı harf harf geri alma işlevi için “Memento” tasarım deseni kullanıldı.

“DeleteCommand.java”

```
    public DeleteCommand(JTextArea b){

        this.b = b;
        originator = new Originator();//creating memento
        careTaker = new CareTaker();
    }



    @Override
    public void execute() {//adding memento
            originator.setState(b.getText());
            careTaker.add(originator.saveStateToMemento());
    }

    @Override
    public void undo() {//undo memento
            originator.getStateFromMemento(careTaker.get(0));
            b.setText(originator.getState());
    }

    @Override
    public boolean isReversible() {
        return true;
    }

}
```

1.b “InsertCommand”

Yazma işlevi için kelime yazılıp boşluk bırakıldığında “Memento” tasarım deseninden yararlanarak görüntüsü alınır ve kaydedilir.

“InsertCommand.java”

```
    @Override
    //Usage of memento design pattern for execute process.
    public void execute() {
        originator.setState(b.getText());
        careTaker.add(originator.saveStateToMemento());
        b.append(a);

    }




@Override
    public void undo() {
            //Usage of memento design pattern for undo process.
            originator.getStateFromMemento(careTaker.get(0));
            b.setText(originator.getState());

    }


    @Override
    public boolean isReversible() {
        return true;
    }

}
```

1.c “fixItemCommand”

“ChainofResposnsibility” tasarım deseni kullanılarak bir kelimenin “Single Transposition” hatasına sahip olup olmadığı kontrol edildi.

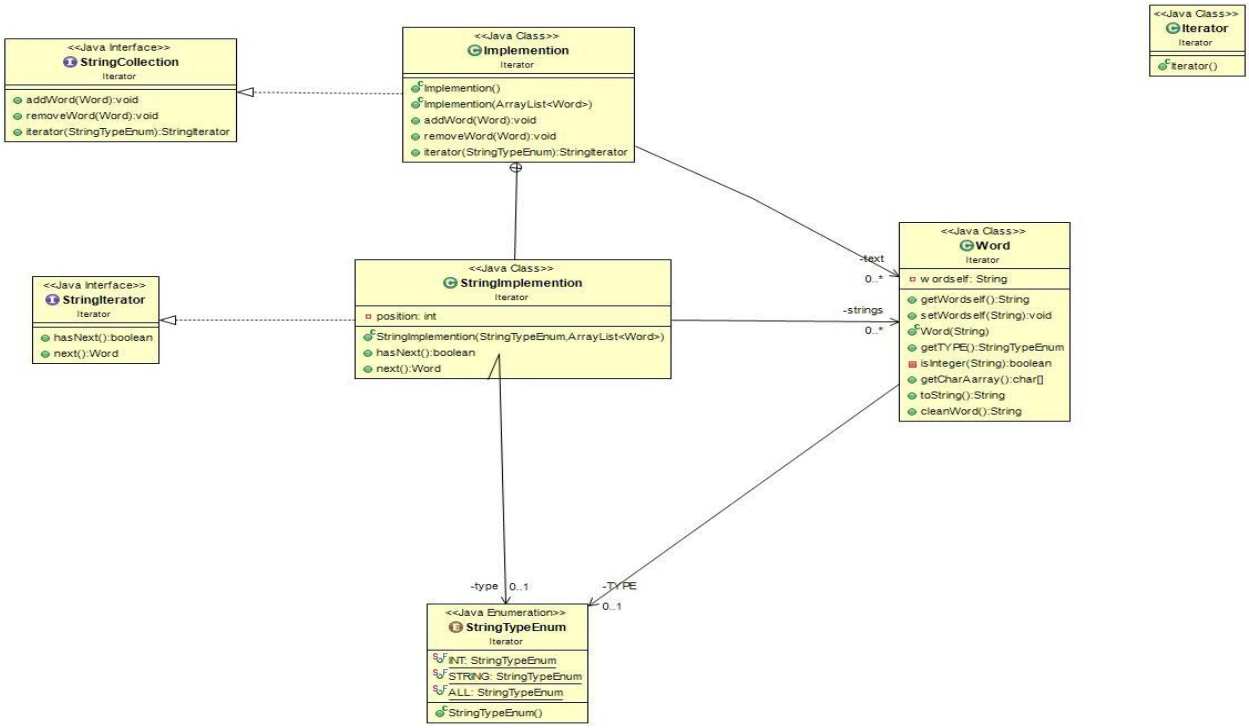
“fixItemCommand.java”

```
    public void execute() {
      FixSingleTr c = new FixSingleTr();//chainofresponsibility creator
      ProceedProcess d = new ValidationLengths();//link procecss
      d.linkWith(new ValidationChars()).linkWith(new
ValidationSingleTrans());//link procecss
      c.setProceedProcess(d);//link procecss
        try{
        for(Word a:words){
            for(int i=0;i<a.cleanWord().length();i++){
                for(String wordarray :
parsed.get(a.cleanWord().toLowerCase().charAt(i)).getWords()){
                String tmp = c.fix(a, new Word(wordarray));//checking for
singletransposition
                if(!"".equals(tmp)){
                    a.setWordself(tmp + " ");//when statement is true
change the words string.

                }

            }
            }
        }
        }catch(StringIndexOutOfBoundsException e){e.printStackTrace();
        }
        String newtext = "";
        for(Word a: words){
            newtext += a;
    }
    main.setText(newtext);
    }


    @Override
    public void undo() {

            main.setText(old);
```

2. “Iterator” Tasarım Deseni ve Gerçekleştirimi

Iterator tasarım deseni hem paket olarak kodlandı hem Java’nın kendisine bulunan method “override” edilerek kullanıldı.

