Name: *Matrikelnummer:*

Programmierung

18. Dezember 2018

Probeklausur (Wintersemester 2018/19)

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 (5 Punkte)

1. Was ist ein *Ausdruck*?

   What is an *expression*?

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

2. Was ist ein abstrakter Datentyp?

   What is an abstract data type?

   Answer: *Abstract data types consist of a set of variables + set of operations + specification*

3. Welche `for`-Kontrollzeile würden Sie benutzen, um in Fünferschritten von 0 aufwärts zu zählen solange die Zahl höchstens drei Stellen hat?

   What `for` loop control line would you use to count by fives starting at 0 as long as the number has at most three digits?

   Answer: `for (int i = 0; i < 1000; i += 5)`

4. 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.*

5. Deklarieren und initialisieren Sie eine Variable `inUse`, die ein Array enthält, das genau 16 Werte vom Typ `boolean` enthalten kann.

   Declare and initialize a variable called `inUse` that holds an array of 16 `boolean` values.

   Answer: `boolean[] inUse = new boolean[16]`

Name: *Matrikelnummer:*

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. Wodurch ist eine Variable charakterisiert, aus Sicht des Programmierers?

   What characterizes a variable, from the perspective of a programmer?

   ☒ Typ (Type) ☐ Speicheradresse (Memory address) ☒ Name (Name) ☒ Wert (Value)

2. Was ist ein Objekt?

   What is an object?

   ☐ Das einzige, was eine Methode zurückgeben darf
   The only thing methods may return

   ☒ Eine Instanz einer Klasse
   An instance of a class

   ☒ Das, was von einem Konstruktor initialisiert wird
   That which is initialized by a constructor

   ☐ Ein kompiliertes Programm
   A compiled program

3. Was sind wahre Aussagen?

   What are true statements?

   ☐ Ein Objekt erweitert eine Klasse
   An object extends a class

   ☐ Eine Superklasse erweitert eine Subklasse
   A superclass extends a subclass

   ☐ Eine Variable erweitert einen Wert
   A variable extends a value

   ☐ Fließkommazahlen erweitern rationale Zahlen
   Floating point numbers extend rational numbers

4. Welche der folgenden Statements führen zu Kompilierfehlern?

   Which of the following statements will fail to compile?

   ☒ `byte n1 = 1337;` ☐ `double n2 = 1337;` ☐ `byte[] arr = {1};` ☒ `3++;`

5. 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`

Aufgabe 3 (15 Punkte) Schreiben Sie ein Programm welches ungefüllte „rekursive Quadrate" bis zu einer Schachtelungstiefe d erzeugt. Jedes „äußere Quadrat" enthält ein Paar „innerer Quadrate", in gegenüberliegenden Ecken. Die minimale Schachtelungstiefe ist $d = 1$, für welche nur ein äußeres Quadrat erzeugt wird. Die Größe des initialen äußeren Quadrats ist durch die Konstante `WIDTH` gegeben. Die Größe der inneren Quadrate ist `INNER_FRACTION` mal der Größe des äußeren Quadrats.

Das Programm soll zunächst den Nutzer nach einem Wert für d fragen, welcher zwischen 1 und `MAX_DEPTH_LIMIT` liegt. Wenn der Nutzer einen Wert außerhalb dieses Wertebereiches eingibt, soll das Programm den Nutzer solange wiederholt fragen, bis der Nutzer einen gültigen Wert eingibt.

Die Wahl der Eckenpaare der inneren Quadrate alterniert zwischen oben links/unten rechts auf einer Schachtelungsebene, und oben rechts/unten links auf der nächsten Schachtelungsebene. Das initiale Eckenpaar kann entweder oben links/unten rechts oder oben rechts/unten links sein.

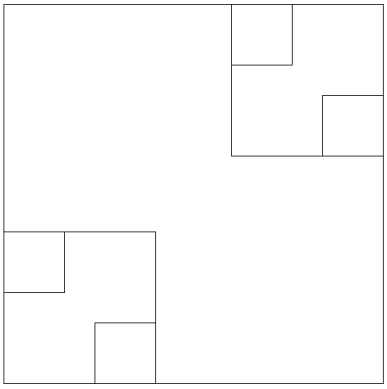
Die resultierende Zeichnung sollte wie in den beiden Beispielen unten aussehen.

Write a program that draws a set of non-filled "recursive squares," up to a given depth d. Each "outer square" contains a pair of "inner squares," in opposite corners. The minimal depth is $d = 1$, which corresponds to just one outer square. The size of the initial outer square is given by the constant `WIDTH`. The size of the inner squares is `INNER_FRACTION` times the size of the outer square.

