Check und Prepare

Vorbereitende Übungen

TECHNISCHE UNIVERSITÄT DARMSTADT

Autoren: Nhan Huynh und Darya Nikitina
Fachbereich: Informatik
Übungsblatt: 03
Version: 20. November 2021
Semesterübergreifend

Dokumentation in Java: Mittels JavaDoc und den Tags @param und @return wollen wir nun im Java-Teil eine geeignete Dokumentation unserer Methoden vornehmen. Das JavaDoc können Sie automatisch vor jeder Methode mittels /** in einer integrierten Entwicklungsumgebung generieren, wenn Sie danach Enter drücken. Ein Beispiel wäre:

```
/**
 * This method accepts two real numbers that belong to a
 * vector and calculates the euclidean norm of the said
 * vector.
 *
 * @param x the first component of a two-dimensional vector (x, y)
 * @param y the second component of a two-dimensional vector (x, y)
 * @return the euclidean norm of the vector (x, y)
 */
double euclid2(float x, float y) {
  return Math.sqrt(x*x + y*y);
}
```

V1 Die Würfel sind gefallen!

★☆☆

Mit der Funktion `Math.random()` können Sie eine Zufallszahl im Bereich `0` (inklusive) und `1` (exklusive) erzeugen. Schreiben Sie nun eine Methode `void diceRoll()`.

Diese soll einen Würfelwurf simulieren und die gewürfelte Augenzahl auf der Konsole zurückgeben. Dabei soll der Würfel fair sein, das heißt alle Augenzahlen sollen mit identischer Wahrscheinlichkeit auftreten.

Hinweis Überlegen Sie sich, wie Sie die erzeugten Zahlen aus dem Intervall $[0, 1)$ auf die diskrete Menge $\{1, 2, 3, 4, 5, 6\}$ abbilden können. Mit der Funktion `Math.ceil()` können Sie zur nächst größeren, ganzen Zahl aufrunden.

Übungsblatt 03: Vorbereitende Übungen von Nhan Huynh und Darya Nikitina

V2 Schleifen

```
int x = 0;
while (Math.pow(x, 2) <= 1000) {
  x++;
}
System.out.println(x);
```

(1) Welche Funktion erfüllt der oben stehende Code?

(2) Ersetzen Sie die `while`-Schleife einmal durch eine `for`- und einmal durch eine `do-while`-Schleife, und zwar so, dass weiterhin die gleichen Schritte ausgeführt werden.

V3 Interfaces

Schreiben Sie ein Interface `I1` mit einer parameterlosen `void`-Methode `m1`. Nun schreiben Sie ein Interface `I2`, das von `I1` erbt. `I2` hat eine zusätzliche `void`-Methode `m2` mit einem `int`-Parameter `i`. Abschließend schreiben Sie eine Klasse `C1`, die das Interface `I2` implementiert. Die Klasse hat ein `int`-Attribut `number`. Beim Aufruf der Methode `m1` soll `number` auf `-1` gesetzt werden. Wird Methode `m2` aufgerufen, so soll `number` auf den übergebenen Wert gesetzt werden.

V4 Geometrische Formen I

Gegeben seien folgende zwei Klassen:

```
public class Circle {

  public double radius;
}

public class Rectangle {

  public double length;
  public double width;
}
```

Schreiben Sie zunächst ein Interface `ComputeArea` mit einer parameterlosen `double`-Methode `computeArea`. Erweitern Sie nun die zwei oben genannten Klassen, sodass beide das Interface `ComputeArea` implementieren. Die Methode `computeArea` soll den Flächeninhalt des Kreises bzw. des Rechtecks berechnen und zurückliefern.

V5 Geometrische Formen II

Schreiben Sie eine Methode `double computeTotalArea` mit zwei Parametern `Circle[] circle` und `Rectangle[] rectangles`. Die Methode summiert die Flächeninhalte aller übergebenen geometrischen Formen in den beiden Arrays auf und gibt diese Summe zurück.

V6 Spieglein, Spieglein ...

Wir nennen eine Gruppe von Elementen in einem Array Spiegel, wenn sie irgendwo im Array nochmal auftaucht, nur in umgekehrter Reihenfolge. Beispielsweise ist im Array `[7, 6, 5, 1, 9, 8, 5, 6, 7]` ein Spiegel vorhanden und zwar `[7,6,5]`. Schreiben Sie eine Methode `int maxMirror(int[] arr)`. Diese bekommt ein Array übergeben und gibt die Länge des größten Spiegels im übergebenen Array zurück. Gibt es keinen Spiegel so wird einfach `0` zurückgeliefert

> **Hinweis** Sie mit zwei Zeigern auf dem ersten und dem letzten Element. Vergleichen Sie nun paarweise die Elemente und überlegen Sie sich, wann Sie die beiden Zeiger weiter in die Mitte bewegen.

V7 Matrix-Multiplikation

Der folgende Code stellt beispielsweise die Matrix

$$\begin{pmatrix} 5 & 8 \\ 1 & -3 \end{pmatrix}$$

dar.

```
1 int[][] matrix = new int[2][2];
2 matrix[0][0] = 5;
3 matrix[0][1] = 8
4 matrix[1][0] = 1
5 matrix[1][1] = -3;
6
7 // alternativ und kuerzer: int[][] matrix = {{5,8},{1,-3}};
```

Sie sehen also, dass Sie einem Array in Java beliebig viele Dimensionen geben können.

Schreiben Sie eine Methode `int[][] matrixMul(int[][] mat1, int[][] mat2)`.

