Aufgaben zur Veranstaltung
Programmierung mit Java, WS 2023/2024

H. Pflug, J. Dietel FH Aachen, Campus Jülich; IT Center, RWTH Aachen

Klausurvorbereitung Blatt 2 17./18.01.2024

1.) Gegeben sei ein Programm, das aus den folgenden vier Klassen besteht. Welche Ausgabe erzeugt der Aufruf der Klasse `Main`?

```
public class Vogel {
    public Vogel() {
        super();
        System.out.println("Ich␣bin␣ein␣Vogel!");
    }

    public void zwitschert() {
        System.out.println("Vogel␣zwitschert!");
    }
}

public class Singvogel extends Vogel {
    public Singvogel() {
        super();
    }

    public void zwitschert() {
        System.out.println("Singvogel␣zwitschert!");
        super.zwitschert();
        singen();
        System.out.println("Singvogel␣singt␣nicht␣mehr!");
    }

    public void singen() {
        System.out.println("Singvogel␣singt␣ein␣Lied.");
    }
}

public class Star extends Singvogel {
    public Star() {
        super();
    }

    public void zwitschert() {
        System.out.println("Star␣zwitschert!");
        super.zwitschert();
        singen();
        System.out.println("Star␣zwitschert␣nicht␣mehr!");
    }

    public void singen() {
        System.out.println("Star␣singt␣ein␣Lied");
    }
}

public class Main {
    public static void main(String[] args) {
        Vogel d = new Star();
        d.zwitschert();
    }
}
```

2.) Gegeben sei folgender Code:

```
public class Bruch {
  public int zaehler;
  public int nenner;

  public Bruch(int zaehler, int nenner) {
    this.zaehler=zaehler;
    this.nenner = nenner;
  }

  public static void main(String[] args) {
    Bruch b = new Bruch(2,3);
    System.out.println(b);
  }
}
```

a) Welche Bildschirmausgabe erwarten Sie (ungefähre Beschreibung)?

b) Fügen Sie eine weitere Methode hinzu, die eine sinnvolle Bildschirmausgabe bewirkt. Wie heißt diese Methode?

c) Es sei ferner die Funktion

```
public static void multipliziere(Bruch b, int x)
```

gegeben, die den Bruch b mit x multipliziert. Warum ist kein Rückgabewert nötig?

d) Schreiben Sie die Methode

```
public void multipliziere(int x)
```

als nicht-statische Methode der Klasse Bruch.

3.) Die untenstehende `main`-Funktion läuft fehlerfrei. Schreiben Sie an alle mit `AUSGABE` bezeichneten Stellen die Bildschirmausgabe des jeweiligen Methodenaufrufs.

```
public class A {
  public void magic(double d) {
    System.out.println("A:magicD");
  }

  public void plus(int d) {
    System.out.println("A:plusI");
  }

  public void plus(double d) {
    System.out.println("A:plusD");
  }
};

public class B extends A {
  public void magic(double d) {
    System.out.println("B:magicD");
    super.plus(d);
  }

  public void magic(int d) {
    super.magic(d);
    System.out.println("B:magicI");
  }

  public void plus(double d) {
    System.out.println("B:plusD");
  }
}
```

```
public class Test {
  public static void main(String[] args) {
    A a = new B();
    B b = new B();

    b.plus(6.2);  //AUSGABE


    b.magic(3);   //AUSGABE


    a.magic(0.5); //AUSGABE


    a.magic(2);   //AUSGABE


    a.plus(3);    //AUSGABE


  }
}
```

4.) Schreiben Sie eine Methode

```
public static boolean pruefeMatrix(int[][] matrix)
```

Die Methode prüft, ob in der übergebenen $m \times n$-Matrix alle Zeilen und alle Spalten ein- und dieselbe Summe haben. Falls dies der Fall ist, wird *true* zurückgegeben, ansonsten *false*.

5.) Schreiben Sie eine Funktion

```
public static String verschoenere(String s)
```

s enthält einen Text, in dem Wörter durch ein oder mehrere Leerzeichen getrennt sind. Die Funktion modifiziert den Text, so dass die Wörter nur durch genau ein Leerzeichen getrennt sind. Das Ergebnis wird zurückgegeben.

6.) Schreiben Sie eine Funktion

```
public static boolean stimmenUeberein(String s1, String s2)
```

Die Funktion gibt `true` zurück wenn die beiden Strings `s1` und `s2` (siehe auch Tabelle 1)

- entweder gleich sind
- oder genau ein Paar benachbarter Zeichen vertauscht sind.

