Westfälische Hochschule

Fachbereich Informatik und Kommunikation
Prof. Dr. Marcel Luis

22. September 2021, 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] (5 Punkte)

Es sei `k` eine Variable des Typs `int`. Geben Sie einen Ausdruck an, der zu folgendem Ausdruck äquivalent (d.h. gleichwertig) ist und keinen bedingten Ausdruck enthält. Ein Tipp: denken Sie an logische Operatoren.

```
k % 50 == 0
    ? true
    : k % 7 > 3
          ? true
          : k > 100
```

Aufgabe 2 [Ausdrücke realisieren] (11 Punkte)

In dieser Aufgabe realisieren Sie kleine 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 `a` und `b` Variablen des Typs `long` mit positiven Werten.

   Schreiben Sie einen Ausdruck des Typs `boolean`, der genau dann `true` liefert, wenn `a + b` (mathematisch korrekt) innerhalb des Wertebereichs des Typs `long` liegt.

3. Es sei `k` eine Variable des Typs `long`.

   Schreiben Sie einen Ausdruck des Typs `float`, der mathematisch korrekt (soweit möglich) ein Fünftel des Werts von `k` berechnet.

4. Es sei `f` eine Variable des Typs `long`.

   Schreiben Sie einen Ausdruck des Typs `int`, der die drittletzte Ziffer des Werts von `f` berechnet. Hat z. B. `f` den Wert 125475, ist der erwartete Wert des Ausdrucks 4. Hat `f` den Wert 12, ist der erwartete Wert 0.

5. Es seien `a` und `b` Variablen des Typs `char`. Das Zeichen in `b` liegt im Zeichenalphabet hinter dem Zeichen in `a`.

   Schreiben Sie einen Ausdruck des Typs `char`, der das Zeichen in der Mitte zwischen diesen beiden Zeichen berechnet. Enthält z. B. `a` das Zeichen `'c'` und `b` das Zeichen `'g'`, dann soll der Ausdruck `'e'` berechnen. Gibt es kein exakt mittleres Zeichen (wie z. B. zwischen `'c'` und `'f'`), dann kann der Ausdruck das Zeichen vor oder hinter der Mitte berechnen.

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

   Schreiben Sie einen Ausdruck des Typs `char`, der das drittletzte Zeichen der Zeichenkette in `s` berechnet. Ist die Zeichenkette nicht ausreichend lang, soll der Ausdruck `'@'` liefern.

Aufgabe 3 [Einfache Methoden] (5 Punkte)

Schreiben Sie eine statische Methode `int cent(double e)`, die für einen positiven Geldbetrag (in Euro) den Centanteil berechnet.

Beispiele:

- Der Aufruf `cent(1.8)` soll 80 ergeben.
- Der Aufruf `cent(124.99)` soll 99 ergeben.

Aufgabe 4 [Rekursion] (10 Punkte)

Realisieren Sie eine Methode statische `int ziffernMinus1(int zahl)`, die zu einer Zahl diejenige Zahl liefert, in der jede Ziffer außer der 0 um 1 verkleinert ist.

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

Beispiele:

Zahl	Ergebnis
0	0
1	0
5	4
961	850
196	85
1020345678	10234567

Aufgabe 5 [Rekursion] (5 Punkte)

Gegeben sei folgende Methode `m`.

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

    return b * b < a
              ? m(a, b + 1)
              : b * b;
    }
```

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

Der Aufruf $m(a, 0)$ berechnet ...

- das Quadrat von a
- die kleinste Quadratzahl, die größer oder gleich a ist
- die größte Quadratzahl, die kleiner oder gleich a ist
- den Vorkommaanteil der Wurzel von a

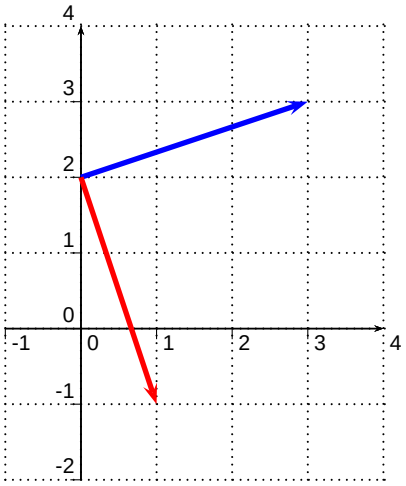
Aufgabe 6 [Klassen] (16 Punkte)

Realisieren Sie eine Klasse `Strecke` zur Repräsentation von Strecken in der zweidimensionalen Ebene. Implementieren Sie darin folgende Methoden (s. auch nächste Seite):

- Konstruktor `Strecke(Punkt start, Punkt ziel)`, mit dem eine Strecke von einem Start- zu einem Zielpunkt erzeugt wird.
- Instanzmethode `boolean fuehrtAufwaerts()`, die angibt, ob die Strecke aufwärts führt. Dies ist dann der Fall, wenn der Zielpunkt echt oberhalb des Startpunkts liegt. In der Abbildung führt die blau dargestellte Strecke aufwärts, die rote nicht.
- Instanzmethode `Strecke drehe90()`, die ein neues Objekt der Klasse `Strecke` liefert. Die neu erzeugte Strecke ist gegenüber der Strecke, die die Methode ausführt, um 90 Grad im Uhrzeigersinn um den Startpunkt gedreht.

  Beispiel: Wird die blau dargestellte Strecke gedreht, ergibt sich die rote Strecke. Der Startpunkt beider Strecken ist gleich. Wie man den Zielpunkt der gedrehten Strecke berechnet, können Sie an der Abbildung erkennen.

  1. Der Zielpunkt der blauen Strecke ist gegenüber dem Startpunkt um $dx = 3$ *horizontal* und $dy = 1$ *vertikal* verschoben.
  2. Dann ist der Zielpunkt der gedrehten Strecke gegenüber dem Startpunkt um $dy = 1$ *horizontal* und $-dx = -3$ *vertikal* verschoben.



- Methode `public static void main(String[] args)`, in der sich genau die folgenden drei Anweisungen befinden:

