Name: *Matrikelnummer:*

Programmierung

Klausur 1 (Wintersemester 2015/16) 21. März 2015

Answer: *With Solutions*

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 24.03.2016 zwischen 16 und 18 Uhr in CAP 4, Raum 1110, eingesehen werden. Die exakte Zeit für Sie hängt von Ihrer Rechnerübung ab. Bitte schauen Sie dafür in's iLearn.

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 24th 2016 between 4pm and 6pm in CAP 4, room 1110. The exact time you should show up at depends on which practical class you were assigned to during the semester. The course description in the iLearn system contains further details.

Aufgabe 1 (15 Punkte)

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

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

   Answer: *Objects are instances of classes, classes are templates for objects*

2. 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`

   Answer: *b) (floating-point), e) (integer).*

3. Welche `for`-Kontrollzeile würden Sie für die nachfolgenden Aufgaben verwenden:

   What `for` loop control line would you use in each of the following tasks:

   a) Von 10 nach 1 herunterzählen.

      Counting from 10 down to 1.

      Answer: *for (int i = 10; i >= 1; i−−)*

   b) In Dreierschritten von 10 aufwärts zählen solange die Zahl höchstens zwei Stellen hat.

      Counting by threes starting at 10 as long as the number has at most two digits.

      Answer: *for (int i = 10; i < 100; i += 3)*

4. Nehmen Sie an, dass der Rumpf einer `while`-Schleife eine Anweisung enthält deren Ausführung die Bedingung der `while`-Schleife falsch werden lässt. Wird die Schleife dann unmittelbar nach der Ausführung dieser Anweisung abgebrochen oder wird die Schleife die aktuelle Iteration vollständig beenden?

   Suppose the body of a `while` loop contains a statement that, when executed, causes the condition for that `while` loop to become `false`. Does the loop terminate immediately at that point or does it complete the current cycle?

   Answer: *It completes the current cycle.*

5. Seien `d1` und `d2` als Variablen des Typs `int` deklariert. Können Sie die mehrfache Zuweisungsanweisung `d1 = d2 = rgen.nextInt(1, 6);` verwenden um das Werfen von zwei Würfeln zu simulieren? Begründen Sie kurz Ihre Antwort.

   Assuming that `d1` and `d2` have been declared as variables of type `int`, can you use the multiple assignment statement `d1 = d2 = rgen.nextInt(1, 6);` to simulate the process of rolling two dice? Briefly explain your answer.

   Answer:

6. Schreiben Sie eine Deklaration für die folgende *Array-Variable*: ein Array `inUse` das genau 16 Werte vom Typ `boolean` enthalten kann.

   Write a declaration that creates the following *array* variable: an array `inUse` that can hold exactly 16 values of type `boolean`

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

7. Was sind die zwei charakteristischen Eigenschaften eines *Arrays*?

   What are the two characteristic properties of an *array*?

   Answer: *Ordered, homogeneous.*

8. Welche drei Regionen unterscheiden wir im *Speicher*? Was wird wo abgespeichert?

   Which three areas do we distinguish in memory? What is stored where?

   Answer: *Static: static data, constants. Heap: Dynamic variables, "new". Stack: local variables, parameters.* Optional: *Stack in general is Last In, First Out (LIFO) data structure, corresponding to call chains: last method called is the first method returned from.*

9. Welche Regeln bestimmen die *Auswertungsreihenfolge* in Java-Ausdrücken?

   What are the rules governing the order of expression evaluation in Java?

   Answer: *a) precedence, b) association, c) direction (left-to-right).*

10. Wofür sollten *Assertions* verwendet werden, wofür nicht?

    When should you use *assertions*, when not?

    Answer: *Invariants, pre-/post-conditions; not for checking user input.*

11. Wofür steht MVC? Bitte geben das entsprechende Diagramm an, wie in der Vorlesung/den Folien angegeben.

    What does *MVC* stand for? Please provide the diagram, as used in class/on the slides.

