

Autoren: Marius Birk
Pieter Vogt
Tutor: Florian Brandt

Abgabe: 26.05.2020, 12:00 Uhr

Smileys:

A1	A2	A3	Σ
3.5	3	5	11.5

Objektorientierte Modellierung und Programmierung

Abgabe Übungsblatt Nr.05

(Alle allgemeinen Definitionen aus der Vorlesung haben in diesem Dokument bestand, es sei den sie erhalten eine explizit andere Definition.)

Aufgabe 1

```
1  import java.lang.reflect.Method;
2
3  class Util {
4
5      // liefert die kleinste Zahl des uebergebenen Arrays
6      public static int minimum(int[] values) throws
          ArrayNull {
7          int min = 0;
8          if(values == null){
9              throw new ArrayNull();
10         }else {
11             min = values[0];
12             for (int i = 1; i < values.length; i++) {
13                 if (values[i] < min) {
14                     min = values[i];
15                 }
16             }
17         }
18         return min;
19     }
20     // konvertiert den uebergebenen String in einen int-
21     // Wert
22     public static int toInt(String str) throws emptyString
23     {
24         int result = 0, factor = 1;
25         if(str.equals("")){
26             throw new emptyString();
27         }else {
28             char ch = str.charAt(0);
29             switch (ch) {
30                 case '-':
31                     factor = -1;
32                     break;
33                 case '+':
34                     factor = 1;
```

```
33         break;
34     default:
35         result = ch - '0';
36     }
37     for (int i = 1; i < str.length(); i++) {
38         ch = str.charAt(i);
39         int ziffer = ch - '0';
40         result = result * 10 + ziffer;
41     }
42 }
43 return factor * result;
44 }
45
46 // liefert die Potenz von zahl mit exp,
47 // also zahl "hoch" exp (number to the power of exp)
48 public static long power(long number, int exp) {
49     if (exp == 0) {
50         return 1L;
51     }
52     return number * Util.power(number, exp - 1);
53 }
54 }
55
56 public class UtilTest {
57     // Testprogramm
58     public static void main(String[] args) throws
59         ClassNotFoundException, NoSuchMethodException {
60         String eingabe="";
61         try{
62             Class IO = Class.forName("IO");
63             ClassLoader classLoader = IO.getClassLoader();
64             Class IO2 = Class.forName("IO", true, classLoader
65                 );
66
67             for(Method readString: IO.getDeclaredMethods()){
68                 if(readString.getName().equals("readString")){
69                     //eingabe = IO.readString("Zahl: ");
70                 }
71             }
72         }catch(ClassNotThere e){
73             e.printStackTrace();
74             System.out.print(e);
75         }
76         int zahl = Util.toInt(eingabe);
77         System.out.println(zahl + " hoch " + zahl + " = " +
78             Util.power(zahl, zahl));
79         System.out.println(Util.minimum(new int[] { 1, 6, 4,
80             7, -3, 2 }));
81         System.out.println(Util.minimum(new int[0]));
```

```
78         System.out.println(Util.minimum(null));
79     }
80 }
81
82 class ArrayNull extends RuntimeException {
83     public ArrayNull() {
84         super("Das Array ist leer");
85     }
86
87     public ArrayNull(String fehlermeldung) {
88         super(fehlermeldung);
89     }
90 }
91 class EmptyString extends RuntimeException {
92     public EmptyString() {
93         super("Der String ist leer");
94     }
95
96     public EmptyString(String fehlermeldung) {
97         super(fehlermeldung);
98     }
99 }
100 class ClassNotThere extends ClassNotFoundException {
101     public ClassNotThere() {
102         super("Klasse existiert nicht.");
103     }
104
105     public ClassNotThere(String fehlermeldung) {
106         super(fehlermeldung);
107     }
108 }
```