Die Methode bekommt zwei Matrizen, dargestellt durch zwei zwei-dimensionale Arrays, übergeben und gibt die resultierende Produktmatrix zurück. Sollte die Multiplikation aufgrund falscher Dimensionen nicht möglich sein, so geben Sie eine entsprechende Nachricht auf dem Bildschirm aus und liefern `null` zurück

> **Hinweis:** Verwenden Sie drei ineinander geschachtelte `for`-Schleifen. Die erste iteriert über die Reihen von `mat1`, die zweite iteriert über die Spalten von `mat1` und die Reihen von `mat2` und die letzte iteriert über die Spalten von `mat2`.

Weitere Roboter-Klassen

In vielen Aufgaben reichen uns die eingeschränkten Methoden eines Roboters der `FopBot`-Werke nicht. Daher definieren wir uns neue Roboter, welche die technischen Anforderungen erfüllen. In den Foliensätzen zu `FopBot` haben Sie bereits Beispiele wie den `SymmTurner` Roboter dazu gesehen.

Übungsblatt 03: Vorbereitende Übungen von Nhan Huynh und Darya Nikitina

V8 CoinPutter ★☆☆

Implementieren Sie zunächst in der aus der Vorlesung bekannten Roboter-Klasse `SymmTurner` die `public void`-Methode `coinMove(int countOfSteps)`: Diese soll `countOfSteps` Schritte nach vorne gehen und vor jedem Schritt einen Coin ablegen. Sollte die geforderte Anzahl an Schritten größer sein als die Anzahl an Coins soll der Roboter einfach stehen bleiben und sich ausschalten. Verwenden Sie dazu die Ihnen aus der Vorlesung bekannte `public`-Methode `void turnOff()`. Sollten mehr Coins vorhanden sein als Schritte gefordert sind, soll er an seiner finalen Position alle verbleibenden Coins ablegen.

V9 Richtungsdreher ★☆☆

In dieser Aufgabe soll eine neue Roboterklasse definiert werden, deren Roboter sich mittels eines einzigen Aufrufs in eine beliebige Richtung drehen können. Erstellen Sie dafür die Klasse `DirectionTurner`, die direkt von der Klasse `Robot` erbt und die parameter- und rückgabelosen `public`-Methoden `turnUp`, `turnRight`, `turnDown` und `turnLeft` so implementiert,dass der Roboter nach Aufruf einer dieser Methoden in die entsprechende Richtung blickt.

V10 TeamRobot ★★☆

In dieser Aufgabe sollen Sie ihre erste eigene Roboterklasse von Grund auf implementieren. Erstellen Sie dazu eine neue Klasse `TeamRobot`, die die Klasse `Robot` erweitert, also von ihr erbt. Der Konstruktor der Klasse `TeamRobot` übernimmt die Parameter des Konstruktors der Oberklasse `Robot` und besitzt zusätzlich die Parameter `int left` und `int right`. Der Parameter `int left` gibt an, wie viele zusätzliche Roboter beim Aufruf des Konstruktors links neben des TeamRobots platziert werden. Der Parameter `int right` ist analog, für die Roboter rechts. Der `TeamRobot`, sowie die Roboter links und rechts von ihm bilden ein Team. Die zusätzlichen Roboter werden vom `TeamRobot` im Konstruktor erzeugt. Bekommt der `TeamRobot` einen Befehl, so soll dieser von allen Robotern im Team ausgeführt werden. Die zusätzlichen Roboter selbst sind dabei nicht ansprechbar, das heißt auf ihnen können keine Methoden aufgerufen werden. Überlegen Sie sich, wie Sie die Roboter des Teams in der `TeamRobot`-Klasse speichern können und wie Sie die Befehle die ein `TeamRobot` erhält, an alle Roboter im Team weiterreichen können. Die Befehle meinen hier die Methoden: `move()`, `turnLeft()`, `pickCoin()` und `putCoin()`.

> **Beispiel:** Beim Erstellen eines TeamRobots mit den Parametern `right = 1` und `left = 2` an der Position `(4, 4)`, werden zusätzlich 3 Roboter erstellt, die dem Team angehören, nämlich an der Position `(2, 4)`, `(3, 4)` (links) und `(5, 4)` (rechts).

V11 The final Countdown ★★☆

In Unix-basierten Systemen wird die Zeit traditionell als vorzeichenbehaftete 32 Bit Ganzzahl gespeichert, die seit dem 1. Januar 1970 vergangenen Sekunden repräsentiert.

Schauen Sie folgenden Java-Code an. Beheben Sie sämtliche eingebauten Fehler, um den Code lauffähig zu machen. Was müssen Sie im Code ändern, um die folgende Ausgabe zu erhalten?

"Am 19.1.2038 kommt es zu einem Ueberlauf des Unix Zeitstempels"

```java
public final class A {

  private int value1 = 0, value2 = 0;
  private final int value3 = 0;

  private int getValue1() {
    return value1;
  }

  private int getValue2() {
    return value2;
  }

  private void setValue1(int newValue1) {
    value1 = newValue1;
  }

  private void setValue2(int newValue2) {
    value2 = newValue2;
  }

  private final void changeValue3(final int newValue3) {
    value3 = 0;
  }
}

class B extends A {

  public void changeValue3(final int newValue3) {
    value3 = newValue3;
  }

  public static void main(String args[]) {
    B obj = new B();
    obj.setValue1(19);
    obj.setValue2(1);
    obj.value3 = 2038;

    String result = "Am " +
        getValue1() + "." +
        getValue2() + "." +
        obj.value3 +
        " kommt es zu einem Ueberlauf des Unix Zeitstempels.";

    System.out.println(result);
  }
}
```