Name: *Matrikelnummer:*

Programmierung

Probeklausur (Wintersemester 2016/17)

20. Dezember 2016

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.03.2017 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 2017 between Xpm and Xpm in XXX X, room X.

Aufgabe 1 (10 Punkte)

1. Was sind wesentliche Eigenschaften eines *Algorithmus*?

   What are essential properties of an *algorithm*?

2. Wie unterscheidet sich ein *Interpretierer* von einem *Übersetzer*?

   How does an *interpreter* differ from a *compiler*?

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

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

4. Was ist ein *Term*?

   What is a *term*?

5. Was ist eine *Zuweisung*?

   What is an *assignment*?

6. Entscheiden Sie welche der folgenden Konstanten legale numerische Konstanten in Java sind. Für die legalen Konstanten, geben Sie an ob diese ganzzahlige Konstanten oder Fließkommakonstanten sind.

   Identify which of the following are legal numeric constants in Java. For the ones that are legal, indicate whether they are integers or floating-point constants.

   a) `0,0`
   b) `-12.3`
   c) `2.3E-1.3`
   d) `0x1ga`
   e) `0x0`
   f) `Affe`

7. Welches Problem ist mit *Dangling Else* gemeint?

   What is the *dangling else problem*?

8. Was ist eine *Methode*?

   What is a *method*?

9. Was ist ein *Feld* (*Array*)?

   What is an *array*?

10. Was ist *Rekursion*?

    What is *recursion*?

Aufgabe 2 (15 Punkte) Schreiben Sie eine Klasse `Diamond`, die eine Subklasse von `ConsoleProgram` ist und im Paket `programming.exam.midterm.diamond` liegt. Nach Start des Programms soll der Benutzer nach einer Ganzzahl $n \geq 0$ gefragt werden. Das Programm soll so lange nach n fragen, bis ein gültiger Wert für n eingegeben wurde. Anschließend soll eine Diamantenform auf der Konsole ausgegeben werden, die n Zeilen über und unter der Mittellinie aufweist. Teilen Sie ihren Quelltext in sinnvolle Methoden auf. Sie brauchen keine Import-Anweisungen aufzuschreiben.

Für $n = 3$ würde die Diamantenform wie folgt aussehen:

Write a class `Diamond` in a package called `programming.exam.midterm.diamond` which is a subclass of `ConsoleProgram`. When the program is run, it should ask the user for an integer $n \geq 0$. As long as the user enters an invalid value for n, she should be asked again for a valid value. Then, the program should print a diamond shape to the console that has n lines above and below the center line. Split your code into methods as appropriate. You don't have to add import statements to your code.

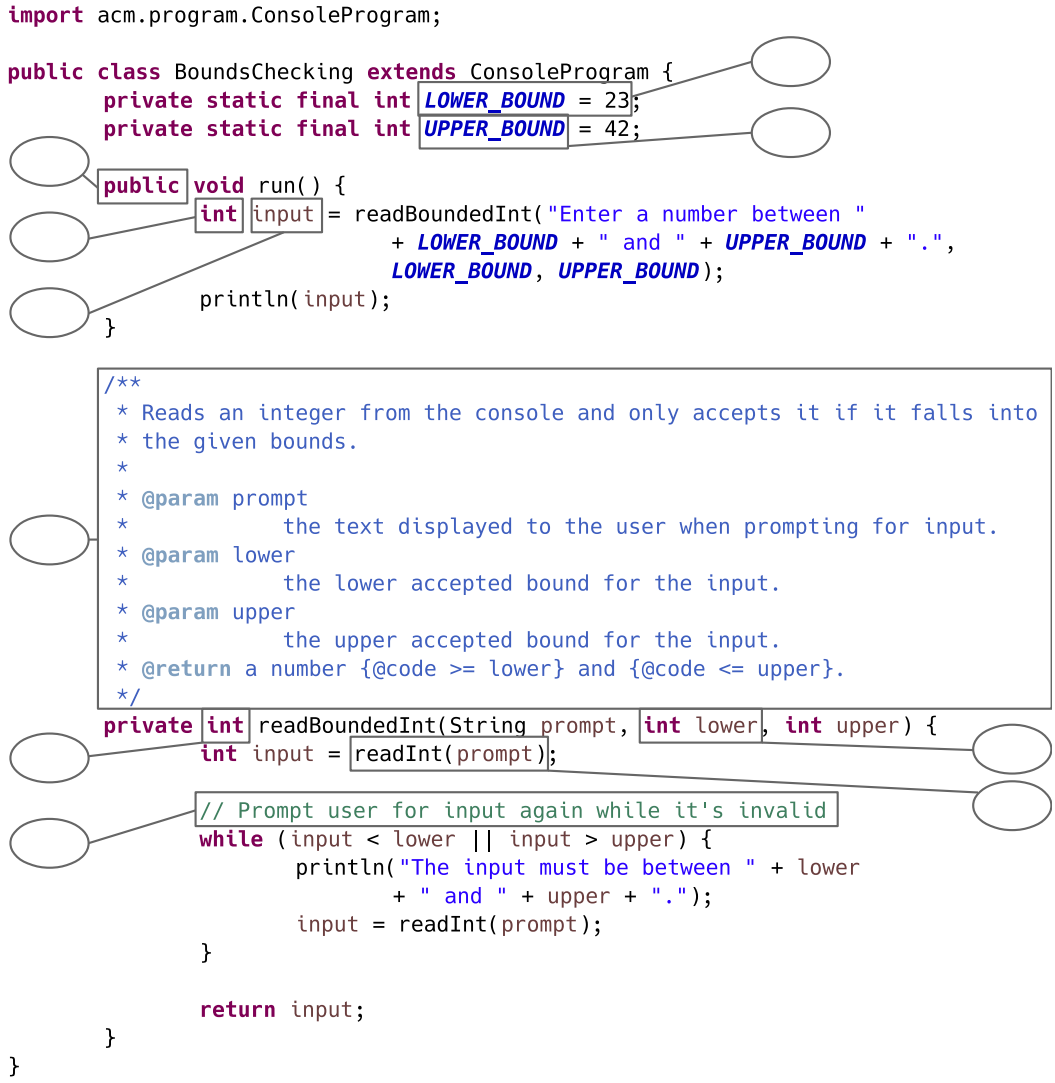
For $n = 3$, the diamond would look like this:

```
   #
  ###
 #####
#######
 #####
  ###
   #
```

