Software-Projekt 1 | SoSe 2014 | Tim Ellhoff
Tutorin: Michaela Bunke | Teilgruppe: ChronoX | Karsten Betjemann

Übungsblatt 5

Lösungsvorschlag
27.07.2014

Aufgabe 1)

```
1  public abstract class Command{
2
3    protected char changedChar; // Wird in den Klassen Edit UND Del benötigt
4
5    public abstract void execute();
6    public abstract void undo();
7    public abstract void redo();
8
9  }
10
11 public class Edit extends Command{
12
13   public Edit(char c){
14     changedChar = c;
15   }
16
17   public void execute(){
18     TextBuffer.add(changedChar);
19   }
20
21   public void undo(){
22     TextBuffer.del();
23   }
24
25   public void redo(){
26     execute();
27   }
28
29 }
30
31 public class Del extends Command{
32
33   public void execute(){
34     changedChar = TextBuffer.del();
35   }
36
37   public void undo(){
38     TextBuffer.add(changedChar);
39   }
40
41   public void redo(){
42     execute();
43   }
44
45 }
```

Aufgabe 2) und Aufgabe 3)

Hinweis. Die für Aufgabe 3 relevanten Stellen sind in den Kommentaren im Java-Quellcode vermerkt.

```
1  public class InvalidArgumentException extends Throwable{};
```

```
2
3  public abstract class Medium{
4
5    private String titel;
6    private String verlag;
7    // --- Aufgabe 3 (section-start)
8    protected double rating;
9    protected int ratings;
10   // --- Aufgabe 3 (section-end)
11
12   public void setTitel(String titel){
13     if ( titel!=null & !titel.equals("") ){
14       this.titel = titel;
15     } else {
16       throw new InvalidArgumentException();
17     }
18   }
19
20   public void setVerlag(String verlag){
21     if ( verlag!=null & !verlag.equals("") ){
22       this.verlag = verlag;
23     } else {
24       throw new InvalidArgumentException();
25     }
26   }
27
28   public String getTitel(){
29     return titel;
30   }
31
32   public String getVerlag(){
33     return verlag;
34   }
35
36   // ------------ Aufgabe 3 (Aufgabenbereich-Beginn)
37   public abstract void setRating( int rating );
38   public abstract double getRating();
39   // ---------- Aufgabe 3 (Aufgabenbereich-Ende)
40
41 }
42
43
44 public class Buch extends Medium{
45
46   private List<String> authors;
47   private Buchserie bookSet;
48
49   public Buch( String titel, String verlag, String author ){
50     setTitel(titel);
51     setVerlag(verlag);
52     if ( author!=null & !author.equals("") ){
53       this.authors.add(author);
54     } else {
55       throw new InvalidArgumentException();
56     }
57   }
58
59   public List<String> getAuthors(){
60     return authors;
61   }
62
63   public addAuthor( String author ){
64     if ( !authors.contains(author) ){
65       authors.add(author);
66     }
67   }
68
69   public delAuthor( String author ){
70     if ( authors.contains(author) ){
71       authors.remove(author);
72     }
```

```
73    }
74
75    public getBookSet(){
76      return bookSet;
77    }
78
79    public setBookSet( Buchserie bs ){
80      if ( !bs.getBooks.contains(this) ){
81        bs.getBooks.add(this);
82      }
83      bookSet = bs;
84    }
85
86    // ----------- Aufgabe 3 (Aufgabenbereich-Beginn)
87    public void setRating( int rating ){
88      if ( rating>=1 & rating <=5 ){
89        this.rating = (this.rating*ratings+rating)/(++ratings);
90      } else {
91        throw new InvalidArgumentException();
92      }
93    }
94
95    public double getRating(){
96      return rating;
97    }
98    // ----------- Aufgabe 3 (Aufgabenbereich-Ende)
99
100 }
101
102 public class Buchserie extends Medium{
103
104   private List<Buch> books;
105
106   public Buchserie( String titel, String verlag, Buch firstBook ){
107     setTitel(titel);
108     setVerlag(verlag);
109     if ( firstBook!=null ){
110       this.books.add(firstBook);
111     } else {
112       throw new InvalidArgumentException();
113     }
114   }
115
116   public List<Buch> getBooks(){
117     return books;
118   }
119
120   // ----------- Aufgabe 3 (Aufgabenbereich-Beginn)
121   public void setRating( int rating ){
122     if ( rating>=1 & rating <=5 ){
123       this.rating = (this.rating*ratings+rating)/(++ratings);
124     } else {
125       throw new InvalidArgumentException();
126     }
127   }
128
129   public double getRating(){
130     double averageRating = 0;
131
132     for ( Buch b: books ){
133       averageRating += b.getRating();
134     }
135     averageRating = averageRating / books.size();
136
137     return averageRating;
138   }
139   // ----------- Aufgabe 3 (Aufgabenbereich-Ende)
140
141 }
```