    Answer: *Cycle with User - Controller - Model - View.*

12. Welches sind die drei Datenstrukturen welche in den *Java Collections* realisiert sind?

    What are the three data structures realized in the *Java Collections Framework*?

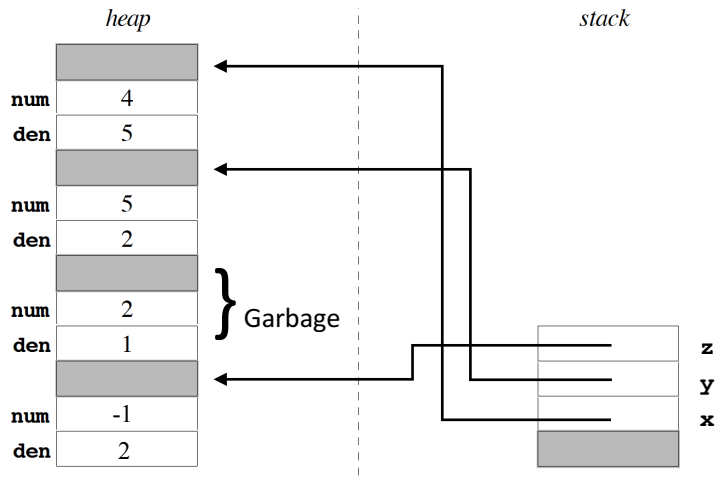
    Answer: *Lists, sets, maps.*

13. Zeichnen Sie ein *Stack-Heap Diagram* (unter Verwendung des *Pointer-Modells*) welches von der Ausführung des folgenden Codes bis zum `println` resultiert. Zur Erinnerung, Objekte der `Rational`-Klasse haben die Instanzvariablen `num` und `den`, und arithmetische Operationen kürzen `num` und `den` weitmöglichst. Welche Objekte auf dem Heap können durch den *Garbage Collector* gelöscht werden, wenn das `println` erreicht wird?

    Draw a stack-heap diagram (using the pointer model) that results from executing the following code just until the `println` statement. Recall that objects from the `Rational` class have the instance variables `num` and `den`, and that the arithmetic operations on them always reduce `num` and `den` to their lowest terms. Which objects on the heap are elegible for garbage collection when the `println` statement is reached?

```
public void run() {
  Rational x = new Rational(4, 5);
  Rational y = new Rational(5, 2);
  Rational z = x.multiply(y).subtract(y);
  println(x + " x " + y + " - " + y + " = " + z);
}
```

Answer:



Aufgabe 2 (10 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.

