Westfälische Hochschule

Fachbereich Informatik und Kommunikation
Prof. Dr. Marcel Luis

23. März 2022, 8.30 - 10.30 Uhr

Klausur zu Einführung in die Programmierung

Die Prüfungsleistung muss **eigenständig** erbracht werden.

Es sind nur die **Hilfsmittel** zulässig, die in dem Dokument „Vorbereitung und Durchführung der Klausur“ auf der Moodle-Seite dieser Klausur aufgeführt sind.

Falls Sie Lösungsbestandteile aus anderen als Ihren eigenen Quellen übernehmen (z. B. Internet, Bücher, etc.), müssen Sie die **verwendeten Quellen angeben**.

Vergessen Sie nicht, die von Ihnen unterschriebene **Erklärung zur Eigenständigkeit** zusammen mit Ihrer Prüfungsbearbeitung hochzuladen.

Hinweise zu den Programmieraufgaben

Die Verwendung von *Sprunganweisungen* ist *nicht* zulässig.

Der *Stil* der Realisierung wird mit einem Anteil von 15 % bei der Bewertung berücksichtigt.

Aufgabe 1 [Ausdrücke] (12 Punkte)

Für diese Aufgabe seien `b1` und `b2` Variablen des Typs `boolean`, `s` eine Variable des Typs `String`, `f` eine Variable des Typs `float` und `i` eine Variable des Typs `int`.

Untersuchen Sie die Ausdrücke auf der nächsten Seite und verbinden Sie die Nummern der Ausdrücke, die äquivalent ist. Wenn ein Ausdruck mit keinem anderen äquivalent ist, umkreisen Sie seine Nummer.

Zwei Ausdrücke sind *äquivalent*, wenn sie denselben Typ besitzen und sich hinsichtlich der Auswertung völlig gleichwertig verhalten, d. h. sie haben stets den gleichen Wert oder es kommt zum gleichen Laufzeitfehler.

Beispiele: Die Ausdrücke

```
i < 10
10 > i
!(i >= 10)
```

sind äquivalent, denn sie haben denselben Typ und für *jeden* möglichen Wert der Variablen `i` denselben Wert.

Die Ausdrücke

```
i < 10
10 >= i
```

sind *nicht* äquivalent, denn es gibt einen Wert der Variablen `i`, für den beide Ausdrücke *verschiedene* Werte haben.

Übertragen Sie auf Ihr Lösungsblatt nur die Nummern der Ausdrücke. Verbinden oder umkreisen Sie die Nummern wie angegeben. Schreiben und zeichnen Sie deutlich. Benutzen Sie ggfs. verschiedene Farben.

1 `i < 10`

2 `10 > i`

3 `10 >= i`

4 `!(i >= 10)`

5 `2 * f`

6 `2f * f`

7 `((float) 2) * f`

8 `f * 2 / 5`

9 `f * (2 / 5)`

10 `f * 2 / 5f`

11 `f * 2 / 5.0`

12 `b1 || (10 / i == 0)`

13 `b1 && !(10 / i == 0)`

14 `(10 / i == 0) || b1`

15 `b1 ? !b2 : false`

16 `b2 ? false : (b1 ? true : false)`

17 `b1 && !b2`

18 `(s.charAt(i) == 'x') && (i < s.length())`

19 `(i < s.length()) && (s.charAt(i) == 'x)`

20 `(i % 2 == 0) ? true : (i <= 0 ? true : i > 100)`

21 `i % 2 == 0 || i <= -1 || i > 100`

Aufgabe 2 [Ausdrücke realisieren] (11 Punkte)

In dieser Aufgabe realisieren Sie Java-Ausdrücke, die bestimmte Berechnungen durchführen. Der erste Ausdruck dient als Beispiel.

1. Es sei `n` eine Variable des Typs `int` mit positivem Wert.

   Realisieren Sie einen Ausdruck, der genau dann `true` liefert, wenn der Wert von `n` gerade ist.

   ```
   n % 2 == 0
   ```

2. Es seien `m` und `n` Variablen des Typs `long`.

   Schreiben Sie einen Ausdruck des Typs `float`, der mathematisch korrekt (soweit möglich) den Mittelwert der Werte von `m` und `n` berechnet.

