Name: *Matrikelnummer:*

Programmierung

Klausur 1 (Wintersemester 2016/17) 15. März 2017

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 20.03.2017 zwischen 16 und 18 Uhr in CAP 4, Raum 1304a, 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 20 2017 between 4pm and 6pm in CAP 4, room 1304a. 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 (16 Punkte)

1. Welche Arten von *Programmierfehler* unterscheiden wir?

   What types of *programming error* do we distinguish? **Answer:** *Syntax Errors, Bugs.*

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 *Präzedenz*?

   What is *precedence*? **Answer:** *Precedence rules determine priority of operators for order of evaluation in expressions with different operators.*

4. Welche logische Operationen können mit boolschen Datentypen in Java durchgeführt werden?

   Which logical operations can be applied to boolean data types in Java? **Answer:** *Can apply logical operations (and, not, or, xor).*

5. Was sind *iterative Anweisungen* in Java?

   What are *iterative statements* in Java? **Answer:** *for, while, do-while.*

6. Warum sollten wir in bedingten Verzweigungen zusammengesetzte Anweisungen anstatt einfacher Anweisungen verwenden?

   In conditionals, why should we use compound statements instead of simple statements? **Answer:** *To avoid the* dangling else problem.

7. Was ist die Motivation für Methoden?

   What is the motivation for having methods? **Answer:** *Code re-use, decomposition, information hiding.*

8. Welche vier Elemente kann der *body* einer Klasse laut Vorlesung enthalten?

   Which four elements may a class body contain according to the lecture? **Answer:** *Constructors, methods, named constants, instance variables.*

9. Was ist ein *mehrdimensionales Array*?

   What is a *multi-dimensional array*? **Answer:** *Array with multiple indices.*

10. Was ist eine *generische Klasse*?

    What is a *generic class*? **Answer:** *Class parameterized over types.*

11. Wie unterscheiden sich *ArrayList* und *LinkedList*? Wann sollte man was verwenden?

    How do *ArrayLists* and *LinkedLists* differ? When should you use which one? **Answer:** *Differ in implementation of the list. Array accessible over index, LinkedList over linked elements. Index/Ordered access better, resize bad in ArrayLists. Add in between, remove, resize better in linked list.*

12. Was ist das Konzept eines Strings?

    What is a *string*, conceptually? **Answer:** *Ordered collection of characters.*

13. Sei `s` ein String. Warum ergibt `if (s.equals(null)) { ... }` keinen Sinn?.

    Let `s` be a string. Why is `if (s.equals(null)) { ... }` unreasonable?. **Answer:** *if s were null, this would cause a null pointer exception*

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

    Which four *debugging strategies* do you know? **Answer:** *println, debugger, unit testing, assertions.*

15. Wie werden *Exceptions* gefangen und bearbeitet?

    How are *exceptions* caught and handled? **Answer:** *With try/catch block; exceptions are propagated method call chain up to next enclosing handler; application must handle exceptions outside of RuntimeException hierarchy.*

16. Wofür steht MVC? Bitte geben Sie 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.*

Aufgabe 2 (12 Punkte)

1. (4 Punkte) Im Folgenden sehen Sie eine Liste von Ausdrücken. Nehmen Sie in jeder Zeile an, dass `i` den Typ **`int`** und zu Beginn den Wert `0` hat. 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. Assume in each line that `i` is of the type **`int`** and its value is set to `0` at the beginning. Compute the value of each expression as interpreted by Java. If an error occurs during any of these evaluations, write "ERROR" on that line.

`50 % 4 - 2 / 4`	***Answer:*** *2*
`(char)('S' + 'T' / 'S')`	***Answer:*** *'T'*
`++i + 2 * ++i`	***Answer:*** *5*
`3 << 3`	***Answer:*** *24*

2. (4 Punkte) Ergänzen Sie die folgenden Expressions mit Klammern, so dass das angegebene Ergebnis mit der Auswertung in Java überein stimmt. Nehmen Sie in jeder Zeile an, dass `i` den Typ **`int`** und zu Beginn den Wert `0` hat.

   Add parentheses to the following expressions such that the Java evaluation of each expression matches the specified result. Assume in each line that `i` is of the type **`int`** and its value is set to `0` at the beginning.