Aufgabe 3 (10 Punkte) Im Folgenden sehen Sie ein Stück Java-Code. Schreiben Sie in jedes der vorgesehenen Felder die Zahlen der dazu passenden Begriffe aus der Liste. Es müssen nicht alle Begriffe Verwendung finden. Pro Feld können mehrere Zahlen zutreffen.

Consider the following piece of Java code. Annotate each of the highlighted parts of code with the corresponding term or terms from the list of terms below by writing the appropriate number(s) into the respective circle.

1. Argument
2. Assignment
3. Block
4. Boolean expression
5. Class
6. Comment
7. Constant
8. Constructor
9. Expression
10. Integer expression
11. Iterative statement
12. Javadoc comment
13. Method call
14. Object
15. Package
16. Parameter
17. Return statement
18. Return type
19. Statement
20. String
21. Term
22. Type
23. Variable
24. Visibility



```
import acm.program.ConsoleProgram;

public class BoundsChecking extends ConsoleProgram {
        private static final int LOWER_BOUND = 23;
        private static final int UPPER_BOUND = 42;

        public void run() {
                int input = readBoundedInt("Enter a number between "
                                + LOWER_BOUND + " and " + UPPER_BOUND + ".",
                                LOWER_BOUND, UPPER_BOUND);
                println(input);
        }

        /**
         * Reads an integer from the console and only accepts it if it falls into
         * the given bounds.
         *
         * @param prompt
         *            the text displayed to the user when prompting for input.
         * @param lower
         *            the lower accepted bound for the input.
         * @param upper
         *            the upper accepted bound for the input.
         * @return a number {@code >= lower} and {@code <= upper}.
         */
        private int readBoundedInt(String prompt, int lower, int upper) {
                int input = readInt(prompt);

                // Prompt user for input again while it's invalid
                while (input < lower || input > upper) {
                        println("The input must be between " + lower
                                        + " and " + upper + ".");
                        input = readInt(prompt);
                }

                return input;
        }
}
```

Aufgabe 4 (10 Punkte) Untenstehend sehen Sie die unvollständige Definition einer simplen, Array-basierten Listenklasse. Ergänzen Sie die Implementierung der Methoden `remove(int index)` sowie `ensureArrayCapacity()`.
You will find the incomplete definition of an array-based list class below. Add the implementation of the two methods `remove(int index)` and `ensureArrayCapacity()`.

```
public class DoubleArrayList {
  /** The array we will be placing our items in. */
  private double[] items = new double[10];
  /** The number of items we have. */
  private int count = 0;

  /**
   * Adds the given item to the end of the list.
   * @param item the item to be added.
   */
  public void add(double item) {
    ensureArrayCapacity();
    items[count++] = item;
  }

  /**
   * Removes the item at the given zero-based index from the list.
   * @param index the index of the item to remove.
   * @throws IllegalArgumentException
   *         if there is no element with the given index in the list.
   */
  public void remove(int index) {









  }

  /**
   * Enlarges the items array if it is full. After calling this method,
   * the array must have enough space left for at least one more item.
   */
  private void ensureArrayCapacity() {









  }
}
```

Aufgabe 5 (10 Bonus Punkte) Bestimmen Sie für jedes der nachfolgenden Code-Beispiele ob es eine Eingabe für x gibt, so dass der Code ausschließlich „success" ausgibt und dabei keinen Fehler verursacht. Wenn es mehrere gültige Eingaben für x gibt, reicht es, eine anzugeben. Wenn es keine Eingabe gibt, schreiben Sie „no solution".

For each of these code snippets, determine whether there are any values of x that can be entered so that the code will print "success" 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, write "no solution."

```
int x = readInt();
if (x != 0 && x / 2 == 0) {
  println("success");
}
```

_____

```
int x = readInt();
if (x >= 10000) {
  while (x != 0) {
    if (x % 10 != 9) {
      println("failure");
    }
    x /= 10;
  }
  println("success");
}
```

_____

```
int x = readInt();
if (x != 0 || x != 1) {
  println("failure");
}
println("success");
```

_____

```
int x = readInt();
if (x / 2 * 3 == 6) {
  println("success");
}
```

_____

```
int x = readInt();
if (x > 0) {
  x /= 10;
  if (x == 0) {
    println("success");
  }
  if (10 / x == 0) {
    println("failure");
  }
}
```

_____

```
int x = readInt();
String s = "failsuccurecess";
println(
  s.substring(x / 1000, 8) +
  s.substring(x % 100, 15));
```

_____