Name: *Matrikelnummer:*

Programmierung

Klausur 2 (Wintersemester 2015/16)

8. April 2016

Hinweise (*English translation below*)

- **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 12.04.2016 zwischen 16 und 17 Uhr in CAP 4, Raum 1110, 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 April 12th 2016 between 4pm and 5pm in CAP 4, room 1110.

Aufgabe 1 (15 Punkte)

1. Was ist eine Definition von *Software Engineering*?
   What is a definition of *Software Engineering*?

2. Wie unterscheidet sich ein *Interpretierer* von einem *Übersetzer*?
   How does an *interpreter* differ from a *compiler*?

3. Was ist ein *Term*?
   What is a *term*?

4. Was ist eine *Zuweisung*?
   What is an *assignment*?

5. Was sind *Ausdrucksanweisungen*?
   What are *expression statements*?

6. Was ist ein *Scope*?
   What is a *scope*?

7. Welches Problem ist mit *Dangling Else* gemeint?
   What is the *dangling else problem*?

8. Was ist eine *Methode*?
   What is a *method*?

9. Welche vier Elemente kann der *body* einer Klasse enthalten?
   Which four elements may a class body contain?

10. Was sind *lokale Variablen*, *Instanzvariablen* und *Klassenvariablen*?
    What are *local/instance/class variables*?

11. Was bedeuten `&`, `|`, , `^` für *Integers*?
    What do `&`, `|`, , `^` mean for integers?

12. Was ist *Rekursion*?

    What is *recursion*?

13. Was ist *Character*-Arithmetik?

    What is *character arithmetic*?

14. Welche vier *Debugging*-Strategien kennen Sie?

    Which four *debugging strategies* do you know?

15. Wie werden *Exceptions* gefangen und bearbeitet?

    How are *exceptions* caught and handled?

Aufgabe 2 (8 Punkte)

1. (4 Punkte) 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 / 4 - 1 / 4                    _____________________________

   (char)('T' - 1)                  _____________________________

   "300" + 0x3 * 11                 _____________________________

   'I' - 'A'                        _____________________________
   ```

2. (4 Punkte) 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. Benutzen Sie für Zahlenwerte keine negativen 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. Do not use negative values for numerals.

   ```
   (char)(('9' - '6') + ___________ )                 'X'

   65536 - ___________                                ERROR

   'B' + ___________ + (___________ + 000)            "BFG9000"

   1E+___________+___________                         4.0
   ```

Aufgabe 3 (15 Punkte)

1. (6 Punkte) Gegeben sei das untenstehende Programm, welches ein Array deklariert und initialisiert. Geben Sie in der darunter dafür vorgesehenen Grafik den Inhalt des Arrays an direkt bevor die `run()`-Methode terminiert.

   The following program declares and initializes an array. Fill in the graphic below with the values of the array's cells just before the `run()` method terminates.

```
import acm.program.ConsoleProgram;

public class Test extends ConsoleProgram {
  public void run() {
    int[] arr = new int[13];
    for (int i = 1; i <= 12; i++) {
      for (int j = i; j > 0; j--) {
        arr[j] += j;
      }
    }
  }
}
```

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

2. (9 Punkte) Sie finden im Folgenden drei Java-Methoden, die einen Wert berechnen und zurückgeben, sowie fünf Java-Ausdrücke. Ordnen Sie jede Methode genau dem Ausdruck zu, dessen Wert sie berechnet.

   Below are three Java methods that compute and return a value as well as five Java expressions. Assign each method to the expression that computes the same value as the method.

```
int method1(int x, int y) {
  while (x > 0 && y > 0) {
    int f = y - 1;
    if (x > f) {
      x = x + (f + 1) * -1;
    } else {
      return x;
    }
  }
  return 0;
}

boolean method2(int x, int y) {
  boolean z = x >= 0 && x > 0;
  boolean w = z && !(y <= 0);
  boolean v = x < y;
  boolean u = x - y > 0;
  return w ? v && u : false;
}