`2 + 3 * 4 / i + 5`	`0`	***Answer:*** *(2 + 3) * (4 / (i + 5))*
`1 + 1E100 - 1E100`	`1`	***Answer:*** *1 + (1E100 - 1E100)*
`"" + ! true & false | true`	`"false"`	***Answer:*** *"" + !(true & false | true)*
`15 & 8 | 1 << 2`	`4`	***Answer:*** *12 & ((8 | 1) « 2)*

3. (4 Punkte) Schreiben Sie unter jede der nachfolgenden Zeilen, um was für ein Java-Konstrukt es sich nach der gängigen Java-Namenskonvention handeln müsste.

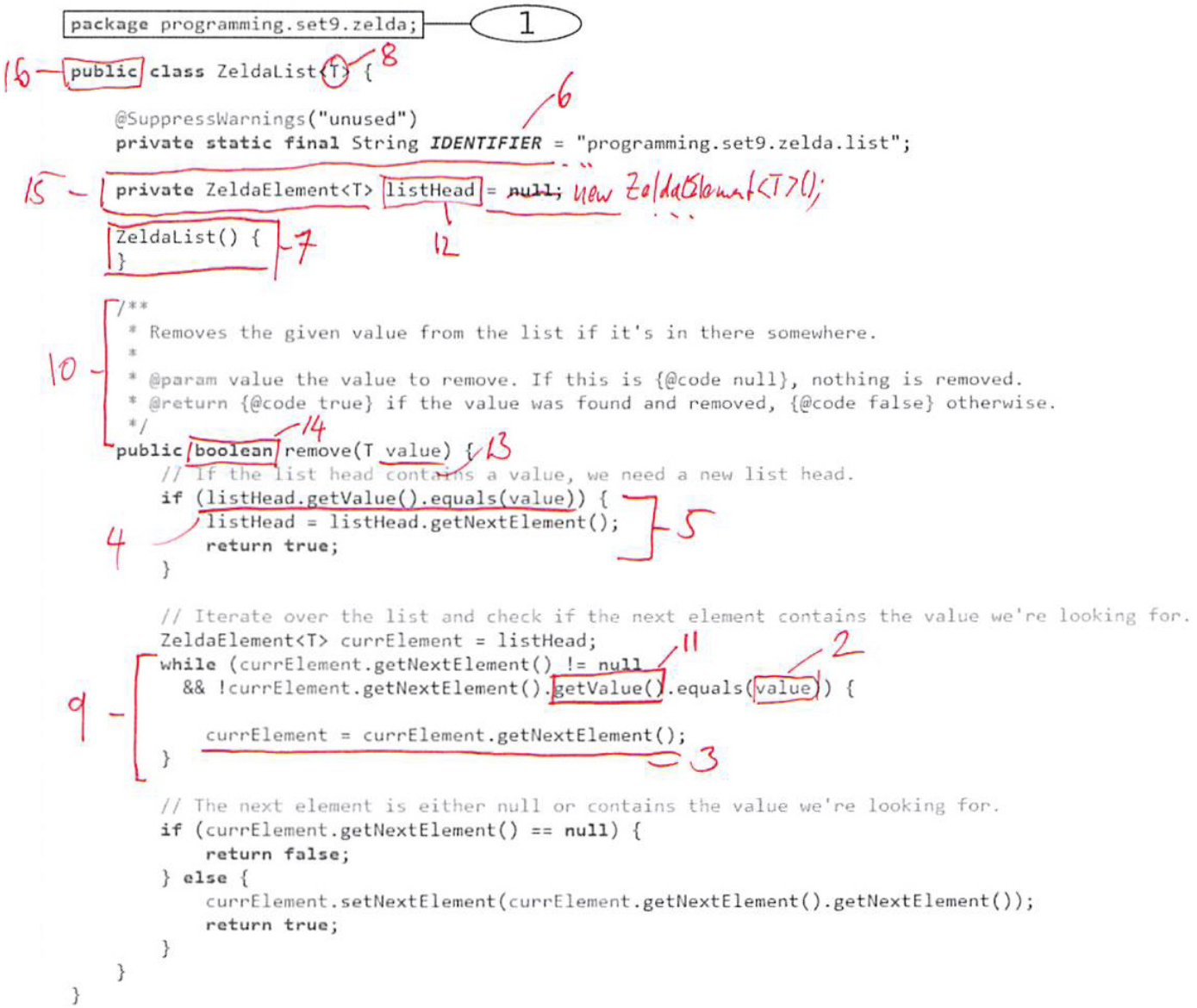
   For each line, write down what kind of Java construct is meant following the common Java naming convention.