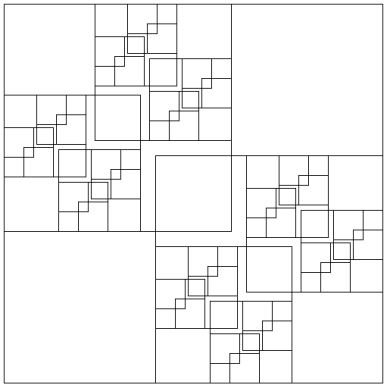
The program first asks the user for a value for d, which must be between 1 and `MAX_DEPTH_LIMIT`. If the user does not provide a number in that range, the program keeps asking the user until the user enters a suitable number.

The choice of corner pairs alternates between upper left/lower right at one nesting level, and upper right/lower left at the next nesting level. The initial choice of corner pairs can be either upper left/lower right or upper right/lower left.

The resulting drawing should look like the examples below.



(a) $d = 3$, `INNER_FRACTION`= 0.4



(b) $d = 6$, `INNER_FRACTION`= 0.6

Für maximale Punktzahl muss Ihre Lösung ein vollständiges Programm sein, einschließlich `import` Statements, Definition der in der Aufgabenstellung erwähnten Konstanten und Kommentaren.

To get maximum credit, your solution should be a complete program, including `import` statements, definitions of the constants referenced in the problem statement above and comments.

Ihre Lösung / *your solution*

Answer:

```
package misc;

import acm.graphics.GRect;
import acm.program.GraphicsProgram;

/**
 * Draw a set of non-filled "recursive squares", up to a given depth. Each
 * "outer square" contains a pair of "inner squares", in opposite corners.
 */
public class RecursiveSquares extends GraphicsProgram {

  /**
   * Run method, starts recursion
   *
   */
  public void run() {
```

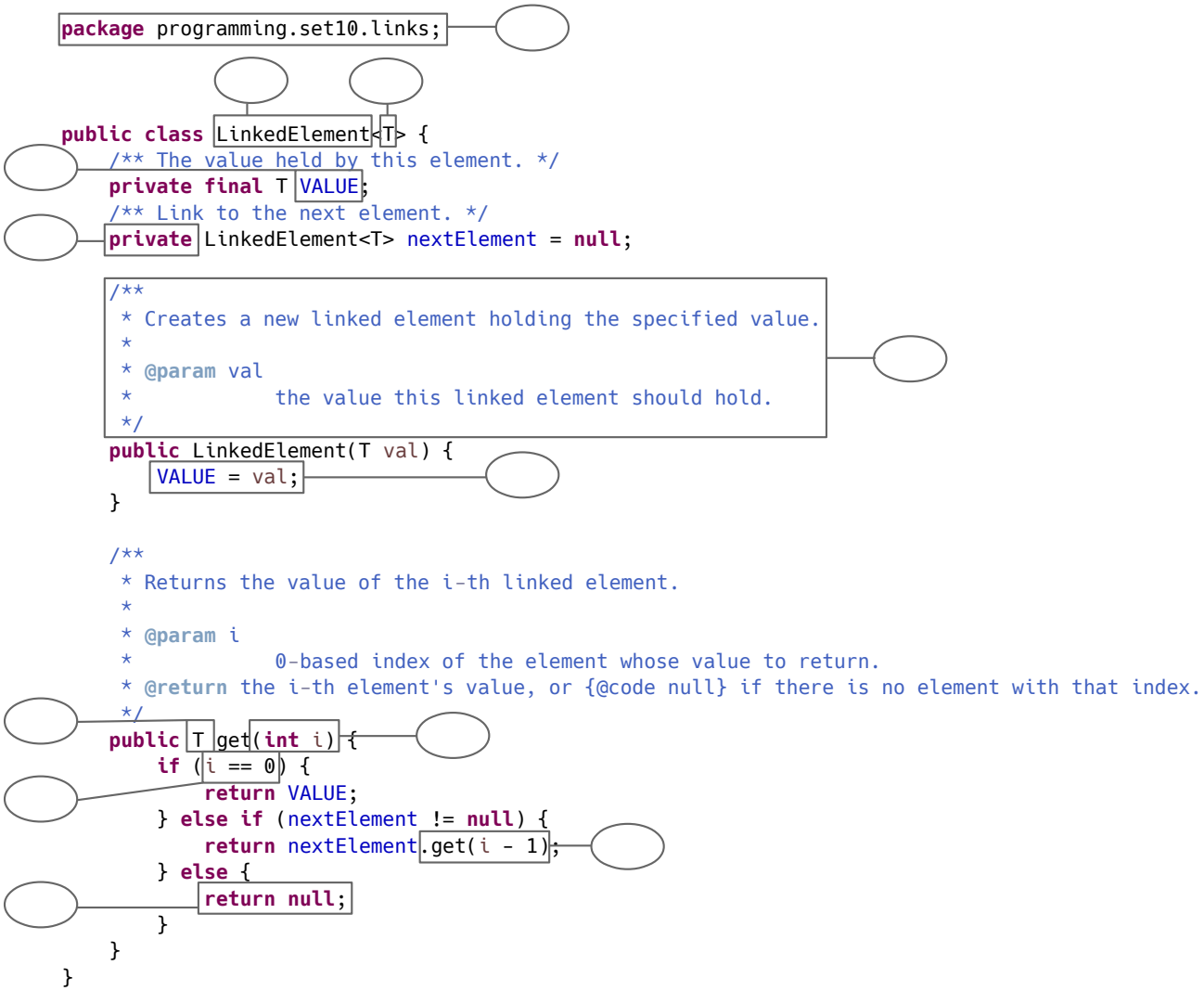
```
    int maxDepth;
    do {
      maxDepth = readInt("Enter maximum depth (between 1 and "
          + MAX_DEPTH_LIMIT + ": ");
    } while (maxDepth < 1 || maxDepth > MAX_DEPTH_LIMIT);

    drawSquares(maxDepth, 0, 0, WIDTH);
  }

  /**
   * Draw outer square and recursively draw inner squares
   *
   * @param depth
   *            Remaining nesting depth
   * @param x
   *            X-coordinate of upper left corner of outer square
   * @param y
   *            Y-coordinate of upper left corner of outer square
   * @param width
   *            Width of outer square
   */
  private void drawSquares(int depth, double x, double y, double width) {
    GRect rect = new GRect(x, y, width, width);
    add(rect); // Draw outer square

    if (depth > 1) {
      double innerWidth = INNER_FRACTION * width;
      double innerShift = width - innerWidth;
      int innerDepth = depth - 1;

      // If the remaining nesting depth is even, start with upper
      // left/lower right
      boolean upperLeft = depth % 2 == 0;

      if (upperLeft) {
        // Upper left
        drawSquares(innerDepth, x, y, innerWidth);
        // Lower right
        drawSquares(innerDepth, x + innerShift, y + innerShift,
            innerWidth);
      } else {
        // Upper right
        drawSquares(innerDepth, x + innerShift, y, innerWidth);
        // Lower left
        drawSquares(innerDepth, x, y + innerShift, innerWidth);
      }
    }
  }

  // Instance variables
  private static final RandomGenerator rgen = RandomGenerator.getInstance();

  // Constants
  private static final double WIDTH = 500; // Width of outer square
  private static final double INNER_FRACTION = 0.6; // Size of inner square,
              // relative to outer
  private static final int MAX_DEPTH_LIMIT = 20; // Max nesting depth
}
```

