Name: *Matrikelnummer:*

Programmierung

Probeklausur (Wintersemester 2019/20)

17. Dezember 2019

Hinweise (*English translation below*) - *Endklausur-MUSTER*

- **Bitte schlagen Sie die Klausur erst auf, sobald Sie dazu aufgefordert werden.**
- Einlesezeit: 15 Minuten. Fragen zur Klausur sollten innerhalb dieser Zeit gestellt werden. Wenn Sie eine Frage haben, melden Sie sich, und es wird jemand zu Ihnen kommen. **Anschließende** Bearbeitungszeit: 120 Minuten
- Mögliche Gesamtpunktzahl: 100 Punkte (+ 10 Bonuspunkte). Die Punktzahl jeder Aufgabe entspricht etwa der vorgesehenen maximalen Bearbeitungszeit in Minuten.
- Überzeugen Sie sich davon, dass Ihre Klausur vollständig ist.
- Legen Sie Ihren Studentenausweis sowie einen Lichtbildausweis vor sich auf den Tisch. Während der Klausur werden wir Ihre Identität kontrollieren.
- Es sind keinerlei Hilfsmittel (Taschenrechner, Skripte, Bücher, Notizen, Telefone, etc.) für diese Klausur erlaubt. Falls Sie Papier brauchen geben Sie uns Bescheid statt eigenes zu benutzen. Bitte schalten Sie Ihre Telefone ab und verstauen Sie Smart Watches in Ihrem Rucksack. Wenn Sie gegen diese Regeln verstoßen, riskieren Sie, von der Prüfung ausgeschlossen zu werden und durchzufallen.
- Schreiben Sie Ihre Antworten auf die Aufgabenzettel. Sie können Antworten ggf. auf der letzten leeren Seite fortsetzen, bitte weisen Sie dann entsprechend darauf hin. Sollte der Platz immer noch nicht ausreichen fragen Sie uns nach zusätzlichem Papier.
- Sie dürfen die Fragen auf deutsch oder englisch beantworten.
- Bitte schreiben Sie auf **jedes Blatt**, das Sie abgeben, Ihren Namen und (falls vorhanden) Ihre Matrikelnummer.
- Bis 20 Minuten vor Ende der Klausur dürfen Sie vorzeitig abgegeben. Wenn Sie nicht vorzeitig abgeben, warten Sie bitte an Ihrem Platz bis Ihre Klausur zusammengeheftet ist und eingesammelt wird.
- Die korrigierten Klausuren können am XX.03.2018 zwischen XX und XX Uhr in XXX X, Raum X, eingesehen werden.

English translation of instructions above — such translations are also provided for the exam problems

- **Please do not turn the page before you are told to do so.**
- Reading time: 15 minutes. Questions concerning the problems should be asked during that time. If you have a question, raise your hand, and somebody will come to you to listen to your question. **Afterwards** time for problem solving: 120 minutes.
- The total maximum score: 100 points (+ 10 bonus points). The points given to each problem roughly correspond to the assumed maximum time to work on that problem.
- Ensure that your copy of the exam is complete.
- Put a photo ID card in front of you. We will examine it during the exam.
- You are not allowed to use anything other than a pen to write with. In particular, you are not allowed to use the book, any printouts, or electronic devices (which includes phones and smart watches). Do not use your own paper to write on; instead, ask us for paper, we have plenty. Breaking these rules means risking a failing grade.
- Use the space provided in the assignments to write down your answers. You may continue your answers on the very last, empty page, please make a note if you do so. If you still run out of space, ask us for additional paper.
- You may answer in English or German.
- Please put your name and immatriculation number (*Matrikelnummer*, if you have one) onto **every sheet** that you hand in.
- You are allowed to hand in early until 20 minutes before the end of the exam. If you do not hand in early, please wait at your seat until your exam has been collated and collected.
- You can examine your corrected exam on March XXth 2018 between Xpm and Xpm in XXX X, room X.

Aufgabe 1 (5 Punkte)

1. Was sind wesentliche Eigenschaften eines *Algorithmus*?

   What are essential properties of an *algorithm*?

2. Was ist der Unterschied zwischen einer *Klasse* und einem *Objekt*?

   What is the difference between a *class* and an *object*?

3. Was ist das Ergebnis der folgenden Ausdrücke?

   What is the result of the following expressions?

   a) `1 & 2`

   b) `11 | 0x11`

   c) `(0xFF » 6) | 011`

4. Was ist ein abstrakter Datentyp?

   What is an abstract data type?

5. Welche `for`-Kontrollzeile würden Sie für die nachfolgenden Aufgaben verwenden:

   What `for` loop control line would you use in each of the following situations:

   a) Von 1 bis 10 zählen.

      Counting from 1 to 10.

   b) In Fünferschritten von 0 aufwärts zählen solange die Zahl höchstens dreistellig ist.

