Woche 3

1. tl;dr Kapitel 1 bis 3

- Java Programme werden mit *Klassen* erstellt
- Klassen enthalten *Methoden* (Verhalten) und *Variablen* (Eigenschaften)
- Die Methode `main` ist der Startpunkt eines jeden Java Programmes
- *Bezeichner* gehören zu einer der drei Kategorien
    - Wörter, die für einen *bestimmten Zweck reserviert* sind(`class`, `int`, ...)
    - Wörter, die etwas aus *diesem* Programm bezeichen (*eigene* Methode oder *eigene* Variable)
    - Wörter, die etwas aus dem *Java API* bezeichnen (`System`, `main`, `println`, …)
- Konventionen für Bezeichner:
    - Klassen: `Student` oder `StudentActivity`
    - Methoden: `start` oder `findMin`
    - Variablen: `grade` oder `nextItem`
    - Konstanten: `MIN` oder `MAX_CAPACITY`
- *Java Quellcode* wird mit `javac` in *Bytecode übersetzt*(kompiliert)
- *Java Bytecode* wird mit `java` *ausgeführt* (interpretiert)
- Fehler:
    - Fehler beim Kompilieren (*Kompilier- oder Syntaxfehler*)
    - Fehler beim Interpretieren (*Laufzeitfehler*)
    - Fehler in der Semantik (*Logische Fehler*)
- Variable := Speicherort für einen Wert oder einen Wert
- Variablen müssen mit *Datentyp* und *Bezeichner* **deklariert** werden
- Sobald einer Variable ein Wert zugewiesen wurde ist sie **definiert**
- Definierte Variablen können gelesen (*referenziert*) werden

```
int pages;
pages = 256;
int figures = 46, tables;
tables = 17;

System.out.println("Anzahl Seiten des Buches: " + pages);
System.out.println("Anzahl Abbildungen: " + figures
+ "; Anzahl Tabellen: " + tables);
```

- *Konstanten* werden mit `final` modifiziert: `final int MIN = 0`
- Java kennt acht primitive Datentypen: (`byte`, `short`, `int`, `long`, `float`, `double`, `char`, `boolean`)
- Wechsel von "kleinen" zu "grossen":

```
int count = 17;
double num = count; // num = 17.0
```

- Wechsel von "grossen" zu "kleinen" via *Cast*:

```
double num = 12.34;
int count = (int) num; // num = 12.34; count = 12
```

- Ausdruck := Kombination von einem oder mehreren *Operanden* und *Operatoren*
- Operanden sind Werte, Variablen oder Konstanten
- Arithmetische Ausdrücke:

```
double grade = (double) points / MAX_POINTS * 5 + 1;
```

- Lesen verändert Variablen niemals: `MAX_POINTS * 5`
- Zuweisungsoperatoren und das Inkrement/Dekrement machen das Leben einfacher:

```
points = points * 2;
points *= 2;
points = points + 1;
points++;
```

- **Boolesche Ausdrücke** sind entweder `true` oder `false`
- Boolesche Ausdrücke oder Boolesche Variablen können kombiniert und negiert werden

```
boolean smaller = hours < MAX;
boolean decision = (hours < MAX || hours > MIN) && !complete;
```

- Die `if`-*Anweisung* ist eine "Verzweigung", die auf einem Boolschen Ausdruck basiert:

```
if (hours < MAX) {
    hours += 10;
```

```
    System.out.println("10 Stunden hinzugefügt.");
} else
    System.out.println("ACHTUNG: Maximum erreicht!");
```

- Das **Java API** besteht aus verschiedenen Packages, welche Klassen beinhalten, die Lösungen für häufige Aufgaben bereitstellen
- Sie kennen verschiedene Klassen: `String, Scanner, Random, DecimalFormat`
- Der `new`-Operator *instanziiert* mit dem Aufruf des Konstruktors ein Objekt aus einer *Klasse* (Datentyp einer Objektvariablen := Klasse)

```
String str = new String("Hallo Welt");
Scanner scan = new Scanner(System.in);
Random rand = new Random();
```

- Methoden können mit dem Punkt-Operator auf instanziierten Objekten aufgerufen werden

```
int length = str.length();
int number = scan.nextInt();
double randomNumber = rand.nextFloat();
```

- *Primitive Datentypen*: Kopien von Variablen sind unabhängig

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

- Objektvariablen: Kopien von Variablen sind *abhängig* (*Aliase*)

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

2. Aufgaben zu Kapitel 1 bis 3

- Begründen Sie, weshalb folgender Kommentar nicht optimal ist:

```
System.out.println("test"); // muss später geändert werden
```

Was, Wann, wo, weshalb muss geändert werden?

