ulm university universität
uulm

Praktische Zulassungsklausur (B) 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 alle geraden Zahlen zwischen 1 und 10 in eine Zeile ausgibt und dann in einer weiteren Zeile alle ungeraden Zahlen zwischen 1 und 10. Formatieren Sie ihre Ausgabe wie die folgende Ausgabe.

Ausgabe:

```
2 4 6 8 10
1 3 5 7 9
```

Lösung

```java
public static void firstSteps() {
    System.out.println("2 4 6 8 10");
    System.out.println("1 3 5 7 9"); //print auch i.O.
}
```

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

Beispiel: `toUpperCase("Ananas")` → `"ANANAS"`

Lösung

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

c) Eine Zahl ist durch 4 teilbar, wenn ihre beiden letzten Stellen durch 4 teilbar sind. Implementieren Sie eine Methode `public static boolean isDivBy4(int n)`, welche genau dann `true` zurückliefert, wenn `n` durch 4 teilbar ist.

Beispiele

- `isDivBy4(3384)` → `true`
- `isDivBy4(12)` → `true`
- `isDivBy4(22)` → `false`

Lösung

```java
public static boolean isDivBy4(int n) {
    return n % 4 == 0;
}
```

d) Implementieren Sie eine Methode `public static void printMostCommon(String[] strings))`, welches den am häufigsten vorkommenden String im Array `strings` sucht und diesen gemeinsam mit seiner Häufigkeit in die Konsole ausgibt. Formatieren Sie ihre Ausgabe wie das Beispiel.

Ausgabe für `printMostCommon(new String[]{"Apfel", "A", "Brot", "A"})`

```
A 2
```

Lösung

```java
public static void printMostCommon(String[] strings) {

    String best = strings[0];
    int bestCount = 1;
    for (int i = 0; i < strings.length; i++) {

        int curCount = 1;
        for (int j = i + 1; j < strings.length; j++) {
            if (strings[i].equals(strings[j])) {
                curCount++;
            }
        }

        if (curCount > bestCount) {
            best = strings[i];
            bestCount = curCount;
        }
    }

    System.out.printf("%s %d%n", best, bestCount);
}
```

Aufgabe 2 - Graphische Oberfläche **12 Punkte**

a) Implementieren Sie eine graphische Oberfläche, welche aussieht, wie die in Abb. 1 abgebildete.

b) Wenn jeder Button mindestens einmal gedrückt wurde, soll das Programm beendet werden.

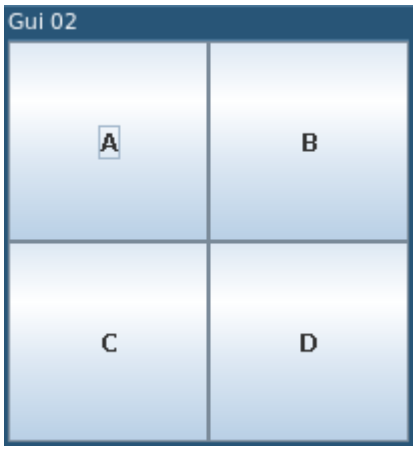

Abb. 1

Lösung

```java
//Importe (javax.swing.*, import java.awt.*,
//         java.awt.event.ActionEvent, java.awt.event.ActionListener)

public class Gui02 {

    private final static String[] BUTTONS = {"A", "B", "C", "D"};

    public static void main(String[] args) {

        JFrame frame = new JFrame("Gui 02");
        frame.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);

        frame.setPreferredSize(new Dimension(200, 200));

        frame.getContentPane().setLayout(new GridLayout(2, 2));

        boolean[] arr = new boolean[4];

        ActionListener ctrl = new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent actionEvent) {

                char c = actionEvent.getActionCommand().charAt(0);

                arr[c - 'A'] = true;

                for (int i = 0; i < arr.length; i++) {
                    if (!arr[i]){
                        return;
                    }
                }

                frame.dispose();
            }
        };

        JButton[] buttons = new JButton[4];
        for (int i = 0; i < buttons.length; i++) {
            buttons[i] = new JButton(BUTTONS[i]);
            buttons[i].setActionCommand(BUTTONS[i]);
            buttons[i].addActionListener(ctrl);
            frame.getContentPane().add(buttons[i]);
        }

        frame.pack();
        frame.setVisible(true);
    }
}
```