      Counting by fives starting at 0 as long as the number has at most three digits.

Aufgabe 2 (5 Punkte) Kreuzen Sie bei den folgenden Fragen bitte alle zutreffenden Optionen an. Für jede richtig beantwortete Frage, bei der also genau die zutreffenden Optionen angekreuzt sind, gibt es einen Punkt. Die Zahl der jeweils zutreffenden Optionen kann zwischen "keine" und "alle" variieren; das heißt, zufälliges ankreuzen lohnt sich nicht.

Tick all correct options in the following questions. For each correctly answered question, where exactly the correct options are ticked, you earn one point. The number of correct options may vary arbitrarily between "none" and "all." It thus does not pay to randomly tick options.

1. Welche der folgenden Schlüsselworte sorgen dafür, dass der Wert einer Variablen nach der ersten Zuweisung nicht mehr veränderbar ist?

   Which of the following keywords ensure that the value of a variable, once assigned, cannot be changed anymore?

   ☐ `static` ☐ `private` ☐ `import` ☐ `protected`

2. Welche der folgenden Codeausschnitte sind valide Java-Anweisungen?

   Which of the following statements are valid Java code?

   ☐ `int pi = 3.14;` ☐ `double pi = 3;` ☐ `char pi = 3;` ☐ `String pi = 3;`

3. Welche der folgenden Klassen erweitern, direkt oder indirekt, die Klasse `Object`?

   Which of the following classes extend the `Object` class, either directly or indirectly?

   ☐ `ConsoleProgram` ☐ `GObject` ☐ `RandomGenerator` ☐ `Integer`

4. Wie muss die Deklaration einer `main`-Methode aussehen?

   How does a `main` method have to be declared?

   ☐ `public void main(String[] args){ ... }`

   ☐ `public static void main(char[][] args){ ... }`

   ☐ `public static final void main(String[] args){ ... }`

   ☐ `public static void main(String[] args){ ... }`

5. Welche der folgenden Aussagen über Variablen sind wahr?

   Which of the following statements about variables are correct?

   ☐ Parameter können denselben Namen wie Instanzvariablen in derselben Klasse haben.
   Parameters can have the same name as instance variables in the same class.

   ☐ Lokale Variablen können denselben Namen wie Instanzvariablen in derselben Klasse haben.
   Local variables can have the same name as instance variables in the same class.

   ☐ Lokale Variablen können denselben Namen wie andere lokale Variablen im selben *Scope* haben.
   Local variables can have the same name as local variables in the same scope.

   ☐ Die Namen statischer Variablen müssen über alle Klassen hinweg eindeutig sein.
   The names of static variables must be unique over all classes.

Aufgabe 3 (10 Punkte) Im Folgenden sehen Sie ein Stück Java-Code. Finden Sie für jeden der nachfolgenden Begriffe ein Vorkommen in dem Code. Markieren Sie die Stelle im Code entsprechend mit der Nummer des Begriffs (siehe 1 als Beispiel). Es reicht aus, pro Begriff **ein Vorkommen im Code zu markieren**.

Consider the following piece of Java code. For each of the listed terms, annotate an appropriate part of code with the number of the term (see term 1 as example). It is sufficient to mark **one example per term**.

1. Package declaration
2. Access modifier
3. Argument
4. Assignment
5. Class variable
6. Conditional statement
7. Constant
8. Expression
9. Javadoc comment
10. Literal
11. Local variable
12. Method call
13. Parameter
14. Return type
15. Type
16. Operator
17. Superclass

```java
package org.eclipse.elk.alg.sequence.graph;
```

1

```java
public final class SLifeline extends SShape implements Comparable<SLifeline> {
    /** How many lifelines exist. */
    private static int lifelineCount = 0;

    /** The owning graph. */
    private final SGraph graph;

    public SLifeline(SGraph graph) {
        this.graph = graph;
        lifelineCount = lifelineCount + 1;
        graph.addLifeline(lifeline);
    }

    public boolean addMessage(SMessage newMsg) {
        ListIterator<SMessage> iterator = messages.listIterator();
        while (iterator.hasNext()) {
            SMessage currMsg = iterator.next();

            if (newMsg.getSourceYPos() < currMsg.getSourceYPos()) {
                iterator.previous();
                iterator.add(newMsg);
                return false;
            }
        }

        messages.add(newMsg);
        return true;
    }
}
```

Aufgabe 4 (10 Punkte) Unten sehen Sie die unvollständige Definition einer Klasse, die einen **mindestens zweidimensionalen** Vektor repräsentiert. Implementieren Sie die folgenden Methoden:

- `add(Vector)` soll die Summe zweier Vektoren berechnen und als neuen Vektor zurückliefern.
- `dotProduct(Vector)` soll das Skalarprodukt zweier Vektoren berechnen (die Summe der komponentenweisen Produkte).
- `toString()` soll eine textuelle Repräsentation des Vektors liefern. Dazu sollen seine Komponenten jeweils durch ein Komma getrennt hintereinander im String stehen.