Aufgabe 2


1) 3.5/6

Bill.java

```
1 import java.util.ArrayList;
2
3 public class Bill {
4
5     //fields
6
7     String name;
8     double billPrice = 0;
9     ArrayList<BillItem> items = new ArrayList<>();
10
11     //methods
12 }
```

```
13     public void add(CarPart part) {
14         items.add(new BillItem(part));
15     }
16
17     //getter - setter
18
19     public double getTotalPrice() {
20         return billPrice;
21     }
22
23     public String toString() {
24         StringBuffer tempString = new StringBuffer("Receipt for
25             Bill: ");
26         double receiptTotal = 0;
27         tempString.append(this.name);
28         tempString.append("\n");
29         for (int i = 0; i < items.size(); i++) {
30             tempString.append(items.get(i).item.getName()); //add
31                 ItemName
32             tempString.append("\t");
33             tempString.append(items.get(i).item.getPrice()); //
34                 add ItemPrice
35             tempString.append("\n");
36             receiptTotal = receiptTotal + items.get(i).item.
37                 getPrice();
38         }
39         tempString.append("\n");
40         Math.nextUp(receiptTotal); //doesn't work for some reason
41
42         tempString.append("In Total this receipt is: " +
43             receiptTotal);
44         String output = tempString.toString();
45         return output;
46     }
47
48     //constructors
49
50     public Bill(String name) {
51         this.name = name;
52     }
53
54     //nested classes
55
56     private class BillItem {
57
58         //fields
59
60         CarPart item;
```

```
56         //methods
57
58         //getter - setter
59
60         public CarPart getItem() {
61             return item;
62         }
63
64         public void setItem(CarPart item) {
65             this.item = item;
66         }
67
68         public BillItem(CarPart item) {
69             this.item = item;
70         }
71     }
72
73 }
```




Car.java

```
1 import java.util.ArrayList;
2
3 public class Car {
4     ArrayList<CarPart> parts = new ArrayList<>();
5 }
```

CarComponent.java

```
1 public interface CarComponent {
2     public String getName();
3 }
```



CarPart.java

```
1 public class CarPart implements CarComponent {
2     String name;
3     double price;
4
5     @Override
6     public String getName() {
7         return null;
8     }
9
10    public double getPrice() {
11        return price;
12    }
```

```
12     }
13
14     public static class Seat extends CarPart {
15         String name = new String("Seat");
16         double price = 2000.0;
17
18         @Override
19         public String getName() {
20             return name;
21         }
22
23         public double getPrice() {
24             return price;
25         }
26     }
27
28     public static class Wheel extends CarPart {
29         String name = new String("Wheel");
30
31         double price = 1000.0;
32
33         @Override
34         public String getName() {
35             return name;
36         }
37
38         public double getPrice() {
39             return price;
40         }
41     }
42
43     public static class Motor extends CarPart {
44         String name = new String("Motor");
45
46         double price = 100000;
47
48         @Override
49         public String getName() {
50             return name;
51         }
52
53         public double getPrice() {
54             return price;
55         }
56     }
57 }
```

Main.java

```

1 public class Main {
2     public static void main(String[] args) {
3         Bill bill = new Bill("Rolls Royce");
4         bill.add(new CarPart.Motor());
5         bill.add(new CarPart.Seat());
6         bill.add(new CarPart.Wheel());
7         bill.add(new CarPart.Wheel());
8         bill.add(new CarPart.Wheel());
9         bill.add(new CarPart.Wheel());
10        System.out.println(bill.toString());
11    }
12 }

```

2) 3/6

Aufgabe 3

0.1 Class Person

```

1 import java.util.ArrayList;
2
3 class Person<T> implements Older<T>{
4     private String name;
5     private int age;
6     private Object other;
7
8
9     public Person( String name, int i) {
10        this.name = name;
11        this.age = i;
12    }
13
14    public String getName() {
15        return name;
16    }
17
18    public int getAge(){
19        return age;
20    }
21
22    @Override public <T extends Person> boolean isOlder(T
23        other) {
24        if(this.getAge() > other.getAge()){
25            return true;
26        }else{
27            return false;
28        }
29    }

```

```

29     }
30
31     interface Older<T > {
32         public <T extends Person> boolean isOlder(T other);
33     }
34
35     class Group<T extends Person> {
36         public Group() {
37
38         }
39         ArrayList<T> group = new ArrayList<>();
40
41         public void add(Person member) {
42             group.add(((T) member);
43         }
44         public T getOldest(){
45             int index=0;
46             for(int i = 0; i<=group.size()-1; i++){
47                 try{
48                     if(group.get(i+1)!=null){
49                         if(group.get(i).getAge()>group.get(i+1)
50                            .getAge() ){
51                             index=i;
52                         }else{
53                             index+=1;
54                         }
55                     }catch(Exception e){
56                         if(group.get(i).getAge() > group.get(index)
57                            .getAge()){
58                             index=index+1;
59                         }
60                     }
61                 }
62             }
63
64             return group.get(index);
65         }
66
67         public String get(int i) {
68             return group.get(i).getName();
69         }
70     }

```

0.2 Class TestGroup

```

1     public class TestGroup {
2         public static void main(String[] args) {
3             Group<Person> group = new Group<>();
4             group.add(new Person("Alice", 25));

```



```
5         group.add(new Person("Bob", 23));
6         group.add(new Person("Carl", 26));
7         System.out.println(group.getOldest().getName());
8     }
9 }
```

3) 5/8