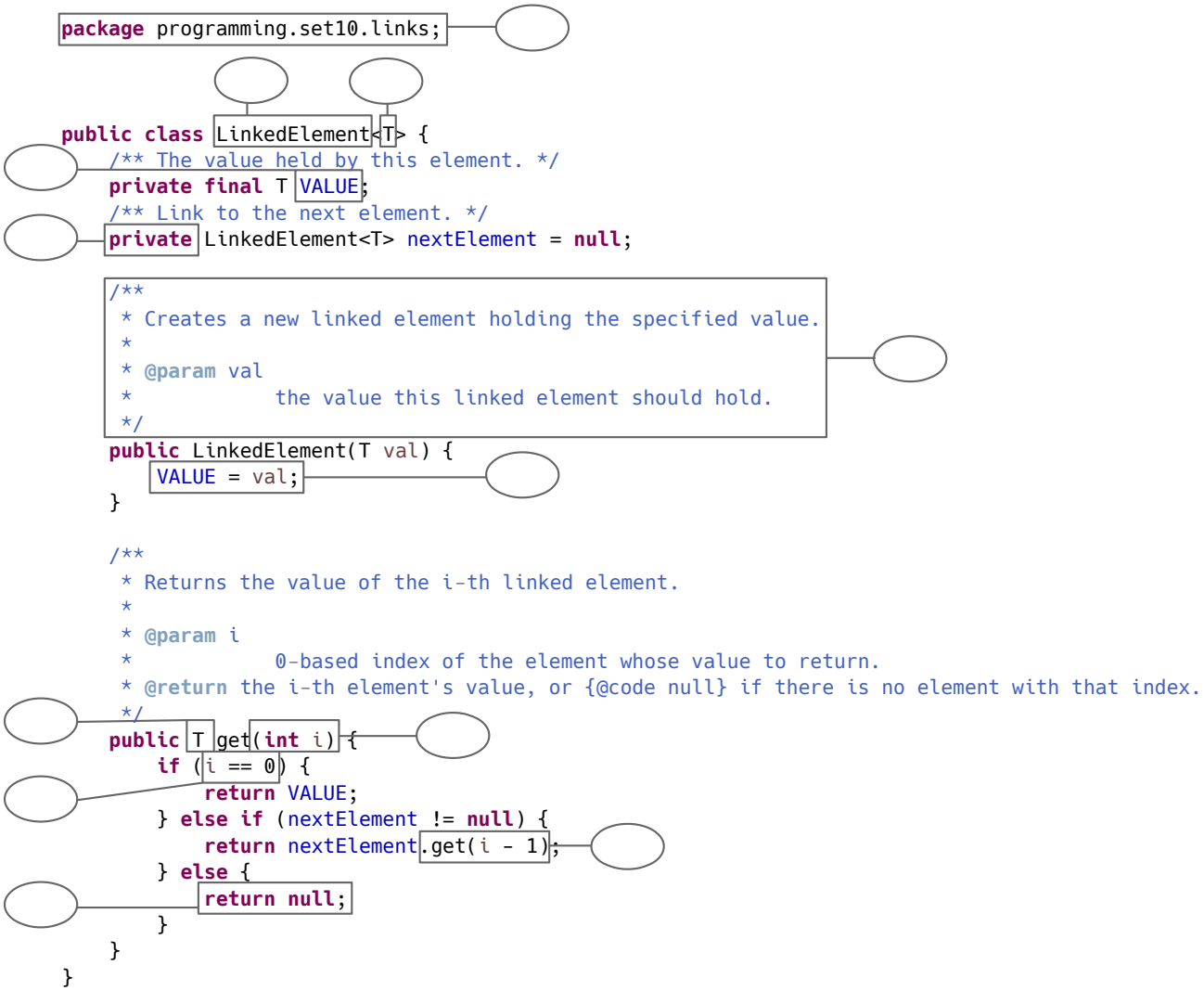
`'H' +` ***Answer:*** *`"A"` `+ 'L' + 9000*	`"HAL9000"`
`!!!` ***Answer:*** *true* `|| !!` ***Answer:*** *false*	`false`
Answer: *5f* `/ 10 * 3`	`1.5`
`(111 %` ***Answer:*** *100* `+ 1) /` ***Answer:*** *10f*	`1.2`
`false ==/* true */` ***Answer:*** *false*	`true`

Aufgabe 3 (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. 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



```java
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 4 (13 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 (8 Punkte)

1. Gegeben ist das folgende Program, welches vier Ausgaben auf der Konsole produziert. Geben Sie an, welche Ausgaben in welcher Reihenfolge gemacht werden.

   The following program prints four outputs to the console. Write them down in the correct order in the space below.

```
public class Mystery {
    public int x;
    private static int y = 0;

    Mystery(int x, int y) {
        this.x = x;
        this.y = y + this.x;
    }

    public static void main(String[] args) {
        Mystery mystery1 = new Mystery(3, 9);
        Mystery mystery2 = new Mystery(2, 4);