   - `fooBar`
   - `fooBar(...)`
   - `FOO_BAR`
   - `FooBar(...)`

Aufgabe 3 (8 Punkte) Im Folgenden sehen Sie ein Stück Java-Code. Finden Sie für jeden der nachfolgenden Begriffe ein Vorkommen in dem Code. Markieren Sie die Stelle im Code entsprechend mit der Nummer des Begriffs (siehe 1 als Beispiel). Es reicht aus, pro Begriff **ein Vorkommen im Code zu markieren**.

Consider the following piece of Java code. For each of the listed terms, annotate an appropriate part of code with the number of the term (see term 1 as example). It is sufficient to mark **one example per term**. ***Answer:***

1. Package Declaration
2. Argument
3. Assignment
4. Boolean expression
5. Compound Statement
6. Constant
7. Constructor
8. Generic Type Parameter
9. Iterative statement
10. Javadoc comment
11. Method call
12. Object
13. Parameter
14. Return type
15. Instance variable
16. Visibility



```
package programming.set9.zelda;

public class ZeldaList<T> {

    @SuppressWarnings("unused")
    private static final String IDENTIFIER = "programming.set9.zelda.list";

    private ZeldaElement<T> listHead = null;

    ZeldaList() {
    }

    /**
     * Removes the given value from the list if it's in there somewhere.
     *
     * @param value the value to remove. If this is {@code null}, nothing is removed.
     * @return {@code true} if the value was found and removed, {@code false} otherwise.
     */
    public boolean remove(T value) {
        // If the list head contains a value, we need a new list head.
        if (listHead.getValue().equals(value)) {
            listHead = listHead.getNextElement();
            return true;
        }

        // Iterate over the list and check if the next element contains the value we're looking for.
        ZeldaElement<T> currElement = listHead;
        while (currElement.getNextElement() != null
          && !currElement.getNextElement().getValue().equals(value)) {

            currElement = currElement.getNextElement();
        }

        // The next element is either null or contains the value we're looking for.
        if (currElement.getNextElement() == null) {
            return false;
        } else {
            currElement.setNextElement(currElement.getNextElement().getNextElement());
            return true;
        }
    }
}
```

Aufgabe 4 (10 Punkte) Die Funktion *facSum*(n) für $n \in \mathbb{N}$ sei für $n \geq 0$ wie unten angegeben definiert. Implementieren Sie die methode `facSum`, welche *facSum* berechnet, und geben Sie an, was nach Starten des Programms auf der Konsole ausgegeben wird. Sie dürfen für die Implementierung weitere private Methoden hinzufügen. Sie müssen Ihren Code nicht kommentieren. Achten Sie darauf, dass Ihre Implementierung die Javadoc-Kommentare umsetzt.

The function *facSum*(n) for $n \in \mathbb{N}$ is defined below for $n \geq 0$. Implement the method `facSum`, which calculates *facSum* and state the console output generated by the program. You are allowed to add further private methods to help implement `facSum`. You don't have to comment your code. Make sure that you consider the Javadoc comments.

$$facSum(n) = \begin{cases} n! + facSum(n-1) & \text{if } n > 0 \\ 1 & \text{otherwise (sonst)} \end{cases}$$

```
public class FacSum {
  /**
   * The program's main entry point.
   *
   * @param args command-line arguments.
   */
  public static void main(String[] args) {
    for (int i = 0; i < 6; i++)
        System.out.println(facSum(i));
  }

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

Answer:

```
public class FacSum3 {

    public static void main(String[] args) {
        for(int i = 0; i < 6; i++)
        System.out.println(facSum(i));
    }

    private static int facSum(int x) {
        if (x <= 0) return 1;
        else return fac(x) + facSum(x - 1);
    }

    private static int fac(int x) {
        if (x < 1) return 1;
        else return x * fac(x - 1);
    }