  ```
  Strecke s = ...;
  boolean fuehrtAufwaerts = ...;
  System.out.println(fuehrtAufwaerts);
  ```

  Setzen Sie hierbei an die Stelle der ersten drei Punkte einen Ausdruck, durch den eine Strecke vom Punkt (0, 2) zum Punkt (3, 3) erzeugt wird.

  Setzen Sie an die Stelle der zweiten drei Punkte einen Ausdruck, durch den die erzeugte Strecke gedreht wird. Auf das resultierende Objekt wird die Methode `fuehrtAufwaerts` angewendet. Das erwartete Ergebnis ist `false`.

Die Klasse `Punkt` ist wie folgt vorgegeben. Sie darf durch Methoden erweitert werden, was jedoch nicht notwendig ist.

```
public class Punkt {

    private final int x;
    private final int y;

    public Punkt(int x, int y) {
        this.x = x;
        this.y = y;
    }

    public int gibX() {
        return x;
    }

    public int gibY() {
        return y;
    }
}
```

Aufgabe 7 [Felder] (8 Punkte)

Realisieren Sie eine statische Methode `int[] spiegelsumme(int[])`. Die Methode addiert die erste und letzte Komponente des Felds, die zweite und vorletzte usw. und liefert ein Feld mit genau diesen Summen.

Beispiel: Enthält das übergebene Feld (in Richtung aufsteigender Indizes) die Werte **14, 5, 0, -6, 7, -4, 2, 16**, dann hat das zurückgegebene Feld die Länge 4 und enthält die Werte **30, 7, -4, 1**.

Ist das übergebene Feld ungeradzahlig lang, wird der mittlere Wert mit sich selbst addiert.

Beispiel: Für das übergebene Feld mit den Werten **14, 5, 3, 2, 16** wird ein Feld der Länge 3 mit den Werten **30, 7, 6** zurückgegeben.

```
public static int[] spiegelsumme(int[] zahlen)
```

Aufgabe 8 [Mehrdimensionale Felder] (10 Punkte)

Realisieren Sie eine Methode `int[] haeufigkeiten(int[][], int)`, die zählt, wie häufig der übergebene Wert in den einzelnen Feldern der übergebenen Matrix vorkommt und diese Zählwerte in einem neuen Feld zurückgibt.

Beispiel: Es sei `m` wie folgt deklariert und initialisiert:

```
int[][] m = new int[3][];
m[0] = new int[]{5, 7, 2, -1, 5, 0};
m[1] = new int[]{0, 1, 2};
m[2] = new int[]{5, 5, 5};
```

Bei Aufruf von `haeufigkeiten(m, 5)` hat das zurückgegebene Feld die Länge 3 mit den Werten 2, 0, 3, denn die Zahl 5 kommt in `m[0]` zweimal, in `m[1]` kein mal und in `m[2]` dreimal vor.

```
public static int haeufigkeiten(int[][] m, int wert)
```

Aufgabe 9 [Felder und Schleifen] (7 Punkte)

Schreiben Sie eine Methode `boolean istLaenger(String[])`, die genau dann `true` liefert, wenn es ein Paar aufeinander folgender Indizes gibt, sodass die zweite Komponente echt länger als die erste Komponente ist.

Beispiel: Für ein Feld mit den Werten `"Ohne"`, `"Aha"`, `"Onkel"`, `"Nord"` liefert die Methode `true`, denn `"Onkel"` ist länger als `"Aha"`.

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 istLaenger(String[] s) {

    boolean istLaenger = ________ /*1*/;

    int i = ________ /*2*/;

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

        istLaenger = ___________________________________ /*4*/;

        _________________ /*5*/;

    }

    return istLaenger;
}
```

Aufgabe 10 [Binäre Bäume] (15 Punkte)

Diese Aufgabe basiert auf den Klassen `BinaererSuchbaum` und `Baumknoten`. Ein Objekt der Klasse `BinaererSuchbaum` 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.

```
public class BinaererSuchbaum

    private Baumknoten wurzel;
    ...
}

public class Baumknoten {

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

    /**
     * Liefert genau dann true, wenn dieser Knoten oder einer der
     * darunter liegenden Knoten den übergebenen Wert enthaelt.
     */
    public boolean enthaelt(int wert) {
        ...
    }
}
```

Realisieren Sie in der Klasse `BinaererSuchbaum` eine Methode `boolean enthaeltDoppelte()`, die angibt, ob der Suchbaum gleiche Werte mehrfach enthält. Sie müssen hierfür auch die Klasse `Baumknoten` erweitern.

Da es sich um Suchbäume handelt, können Sie sich darauf verlassen, dass für jeden Knoten gilt, dass sein Wert größer oder gleich dem Wert aller linken Knoten ist, und dass sein Wert kleiner als der Wert aller rechter Knoten ist. Der Baum auf der nächsten Seite ist ein Beispiel dafür.

Beachten Sie die Methode `enthaelt` in der Klasse `Baumknoten`. Sie dürfen davon ausgehen, dass diese Methode vollständig implementiert ist. Sie können sie für Ihre Lösung verwenden.

Beispiel: Dieser Suchbaum enthält doppelte Werte.