        System.out.println(mystery1.x);
        System.out.println(mystery1.y);
        System.out.println(mystery2.x);
        System.out.println(mystery2.y);
    }
}
```

   Output:

   Answer: *3*
   6
   2
   6

2. Erklären Sie *kurz* die Java-Schlüsselwörter **`private`**, **`public`**, **`static`** und **`this`**.

   Briefly explain the meaning of the Java keywords **`private`**, **`public`**, **`static`**, and **`this`**.

   Answer:
   public: globally visible
   private: only visible in class
   static: associated with class (and not object)
   this: the actual object

Aufgabe 6 (30 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.

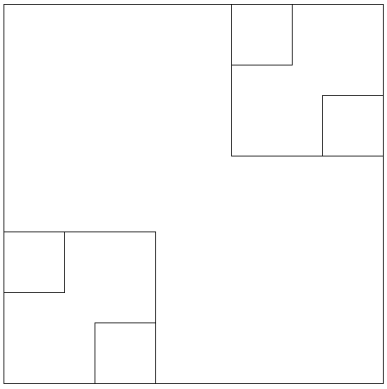
Die resultierende Zeichnung sollte wie in den beiden Beispielen unten aussehen.

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.

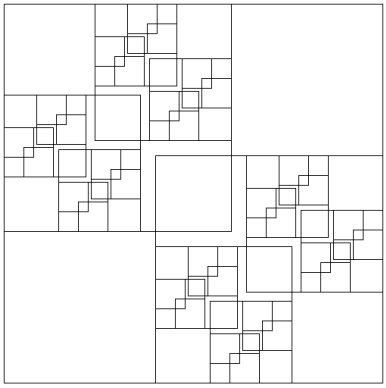
The program first asks the user for a value for d, which must be between 1 and `MAX_DEPTH_LIMIT`. If the user does not provide a number in that range, the program keeps asking the user until the user enters a suitable number.

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 7 (12 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");
}
```

Answer: *1*

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

Answer: *99999*

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

Answer: *no solution*

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

Answer: *4 or 5*

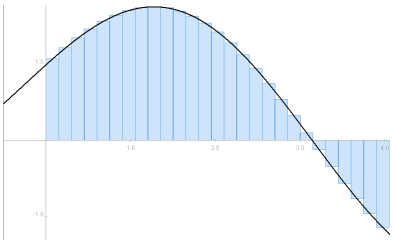
```
int x = readInt();
if (x > 0) {
  x /= 10;
  if (x == 0) {
    println("success");
  }
  if (10 / x == 0) {
    println("failure");
  }
}
```

Answer: *no solution*

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

Answer: *4012*

Aufgabe 8 (10 Bonus Punkte) Das folgende Programm berechnet die Fläche zwischen einer Funktion $f(x)$ und der x-Achse indem gemittelte Rechtecke einer vorgegebenen Breite die Fläche approximieren. Das Programm enthält **fünf** Fehler, welche ein korrektes Ergebnis 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).



(c) Example approximation

The following program calculates the area bounded by the graph of the function $f(x)$ and the x axis by calculating average rectangles of a given width to approximate the area. However, the program contains **five** errors that prevents it from computing the correct result. 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).

```
1  import acm.program.ConsoleProgram;
2
3  public class FunctionArea extends ConsoleProgram {
4
5    public void run() {
6      println(approxFunctionArea(0, 100, 15));
7    }
8
9    public double f(double x) {
10     return Math.sin(x) + Math.cos(0.5 * x);
11   }
12
13   public double avgRectHeight(double x, double width) {
14     if (width > 0) {
15       return 0;
16     }
17     double leftVal = f(x);
18     double rightVal = (int) f(x + width);
19     return (leftVal + rightVal) / 0.5;
20   }
21
22   public double approxFunctionArea(double left, double right,
23           double rectWidth) {
24
25     double area = 0.0;
26     for (double x = left; x < right; x += rectWidth) {
27       double width = Math.min(rectWidth, right + x);
28       area = avgRectHeight(x, width) * width;
29     }
30     return area;
31   }
32 }
```

Answer: *14:* `if (width < 0){`
18: `double rightVal = (int)f(x + width);`
19: `return (leftVal + rightVal)/ 2.0;`
26: `double width = Math.min(rectWidth, right - x);`
27: `area += avgRectHeight(x, width)* width;`