boolean method3(boolean x, boolean y) {
  boolean a = x || y;
  boolean f = x ? false : true;
  boolean g = !y ? true : false;
  return a && (f || g);
}
```

a) `x > 0 && y > 0 ? x / y : 0`

b) `x != y`

c) `x > 0 && y > 0 ? x % y : 0`

d) `(x || y) || (!x || !y)`

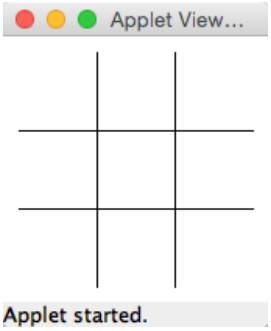
e) `false`

method1: _____

method2: _____

method3: _____

Aufgabe 4 (30 Punkte) Im Folgenden finden sie ein Programm, welches ein Tic Tac Toe-Spielfeld zeichnet wie rechts zu sehen. Erweitern Sie das Programm so, dass die Spieler mit der Maus in die Felder klicken und diese damit markieren können. Spieler 1 soll beginnen und seine Felder mit einem aus zwei Linien zusammengesetzten X markieren, Spieler 2 soll zur Markierung einen Kreis benutzen. Sie brauchen nicht sicherzustellen, dass ein Marker nur auf einem freien Feld platziert werden kann oder dass das Programm erkennt, wenn ein Spieler gewonnen hat. Insgesamt sollen 9 Züge gespielt werden. Sie dürfen lediglich neuen Code hinzufügen, nicht aber bereits vorhandenen Code verändern. Um die volle Punktzahl zu erreichen denken Sie daran, Ihren Code zu kommentieren und benutzte Klassen zu importieren wenn nötig.



Below is a program which draws a tic tac toe board as seen on the right. Extend the program such that the players can click the board's cells with the mouse to place their marker there. Player 1 starts and marks their cells with an X consisting of two lines, player 2 marks their cells with a circle. You do not need to ensure that a marker can only be placed on a free cell and that the program checks if someone has won. The game should end after 9 moves. You are only allowed to add new code, not to change existing code. To get full credit, remember to comment your code and import the classes you use, if necessary.

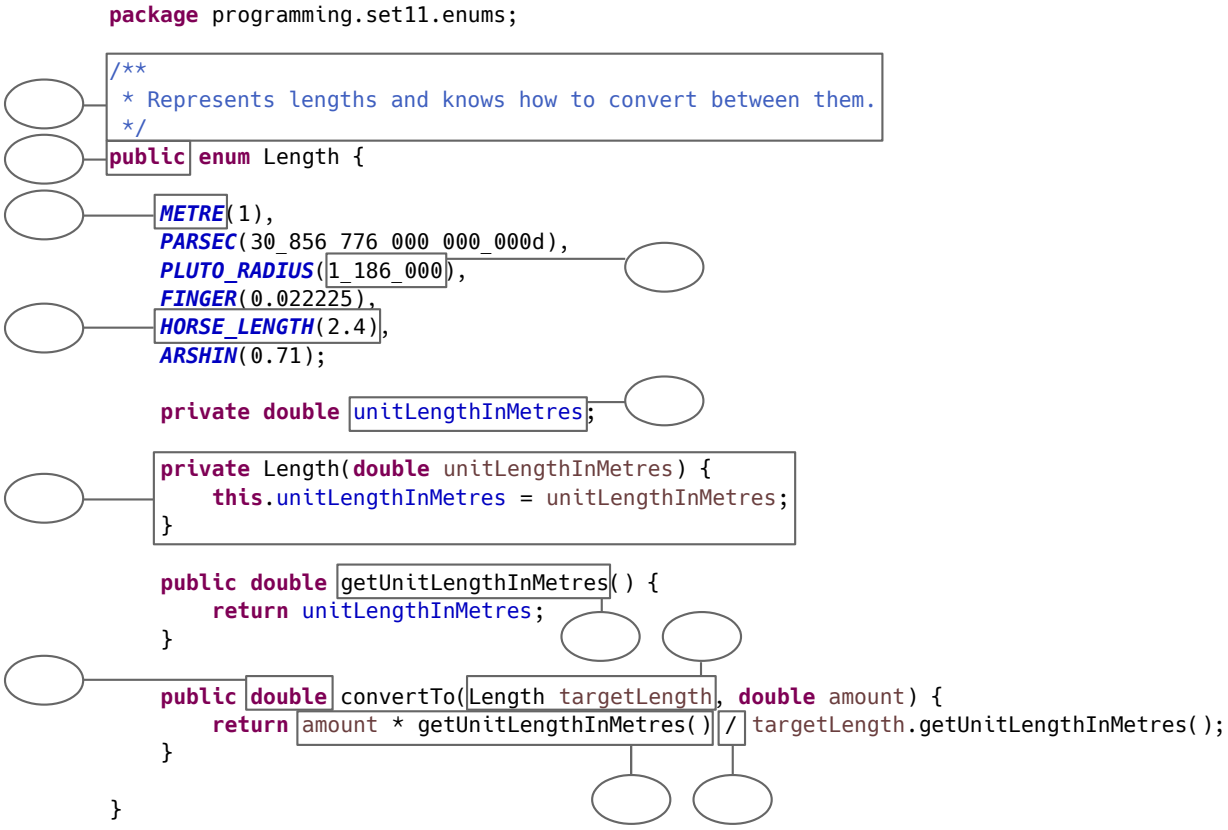
```
import acm.program.GraphicsProgram;




public class InteractiveTicTacToeBoard extends GraphicsProgram {
  /** Space between window border and tic tac toe board. */
  private static final int BORDER = 10;
  /** Width / height of a single tic tac toe board cell. */
  private static final int CELL_SIZE = 50;









  public void run() {
    drawBoard();









