



Praktische Zulassungsklausur (C) zur Vorlesung

# Einführung in die Informatik II - Vertiefung

im Sommersemester 2019

Dr. J. Kohlmeyer, T. Heß

08.06.2019

# Musterlösung

### Aufgabe 1 - Allgemeines

### 3 + 3 + 3 + 3 = 12 Punkte

a) Implementieren Sie eine Methode public static void firstSteps(), welche zuerst alle geraden Zahlen zwischen 1 und 10 und dann alle ungeraden Zahlen zwischen 1 und 10 in die gleiche Zeile ausgibt. Im Anschluss soll umgebrochen werden. Formatieren Sie ihre Ausgabe wie die folgt:

### Ausgabe:

2 4 6 8 10 1 3 5 7 9

### Lösung

```
public static void firstSteps() {
       for (int i = 1; i <= 10; i++) {
2
            if (i % 2 == 0) {
3
                System.out.print(i + " ");
 4
            }
5
        }
6
7
       for (int i = 1; i <= 10; i++) {
            if (i % 2 != 0) {
8
9
                System.out.print(i + " ");
            }
10
11
        System.out.println();
12
   }
13
```

**b)** Implementieren Sie eine Methode **public static** String toLowerCase(String str), welche einen String übergeben bekommt und alle Großbuchstaben durch Kleinbuchstaben ersetzt.

**Beispiel:** toLowerCase("Ananas")  $\rightarrow$  "ananas"

#### Lösung

```
public static String toLowerCase(String str) {
   return str.toLowerCase();
}
```

c) Eine Zahl ist durch 9 teilbar, wenn ihre Quersumme durch 9 teilbar ist. Implementieren Sie eine Methode public static boolean isDivBy9(int n), welche genau dann true zurückliefert, wenn n durch 9 teilbar ist.

### **Beispiele**

- isDivBy9(9)  $\rightarrow$  true
- isDivBy9(423)  $\rightarrow$  true
- isDivBy9(26)  $\rightarrow$  false

### Lösung

```
public static boolean isDivBy9(int n) {
   return n % 9 == 0;
}
```

d) Implementieren Sie eine Methode public static void printDuplicates(String[] strings)), welche alle Strings ausgibt, die im Array strings mehr als einmal vorkommen. In der Ausgabe soll jeder String maximal einmal vorkommen, die Reihenfolge ist egal.

```
Ausgabe für printDuplicates(new String[]{"A", "A", "C", "B", "A", "B"})
A
B
```

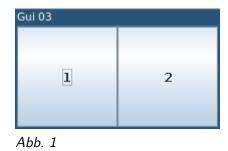
### Lösung

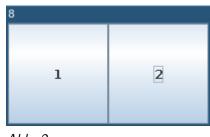
```
public static void printDuplicates(String[] strings) {
2
        ArrayList<String> duplicates = new ArrayList<>();
        for (int i = 0; i < strings.length; i++) {</pre>
3
            for (int j = i + 1; j < strings.length; j++) {
4
5
                if (strings[i].equals(strings[j])) {
                    if (!duplicates.contains(strings[i])) {
6
                         duplicates.add(strings[i]);
7
8
                         break;
9
                    }
10
                }
            }
11
        }
12
        for (String duplicate : duplicates) {
13
14
            System.out.println(duplicate);
        }
15
16
  | }
```

# Aufgabe 2 - Graphische Oberfläche

12 Punkte

- a) Implementieren Sie eine graphische Oberfläche, welche aussieht, wie die in Abb. 1 abgebildete.
- b) Implementieren Sie eine Funktionalität welche für jeden Button zählt wie oft dieser angeklickt wurde. Die Summe der beiden Zähler soll im Fenstertitel stehen. In Abb. 2 ist die Gui dargestellt, nachdem Button 1 dreimal und Button 2 fünfmal geklickt wurde.





### Lösung

```
//Importe (javax.swing.*, import java.awt.*,
               java.awt.event.ActionEvent, java.awt.event.ActionListener)
2
   public class Gui03 {
3
       private static int a;
 4
5
       private static int b;
       public static void main(String[] args) {
6
 7
            JFrame frame = new JFrame("Gui 03");
            frame.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
8
9
            frame.setPreferredSize(new Dimension(200, 100));
10
            JButton button1 = new JButton("1");
            button1.setActionCommand("A");
11
            JButton button2 = new JButton("2");
12
            button2.setActionCommand("A");
13
14
            ActionListener ctrl = new ActionListener() {
                @Override
15
                public void actionPerformed(ActionEvent actionEvent) {
16
                    if (actionEvent.getActionCommand().equals("A")) {
17
18
                        a++;
19
                    } else {
20
                        b++;
21
                    frame.setTitle((a + b) + "");
22
                }
23
            };
24
25
            button1.addActionListener(ctrl);
            button2.addActionListener(ctrl);
26
            frame.getContentPane().setLayout(new GridLayout(1, 2));
27
            frame.getContentPane().add(button1);
28
29
            frame.getContentPane().add(button2);
30
            frame.pack();
31
            frame.setLocationRelativeTo(null);
32
            frame.setResizable(false);
33
            frame.setVisible(true);
34
       }
   }
35
```

### Aufgabe 3 - Listen

12 Punkte

In dieser Aufgabe sollen Sie mit der vordefinierten Datenstruktur LinkedList<sup>1</sup> aus dem Package java.util arbeiten.