“Iterator Paketinin UML Diyagrami”



2.a “RangedWords”

Java’nın kendisinde bulunanı “override” edilerek kullanıldı. Bu sınıfın kullanım amacı sözlüğün “words.txt” dosyasının gruplanan verilerini depolamak.

“RangedWords.java”

```
public class RangedWords implements Iterable<String> {
    private String[] words;

    public String[] getWords() {
        return words;
    }

    public void setWords(String[] words) {
```

```
        this.words = words;
    }
    private int currentSize;
    public RangedWords(Object[] words){
        this.words = objecttoString(words);
        this.currentSize = words.length;
    }

    @Override
    public Iterator<String> iterator(){
        Iterator<String> it = new Iterator<String>() {

            private int currentIndex = 0;

            @Override
            public boolean hasNext() {
                return currentIndex < currentSize && words[currentIndex] !=
null;
            }

            @Override
            public String next() {
                return words[currentIndex++];
            }

            @Override
            public void remove() {
                throw new UnsupportedOperationException();
            }
        };
        return it;
    }
private String[] objecttoString(Object[] a){
    String[] stringArray = Arrays.copyOf(a, a.length, String[].class);
    return stringArray;
}

}
```

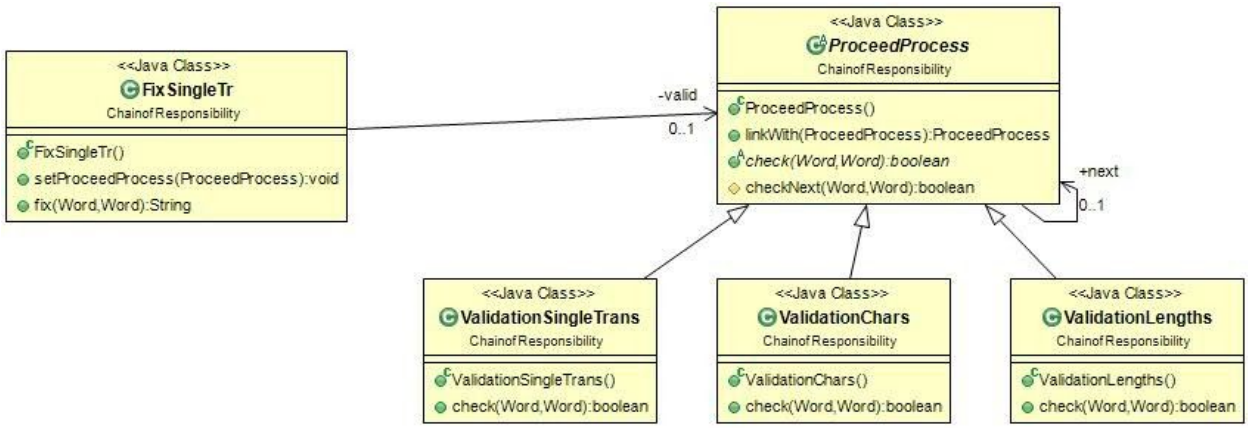
3.”Memento” Tasarım Deseni ve Gerçekleştirimi

Yazı editöründe “Snapshot” özelliği getirildi ve gerçekleştirim memento tasarım deseni kullanılarak yapıldı.

4.”ChainOfResponsibility” Tasarım Deseni ve Gerçekleştirimi

Herhangi bir kelimenin sözlüğe göre “Single Position” sahip olup olmadığını kontrol etmek için bu tasarım deseni kullanıldı. Bu parçaların birbirine bağlanması “Command” paketinde gerçekleştirildi.

“ChainofResponsiblity UML Diyagramı”



5.”Singleton” Tasarım Deseni ve Gerçekleştirimi

Yazı editöründe kelime hatalarını kontrol etmek için çağırılan sözlüğün her zaman tek ve aynı “instance” sahip olması için “Singleton” kullanıldı.

```
private static StringIterator a = getDictionary("words.txt");
      private Dictionary(){}
      public static StringIterator getInstance(){
       return a;
  }
      private static StringIterator getDictionary(String filename){
       Scanner sc2 = null;
       StringCollection text = new Implemention();
       StringIterator baseIterator = text.iterator(StringTypeEnum.STRING);
```

```
//for passing by int values
        try {
        sc2 = new Scanner(new File(filename));
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        }
        while (sc2.hasNextLine()) {
            Scanner s2 = new Scanner(sc2.nextLine());
            while (s2.hasNext()) {
                String s = s2.next();
                text.addWord(new Word(s));
            }
        }
       return baseIterator;
       }
}
```