Objektorientierte Programmierung: Aufgabenblatt 7

Florian Ludewig (185722)

30. Juni 2020

Aufgabe 1

1.a)

Aufruf	Methode	Begründung
1	-	`1.0` und `2.0` werden als `double` interpretiert, aber keine der Methoden nimmt zwei `double` als Parameter
2	1	es werden nur ganze Zahlen übergeben und, wodurch Methode 1 mehr spezifischer ist alse Methode 2
3	1	es werden nur ganze Zahlen übergeben und, wodurch Methode 1 mehr spezifischer ist alse Methode 2
4	3	`C` ist Unterklasse von `B` und implementiert somit zwingend alle Methoden und Attribute von B
5	4	`A` ist allgemeiner als `B`, weile es eine Oberklasse von `B` ist
6	4	`B` als zweiter Parameter von Aufruf 6 kann nicht in ein `C` aus Methode 3 umgewandelt werden
7	-	Die Klasse `A` kann weder in ein `B` noch in ein `C` umgewandelt werden

1.b

Aufruf	Ausgabe	Begründung
1	1	`p1` wurde als `Print1` erstellt und somit wird das `x` von `Print1` ausgegeben
2	4	`print(B b)` aus `Print2` wird aufgerufen
3	5	`p1` ist vom Typ `Print2`, wodurch `print(C c)` aus `Print2` aufgerufen wird
4	4	`p2` wird nicht in `Print1` umgewandelt, wodurch `print(B b)` aus `Print2` aufgerufen wird
5	4	`p2` wird nicht in `Print1` umgewandelt, wodurch `print(B b)` aus `Print2` aufgerufen wird
6	5	`print(C c)` in `Print2` wird aufgerufen, weil `C` nicht in `B` umgewandelt werden kann
7	Exception	`Print1` kann nicht in `Print2` umgewandelt werden, weil `p1` mit `new new Print1()` überschrieben wurde

Aufgabe 2

`AbstrakteStringMenge.java`

```java
import java.util.LinkedList;
import java.util.Iterator;

public abstract class AbstrakteStringMenge implements StringMenge {
  protected LinkedList<String> list = new LinkedList<String>();

  public int getCharCount() {
    int count = 0;
    Iterator<String> iterator = list.iterator();
    while(iterator.hasNext()) {
      count += iterator.next().length();
    }
    return count;
  }
```

```
15
16    public boolean isEmpty() {
17      return list.size() <= 0;
18    }
19
20    public void print() {
21      Iterator<String> iterator = list.iterator();
22      while(iterator.hasNext()) {
23        System.out.printf(iterator.next());
24      }
25    }
26  }
```

StringMenge.java

```
1   public interface StringMenge {
2     void add(String s);
3     void remove(String s);
4     boolean contains(String s);
5     boolean isEmpty();
6     int getSize();
7     String[] getElements();
8     int getCharCount();
9     void print();
10  }
```

StringMengeImpl.java

```
1   import java.util.Iterator;
2
3   public class StringMengeImpl extends AbstrakteStringMenge {
4     public void add(String s) {
5       if (!contains(s)) list.add(s);
6     }
7
8     public void remove(String s) {
9       list.remove(list.indexOf(s));
10    }
11
12    public boolean contains(String s) {
13      return list.indexOf(s) >= 0;
14    }
15
16    public int getSize() {
17      return list.size();
18    }
19
20    public String[] getElements() {
21      String[] array = new String[getSize()];
22      int i = 0;
23      Iterator<String> iterator = list.iterator();
24      while(iterator.hasNext()) {
25        array[i] = iterator.next();
26        i++;
27      }
28      return array;
29    }
30  }
```