ulm university universität
uulm

Praktische Zulassungsklausur (D) 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 ungeraden Zahlen zwischen 1 und 10 und dann alle geraden Zahlen zwischen 1 und 10 in die gleiche Zeile ausgibt. Im Anschluss soll umgebrochen werden. Formatieren Sie ihre Ausgabe wie folgt:
Ausgabe:
`1 3 5 7 9 2 4 6 8 10`

— **Lösung**

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

b) Implementieren Sie eine Methode

`public static String replaceAll(String str, String old, String replace)`

welche alle Vorkommen von `old` in `str` durch `replace` ersetzt.

Beispiel: `replaceAll("AI2 ist einfach. #AI2", "AI2", "EidI2")`
→ *"EidI2 ist einfach. #EidI2"*

— **Lösung**

```
1  public static String replaceAll(String str, String old, String repl){
2      return str.replaceAll(old, repl);
3  }
```

c) Eine Zahl ist durch 8 teilbar, wenn ihre letzten drei Stellen durch 8 teilbar sind. Implementieren Sie eine Methode `public static boolean isDivBy8(int n)`, welche genau dann `true` zurückliefert, wenn `n` durch 8 teilbar ist.
Beispiele

— **Lösung**

```
1  public static boolean isDivBy8(int n) {
2      return n % 8 == 0;
3  }
```

d) Implementieren Sie eine Methode `public static void mulPairs(int n)`, welche alle Zahlen $a \leq b < n$ in die Konsole ausgibt, für welche $a \cdot b = n$ gilt. Formatieren Sie ihre Ausgabe wie im Beispiel.
Beispiel: `mulPairs(24);`

```
2 12
3 8
4 6
```

Lösung

```
1 public static void mulPairs(int n) {
2     for (int i = 0; i < n; i++) {
3         for (int j = i; j < n; j++) {
4             if (i * j == n) {
5                 System.out.printf("%d %d%n", i, j);
6             }
7         }
8     }
9 }
```

Aufgabe 2 - Graphische Oberfläche **12 Punkte**

a) Implementieren Sie eine graphische Oberfläche, welche aussieht, wie die in Abb. 1 abgebildete (die weiße Fläche ist ein `JTextField`).

b) Implementieren Sie die Funktionalität, das bei einer Eingabe von „Passwort: 42“ und dem anschließenden Drücken der Enter-Taste das Programm beendet wird. Wird ansonsten die Enter-Taste gedrückt, soll „Falsche Eingabe!“in die Konsole ausgegeben werden.

Hinweis: Beim Drücken der Enter-Taste wird ein `ActionEvent` ausgelöst.

Abb. 1

Lösung

```
1  import javax.swing.*;
2  import java.awt.event.ActionEvent;
3  import java.awt.event.ActionListener;

4  public class Gui04 {

5      public static void main(String[] args) {

6          JFrame frame = new JFrame("Gui 04");
7          frame.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);

8          JTextField textField = new JTextField(16);

9          textField.addActionListener(new ActionListener() {
10             @Override
11             public void actionPerformed(ActionEvent actionEvent) {
12                 if (textField.getText().equals("Passwort: 42")) {
13                     frame.dispose();
14                 }else{
15                     System.out.println("Falsche Eingabe!");
16                 }
17             }
18         });

19         frame.getContentPane().add(textField);

20         frame.pack();
21         frame.setLocationRelativeTo(null);
22         frame.setResizable(false);

23         frame.setVisible(true);
24     }
25 }
```

Aufgabe 3 - Stacks **12 Punkte**

In dieser Aufgabe sollen Sie mit der vordefinierten Datenstruktur `Stack`[1] aus dem Package `java.util` arbeiten.

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

```
1 Stack<Integer> stack = new Stack<>();
2 stack.push(1);
3 stack.push(2);
4 stack.push(3);
5 System.out.println(stack); //-> [1, 2, 3]
```

a) Implementieren Sie eine Methode `public static void squareIt(Stack<Integer> stack)`, welche **iterativ** alle Werte auf dem Stack quadriert.
Beispiel:

```
1 squareIt(stack);
2 System.out.println(stack); //-> [1, 4, 9]
```

b) Implementieren Sie eine Methode `public static void squareRec(Stack<Integer> stack)`, welche **rekursiv** alle Werte auf dem Stack quadriert.
Beispiel:

```
1 squareRec(stack);
2 System.out.println(stack); //-> [1, 4, 9]
```

c) Implementieren Sie eine Methode

`public static void removeDuplicates(Stack<Integer> stack)`

welche alle Duplikate aus dem Stack entfernt.
Beispiel:

```
1 stack.push(1);
2 stack.push(3);
3 System.out.println(stack); //-> [1, 2, 3, 1, 3]
4 removeDuplicates(stack);
5 System.out.println(stack); //-> [1, 2, 3]
```