Alle Methoden sollen geeignete *Exceptions* werfen. Ergänzen Sie den Konstruktor ebenfalls um geeignete *Exceptions*. Sie dürfen weitere Methoden hinzufügen, welche dann aber `private` sein müssen. Kommentieren Sie Ihren Code.

The `Vector` class below represents a vector that has at least two dimensions. Implement the following methods:

- `add(Vector)` calculates the sum of two vectors and returns it as a new vector.
- `dotProduct(Vector)` calculates the dot product of two vectors (the sum of their component-wise products).
- `toString()` returns a textual representation of the vector. Its components should be listed one after the other, divided by a comma.

All methods should throw appropriate exceptions. Extend the constructor to throw appropriate exceptions as well. You may add further `private` methods. Comment your code.

```
public class Vector {

  private final double[] vector;

  public Vector(int dimensions) {




    vector = new double[dimensions];
  }

  public double[] getVector() {
    return vector;
  }

  public Vector add(Vector other) {









  }
```

```
  public double dotProduct(Vector other) {









  }

  public String toString() {









  }
}
```

Aufgabe 5 (15 Punkte) Bestimmen Sie für jedes der nachfolgenden Code-Beispiele, ob es eine Eingabe für `x` so gibt, dass der Code ausschließlich ein einziges Mal „success" ausgibt und dabei keinen Fehler verursacht. Wenn es mehrere gültige Eingaben für `x` gibt, reicht es, eine anzugeben. Wenn es keine Eingabe gibt, streichen Sie den Lösungsbereich erkennbar durch.

For each of the following code snippets, determine whether there are any values of `x` that can be entered so that the code will print "success" only once, without causing any errors and without printing anything else. If there are multiple values of `x` that will work, just give one of them. If there are no values of `x` that will work, strike through the solution area.

```
int x = readInt();
if (x / 0 == x || x > 7) {
  println("success");
}
```

```
int x = readInt();
while (x % 2 != 0) {
  print("succ");
  x--;
}
println("ess");
```

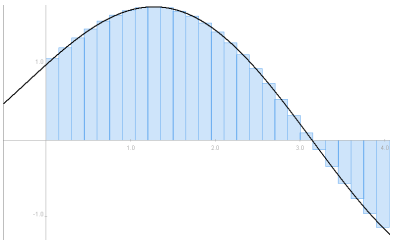
```
int x = readInt();
boolean b = false;
int i = 1;
while (i < Integer.MAX_VALUE) {
  b |= x % ++i == 0 && i != x;
}
if (!b) println("success");
```

```
int x = readInt();
for (int i = 0; i < x; ) {
  println("success");
}
```

```
int x = readInt();
int n = 1;
while (n > 0 && n != x) {
  n *= 2;
}
if (n == x) {
  println("success");
}
```

```
int x = readInt();
x = x * 0.5;
if (x % 2 == 0) {
  println("success");
}
```

Aufgabe 6 (10 Bonus Punkte) Das folgende Programm berechnet die Fläche zwischen einer Funktion $f(x)$ und der x-Achse indem gemittelte Rechtecke einer vorgegebenen Breite die Fläche approximieren. Das Programm enthält **fünf** Fehler, welche ein korrektes Ergebnis verhindern. Finden Sie die Fehler (markieren Sie die Fehler im Quellcode und erläutern Sie sie kurz im freien Bereich unter dem Progamm). Korrigieren Sie anschließend die Fehler (schreiben Sie die korrekte Zeile nieder).



(a) Example approximation

The following program calculates the area bounded by the graph of the function $f(x)$ and the x axis by calculating average rectangles of a given width to approximate the area. However, the program contains **five** errors that prevents it from computing the correct result. Find the errors (mark them in the source code and explain the issue in the space below) and correct them (write down the corrected line).

```
1  import acm.program.ConsoleProgram;
2
3  public class FunctionArea extends ConsoleProgram {
4
5    public void run() {
6      println(approxFunctionArea(0, 100, 15));
7    }
8
9    public double f(double x) {
10     return Math.sin(x) + Math.cos(0.5 * x);
11   }
12
13   public double avgRectHeight(double x, double width) {
14     if (width > 0) {
15       return 0;
16     }
17     double leftVal = f(x);
18     double rightVal = (int) f(x + width);
19     return (leftVal + rightVal) / 0.5;
20   }
21
22   public double approxFunctionArea(double left, double right,
23           double rectWidth) {
24
25     double area = 0.0;
26     for (double x = left; x < right; x += rectWidth) {
27       double width = Math.min(rectWidth, right + x);
28       area = avgRectHeight(x, width) * width;
29     }
30     return area;
31   }
32 }
```