Tabelle 1: Beispiele (geänderte Buchstaben sind unterstrichen):

s1	s2	Ergebnis
funktion	funktion	true (stimmen überein)
funkti<u>o</u>n	funkti<u>a</u>n	false (ein Zeichen ersetzt)
funk<u>ti</u>on	funki<u>t</u>on	true (benachbarte Zeichen vertauscht)
funk<u>ti</u>on	funko<u>i</u>t<u>n</u>	false (vertauschte Zeichen nicht benachbart)
f<u>un</u>kti<u>o</u>n	f<u>nu</u>kt<u>o</u>in	false (zwei Vertauschungen)

7.) Schreiben Sie eine Methode

```
public static double[] liesDaten(String filename)
```

die eine Datei folgenden Formats ausliest:

- Steht am Anfang der Zeile ein #, so handelt es sich um eine Kommentarzeile, die nicht ausgewertet werden darf. In der Datei können sich an jeder Stelle Kommentarzeilen befinden.
- Die erste gültige Zeile hat das Format: „Anzahl der Messdaten: n“. Dabei ist n die Anzahl der folgenden Zeilen mit jeweils einem Messwert.
- Diese Messwerte sollen eingelesen und in einem double-Feld zurückgegeben werden. Beispiel:

Beispiel: Eine Datei hat folgendes Aussehen:

```
#Versuch vom 11.3.05
#Probe Nr. 4b
Anzahl der Messdaten: 5
34.689
37.680
40.013
45.221
48.518
#Versuchsdauer 23 min
```

8.) Eine Primzahl p ist eine natürliche Zahl mit genau zwei unterschiedlichen natürlichen Zahlen als Teiler, nämlich der Zahl 1 und sich selbst.

Um Primzahlen zu ermitteln, gibt es einen Algorithmus mit dem Namen „Das Sieb des Eratosthenes“. Man kann damit alle Primzahlen kleiner n ermitteln, indem man aus einem fortlaufend von 1 bis n durchnummerierten Feld alle Werte entfernt, die Vielfache der ganzen Zahlen von 2 bis $\sqrt{n}$ sind. Beispiel für n = 20: Schranke = Ganzahlanteil ($\sqrt{20}$) = 4

```
Feld:            1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20
Vielfache von 2: 1,2,3,x,5,x,7,x,9, x,11, x,13, x,15, x,17, x,19, x
Vielfache von 3: 1,2,3,x,5,x,7,x,x, x,11, x,13, x, x, x,17, x,19, x
Vielfache von 4: 1,2,3,x,5,x,7,x,x, x,11, x,13, x, x, x,17, x,19, x
```

Die Primzahlen < 20 sind somit 2,3,5,7,11,13,17,19

Schreiben Sie eine Funktion, die nach obigem Algorithmus bei Eingabe einer Zahl n alle Primzahlen ermittelt, die kleiner als n sind. Die berechneten Primzahlen sollen in einer `ArrayList` zurückgegeben werden. Die Signatur der Funktion sieht aus wie folgt:

```
public static ArrayList<Integer> primZahlTest(int n){...}
```

9.) a) Schreiben Sie eine Klasse `Dominostein`, die einen Dominostein repräsentiert. Beide Seiten des Dominosteins können die Zahlen 0-6 enthalten. Die Klasse soll enthalten:

- Einen passenden Konstruktor. Werfen Sie dort gegebenenfalls eine `NumberFormatException`.
- Eine toString-Methode, die den Dominostein in der Form `[a,b]` ausgibt. Der äußerste linke Stein in der Abbildung würde also als `[4,2]` ausgegeben.
- Eine Methode `public void dreheUm()`

  die den Dominostein herumdreht (also die beiden Seiten vertauscht).
- Eine Methode `public int[] getValues()`

  die die Werte des Dominosteins in einem Integer-Feld der Größe 2 zurückgibt.

b) Schreiben Sie ferner eine Klasse `Dominokette`, die eine Kette von Dominosteinen repräsentiert (siehe Abbildung 1).

Es ist möglich, an beide Enden der Kette weitere Steine anzufügen, wenn die Symbole passen (siehe Abbildung). Implementieren Sie folgende Konstruktoren und Methoden:

- Einen Konstruktor, der den ersten Dominostein als Übergabeparameter erhält.
- Eine `toString`-Methode. Das Ausgabeformat des abgebildeten Beispiels soll dabei sein:

  `[4,2][2,5][5,0][0,6][6,3][3,3]`
- Eine Methode

  `public void fuegeRechtsAn(Dominostein d)`

die den Dominostein am rechten Ende anfügt. Gegebenenfalls muss der Stein vorher herumgedreht werden. Die Methode löst eine `NumberFormatException` aus, falls der Stein nicht angelegt werden kann. Die Methode `fuegeLinksAn` muss aus Zeitgründen nicht implementiert werden.