[1] https://docs.oracle.com/javase/7/docs/api/java/util/Stack.html

Lösung

```java
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;
import java.util.Stack;

public class StackUtil {
    public static void squareIt(Stack<Integer> stack) {

        Stack<Integer> stack2 = new Stack<>();
        while (!stack.isEmpty()) {
            int val = stack.pop();
            val *= val;
            stack2.push(val);
        }

        while (!stack2.isEmpty()) {
            stack.push(stack2.pop());
        }
    }

    public static void squareRek(Stack<Integer> stack) {
        if (stack.isEmpty()) {
            return;
        }

        int val = stack.pop();

        squareRek(stack);

        stack.push(val * val);
    }

    public static void removeDuplicates(Stack<Integer> stack) {

        List<Integer> list = new ArrayList<>();

        for (Integer i : stack) {
            if (!list.contains(i)) {
                list.add(i);
            }
        }

        stack.clear();
        stack.addAll(list);
    }
}
```

Aufgabe 4 - Bäume **12 Punkte**

Im Ordner `~/material` finden Sie eine Datei `materialTree04.zip`, welche eine rudimentäre Implementierung eines Binärbaumes beinhaltet.

Beachten Sie den folgenden Code für die Beispiele:

```
1 IntTree tree = new IntTree();
2 tree.add(5).add(3).add(3).add(1).add(7).add(9);
3 System.out.println(tree); //-> 1 3 3 5 7 9
```

a) Importieren Sie das Material.

b) Implementieren Sie eine Methode **`public int`** `size()`, welche die im Baum abgespeicherten Werte zählt.

Hinweis: Sie dürfen eine Hilfsmethode benutzen.

Beispiel: `tree.size()` → 6

c) Implementieren Sie eine Methode **`public int`** `countLeaves()`, welche die Anzahl der Blätter (Knoten ohne Nachfolger) zählt und zurückgibt.

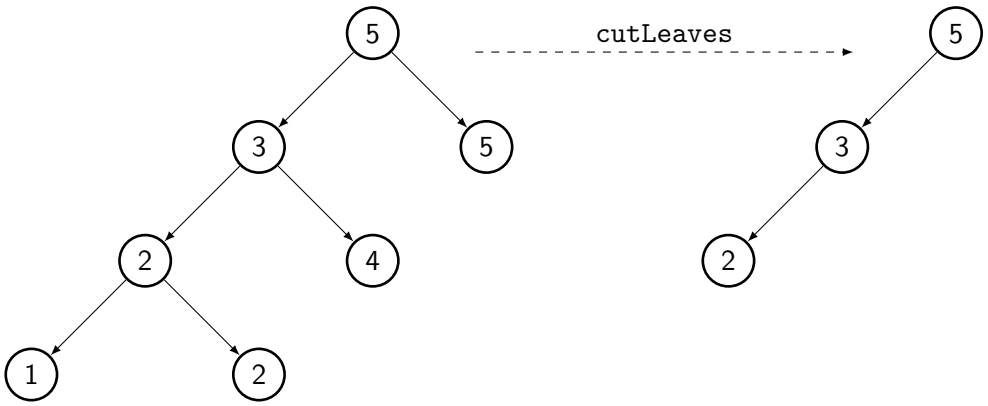
Beispiel: `tree.countLeaves()` → 3

d) Implementieren Sie eine Methode **`public int`** `sum()`, welche die Summe aller im Baum gespeicherten Werte berechnet und zurückgibt.

Beispiel: `tree.sum()` → 28

e) Implementieren Sie eine Methode **`public void`** `cutLeaves()`, welche alle Blätter des Baumes entfernt, die zu Zeitpunkt des Aufrufs im Baum existieren.

Beispiel:



Lösung

```java
public int size() {
    return size(root);
}

private int size(IntNode node) {
    if (node == null) {
        return 0;
    }
    return size(node.left) + 1 + size(node.right);
}

public int countLeaves() {
    return countLeaves(root);
}

private int countLeaves(IntNode node) {
    if (node == null) {
        return 0;
    }

    if (node.left == null && node.right == null) {
        return 1;
    }

    return countLeaves(node.left) + countLeaves(node.right);
}

public int sum() {
    return sum(root);
}

private int sum(IntNode node) {
    if (node == null) {
        return 0;
    }

    return node.value + sum(node.left) + sum(node.right);
}
```

cutLeaves() auf der nächsten Seite

Lösung

```
1  public void cutLeaves() {
2      root = cutLeaves(root);
3  }

4  private IntNode cutLeaves(IntNode node) {
5      if (node == null) {
6          return null;
7      }

8      if (node.left == null && node.right == null) {
9          return null;
10     }

11     node.left = cutLeaves(node.left);
12     node.right = cutLeaves(node.right);

13     return node;
14 }
```