P1 - Serie 02

Lukas Batschelet - 16-499-733

Theorieaufgaben

1. `new`-Operator

> **Task**
>
> *Der `new`-Operator zusammen mit dem Konstruktor einer Klasse erledigt zwei Dinge. Was genau?*

- Reservierung von Speicher
- Initialisierung des Objekts zusammen mit dem Konstruktor

2. `ArrayList` in der Java API

> **Task**
>
> *Informieren Sie sich in der Java API Dokumentation über die Klasse `ArrayList` (welche eine Liste für Objekte repräsentiert).*

- Wie instanziieren Sie eine solche Liste?
  - `ArrayList list = new ArrayList();`
  - Oder mit einer Zuordnung des Typs (hier mit dem Beispiel `String`)
    - `ArrayList<String> list = new ArrayList<>();`
- Wie fügen Sie ein Objekt zur Liste hinzu?
  - mit der Methode `.add()` wird ein Objekt am Ende der Liste eingefügt
    - Möglicher Parameter ist der Index vom Typ `int`, welcher angibt an welcher Position das Objekt in die Liste eingefügt werden soll
- Wie greifen Sie auf ein Objekt an Position `i` zu?
  - mit der Methode `.get(i)`
- Wie löschen Sie den gesamten Inhalt der Liste?
  - mit der Methode `.clear()`
- Wie können Sie überprüfen, ob ein bestimmtes Objekt in der Liste vorhanden ist?
  - mit der Methode `.contains()` wird ein `boolean` Wert zurückgegeben, ob das Objekt welches als Parameter "mitgegeben wurde" in der Liste ist

3. Unterschied zwischen Klassen und Objekten

Task

Erläutern Sie anhand der Klasse `String` und des Objektes `"String"` den Unterschied zwischen Klasse und Objekt.

- Die Klasse `String` gibt den "Bauplan" für alle Objekte in ihr vor
  - Sie definiert die Methoden (`length()` , `substring()` , etc.) welche auf einem Objekt der Klasse ausgeführt werden können
- Das Objekt `"String"` wurde mit dem "Bauplan" der Klasse `String` "gebaut"
  - Methoden (in der Klasse definiert) die darauf angewendet werden

4. Ausgabe

Task

Welche Ausgabe erzeugen folgende Anweisungen?

```
String testString = "Think different";
System.out.println(testString.length());
System.out.println(testString.substring(0, 4));
System.out.println(testString.toUpperCase());
System.out.println(testString.charAt(7));
System.out.println(testString);
```

```
//Ausgabe
15
Thin
THINK DIFFERENT
i
Think different
```

5. Random

Task

Gegeben sei eine Objektvariable vom Typ Random mit Bezeichner `rand`*. In welchen Intervallen suchen die folgenden Anweisungen eine Zufallszahl?*

- `rand.nextInt(100) + 1;`
  - `[1, 100]`
- `rand.nextInt(51) + 100;`
  - `[100, 150]`
- `rand.nextInt(10) - 5;`
  - `[-5, 4]`
- `rand.nextInt(3) - 3;`
  - `[-3, -1]`

6. Aliase

> **Task**
>
> *Was sind Aliase und weshalb können Aliase problematisch sein?*

- Aliase sind Variablen, die auf dasselbe Objekt zeigen. Bei primitiven Datentypen passiert das nicht, da der Wert kopiert wird.

```
int num1 = 17;
int num2 = num1;
num2 = 99; System.out.println(num1); // 17
System.out.println(num2); // 99
```

Bei Objektvariablen sieht es anders aus:

```
Integer num1 = new Integer(17); Integer num2 = num1; num2.setValue(99);
System.out.println(num1); // 99
System.out.println(num2); // 99
```

Hier greifen beide Variablen auf das selbe Objekt zu. Das ist nicht immer wünschenswert, weil:

- Es ist unklar, welche Variable für was zuständig ist.
- Änderungen an einer Variable beeinflussen die andere.
- Der Code wird unübersichtlicher.
- Es kann sogenannter *garbage* und damit einhergehender Datenverlust entstehen.
    - Im obigen Beispiel aus der Vorlesung ist dem Objekt `num1` selber kein Wert mehr zugewiesen, sondern nur noch die Verweisung auf `num2`. Der Wert `17` wird damit zu *garbage*