Aufgabe 4 (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. There may be terms that will go unused.

Answer: *Top to bottom, left to right: 9; 3; 16; 4 (17 also ok, but not required); 18; 6; 1, 13; 12; 10; 2; 7; 11 (13 also ok, but not required)*

1. Assignment
2. Boolean expression
3. Class
4. Constant
5. Constructor
6. Javadoc comment
7. Method call
8. Object
9. Package declaration
10. Parameter list
11. Return statement
12. Return type
13. Statement
14. Term
15. Type
16. Type parameter
17. Variable
18. Visibility modifier



```
package programming.set10.links;

public class LinkedElement<T> {
    /** The value held by this element. */
    private final T VALUE;
    /** Link to the next element. */
    private LinkedElement<T> nextElement = null;

    /**
     * Creates a new linked element holding the specified value.
     *
     * @param val
     *            the value this linked element should hold.
     */
    public LinkedElement(T val) {
        VALUE = val;
    }

    /**
     * Returns the value of the i-th linked element.
     *
     * @param i
     *            0-based index of the element whose value to return.
     * @return the i-th element's value, or {@code null} if there is no element with that index.
     */
    public T get(int i) {
        if (i == 0) {
            return VALUE;
        } else if (nextElement != null) {
            return nextElement.get(i - 1);
        } else {
            return null;
        }
    }
}
```

Aufgabe 5 (10 Punkte) Die Funktion facSum(n) für $n \in \mathbb{N}$ ist für $n \geq 0$ wie unten angegeben definiert. Implementieren Sie die methode `facSumIter`, welche facSum **iterativ** (nicht rekursiv) berechnet, und geben Sie an, was nach Starten des Programms auf der Konsole ausgegeben wird. Sie dürfen für die Implementierung weitere private Methoden hinzufügen. Denken Sie daran, Ihren Code zu kommentieren und benutzte Klassen zu importieren wenn nötig.

