Name: *Matrikelnummer:*

Programmierung

10. Dezember 2017

Probeklausur (Wintersemester 2017/18)

Answer: *With Solutions*

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 (10 Punkte)

1. Was ist eine Definition von *Software Engineering*?

   What is a definition of *Software Engineering*?

   Answer: *Discipline of writing programs so that they can be understood and maintained by others.*

2. Was ist eine *Klassenhierarchie*?

   What is a *class hierarchy*?

   Answer: *Classes form hierarchies; each class can have one (in Java) superclass and an arbitrary number of subclasses.*

3. Was ist ein *Ausdruck*?

   What is an *expression*?

   Answer: *Terms + operators, evaluates to a value.*

4. Nennen Sie die zwei Attribute, die einen Datentypen definieren.

   What are the two attributes that define a data type?

   Answer: *A domain (valid values) and a set of operations.*

5. Was sind *iterative Anweisungen* in Java?

   What are *iterative statements* in Java?

   Answer: *for, while, do-while.*

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

   What is the *dangling else problem*?

   Answer: *Ambiguity of matching an else to one of the preceding if statements.*

7. Schreiben Sie eine Endlosschleife mit `for`.

   Write an infinity loop with `for`.

   Answer: *for(;;) or for(;true;)*

8. Was ist eine *statische Methode*? Nennen Sie ein Beispiel.

   What is a *static method*? Name an example.

   Answer: *Method that does not require to instantiate class into an object, see for example methods in Math class.*

9. Was passiert bei einem Methodenaufruf?

   What are the mechanics of method calling?

   Answer: *Evaluate arguments, copy argument values to parameters (stack frame), evaluate statements in method body, return substitutes value in place of call, discard stack frame, return to caller*

10. Was bedeutet *geschichtete Abstraktion*?

    What does *layered abstraction* mean?

    Answer: *Clients don't care where methods are implemented. Eg, some methods are implemented in Random, some in RandomGenerator.*

Aufgabe 2 (15 Punkte) Schreiben Sie eine Klasse `CrossedSquare`, die eine Subklasse von `ConsoleProgram` ist und im Paket `programming.exam.midterm.square` liegt. Nach Start des Programms soll der Benutzer nach einer Ganzzahl $n > 0$ gefragt werden. Das Programm soll so lange nach n fragen, bis ein gültiger Wert für n eingegeben wurde. Anschließend soll ein Quadrat mit Seitenlänge $n$mit #-Zeichen auf der Konsole ausgegeben werden, durch welches ein Strich von oben links nach unten rechts läuft. Teilen Sie ihren Quelltext in sinnvolle Methoden auf. Sie brauchen keine Import-Anweisungen aufzuschreiben.

Für $n = 7$ würde die Diamantenform wie folgt aussehen:

Write a class `CrossedSquare` in a package called `programming.exam.midterm.square` which is a subclass of `ConsoleProgram`. When the program is run, it should ask the user for an integer $n > 0$. As long as the user enters an invalid value for n, they should be asked again for a valid value. Then, the program should print a square shape with side length n to the console using the # character and a line that runs from the top left to the bottom right corner. Split your code into methods as appropriate. You don't have to add import statements to your code.

For $n = 7$, the diamond would look like this:

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

Answer: *The sample solution can be downloaded from the course's iLearn site.*

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.

Answer: *Left column, top to bottom: 1; 23; 24; 7, 14; 19, 23; 7. Right column, top to bottom: 10, 20; 8; 17; 11, 12, 15, 22.*

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

Aufgabe 4 (10 Punkte) Untenstehend sehen Sie einen Teil einer Klasse, die ein Array so benutzen möchte, dass Einträge an beliebiger Stelle eingefügt und entfernt werden können. Zu jeder Zeit sollten alle Einträge von Index 0 bis `count - 1` valide Einträge sein, die per Aufruf der `add`-Methode hinzugefügt wurden. Ergänzen Sie die Implementierung der Methoden `add(int index, double value)` sowie `remove(int index)`.

Below is a part of a class that wants to use an array as a random-access memory by inserting and removing values at arbitrary indices. At any given time, the values from index 0 to index `count - 1` should be valid entries added through calls to the `add` method. Add the implementation of the two methods `add(int index, double value)` and `remove(int index)`.

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

  /**
   * Inserts the given value at the given index. After this method returns,
   * values[index] == item will be true. The order of existing values must
   * remain stable.
   * @param index the index to insert the value at.
   * @param value the value to be inserted.
   * @throws IllegalArgumentException
   *         if index < 0 or index > count.
   */
  public void add(int index, double value) {
    if (index < 0 || index > count) {
      throw new IllegalArgumentException("invalid index");
    }

    // Check if the array is large enough for one more value
    if (count == values.length) {
      double[] newValues = new double[values.length * 2];
      for (int i = 0; i < count; i++) {
        newValues[i] = values[i];
      }
      values = newValues;
    }

    // Move values to the right
    for (int i = count - 1; i >= index; i--) {
      values[i + 1] = values[i];
    }

    // Add the new value
    values[index] = value;
    count++;
  }

  /**
   * Removes the value at the given zero-based index from the array.
   * @param index the index of the value to remove.
   * @throws IllegalArgumentException
   *         if there is no element with the given index in the list.
   */
  public void remove(int index) {
    if (index < 0 || index >= count) {
      throw new IllegalArgumentException("invalid index");
    }
```

```
    for (int i = index; i < count - 1; i++) {
      values[i] = values[i+1];
    }
    count--;
  }
}
```

Aufgabe 5 (10 Bonus Punkte)

1. Im Folgenden sehen Sie eine Liste von Ausdrücken. Schreiben Sie hinter jeden Ausdruck das Ergebnis seiner Auswertung in Java. Sollte während der Berechnung ein Fehler auftreten markieren Sie die Zeile stattdessen mit „ERROR".

   Consider the following list of expressions. Compute the value of each expression as interpreted by Java. If an error occurs during any of these evaluations, write "ERROR" on that line.

   `5.0 / 4 - 1 / 4` ***Answer:*** *1.25*

   `7 < 9 - 5 && 3 % 0 == 3` ***Answer:*** *false*

   `('6' - '2') + 'A'` ***Answer:*** *69 or 'E'*

   `"E" + 1 + 5 + 5` ***Answer:*** *"E155"*

   `"I" - "A"` ***Answer:*** *ERROR*

2. Vervollständigen Sie die folgenden Expressions, so dass das angegebene Ergebnis der Auswertung in Java zutrifft. Sie dürfen nur auf den markierten Flächen Ergänzungen vornehmen. Verwenden Sie ausschließlich **`int`**-, **`double`**-, **`String`**, **`char`**- oder **`boolean`**-Werte.

   Complete the following expressions such that the Java evaluation of each expression matches the specified result. Only write on the marked spaces. Use only **`int`**, **`double`**, **`String`**, **`char`**, or **`boolean`** values to complete the expressions.

`'H' +` ***Answer:*** *`"A"` `+ 'L' + 9000*	`"HAL9000"`
`!!!` ***Answer:*** *true* `|| !!` ***Answer:*** *false*	`false`
Answer: *5.0* `/ 10 * 3`	`1.5`
`(111 %` ***Answer:*** *100* `+ 1) /` ***Answer:*** *10.0*	`1.2`
`false ==/* true */` ***Answer:*** *false*	`true`