}
```

Answer: *1 2 4 10 34 154*

Aufgabe 5 (14 Punkte) Ein Kellerspeicher (*Stack*) ist eine Datenstruktur die nach dem *first-in last-out*-Prinzip funktioniert. Dem Speicher können beliebeig viele Elemente hinzugefügt (*push*) und wieder entnommen (*pop*) werden. Beim Entnehmen wird stets das zuletzt hinzugefügte Element zurückgegeben und dann wieder aus der Struktur entfernt. Beispiel: Ein Aufruf von `push(4)`, `push(3)`, `pop()` würde also 3 zurückgeben. Ein erneutes `pop()` würde danach 4 zurückgeben.

Vervollständigen Sie die Klasse `Stack` um einen Kellerspeicher für generische Typen bereitzustellen. Liefern Sie `null` zurück, wenn `pop()` versucht, ein Element von einem leeren Speicher zurückzugeben. Sie dürfen für Ihre Lösung keine Java-Collections Klassen (wie z.B. *LinkedList* oder *ArrayList*) verwenden. Sie dürfen aber neue nicht-*public* Klassen erstellen und verwenden.

Sie müssen Ihren Code nicht kommentieren.

A stack is a data structure that works according to the *first-in last-out* principle. You can add items with a *push* operation and retrieve items with *pop*. *pop* will always return the item that was pushed last on the stack. The item is then removed. For example: `push(4)`, `push(3)`, `pop()` will return 3. Afterwards, another `pop()` operation will return 4.

Complete the class `Stack` to provide a stack for generic types. Return `null` if `pop()` tries to return from an empty stack. You are not allowed to use Java collection classes, such as *LinkedList* or *ArrayList*. However, you are allowed to create and use new non-*public* classes.

You do not have to comment your code.

Answer:

```
public class Stack<T> {

    private Item<T> head = null;

    public void push(T item) {
        head = new Item<T>(item, head);
    }

    public T pop() {
        T object = head.getObject();
        head = head.getNext();
        return object;
    }

}

class Item<T> {

    private T object;
    private Item<T> next;

    Item(T object, Item<T> next) {
        this.object = object;
        this.next = next;
    }

    T getObject() {
        return object;
    }

    Item<T> getNext() {
        return next;
    }

}
```

Aufgabe 6 (30 Punkte) Schreiben Sie ein Programm in einem Package `diagrams`, welches für ein Feld von doubles ein Diagramm erzeugt. Der Benutzer soll über interaktive Buttons zwischen Balken- und Kreisdiagramm auswählen und beliebig wechseln dürfen. Im Balkendiagramm soll der Balken mit dem höchsten Wert die gesamte zur Verfügung stehende Höhe ausnutzen. Beide Diagrammformen sollen allerdings ein `BORDER_SPACING`, initialisiert mit 10, enthalten.

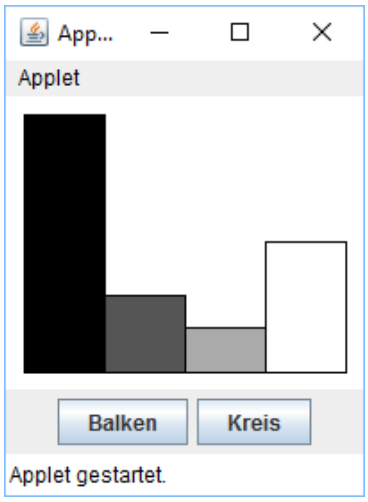
Das Feld mit den Daten kann über eine weitere Klasse `ValueLoader` im gleichen Package geladen werden. Diese Klasse wird Ihnen zur Verfügung gestellt und muss nicht noch implementiert werden. Die Signatur, der Methode, die die Daten zur Verfügung stellt, sieht wie folgt aus: `public static double[] getValues()`. Die Daten werden sich während der Ausführung nicht ändern und müssen nur einmal am Start eingelesen werden. Die Werte sind stets ≥ 0 und ergeben summiert 100. Im Beispiel werden 4 Werte gezeigt. Ihr Programm sollte mit Feldern belieber Größe umgehen können.

Färben Sie die Balken/Kreissegmente gleichmäßig von schwarz (RGB 0,0,0) nach weiß (RGB 255,255,255). Sie müssen sich nicht um *resizing* des Fensters kümmern. Zu Beginn darf das Fenster leer sein. Orientieren Sie sich an den zwei folgenden Beispielbildern.

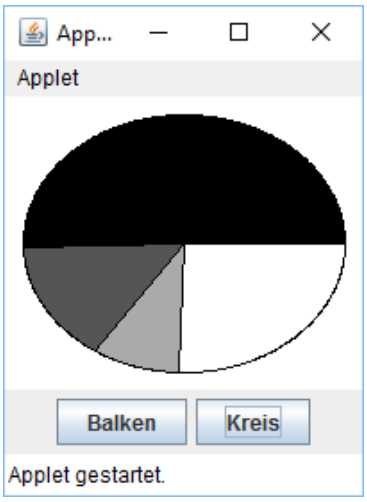
Write a program in a package `diagrams` that creates a diagram for an array of doubles. The user should be able to choose between bar and circle diagrams via interactive buttons. He should also be able to switch between them at any time. In the bar diagram, the bar with the highest value should use the whole height available. However, both diagram types should contain a `BORDER_SPACING` initialized with 10.

Another already finished class `ValueLoader` that resides in the same package can be used to retrieve the data. You don't have to implement `ValueLoader`. The signature of the method to use looks like this: `public static double[] getValues()`. The data will not change during execution. It is sufficient to load them once at startup. The values are always ≥ 0 and sum up to 100. The example shows 4 values. However, your program should be able to handle arbitrary many values.

Color the bars/circle segments gradually from black (RGB 0,0,0) to white (RGB 255,255,255). You do not need to consider frame *resizing*. The frame is allowed to be empty at program start. Use the two following figures as template.



(a) Balken



(b) Kreis

Für maximale Punktzahl muss Ihre Lösung ein vollständiges Programm sein, einschließlich sinnvoller Benennung aller Java-Konstrukte, Definition der in der Aufgabenstellung erwähnten Konstanten und Kommentaren. `Import` Statements dürfen weggelassen werden. Wenn Ihnen an einem Punkt die konkrete Syntax entfallen ist, dürfen Sie um Teilpunkte zu erhalten als Kommentar in Pseudocode aufschreiben, wie es weitergehen müsste.

To get maximum credit, your solution should be a complete program, including meaningful naming of all Java constructs, definitions of the constants referenced in the problem statement above, and comments. You may omit `import` statements. If you can't remember the concrete syntax for a specific task, you may write down pseudo code in comments to get partial points for describing how the code should be continued.

Useful constructors and static methods:

```
public GLine(double x0, double y0, double x1, double y1)
public GRect(double width, double height)
public GRect(double x, double y, double width, double height)
public GArc(double width, double height, double start, double sweep)
public GArc(double x, double y, double width, double height, double start, double sweep)
public GOval(double width, double height)
public GOval(double x, double y, double width, double height)
public Color(int r, int g, int b)
public static double GMath.toDegrees(double radians)
```

Ihre Lösung / *your solution*
Answer:

```
package diagrams;

import java.awt.Color;
import java.awt.event.ActionEvent;

import javax.swing.JButton;

import acm.graphics.GArc;
import acm.graphics.GMath;
import acm.graphics.GRect;
import acm.program.GraphicsProgram;
import acm.util.RandomGenerator;

/*
 * DiagramViewer loads data from {@link ValueLoader} and
 * presents it as bar or circle diagram.
 */
public class DiagramViewer extends GraphicsProgram {

    /** Bar button text */
    private static final String BARS_BUTTON_TEXT = "Balken";
    /** Circle button text */
    private static final String CIRCLE_BUTTON_TEXT = "Kreis";
    /** Border spacing in Pixel */
    private static final double BORDER_SPACING = 10;

    /** Diagram data storage */
    private double[] diagramData;

    /**
     * Initialize the buttons and action listeners.
     * Additionally, load the data.
     */
    @Override
    public void init() {
        add(new JButton(BARS_BUTTON_TEXT), SOUTH);
        add(new JButton(CIRCLE_BUTTON_TEXT), SOUTH);
        addActionListeners();
        diagramData = ValueLoader.getValues();
    }

    /**
     * Listener for actions.
     */
    @Override
    public void actionPerformed(ActionEvent e) {
        if (e.getActionCommand().equals(BARS_BUTTON_TEXT)) {
            // Switch to bar diagram.
            drawBars();
        } else if (e.getActionCommand().equals(CIRCLE_BUTTON_TEXT)) {
            // Switch to circle diagram.
            drawCircle();
        }
    }

    /**
```

```
     * Draw the bars.
     */
    private void drawBars() {
        // Remove what was drawn before.
        removeAll();
        // Set variables for convenience.
        double width = getWidth() - BORDER_SPACING * 2;
        double barWidth = width / diagramData.length;
        double height = getHeight() - BORDER_SPACING * 2;
        double highestBarValue = 0;