3. Es sei `z` eine Variable des Typs `int`. Der Wert von `z` sei eine Zahl zwischen 0 und 9.

   Schreiben Sie einen Ausdruck des Typs `char`, der das Ziffernsymbol zum Wert von `z` berechnet. Enthält `z` den Wert 0, berechnet der Ausdruck das Zeichen `'0'`, für den Wert 1 das Zeichen `'1'` usw.

   Ihre Lösung darf *keinen* bedingten Ausdruck und *keine* Konstante des Typs `int` oder `long` enthalten.

4. Es sei `url` eine Variable des Typs `String`, die eine gültige Web-Adresse enthält (z. B. `https://www.w-hs.de`).

   Schreiben Sie einen Ausdruck des Typs `String`, der die letzten 3 Zeichen der Zeichenkette in `url` berechnet. Für das Beispiel ist der Wert des Ausdrucks `".de"`.

5. Es sei `s` eine Variable des Typs `String`.

   Schreiben Sie einen Ausdruck, der genau dann `true` liefert, wenn der Wert von `s` ein "Doppelwort" ist. Bei einem Doppelwort ist die erste Worthälfte gleich der zweiten Worthälfte, wie z. B. bei `papa` oder `MiuMiu`. `MaMaM` und `Mama` sind dagegen keine Doppelwörter.

6. Es seien `u` und `v` Variablen des Typs `int`. `u` hat einen negativen Wert, `v` einen positiven.

   Schreiben Sie einen Ausdruck des Typs `boolean`, der genau dann `true` liefert, wenn `u - v` (mathematisch korrekt) außerhalb des Wertebereichs des Typs `int` liegt.

Aufgabe 3 [Einfache Methoden] (6 Punkte)

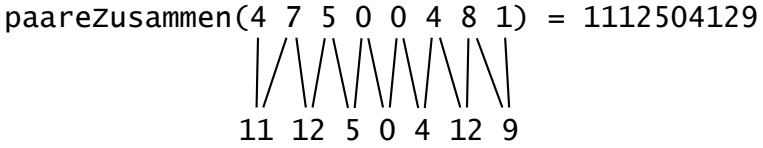
Schreiben Sie eine statische Methode `int inMeilen(int km)`, die eine Anzahl km gerundet in volle Meilen umrechnet. Eine Meile ist gleich 1609,344 m. Verwenden Sie keine Methoden aus Standardklassen des JDK.

Beispiele:

- Der Aufruf `inMeilen(17)` ergibt 11 (denn 17 km sind ca. 10,56 Meilen).
- Der Aufruf `inMeilen(200)` ergibt 124 (denn 200 km sind ca. 124,27 Meilen).

Aufgabe 4 [Rekursion] (10 Punkte)

Realisieren Sie eine statische Methode `long paareZusammen(int zahl)`, die von der übergebenen Zahl jede Ziffer mit ihrer folgenden Ziffer addiert und diese Summen zu einer Zahl zusammenfasst. Sie dürfen davon ausgehen, dass die Methode nur mit positiven, mindestens zweistelligen Zahlen aufgerufen wird.



Verwenden Sie keine Schleifen und keine Klassen oder Methoden des JDK.

Beispiele:

Zahl	Ergebnis
47500481	1112504129
9637	15910
47	11
20	2

Aufgabe 5 [Rekursion] (5 Punkte)

Gegeben sei folgende Methode `m`.

```
public static long m(int a, int b, long c) {

    return b < 0
            ? c
            : m(a, b - 1, c * a);
}
```

Überlegen Sie, was die Methode berechnet, wenn man sie mit einem beliebigen `int`-Wert a als erstem, einem $b \geq 0$ als zweitem und der Zahl 1 als drittem Parameter aufruft. Geben Sie die Antwort, indem Sie den folgenden Satz mit einer der angegebenen Optionen ergänzen.

Der Aufruf $m(a, b, 1)$ berechnet ...

