RECAP Tag 1 – 3 Java Intro Workshop

Aufgabe 1: Schleifen, Arrays und `if` 15- 30 min

Implementieren Sie eine Ausgabe, die für ein gegebenes Array, etwa

```
jshell> int[] values = { 2, 4, 7, 3, 1}
values ==> int[5] { 2, 4, 7, 3, 1 }
```

jeweils eine Zahlfolge von 1 bis zu dem aktuellen Wert des Arrays ausgibt:

```
12
1234
1234567
123
1
```

Als Abwandlung soll n-Mal der Wert ausgegeben werden:

```
22
4444
7777777
333
1
```

Als Kür wollen wir eine Art Status-Indikator erstellen, der je nach Wert eines der folgenden Zeichen – (< 3), = (3 bis 5 inklusive) oder # (>= 6) zur Darstellung nutzt:

```
--
====
#######
===
-
```

Abschließend wollen wir eine Darstellung wie in einem Equalizer erreichen, wo die Wertebereich wie zuvor sind, jedoch die Zeichen innerhalb der Darstellung wechseln:

```
-
--
--==
--===##
--=
-
```

Aufgabe 2: Random und `rollDice()` 15- 30 min

Simulieren Sie ein Würfelspiel. Es soll die Anzahl der Versuche gezählt werden, bis eine 6 gewürfelt wird. Starten Sie mit folgendem Fragment

```
jshell> boolean got6 = false;
got6 ==> false

jshell> while (!got6)
   ...> {
   ...>     // TODO
   ...> }
```

Die Ausgabe sollte in etwa wie folgt sein:

```
4
5
2
6
#trys: 4
```

Kür: Ergänzen Sie die bestehende Funktionalität, sodass die Versuche ermittelt werden, bis zweimal eine 6 gewürfelt wurde. Es soll jeweils die Anzahl der Versuche ausgegeben werden, beispielsweise wie folgt:

```
gotFirst6 with 9
gotSecond6 with 16
```

Aufgabe 3: Rekursion und `powerOf()` 15 min

Implementieren Sie die Berechnung der Potenz mithilfe einer rekursiven Methode `powerOf(int x, int y)`, die x^y berechnet. Das kann man wie folgt prüfen:

```
public static void main(String[] args)
{
    System.out.println(powerOf(4, 2));
    System.out.println(powerOf(4, 3));
    System.out.println(powerOf(8, 2));
    System.out.println(powerOf(8, 3));
    System.out.println(powerOf(8, 4));
    System.out.println(powerOf(16, 2));
}
```

Wie sieht eine iterative Variante aus? Was kann man bei der Rekursion noch verbessern?

Aufgabe 4: String 10 min

Schreiben Sie eine Methode, um Schimpfwörter aus einem Text zu entfernen bzw. genauer durch «– PIEP –» zu ersetzen.

Beispielsweise soll folgende Eingabe

```
"Was für ein Dreck Mist Wetter in Kiel. Scheiss nasskalter Regen."
```

dann dieses Resultat produzieren:

```
Was für ein  - PIEP -   - PIEP -  Wetter in Kiel.  - PIEP -
nasskalter Regen.
```

Aufgabe 5: String 15 – 30 min

In dieser Aufgabe sollen Sie eine vereinfachte Form der Lauflängenkodierung implementieren. Um was geht es? Mitunter möchte man Informationen kompakter transportieren. Dazu kennen wir alle ZIP und ähnliche. Eine frühe und einfache Form der Komprimierung ist das sogenannte RLE (RunLength Encoding = Lauflängenkodierung). Wir vereinfachen das Ganze hier noch weiter, sodass jedes Zeichen nur 1- bis 9-mal wiederholt werden kann. Die Idee ist es, jedes Zeichen mitsamt seiner Anzahl darzustellen:

```
"A1B2C3D4E5"
```

bedeutet also 1 x A, 2 x B, 3 x C, 4 x D und 5 x E:

```
"ABBCCCDDDDEEEEE"
```

a) Implementieren Sie eine Methode `String decode(String input)`, um eine Zeichenfolge in eine kompakte Notation zu konvertieren.
b) Implementieren Sie als Gegenstück zur Dekodierung die Methode `String encode(String input)`.

Aufgabe 6: OO Record 15 min

Implementieren Sie einen Record zur Darstellung von Farben mit Namen und drei Werten für Rot-, Grün- und Blauanteil.

Kür: Ergänzen Sie einen Konsistenzcheck, damit Werte lediglich im Bereich 0 bis 255 erlaubt sind.

Tipp: Schaue einmal auf `Objects.checkIndex()`

Aufgabe 7: OO Klassen 15 min

Schreiben Sie eine Klasse namens Dog mit Attributen für Rasse, Alter und Farbe. Außerdem kann ein Hund bellen, schlafen und fressen. Erstellen Sie passende Methoden (mit Konsolenausgaben).

Überprüfen Sie Ihre Implementierung mit folgendem Ablauf:

```
public static void main(String[] args)
{
    var collie = new Dog("Collie", 11, "tricolor");
    collie.sleep();

    var boxer = new Dog("Boxer", 7, "brown");
    boxer.eat();

    var germanShepherd = new Dog("German Shepherd", 3, "black/brown");
    germanShepherd.bark();
}
```

Aufgabe 8: OO Basisklasse + Polymorphie 30 - 45 min

Schreiben Sie eine Basisklasse namens Shape. Diese soll die Subklassen für Punkte, Dreiecke und Kreise (ggf. auch noch Quadrate) besitzen. Jede Klasse soll die zwei Methoden namens `draw()` und `moveBy(int dx, int dy)` bereitstellen. Denken Sie an Polymorphismus sowie die Möglichkeit, Klassen wiederzuverwenden, etwa für die Eckpunkte von Figuren.

Aufgabe 9: OO Interaktion 30 - 45 min

Schreiben Sie zwei Klassen `Car` und `Driver`. Lassen Sie den Fahrer zunächst fahren, dann bremsen und dann wieder beschleunigen. Nutzen Sie folgenden Ablauf als Beispiel:

```
public static void main(String[] args)
{
    var audi = new Car("Audi", 0, 220);
    var driver = new Driver("Michael", audi);
    driver.drive();
    driver.speedPedalKickStart();
    driver.drive();
    driver.speedPedalDown();
    driver.drive();

    driver.brake();
    driver.brake();
    driver.brake();

    var vw = new Car("VW", 0, 120);
    driver.changeCar(vw);
    driver.speedPedalDown();
    driver.drive();
}
```

Das soll etwa zu folgenden Ausgaben führen:

```
Car stands still
Car acceleracte
Car moving with 50 km/h
Car acceleracte
Car moving with 70 km/h
Car slow down
Car slow down
Car slow down
Car acceleracte
Car moving with 20 km/h
DrivingLicense [assignedTo=Michael, validSince=1990-05-18,
city=Syke]
UUPS: I lost my license and are not allowed to drive
```

Kür: Modellieren Sie eine einfache Verkehrskontrolle, bei der der Führerschein gezeigt werden muss. Implementieren Sie den Führerschein mit Namen und Ausstelldatum sowie Ort. Für das Ausstelldatum soll die Klasse `java.time.LocalDate` zur Modellierung eines Datums genutzt werden – diese wird erst später im Kurs gründlich vorgestellt.