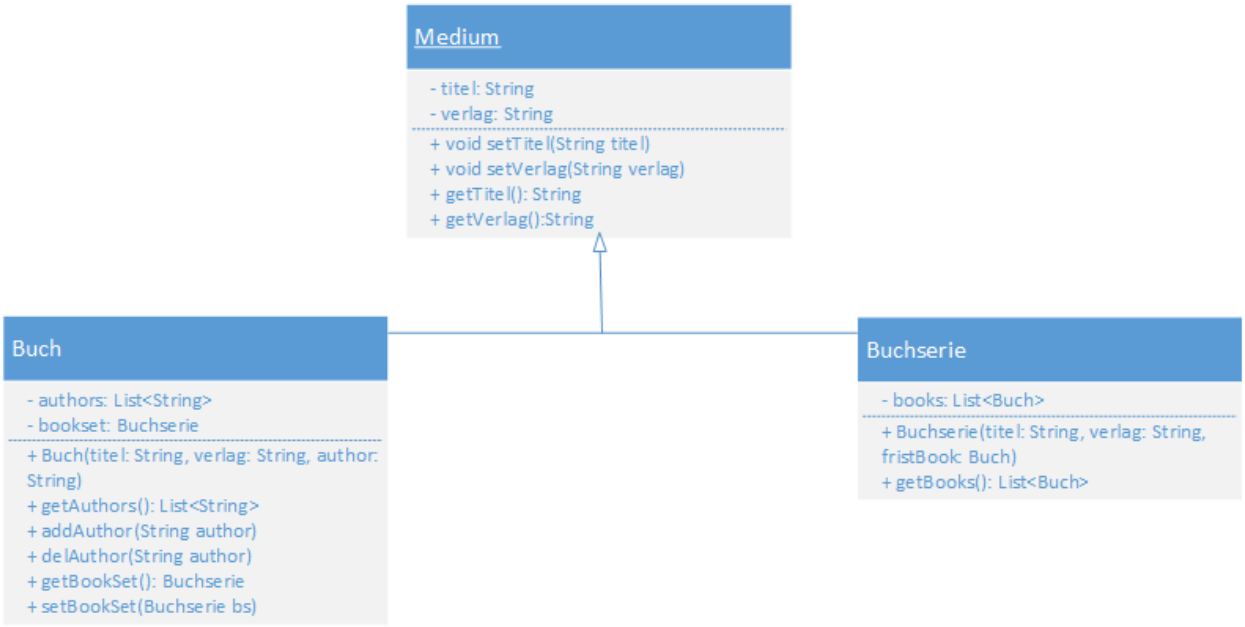

Abbildung 1: UML-Klassendiagramm zum Sachverhalt in Aufgabe 2

Aufgabe 4)

a)

typ(60.0, 30.0, 110.0,160.0)

b)

c)

d)

Bezogen auf die insgesamt zur Verfügung stehenden Programmzweige, beläuft sich die Zweigabdeckung des ersten Testfalles auf 75 %, da in diesem Falle die Konstruktion eines Sehnenvierecks vorliegt und deshalb die ersten beiden if-Zweige *(26-31)* durchlaufen werden *(false)*, welche ausschließlich dazu dienen, im Vorfeld jede Form auszusondern, die aufgrund ihrer Innenwinkel nicht als Viereck in Frage kommen.
Auch das nächste if-Statement *(32-37)*, beziehungsweise der darauf bezogene Pfad wird mit demselben Ergebnis durchlaufen, da hier eben kein Rechteck vorliegt, (nicht alle Winkel sind gleich groß, alpha nicht 90.0).
Daraufhin werden noch zwei else-if-Prüfungszweige *(38-41)* durchlaufen, welche wiederum ausschließen, dass es sich beim Testfall um ein Parallelogramm oder ein Trapez handelt.
Zuletzt schließt das Ganze bei der nächsten, erfolgreichen *(true)* Kontrolle im 6. Programmzweig *(42-43)*, welcher das Ergebnis - nämlich dass der Testfall ein Sehnenviereck darstellt - über *return 3* indiziert und damit den Programmablauf beendet, ohne dass die letzten beiden Programmzweige *(44-47)* durchlaufen wurden.

Der zweite Testfall endet wesentlich früher; hier kann sofort eine Exception *(Kein Viereck)* ausgegeben werden, sobald im ersten Programmzweig *(26-27)* ermittelt wird, dass die Winkel des Testfalles alle gemeinsam keine 360.0 ausmachen *(true)*.
Dadurch ergibt sich eine Zweigabdeckung von 12,5%.

Aufgabe 5)