- die Potenz a^b
- die Potenz a^{b+1}
- den Vorkommaanteil der Wurzel von b
- den Wert $a \cdot b!$ ($b!$ ist die Fakultät von b)
- die Potenz a^{b-1}

Aufgabe 6 [Klasse und Objekte] (14 Punkte)

Realisieren Sie eine Klasse `Rechteck` zur Repräsentation von Rechtecken durch Angabe ihrer Breite und Höhe. Implementieren Sie darin folgende Methoden:

- Konstruktor `Rechteck(int, int)`, mit dem ein Objekt der Klasse `Rechteck` für die übergebene Breite (1. Parameter) und Höhe (2. Parameter) erzeugt wird. Sie dürfen davon ausgehen, dass der Konstruktor nicht mit negativen Werten aufgerufen wird.
- Instanzmethode `double gibSeitenverhaeltnis()`, die das Verhältnis von Breite zu Höhe berechnet. Für ein Rechteck der Breite 6 und Höhe 4 liefert die Methode z. B. 1.5.
- Instanzmethode `boolean istGroesser(Rechteck)`, die angibt, ob das Rechteck, das die Methode ausführt, eine *echt größere* Fläche besitzt als das übergebene Rechteck.
- Instanzmethode `Rechteck drehe90()`, die ein Rechteck liefert, das gegenüber dem Rechteck, das die Methode ausführt, um 90 Grad gedreht ist. Das Rechteck, das die Methode ausführt, bleibt unverändert.
- Methode `public static void main(String[] args)`, in der sich genau die folgenden beiden Anweisungen befinden:

  ```
  double verhaeltnis = ...
  System.out.println(verhaeltnis);
  ```

  Setzen Sie hierbei an die Stelle der drei Punkte einen zusammenhängenden, geschachtelten Ausdruck, durch den ein Rechteck mit Breite 2 und Höhe 1 erzeugt wird, für das anschließend das gedrehte Rechteck erzeugt wird (Methode `drehe90`), von dem anschließend das Seitenverhältnis berechnet wird. Der berechnete Wert ist 0.5.

Aufgabe 7 [Felder und Schleifen] (9 Punkte)

Schreiben Sie eine Methode `boolean bestehtAusPaaren(int[])`, die genau dann `true` liefert, wenn die Werte des Felds (in Richtung aufsteigender Indizes) aus einer Aneinanderreihung gerader oder ungerader Zahlenpaare besteht. Eine Voraussetzung für den Wert `true` ist somit, dass die Länge des Felds geradzahlig ist.

Beispiele:

- Für ein Feld mit den Werten 1, 3, 6, 2, 5, 7, 11, 131 liefert die Methode `true`, denn alle Paare (1, 3), (6, 2), (5, 7), (11, 131) sind gerade bzw. ungerade.
- Für ein Feld mit den Werten 1, 3, 6, 2, 5, 7, 11 liefert die Methode `false`, denn die Anzahl der Zahlen ist ungerade.
- Für ein Feld mit den Werten 1, 3, 6, 2, 8, 0, 6, 7 liefert die Methode ebenfalls `false`, denn das letzte Paar (6, 7) ist weder ein gerades, noch ein ungerades Zahlenpaar.

Ergänzen Sie für die Realisierung das folgende Gerüst der Methode. Vermeiden Sie unnötige Schleifendurchläufe. Notieren Sie auf Ihrem Lösungsblatt nur die Einfügungen an den Stellen 1 bis 5.

```
public static boolean bestehtAusPaaren(int[] a) {

    boolean ausPaaren = ________________ /*1*/;

    int i = ________ /*2*/;

    while (i < ____________________________________ /*3*/) {

        ausPaaren = ____________________________________ /*4*/;

        _________________ /*5*/;

    }

    return ausPaaren;
}
```

Aufgabe 8 [Felder] (8 Punkte)

Realisieren Sie eine Methode `int[] paarsummen(int... a)`, die zu den übergebenen (variabel vielen) Werten ein Feld liefert, dessen Werte die Summen von jeweils zwei aufeinander folgenden Parametern sind.

Beispiel: Für den Aufruf `paarsummen(1, 3, -5, 7, 9, 11)` liefert die Methode ein Feld der Länge 5 mit den Werten 4, -2, 2, 16 und 20.