- Welche der folgenden Bezeichner sind in Java syntaktisch nicht korrekt:

```
int grade, quiz Grade, 2ndGrade, MAX_GRADE, gradeNr, grades1&2,
grade_17;
```

Nicht korrekt sind: `quiz Grade`, `2ndGrade`, `grades1&2`

- Welche Ausgabe produziert das folgende Code-Fragment?

```
int num = 9;
Math.sqrt(num);
System.out.println(num);

double d = 7.99;
int i = (int) d;
System.out.println(d + " " + i);

String name = "emilie";
name.replace("e", "x");
System.out.println(name.charAt(0));

boolean a = false, b = false, c = true;
boolean e = (a || !b) && c;
System.out.println(e);
```

  - `9` da nicht zugewiesen wird
  - `7.99 7`
  - `e` da Strings immutable sind und die Veränderung nicht zugewiesen wurde
  - `true`

- Welche Ausgabe produziert das folgende Code-Fragment?

```
System.out.println("Ich bin " + (5 + 5));
System.out.println("Ich bin " + 5 + 5);
System.out.println(5 + 5 + " bin ich");
System.out.println("Ich bin " + 5 * 5);
```

  - `Ich bin 10`
  - `Ich bin 55`
  - `10 bin ich`
  - `Ich bin 25`

- Welchen Wert enthält die Variable `result`, nachdem folgende Anweisungen durchgeführt worden sind?

```
int result = 10;
result += 10;
result *= result;
result /= 4;
result -= 10;
```

Schlussresultat = `90`

- Welche Ausgabe produziert das folgende Code-Fragment?

```
// Teaser zu Verschachtelungen!
int val1 = 7, val2 = 3;
boolean on = false;
if (val1 > val2 * 2) {
    if (!on)
        System.out.print("Blau ");
    else
        System.out.print("Schwarz ");
} else
    System.out.print("Rot ");
System.out.print("Gelb ");
```

Ausgabe: `Blau Gelb`

- Geben Sie je eine Konfiguration für `val1`, `val2` und `on` an, so dass die Ausgabe lautet:
  - `Schwarz Gelb`
  - `Rot Gelb`

```
int val1 = 7, val2 = 3;
boolean on = true;
```

```
int val1 = 4, val2 = 3;
boolean on = false;
```

- Welche Ausgabe produziert das folgende Code-Fragment?

```
DecimalFormat fmt = new DecimalFormat("000.0#");
System.out.println(fmt.format(3));
System.out.println(fmt.format(13.20));
```

```
System.out.println(fmt.format(2134.201));
System.out.println(fmt.format(0.1346));
```

- `003.0`
- `013.2`
- `2134.2`
- `000.13`

- Gegeben sei eine Objektvariable vom Typ `Random` mit Bezeichner `rand`. Schreiben Sie je eine Programmieranweisung, die Ihnen eine ganzzahlige Zufallszahl aus folgenden Intervallen erzeugt:
  - `[0,100];`
  - `[1,3];`
  - `[5, 10];`
  - `[-10,0]`

```
rand.nextInt(101);
rand.nextInt(3) + 1;        rand.nextInt(1, 4);
rand.nextInt(6) + 5;        rand.nextInt(5, 11);
rand.nextInt(11) -10;       rand.nextInt(-10, 0);
```

3. Vorprogrammieren der Klassen ParkingFee und GuessingGame

```
import java.util.Scanner;

public class ParkingFee {

    public static void main(String[] args) {

        final double FEE_PER_HOUR = 2.5;
        final int MAX_HOURS = 5;
        final double FACTOR = 1.5;

        Scanner scan = new Scanner(System.in);

        System.out.println("Parkzeit in Stunden angeben: ");
        int parkingTime = scan.nextInt();

        scan.close();

        double fee = 0.0;
        if (parkingTime <= MAX_HOURS) {
            fee = parkingTime * FEE_PER_HOUR;
        } else {
            int additionalHours = parkingTime - MAX_HOURS;
            fee = MAX_HOURS * FEE_PER_HOUR;
            fee += additionalHours * FEE_PER_HOUR * FACTOR;
        }
```

```
        System.out.println("Parkgebühr: " + fee);
    }
}
```

```
import java.util.Scanner;
import java.util.Random;

public class GuessingGame {
    public static void main(String[] args) {

        final int MAX = 3;
        int answer, guess;

        Scanner scan = new Scanner(System.in);
        Random rand = new Random();

        answer = rand.nextInt(MAX) + 1;

        System.out.println("Gesucht ist eine Zahl zwischen 1 und " + MAX);

        guess = scan.nextInt();

        scan.close();

        if (guess == answer) {
            System.out.println("Juhuuuu!");
        } else {
            System.out.println("Buhuuu");
            System.out.println("Gesucht war: " + answer);
        }

    }
}
```

4. (Java Code Challenge)

- Schreiben Sie ein Java-Programm, das eine Zeichenkette umkehrt, ohne die Methode `reverse` der Klasse `String` zu verwenden. Es gibt mindestens drei Möglichkeiten, diese Aufgabe zu lösen.
- Schreiben Sie ein Java-Programm, um festzustellen, ob eine Zahl eine Primzahl ist oder nicht. Ihr Programm soll ein Array von Zahlen akzeptieren. Es soll durch das Array iterieren und feststellen, ob jede Zahl eine Primzahl ist oder nicht. Wenn es mit der Verarbeitung fertig ist, soll es alle Zahlen zurückgeben, die Primzahlen sind.

5. VS Code Java Shortcuts

Abkürzung	Beschreibung	Generierter Code
`syso`	System.out.println	`System.out.println("");`

Abkürzung	Beschreibung	Generierter Code
psvm	public static void main	public static void main(String[] args) {}
fori	For-Schleife	for (int i = 0; i < ; i++) {}
fore	Enhanced For-Schleife	for (Type var : iterable) {}
while	While-Schleife	while (condition) {}
do	Do-While-Schleife	do {} while (condition);
if	If-Anweisung	if (condition) {}
ifelse	If-Else-Anweisung	if (condition) {} else {}
switch	Switch-Anweisung	switch (variable) { case value: break; }
sout	System.out.print	System.out.print("");