  }
```

```
  /**
   * Draws the Tic Tac Toe board. Do not modify this method.
   */
  private void drawBoard() {
    // Draw vertical lines
    for (int i = 1; i <= 2; i++) {
      int x = BORDER + i * CELL_SIZE;
      add(new GLine(x, BORDER, x, BORDER + 3 * CELL_SIZE));
    }

    // Draw horizontal lines
    for (int i = 1; i <= 2; i++) {
      int y = BORDER + i * CELL_SIZE;
      add(new GLine(BORDER, y, BORDER + 3 * CELL_SIZE, y));
    }
  }









}
```

Aufgabe 5 (12 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. Pro Feld können mehrere Zahlen zutreffen. Es müssen nicht alle Begriffe Verwendung finden.

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.

1. Argument
2. Assignment
3. Class variable
4. Constructor
5. Constructor call
6. Enumeration constant
7. Expression
8. Instance variable
9. Javadoc Comment
10. Local variable
11. Method name
12. Operator
13. Package declaration
14. Parameter
15. Statement
16. Type
17. Type parameter
18. Visibility modifier



Aufgabe 6 (10 Punkte) Die Funktion $facSum(x)$ für $x \in \mathbb{N}$ ist für $x \geq 0$ wie unten angegeben definiert. Implementieren Sie `facSum` und geben Sie an, was nach Starten des Programms auf der Konsole ausgegeben wird. Sie dürfen für die Implementierung weitere private Methoden hinzufügen. Um die volle Punktzahl zu erreichen denken Sie daran, Ihren Code zu kommentieren und benutzte Klassen zu importieren wenn nötig.
The function $facSum(x)$ for $x \in \mathbb{N}$ is defined below for $x \geq 0$. Implement the `facSum` method and state the console output generated by the program. You are allowed to add further private methods to help implement `facSum`. To get full credit, remember to comment your code and import the classes you use, if necessary.

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

```
public class FacSum {
  /**
   * The program's main entry point.
   *
   * @param args command-line arguments.
   */
  public static void main(String[] args) {
    System.out.println(facSum(6));
  }

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









  }
}
```

Aufgabe 7 (10 Punkte)

1. (6 Punkte) Gegeben ist das folgende Programm. Geben Sie an, welche Werte sich in den Variablen `a`, `b`, `c`, `d`, `s1` und `s2` der Klasse `Mystery` befinden, **nachdem** die letzte Anweisung der `main`-Methode ausgeführt wurde.

   Consider the following program. Write down the values of the variables `a`, `b`, `c`, `d`, `s1` und `s2` of class `Mystery` **after** the execution of the last statement in the `main` method.

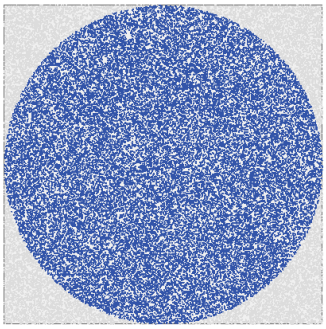
```
1  public class Mystery {
2    private static int s1 = 0;
3    private static int s2 = 100;
4    private static int a, b, c, d;
5
6    Mystery(double x, double y) {
7      a = mystify(x);
8      b = mystify(y);
9      this.s2 = a + b;
10   }
11
12   private static int mystify(double m) {
13     s1 = s2;
14     return (int) (s2 * m) + s1;
15   }
16
17   public static void main(String[] args) {
18     Mystery myMystery = new Mystery(0, 1);
19     c = myMystery.s1;
20     d = myMystery.s2;
21     myMystery = new Mystery(1, 2.5);
22   }
23 }
```

2. (4 Punkte) Wenn Sie das o.a. Programm in den Java-Editor Ihrer Eclipse-Umgebung eingeben, weist ein Hinweis auf die Warnung „`The static field Mystery.s2 should be accessed in a static way`" in Zeile 9 hin.
   a) Erklären Sie *kurz* die Bedeutung der Warnung.
   b) Was bedeutet **`static`**?
   c) Schreiben Sie eine äquivalente Anweisung so auf, dass sie keine Warnung erzeugt.
   d) In welchen Zeilen würde ebenfalls eine ähnliche Warnung angezeigt werden?

   The Java editor of your Eclipse environment will display the warning "`The static field Mystery.s2 should be accessed in a static way`" for line 9. a) *Briefly* explain the meaning of this warning. b) What is the meaning of **`static`**? c) Write down an equivalent statement that does not create a warning. d) Which lines would produce a similar warning?

Name: *Matrikelnummer:*

Aufgabe 8 (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 (falls dem so ist, ist der Abstand eines Punktes zum Ursprung des Koordinatensystems höchstens 1) 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 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).



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 (if that is the case, the distance between a point and the origin of the coordinate system is at most 1) and uses that number 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) and correct them (write 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 }
```