Sie dürfen davon ausgehen, dass die Methode mit mindestens zwei Werten aufgerufen wird.

```
public static int[] paarsummen(int... a) {
```

Aufgabe 9 [Code ausführen] (8 Punkte)

Das folgende Programmstück enthält 3 `println`-Anweisungen, die mit (1) bis (3) gekennzeichnet sind.

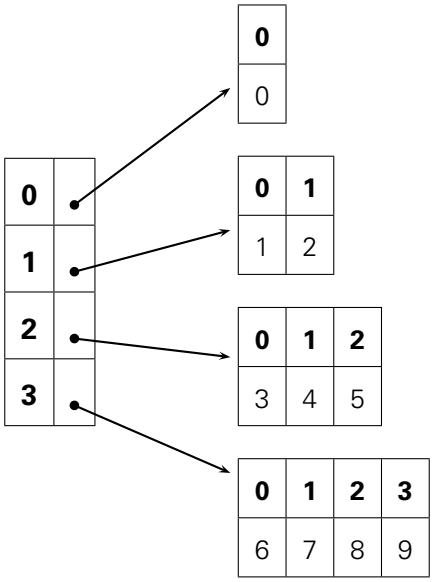
Geben Sie an, welche Ausgaben das Programmstück an diesen Stellen produziert. Schreiben Sie dazu auf das Lösungsblatt die Zahlen (1) bis (3) und notieren Sie dahinter die Ausgabe oder – im Falle von Schleifen – die Ausgabe*n*, die die `println`-Anweisung an der jeweiligen Stelle erzeugt.

```
int[] a = new int[]{10, 8, 6, 5};
int[] b = new int[3];

a[3] = 55;
b[0] = 1;
b[1] = 2;
b[2] = 3;

a = b;

for (int i = 0; i < a.length; i++) {
    System.out.println(a[i]);          // (1)
}

a[0] = 11;
a[1] = 22;

for (int i = 0; i < b.length; i++) {
    System.out.println(b[i]);          // (2)
}

System.out.println(a.length);          // (3)
```

Aufgabe 10 [Mehrdimensionale Felder] (10 Punkte)

Realisieren Sie eine statische Methode `int[][] dreieck(int n)`, die ein dreieckförmiges, zweidimensionales Feld der Größe $n \geq 0$ erzeugt.
Die Komponenten des Felds enthalten ab 0 aufsteigende Zahlen, so wie es das Beispiel zeigt. Die fett gedruckten Zahlen sind die Indizes der Feldkomponenten.

Beispiel: Für den Aufruf `dreieck(4)` erzeugt die Methode folgendes Feld.



Aufgabe 11 [Binäre Bäume] (15 Punkte)

Diese Aufgabe basiert auf den Klassen `Binaerbaum` und `Baumknoten`. Ein Objekt der Klasse `Binaerbaum` verwaltet in seinen Knoten Elemente des Typs `int`. Die Struktur beider Klassen erkennen Sie an folgenden Code-Ausschnitten. Beide Klassen haben *nur* die hier aufgeführten Instanzvariablen. Objekte der Klasse `Binaerbaum` sind *keine* binären Suchbäume.

```
public class Binaerbaum

    private Baumknoten wurzel;
    ...

    public int maxPfadlaenge() {
        ...
    }
}

public class Baumknoten {

    private int inhalt;
    private Baumknoten linkerTeilbaum;
    private Baumknoten rechterTeilbaum;

    public int maxPfadlaenge() {
        ...
    }
}
```

Ergänzen Sie die beiden vorbereiteten Methoden `int maxPfadlaenge()` – *und nur diese!* –, sodass die Anwendung der Methode `maxPfadlaenge` auf ein Objekt der Klasse `Binaerbaum` die Anzahl der Knoten zurückgibt, die auf dem längsten Pfad von der Wurzel zu den Blättern liegen.

Beispiele (die Blätter der Bäume sind grau dargestellt): für den linken Baum ist der Wert 1, für den rechten 4 (es ist egal, dass es zwei gleich lange Pfade gibt). Für den leeren Baum ist der Wert 0.

Denken Sie an die Methode `Math.max`, um auf einfache Art den größeren von zwei Werten zu ermitteln.