        // Calculate highest bar.
        for(double d : diagramData) {
            if (d > highestBarValue)
                highestBarValue = d;
        }
        // Draw bars with variable height and colors.
        for(int i = 0; i < diagramData.length; i++) {
            double barHeight = (diagramData[i] / highestBarValue) * height;
            GRect bar = new GRect(barWidth, barHeight);
            bar.setLocation(BORDER_SPACING + i * barWidth, getHeight() -
               BORDER_SPACING - barHeight);
            int grayValue = i * 255 / Math.max(diagramData.length-1, 1);
            bar.setFillColor(new Color(grayValue, grayValue, grayValue));
            bar.setFilled(true);
            add(bar);
        }
    }

    /**
     * Draw the circle segments.
     */
    private void drawCircle() {
        // Remove what was drawn before.
        removeAll();
        // Start of the circle.
        double startAngle = 0;
        // Draw circle segments with GArc.
        for(int i = 0; i < diagramData.length; i++) {
            // Angle uses degrees.
            double angle = GMath.toDegrees(diagramData[i] / 100.0 * 2 *
               Math.PI);
            GArc arc = new GArc(BORDER_SPACING,
                    BORDER_SPACING,
                    getWidth() - BORDER_SPACING * 2,
                    getHeight() - BORDER_SPACING * 2,
                    startAngle, angle);
            int grayValue = i * 255 / Math.max(diagramData.length-1, 1);
            arc.setFillColor(new Color(grayValue, grayValue, grayValue));
            arc.setFilled(true);
            add(arc);

            startAngle += angle;
        }
    }
}
```

Aufgabe 7 (10 Punkte) Gegeben sei das folgende Programm. Geben Sie an, welche Werte sich im Feld `m` von `foo` und `bar` befinden, nachdem die letzte Anweisung der `main`-Methode ausgeführt wurde. Streichen Sie Zellen durch, die sich außerhalb der Feldgrenzen befinden.

Consider the following program. Write down the values of array `m` of `foo` and `bar` after the execution of the last statement in the `main` method. Cross out cells that are out of the array bounds.

```
public class Mystery {
  private static int[] m;

  Mystery(int n) {
      m = new int[n];
  }

  private void mystify(int i) {
      m[i] = i + 1 + m[i - 1 >= 0 ? i - 1 : m.length - 1];
  }

  public static void main(String[] args) {
    Mystery foo = new Mystery(12);
    Mystery bar = new Mystery(6);
    for (int i = 0; i < 12; i++) {
        if (i < foo.m.length) foo.mystify(i);
        if (i < bar.m.length - 1) bar.mystify(i + 1);
    }
}
```

foo.m												
bar.m												
	0	1	2	3	4	5	6	7	8	9	10	11

Answer: *both: 1 3 6 10 15 21*
Answer: *if not static (half points):*
foo: 1 3 6 10 15 21 28 36 45 55 66 78
bar: 0 2 5 9 14 20

Aufgabe 8 (10 Bonus Punkte)

1. (3 Punkte) Wie lautet die Definition einer kontextfreien Grammatik?

   Write down the definition of a *context free grammar*.

   Answer: *A tuple of G = (V, Sigma, R, S) consisting of set of Variables V, an alphabet/set of terminals Sigma, a set of productions R which are a relation from V to (V union Sigma)*, and a start variable S in V.*

2. (4 Punkte) Spezifizieren Sie eine kontextfreie Grammatik, dessen Sprache aus balancierten runden und eckigen Klammern besteht. Zum Beispiel soll die Sprache (), [], ([]) enthalten, aber nicht [) oder ([)].

   Specify a context free grammar whose language consists of balanced parentheses and square brackets. E.g., the language should contain strings like (), [], ([]), but not [) or ([)].

   Answer: *G = (V, Sigma, R, S) with Variables V = S , Alphabet Sigma = (,), [,] , and productions R = S -> SS, S -> (S), S -> (), S -> [S], S -> []*

3. (3 Punkte) Zeigen Sie, dass [()()] in der Sprache, die Sie erstellt haben, enthalten ist.

   Prove that [()()] is in the language that you constructed.

   Answer: *S => [S] => [SS] => [()S] => [()()]*