Softwaretechnologie, Übung 4

Henry Haustein

Aufgabe 1

(a) Problem ist, dass man beim Casten schnell Fehler macht, die beim Kompilieren nicht erkannt werden, aber zur Laufzeit einen Fehler geben:

```
1 Bootle beerBottle = new Bottle();
2 beerBottle.fill (new WhiteWine("Burgunder"));
3
4 Beer beer = (Beer) beerBottle.empty();
5 // ClassCastException
```

(b) Zu jedem Drink gibt es eine Klasse, die die Flasche für diesen Drink bereitstellt.

(c) Datei `Bottle.java`

```
1  public class Bottle<T extends Drink> {
2    private T content;
3
4    public Bottle() {
5      this.content = null;
6    }
7
8    public boolean isEmpty() {
9      if (content == null) {
10       return true;
11     }
12     else {
13       return false;
14     }
15   }
16
17   public void fill(T con) {
18     if (this.isEmpty()) {
19       this.content = con;
20     }
21     else {
22       throw new IllegalStateException("Bottle not empty!");
23     }
24   }
25
26   public T empty() {
27     if (this.isEmpty()) {
```

```
28          throw new IllegalStateException("Bottle must be filled")
                ;
29        }
30        else {
31          T oldcontent = content;
32          content = null;
33          return oldcontent;
34        }
35      }
36    }
```

Datei `Drink.java`

```
1  public abstract class Drink {
2
3  }
```

Datei `Beer.java`

```
1  public class Beer extends Drink {
2    private String brewery;
3
4    public Beer(String brew) {
5      this.brewery = brew;
6    }
7
8    public String getBrewery() {
9      return brewery;
10   }
11
12   public String toString() {
13     return brewery;
14   }
15 }
```

Datei `Wine.java`

```
1  public abstract class Wine extends Drink {
2    private String region;
3
4    public Wine(String reg) {
5      this.region = reg;
6    }
7
8    public String getRegion() {
9      return region;
10   }
11
12   public String toString() {
13     return region;
14   }
15 }
```

Datei `WhiteWine.java`

```java
public class WhiteWine extends Wine {
  public WhiteWine(String reg) {
    super(reg);
  }
}
```

Datei `RedWine.java`

```java
public class RedWine extends Wine {
  public RedWine(String reg) {
    super(reg);
  }
}
```

Datei `Bar.java`

```java
public class Bar {
  public static void main(String[] args) {
    RedWine rw = new RedWine("Barolo");
    WhiteWine ww = new WhiteWine("Burgunder");
    Bottle<Beer> b = new Bottle<>();

    b.fill(new Beer("Uri"));

    Beer beer = b.empty();
  }
}
```

Aufgabe 2

Datei `Book.java`

```java
public class Book implements Comparable<Book>{
        private String isbn;
        private String author;
        private String title;

        public Book(String isbn, String author, String title) {
                if (isbn == null | author == null | title == null)
                     {
                        throw new IllegalArgumentException("
                            arguments for book can't be null");
                }
                this.isbn = isbn;
                this.author = author;
                this.title = title;
        }

        //ueberladen des Konstruktors
        public Book(String isbn) {
                if (isbn == null) {
```

```
18                          throw new IllegalArgumentException("
                                arguments for book can't be null");
19                  }
20                  this.isbn = isbn;
21          }
22
23          public String getIsbn() {
24                  return this.isbn;
25          }
26
27          public void setIsbn(String isbn) {
28                  this.isbn = isbn;
29          }
30
31          public String getAuthor() {
32                  return this.author;
33          }
34
35          public void setAuthor(String author) {
36                  this.author = author;
37          }
38
39          public String getTitle() {
40                  return this.title;
41          }
42
43          public void setTitle(String title) {
44                  this.title = title;
45          }
46
47          public String toString() {
48                  return this.getIsbn() + " " + this.getAuthor() + "
                        " + this.getTitle();
49          }
50
51          // notwendig fuer binaere Suche aus dem Collections-
                Framework
52          public int compareTo(Book book) {
53                  return this.isbn.compareTo(book.getIsbn());
54          }
```

Datei `Library.java`

```
1  public class Library {
2    List<Book> book_list;
3
4    public Library() {
5      book_list = new ArrayList<Book>();
6    }
7
8    // es sollte nach ISBN sortiert eingefuegt werden, da fuer die
         binaere Suche so einfacher
```

```
 9    public sortedInsert(Book newBook) {
10      book_list.add(newBook);
11      Collections.sort(book_list);
12    }
13
14    public Book searchForISBN_new(String isbn) {
15      int index = Collections.binarySearch(book_list, new Book(isbn,
            "",""));
16      // was passiert, wenn die Suche nix findet?
17      // -1 kommt dann zurueck! not fast enough.. :(
18      if (index >= 0) {
19        return book_list.get(index);
20      } else {
21        return null; //besser beim Aufruf dann ueberpruefen
22      }
23    }
24
25    public Collection<Book> searchForAuthor(String author){
26      ArrayList<Book> autorBook_list = new ArrayList<Book>();
27      for (Book book : book_list) {
28        if(book.getAuthor.equals(author)) {
29          autorBook_list.add(book);
30        }
31      }
32      return autorBook_list;
33    }
34  }
```