Implementieren Sie die folgenden Aufgaben in einer Klasse LinkedListUtil. Beachten Sie den folgenden Code für die Beispiele:

```
LinkedList<Integer> list = new LinkedList<>();
list.add(1);
list.add(2);
list.add(3);
```

a) Implementieren Sie eine Methode

```
public static <T> void print(LinkedList<T> list)
```

welche die in der Liste gespeicherten Werte in eine Zeile ausgibt (in  $\mathcal{O}(n)$ ). Im Anschluss soll die Zeile umgebrochen werden.

**Beispiel:** print(list);  $\rightarrow$  1 2 3

### Lösung -

```
public static <T> void print(LinkedList<T> list) {
   for (T val : list) {
      System.out.print(val + " ");
   }
   System.out.println();
}
```

b) Implementieren Sie eine Methode

```
public static <T> void printReversed(LinkedList<T> list)
```

welche die in der Liste gespeicherten Werte in umgekehrter Reihenfolge in eine Zeile ausgibt (in  $\mathcal{O}(n)$ ). Im Anschluss soll die Zeile umgebrochen werden.

**Beispiel:** printReversed(list); → 3 2 1

#### Lösung -

```
public static <T> void printReversed(LinkedList<T> list) {
   Iterator<T> iterator = list.descendingIterator();
   while (iterator.hasNext()) {
       System.out.print(iterator.next() + " ");
   }
   System.out.println();
}
```

<sup>1</sup>https://docs.oracle.com/javase/7/docs/api/java/util/LinkedList.html

### c) Implementieren Sie eine Methode

```
public static <T> void removeDuplicates(LinkedList<T> list) welche alle in der Liste list enthaltenen Duplikate entfernt (in \mathcal{O}(n^2)).
```

### Beispiel:

```
1 list.addFirst(1);
2 list.add(3);
3 list.add(1);
4 print(list); //-> 1 1 2 3 3 1
5 removeDuplicates(list);
6 print(list); //-> 1 2 3
```

### - Lösung -

Lösung (in  $\mathcal{O}(n^2)$ )

```
public static <T> void removeDuplicates(LinkedList<T> list) {
2
       LinkedList<T> list2 = new LinkedList<>();
       for (T val : list) {
3
           if (!list2.contains(val)) {
4
                list2.add(val);
5
           }
6
       }
7
       list.clear();
8
9
       list.addAll(list2);
10 }
```

## Aufgabe 4 - Bäume

12 Punkte

Im Ordner ~/material finden Sie eine Datei materialTree03.zip, welche eine rudimentäre Implementierung eines Binärbaumes beinhaltet.

Beachten Sie den folgenden Code für die Beispiele:

- a) Importieren Sie das Material.
- **b)** Implementieren Sie eine Methode **public int** maxRec(), welche rekursiv nach dem größten im Baum gespeicherten Wert sucht. Falls der Baum keine Werte enthält, soll eine IllegalStateException geworfen werden.

Hinweis: Sie dürfen eine Hilfsmethode benutzen.

**Beispiel:** tree.maxRec()  $\rightarrow$  9

c) Implementieren Sie eine Methode public int maxIt(), welche iterativ nach dem größten im Baum gespeicherten Wert sucht. Falls der Baum keine Werte enthält, soll eine IllegalStateException geworfen werden.

**Beispiel:** tree.maxIt()  $\rightarrow$  9

d) Implementieren Sie eine Methode public int countOcc(int val), welche zählt wie oft der Wert val im Baum vorkommt und diese Häufigkeit zurückgibt.

**Beispiel:** tree.count0cc(3)  $\rightarrow$  2

#### Lösung -

```
//IntTree
   public int maxRec() { //b)
3
       if (root == null) {
            throw new IllegalStateException();
4
5
6
       return maxRec(root);
   }
7
   private int maxRec(IntNode node) {
9
       if (node.right != null) {
            return maxRec(node.right);
10
       }
11
12
       return node.value;
   }
13
   public int maxIt() { //c)
14
       if (root == null) {
15
            throw new IllegalStateException();
16
       }
17
       IntNode temp = root;
18
       while (temp.right != null) {
19
20
            temp = temp.right;
21
       return temp.value;
22
   }
23
   public int countOcc(int val) {
                                        //d)
24
       return countOcc(val, root);
25
   }
26
27
   private int countOcc(int val, IntNode node) {
       if (node == null) {
28
29
            return 0;
       }
30
31
       int count = 0;
32
       if (node.value == val) {
            count++;
33
       }
34
       return count + countOcc(val, node.left)
35
            + countOcc(val, node.right);
36
37 }
```