Abbildung 1: Kette von Dominosteinen

10.) Ihre Aufgabe ist es, eine einfache Terminplanung zu realisieren.

a) Implementieren Sie eine Klasse `Termin` mit den privaten Integer-Attributen `tagImJahr`, `stunde`, `minute` und dem privaten String-Attribut `beschreibung`. Schreiben Sie dann einen Konstruktor, der diese Attribute übergeben bekommt und setzt. Gehen Sie davon aus, dass die Attribute immer im zulässigen Bereich gesetzt werden. Versehen Sie die Klasse `Termin` zudem mit Gettern für jedes Attribut und erweitern Sie die `toString`-Methode aus `java.lang.Object`, so dass der Termin in folgender Formatierung dargestellt wird:

> Sie haben am [tagImJahr]. Tag des Jahres um [uhrzeit] Uhr den folgenden Termin: [beschreibung]

b) Die Ausgabe von `toString` soll sprachabhängig werden. Alle Termin-Objekte sollen eine einheitliche Sprache benutzen. Die Sprache kann für alle Termin-Objekte gemeinsam gesetzt oder während des Programmablaufs geändert werden. Ergänzen Sie dazu die Klasse `Termin` entsprechend und schreiben Sie die Methode `toString` neu. Verwenden Sie das Interface

```
public Interface SprachFormat {
  String format(Termin t);
}
```

c) Schreiben Sie eine Klasse

```
public class FormatDeutsch implements SprachFormat
```

die den in Teil 1.) beschriebenen deutschen Text erzeugt.

d) Schreiben Sie eine Klasse `Test`, deren `main`-Methode einen Termin mit den Werten

Tag	91
Stunde	8
Minute	30
Beschreibung	Java-Klausur

erzeugt, die Sprache auf Deutsch setzt und den Termin auf dem Bildschirm ausgibt.

11.) Schreiben Sie eine Funktion

```
public static int getLevenshteinDistanz(char[] m, char[] n)
```

Die gesuchte Levenshtein-Distanz ist gleich einer Distanz-Funktion D an der Stelle $D(m.length - 1, n.length - 1)$. Die Funktion D ist rekursiv definiert als:

$$
\begin{aligned}
D(0,0) &= 0 \\
D(i,0) &= i \\
D(0,j) &= j \\
D(i,j) &= \min \left\{ \begin{array}{lll} D(i-1,j-1) & \text{falls} & m[i] = n[j] \\ D(i-1,j-1)+1 & \text{falls} & m[i] \neq n[j] \\ D(i-1,j)+1 & & \\ D(i,j-1)+1 & & \end{array} \right| (i \neq 0, j \neq 0)
\end{aligned}
$$

Tipp: Benutzen Sie als Hilfsmethode

```
public static int distanz(char[] m, char[] n, int i, int j)
```

12.) Gegeben seien die drei Funktionen

```
1 public static int fakultaet(int z)
2 public static int dreheUm(String z)
3 public static boolean summe(int[] test)
```

Unten finden Sie den Code der drei Funktionen.

- `fakultaet` berechnet die Fakultät (*factorial*) einer positiven Zahl.
- `dreheUm` dreht einen String herum.
- `summe` berechnet die Summe der Elemente eines int-Felds.

Schreiben Sie drei Funktionen `fakultaet`, `dreheUm` und `summe` neu mit gleicher Funktionalität und gleichen Übergabeparametern, aber einem rekursiven Algorithmus. Insbesondere sind alle Arten von Schleifen (`for, while, ...`) nicht gestattet.

Hinweis: Hilfsfunktionen sind erlaubt und bei `summe` auch sehr sinnvoll.

```
1  public static int fakultaet(int z) {
2      int ret=1;
3      for (int i=1; i<=z; i++) {
4          ret = ret*i;
5      }
6      return ret;
7  }
8
9  public static String dreheUm(String s) {
10     String ret ="";
11     for (int i=0; i<s.length(); i++) {
12         ret = s.charAt(i)+ret;
13     }
14     return ret;
15 }
16
17 public static int summe(int[] test) {
18     int ret = 0;
19     for (int t: test) {
20         ret +=t;
21     }
22     return ret;
23 }
```