Name: *Matrikelnummer:*

Programmierung

Klausur 1 (Wintersemester 2019/20)

3. März 2020

Hinweise (*English translation below*)

- **Bitte schlagen Sie die Klausur erst auf, sobald Sie dazu aufgefordert werden.**
- Einlesezeit: 10 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: 90 Minuten
- Mögliche Gesamtpunktzahl: 100 Punkte (+ 10 Bonuspunkte).
- Überzeugen Sie sich davon, dass Ihre Klausur vollständig ist.
- Legen Sie Ihren Studierendenausweis 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 gegebenenfalls 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 abgeben. Wenn Sie nicht vorzeitig abgeben, warten Sie bitte an Ihrem Platz bis Ihre Klausur zusammengeheftet ist und eingesammelt wird.
- Die korrigierten Klausuren können am 09.03.2020 zwischen 12:15 und 15:15 Uhr in LMS 2, Raum Ü2, eingesehen werden. Die exakte Zeit für Sie hängt von Ihrer Rechnerübung ab. Bitte schauen Sie dafür ins Wiki.

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: 10 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: 90 minutes.
- The total maximum score: 100 points (+ 10 bonus points).
- Ensure that your copy of the exam is complete.
- Put a photo ID card and your student 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 9, 2020 between 12:15pm and 3:15pm in LMS 2, room Ü2. The exact time you should show up at depends on which practical class you were assigned to during the semester. Our wiki contains further details.

Aufgabe 1 (12 Punkte) Kreuzen Sie bei den folgenden Fragen bitte alle zutreffenden Optionen an. Für eine Frage mit vier korrekt markierten Optionen gibt es 1,5 Punkte. Für eine Frage mit drei korrekt markierten Optionen gibt es 0,5 Punkte. Für eine Frage mit weniger korrekt markierten Optionen gibt es 0 Punkte. 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. A question with four correct answers earns 1.5 points. A question with three correct answers earns 0.5 points. A question with fewer correct answers earns 0 points. The number of correct options may vary arbitrarily between "none" and "all." It thus does not pay to randomly tick options.

1. Arrays...

   Arrays...

   ☐ ...stellen eine Methode `length()` zur Verfügung
   ...provide a method `length()`

   ☐ ...stellen ein öffentliches Feld `length` zur Verfügung
   ...provide a public field `length`

   ☐ ...speichern Elemente homogenen Typs
   ...store elements of homogeneous type

   ☐ ...sind in ihrer Größe veränderbar
   ...have a changeable size

2. Welche der folgenden Arten von Variablen werden *immer* auf dem *Stack* gespeichert?

   Which of the following kinds of variables are *always* saved on the *stack*?

   ☐ Instanzvariablen
   Instance variables

   ☐ Objektvariablen
   Object variables

   ☐ Klassenvariablen
   Class variables

   ☐ Lokale Variablen
   Local variables

3. Welche der folgenden Dinge kann direkt innerhalb einer Klasse außerhalb von Methoden deklariert werden?

   Which of the following things can be declared directly inside a class (that is, outside of methods)?

   ☐ Lokale Variablen
   Local variables

   ☐ Parameter
   Parameters

   ☐ Konstanten
   Constants

   ☐ Pakete
   Packages

4. Statische Methoden...

   Static methods...

   ☐ ...müssen auf Objekten aufgerufen werden
   ...must be called on objects

   ☐ ...können auf Objekten aufgerufen werden
   ...can be called on objects

   ☐ ...können über **`this`** auf Instanzvariablen zugreifen
   ...can access instance variables through **`this`**

   ☐ ...können keine Parameter empfangen
   ...can only have empty parameter lists

5. Welche der folgenden **`for`**-Schleifen terminiert?

   Which of the following **`for`** loops terminates?

   ☐
   ```
   for (int i = 0; i != 0; i++){ }
   ```
   ☐
   ```
   for (int i = 0; i <= 0 || i > -1; i++){ }
   ```

☐ `for (int i = 0; i < 5; i++){ }`

☐ `for (int i = 0; i < 42; i *= -1){ }`

6. Welche der folgenden Ausdrücke führen zu Kompilierfehlern unter der Voraussetzung, dass die referenzierten Variablen existieren und passenden Typs sind?

   Which of the following expressions will fail to compile provided that the referenced variables exist and have compatible types?

   ☐ `(n * 2)++`

   ☐ `"The value is "+ 42`

   ☐ `((n + (5))* 7) / 5`

   ☐ `o.equals(null)`

7. Welche Regeln müssen beachtet werden, um aus einem Ausdruck einen vollständig geklammerten Ausdruck zu machen?

   Which rules need to be taken into account in order to turn an expression into a fully parenthesized expression?