The function facSum(n) for $n \in \mathbb{N}$ is defined below for $n \geq 0$. Implement the method `facSumIter`, which calculates facSum in an **iterative** (instead of recursive) way, and state the console output generated by the program. You are allowed to add further private methods to help implement `facSum`. Remember to comment your code and import the classes you use, if necessary.

$$\text{facSum}(n) = \begin{cases} n \cdot \text{facSum}(n-1) & \text{if } n > 0 \text{ and } n \text{ even (gerade).} \\ n + \text{facSum}(n-1) & \text{if } n > 0 \text{ and } n \text{ odd (ungerade).} \\ 1 & \text{otherwise (sonst)} \end{cases}$$

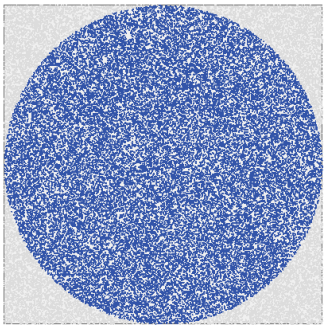
```
public class FacSum {
  /**
   * The program's main entry point.
   *
   * @param args command-line arguments.
   */
  public static void main(String[] args) {
    for (int i = 1; i < 7; i++)
      System.out.println(facSumIter(i));
  }

  /**
   * Calculates facSum as defined above iteratively.
   *
   * @param n value to compute the facSum for.
   * @throws IllegalArgumentException
   *         if facSum is not defined for the value of n.
   */
  private static int facSumIter(int n) {









  }
}
```

Answer:

```
public class FacSum {
    /**
     * The program's main entry point.
     *
     * @param args command-line arguments.
     */
    public static void main(String[] args) {
        for (int i = 1; i < 7; i++)
            System.out.println(facSumIter(i));
    }

    /**
     * Calculates facSum as defined above iteratively.
     *
     * @param n value to compute the facSum for.
     * @throws IllegalArgumentException if facSum is not defined for the
         value of n.
     */
    private static int facSumIter(int n) {
        if (n < 0) {
            throw new IllegalArgumentException("n must be >= 0");
        }

        int result = 1;

        for (int i = 1; i <= n; i++) {
            if (i % 2 == 0) {
                result *= i;
            } else {
                result += i;
            }
        }

        return result;
    }
}
```

Answer: *2 4 7 28 33 198*

Aufgabe 6 (10 Bonus Punkte) Das folgende Programm approximiert den Wert der Konstanten π indem es zufällige Punkte in einem zweidimensionalen Koordinatensystem generiert, deren Koordinaten zwischen -1 und 1 liegen. Es zählt, wie viele dieser Punkte im Einheitskreis landen, also einen Abstand von höchstens 1 zum Ursprung des Koordinatensystems haben, und berechnet daraus einen Näherungswert für π nach der unten angegebenen Formel. Das Programm enthält **fünf** Fehler, welche ein korrektes Ergebnis oder gar das Kompilieren des Programms verhindern. Finden Sie die Fehler, markieren Sie sie im Quellcode und erläutern Sie sie kurz im freien Bereich unter dem Progamm. Korrigieren Sie anschließend die Fehler indem Sie die korrekte Zeile aufschreiben.



The following program approximates the value of the constant π by generating random points in a two-dimensional coordinate system whose coordinates are between -1 and 1. It then counts how many of the points end up in the unit square. The distance of such points to the coordinate system's origin is at most 1. The number of points is used to calculate an approximate value for π according to the formula given below. The program contains **five** errors that prevent it from computing the correct result or even from compiling in the first place. Find the errors, mark them in the source code, and explain the issue in the space below. Then, correct the errors by writing down the corrected line.

$$\frac{\text{points within circle}}{\text{total number of points}} = \frac{\pi}{4}$$

```
1  import acm.graphics.GPoint;
2  import acm.program.ConsoleProgram;
3  import acm.util.RandomGenerator;
4
5  public class PiApproximator extends ConsoleProgram {
6    private static final int POINTS = 50_000;
7    private RandomGenerator rgen = RandomGenerator.getInstance();
8
9    public void run() {
10     int pointsInsideCircle = POINTS;
11
12     int i = 0;
13     while (i < POINTS) {
14       if (isInCircle(randomPoint())) {
15         pointsInsideCircle++;
16       }
17     }
18
19     println(pointsInsideCircle / POINTS * 4);
20   }
21
22   public GPoint randomPoint() {
23     double x = rgen.nextDouble() * (rgen.nextBoolean() ? 1 : -1);
24     double y = rgen.nextDouble() * (rgen.nextBoolean() ? 1 : -1);
25     return null;
26   }
27
28   public void isInCircle(GPoint unitPoint) {
29     return Math.sqrt(unitPoint.getX() * unitPoint.getX()
30         + unitPoint.getY() * unitPoint.getY()) < 1;
31   }
32 }
```

Answer: *9:* `... = 0;`
13: `i++ < POINTS`
19: `(double)pointsInsideCircle`
25: `return new GPoint(x, y);`

28: `boolean isInCircle`