Aufgabe 3 - Dateiverarbeitung **12 Punkte**

Eine csv-Datei enthält durch Semikola getrennte Werte. In unserem Fall in jeder Zeile gleich viele Werte. Im Ordner `~/material` finden Sie eine csv-Datei `data01.csv`, kopieren Sie diese in ihr Projekt (auf die gleiche Ebene, wie auch ihr `src`-Ordner).

a) Implementieren Sie die Funktionalität um die Werte aus der csv-Datei einzulesen.

b) Parsen Sie die Werte als `double`. Finden Sie den größten Wert und geben Sie dessen Zeile, Spalte sowie den Wert selbst in die Konsole aus. Formatieren Sie ihre Ausgabe genau wie die folgende Ausgabe.
Ausgabe:

```
18 8 9.98
```

Lösung

```java
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.List;

public class ExerciseB03 {

    public static void main(String[] args) {

        Path path = Paths.get("data01.csv");
        List<String> raw = null;

        try {
            raw = Files.readAllLines(path);
        } catch (IOException e) {
            e.printStackTrace();
        }

        if (raw == null)
            return;
        }

        double max = Double.NaN;
        int row = 0;
        int col = 0;
        for (int i = 0; i < raw.size(); i++) {
            String[] splitData = raw.get(i).split(";");
            for (int j = 0; j < splitData.length; j++) {
                double d = Double.parseDouble(splitData[j]);
                if (Double.isNaN(max) || d > max) {
                    max = d;
                    row = i;
                    col = j;
                }
            }
        }

        System.out.printf("%d %d %.2f%n", row, col, max);
    }
}
```

Aufgabe 4 - Listen **12 Punkte**

Im Ordner `~/material` finden Sie eine Datei `materialListe02.zip`, welche eine rudimentäre Implementierung einer einfach-verketteten Liste beinhaltet.

a) Importieren Sie das Material.

b) Fügen Sie der Klasse `StringList` ein boolesches Attribut `isSet` hinzu. Implementieren Sie einen Konstruktor um dieses Attribut instanziieren zu können.

c) Passen Sie die Methode `append(...)` derartig an, sodass diese im Falle von `isSet == true` nur Werte an die Liste anhängt, welche noch nicht in der Liste vorkommen.
Beispiel:

```
1 StringList list = new StringList(true);
2 list.append("A").append("B").append("A").append("C").append("A");
3 System.out.println(list); //-> ["A", "B", "C"]
```

d) Implementieren Sie eine Methode `public void toSet()`, welche alle Duplikate aus der Liste entfernt und `isSet` auf `true` setzt.

Lösung

```
1  public class StringList {
2      private boolean isSet;
3      StringElement first;
4
4      public StringList(boolean isSet) {
5          this.isSet = isSet;
6      }
7
7      public StringList append(String val) {
8          StringElement temp = first;
9          if (temp == null) {
10             first = new StringElement(val);
11         } else {
12             while (temp.next != null) {
13                 if (isSet && temp.val.equals(val)) {
14                     return this;
15                 }
16                 temp = temp.next;
17             }
18             temp.next = new StringElement(val);
19         }
20         return this;
21     }
22     public void toSet() {
23         if (isSet) {
24             return;
25         }
26         isSet = true;
27         StringElement temp1 = first;
28         while (temp1 != null) {
29             StringElement temp2 = temp1;
30             while (temp2.next != null) {
31                 if (temp1.val.equals(temp2.next.val)) {
32                     temp2.next = temp2.next.next;
33                 } else {
34                     temp2 = temp2.next;
35                 }
36             }
37             temp1 = temp1.next;
38         }
39     }
40 }
```