   ☐ Kommutativität der Operatoren
   Operator commutativities

   ☐ Präzedenz der Operatoren
   Operator precedences

   ☐ Assoziativität der Operatoren
   Operator associativities

   ☐ Reflexivität der Operatoren
   Operator reflexivities

8. Welche der folgenden Anweisungen sind *Kontrollflussanweisungen*?

   Which of the following statements are *control flow statements*?

   ☐ **`for`** ☐ **`while`** ☐ **`do-while`** ☐ **`switch`**

Aufgabe 2 (12 Punkte) Beantworten Sie die folgenden Fragen *kurz*. Jede korrekte Antwort gibt 1,5 Punkte.
Briefly answer the following questions. Each correct answer earns 1.5 points.

1. Was ist ein *Term*, und welche Arten von Termen kennen wir?

   What is a *term*, and which types of terms do we know?

2. Was ist eine *Zuweisung*?

   What is an *assignment*?

3. Was ist eine *nicht-mutierbare* Klasse?

   What is an *immutable class*?

4. Welche drei Regionen unterscheiden wir im *Speicher*? Wo werden die drei verschiedenen Arten von Variablen abgespeichert, die wir kennen?

   Which three areas do we distinguish in memory? Which of the three kinds of variables we know is stored where?

5. Wie überprüfen wir Strings auf Gleichheit?

   How do we check strings for equality?

6. Warum ist `break` ein wichtiges Schlüsselwort in `switch`-Anweisungen?

   Why is `break` an important keyword in `switch` statements?

7. Was ist der Unterschied zwischen dem `if`- und dem `? :`-Konstrukt?

   What is the difference between the `if` and the `? :` constructs?

8. Was ist ein *Interface*?

   What is an *interface*?

Aufgabe 3 (16 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. Binary operator
5. Boolean expression
6. Class variable
7. Conditional statement
8. Constant
9. Iterative statement
10. Javadoc comment
11. Local variable
12. Method call
13. Parameter
14. String literal
15. Superclass
16. Type
17. Unary operator

```
package programming.set6.optimus;                                    1

public class SieveOfEratosthenes extends ConsoleProgram {

    /** Minimum maximum value. */
    private static final int MIN_UPPER_BOUND = 2;
    /** Actual maximum value. */
    private int upperBound;

    @Override
    public void run() {
        do {
            upperBound = readInt("Enter a value!");
        } while (upperBound < MIN_UPPER_BOUND);

        boolean[] isComposite = new boolean[upperBound + 1];

        for (int i = 2; i < isComposite.length; i++) {
            if (!isComposite[i]) {
                println(i);

                for (int mult = i * 2; mult < isComposite.length; mult += i) {
                    isComposite[mult] = true;
                }
            }
        }
    }

    public static void main(String[] args) {
        new SieveOfEratosthenes().start();
    }
}
```

Aufgabe 4 (40 Punkte) Die Klasse `SubstringCounter` soll die Anzahl der Vorkommen von Zeichenketten in einer Basis-Zeichenkette zählen (`aa` kommt in `aaa` beispielsweise zwei mal vor). Bereits gezählte Vorkommen sollen in der `HashMap` namens `countCache` gespeichert und bei erneutem Abruf von dort bezogen anstatt erneut gezählt zu werden.

Implementieren Sie die folgenden Methoden:

- Der Konstruktor soll die Basis-Zeichenkette speichern falls nicht **`null`** übergeben wurde.
- `isInString(`**`String`**`)` soll überprüfen, ob eine (nicht leere) Zeichenkette in der Basis-Zeichenkette mindestens ein mal vorkommt.
- `occurrencesOf(`**`String`**`)` soll die Anzahl der Vorkommen einer (nicht leeren) Zeichenkette in der Basis-Zeichenkette zurückgeben.

Alle Methoden sollen geeignete *Exceptions* werfen. Sie dürfen weitere Methoden und Variablen hinzufügen, welche dann aber **`private`** sein müssen. Kommentieren Sie Ihren Code (Javadocs sind allerdings nicht nötig). Von der **`String`**-Klasse stehen Ihnen in dieser Aufgabe lediglich die Methoden `length()` und `charAt(`**`int`**`)` zur Verfügung.

The class `SubstringCounter` should count the number of occurrences of strings in a base string (`aa` for example appears twice in `aaa`). Occurrences already counted for a particular substring should be saved in a `HashMap` called `countCache`. If they should be counted again for that substring, they should be retrieved from the map instead of counting anew.

Implement the following methods:

- The constructor should save the base string unless it is **`null`**.
- `isInString(`**`String`**`)` should check whether a (non-empty) string occurs at least once in the base string.
- `occurrencesOf(`**`String`**`)` should return the number of occurrences of a substring in the base string.

All methods should throw appropriate exceptions. You may add further **`private`** methods and variables. Comment your code (Javadocs are not required, though). Out of all methods in the **`String`** class, you may only use `length()` and `charAt(`**`int`**`)`.

```
import java.util.HashMap;

public class SubstringCounter {

    private final HashMap<String, Integer> countCache = new HashMap<>();









    public SubstringCounter(String base) {









    }

    private boolean isInCache(String sub) {
        return countCache.containsKey(sub);
    }
```

```
    public boolean isInString(String sub) {









    }
    public int occurrencesOf(String sub) {









    }

}
```

Aufgabe 5 (20 Punkte) Gegeben sei die folgende Klasse, die eine verlinkte Liste für nicht negative Zahlen implementieren soll.

Consider the following class, which is supposed to implement a linked list of non-negative integers.

```
1  public class StrangeList {
2      private static StrangeElement listHead = null;
3
4      public void add(int value) {
5          if (value < 0) {
6              throw new IllegalArgumentException();
7          }
8
9          StrangeElement newElement = new StrangeElement();
10
11         if (listHead == null) {
12             listHead = newElement;
13         } else {
14             StrangeElement lastElement = listHead;
15             while (lastElement.nextElement != null) {
16                 lastElement = lastElement.nextElement;
17             }
18             lastElement.nextElement = newElement;
19         }
20     }
21
22     public int get(int index) {
23         StrangeElement element = listHead;
24         for (int i = 1; i < index && element != null; i++) {
25             element.nextElement = element.nextElement;
26         }
27
28         if (element == null) {
29             return -1337;
30         } else {
31             return element.value;
32         }
33     }
34
35     public void clear() {
36         listHead.nextElement = null;
37     }
38 }
39
40 class StrangeElement {
41     public int value;
42     public StrangeElement nextElement = null;
43 }
```

Name: *Matrikelnummer:*

Der folgende Code nutzt die Klasse und soll eigentlich die rechts angegebene Ausgabe produzieren.

The following code uses this class and is supposed to produce the given output.

Code:

```
1  StrangeList odd = new StrangeList();
2  StrangeList even = new StrangeList();
3
4  for (int i = 1; i <= 6; i++) {
5      if (i % 2 == 0) {
6          even.add(i);
7      } else {
8          odd.add(i);
9      }
10 }
11
12 for (int i = 0; i < 3; i++) {
13     System.out.println(odd.get(i));
14 }
15
16 odd.clear();
17 System.out.println(odd.get(0));
```

Expected Output:

```
1
3
5
-1337
```

1. (15 Punkte) `StrangeList` enthält **fünf fehlerhafte oder fehlende Zeilen**, welche die erwartete Ausgabe verhindern. Finden Sie zunächst die Fehler (markieren Sie die Fehler im Quellcode und erläutern Sie sie kurz im freien Bereich unter der Klasse). Korrigieren Sie anschließend die Fehler (schreiben Sie die korrekte Zeile nieder).

   `StrangeListStrangeList` contains **five erroneous or missing lines** that prevent our code from producing the expected output. 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).

2. (5 Punkte) Geben Sie die Ausgabe an, die der Code mit der fehlerhaften `StrangeList`-Implementierung tatsächlich erzeugt.

   Write down the output our code actually produces with the faulty version of `StrangeList`.

Aufgabe 6 (10 Bonus Punkte) Die folgenden beiden Programme initialisieren jeweils ein Array. Geben Sie in der dafür vorgesehenen Grafik den Inhalt des Arrays an direkt bevor die `main(...)`-Methode terminiert.
The following programs both declare and initializes an array. In the graphic below, write down the values which the respective arrays hold just before the `main(...)` method terminates.

arr1													
arr2													
	0	1	2	3	4	5	6	7	8	9	10	11	12

1. (5 Punkte)

```
public class ArrayInitializer {
  private static int[] arr1 = new int[13];
  public static void main(String[] args) {
    for (int i = 0; i < arr1.length; i++) {
      new ValueGenerator();
      arr1[i] = ValueGenerator.val;
    }
  }
}

class ValueGenerator {
  public static int val = 0;
  public int mod = 0;
  public ValueGenerator() {
    if (val == 0) {
      mod++;
    }
    val = val * 2 + mod++;
  }
}
```

2. (5 Punkte)

```
public class ArrayFiller {
  private static int[] arr2 = new int[13];

  public static void main(String[] args) {
    for (int i = 0; i < arr2.length; i++) {
      int val1 = 0;
      int val2 = 1;
      for (int j = i; j > 0; j--) {
        int tmp = val1 + val2;
        val1 = val2;
        val2 = tmp;
      }
      arr2[i] = val2 % 2;
    }
  }
}
```