Fachklassen

Eine **Fachklasse** ist eine Klasse, die nur "im Hintergrund" arbeitet und **nicht für die Interaktion mit dem Benutzer zuständig ist**.

Klassenstruktur

Gemäss aktuellem Wissensstand folgen Klassen dem **Schema**:

Klassendeklaration

Instanzvariablen / Attribute

Instanzmethoden / Methoden

```
public class ClassName { // Klassendeklaration Start


    private int myVar; // Instanzvariable / Attribut



    public void setMyVar(int value) { // Methode (setter)
        myVar = value;
    }

    public int getMyVar() { // Methode (getter)
        return myVar;
    }
} // Klassendeklaration Ende
```

Klassendeklaration:

- Die Klassendeklaration definiert den **Namen** der Klasse.
- Der **Namen** der Klasse definiert automatisch auch einen **Datentyp**!

Klassen-Body:

- Der Klassenbody beinhaltet die **Deklaration der Instanzvariablen und Instanz Methoden und wird durch { } begrenzt**
- Innerhalb des Body darf **kein ausführenden Code** stehen, dieser befindet sich **immer** innerhalb vom Methoden-Body

Instanzvariablen / Attribute:

- Instanzvariablen sind Variablen welche im Klassen-Body definiert werden.
- Daher befinden sie sich **nicht innerhalb** einer Methode.
- Instanzvariablen können innerhalb **aller Instanzmethoden** zugegriffen werden.
- Das Wort `private` bedeutet, dass die Variable aber nur innerhalb der Klasse sichtbar ist, **nicht ausserhalb**.

Instanzmethoden:

- sind Methoden innerhalb einer Klasse, welche **nicht static** deklariert sind
- können auf alle Instanzvariablen zugreifen
- In ihnen wird die **Fachlogik** ausprogrammiert. Häufig werden Instanzvariablen mutiert, verarbeitet und oder zurückgegeben

Methoden-Body:

- Im Methoden-Body {} befindet sich der **eigentliche Code für die Fachlogik**
- Werden Variablen innerhalb vom Methoden-Body deklariert, sind diese **ausschliesslich innerhalb diesem Block** verwendbar/sichtbar.

> **KEINE `main` METHODE IN EINER FACHKLASSE**
>
> - Es gibt **keine** Methode `public static void main(String[] args)`.
> - Diese sollte **nur** in der `Starter` Klasse existieren
> - Es ist theoretisch möglich mehrere `main` Methoden zu haben, dies ist jedoch **schlechter Stil**

Instanziierung und Verwendung eines Objekts/Instanz

Objekte lassen sich im Code wie folgt erstellen:

```
// Datentyp    Variable        Objektzuweisung  Objekterstellung
   ClassName   variablenName          =         new ClassName();

// Es können mehrere Variablen mit Objekte derselben Klasse definiert werden
   ClassName   otherObject            =         new ClassName();

// Mit einem Punkt "." wird auf die Instanz-Methoden zugegriffen!
   variablenName.setMyVar(12);

// Der Rückgabewert einer Methode kann in einer Variablen gespeichert werden
   int value = variablenName.getMyVar();

// Der Rückgabewert einer Methode kann auch direkt wiederverwendet werden
   otherObject.setMyVar(variablenName.getMyVar());
```

JE BESSER DIE NAMEN DESTO LESERLICHER WIRD DER CODE!

`ClassName` ist in dem oberen Beispiel generisch gewählt da es sich um ein generelles Beispiel handelt. Anstatt `ClassName` sollte später ein **spezifischer Namen** gewählt werden, wie z.B. `Account`. Der Name der Variable kann beliebig sein. Das gleiche gilt für `Variablen` und `Methoden`

```
Account savingAccount = new Account(); // Toll
Xyz b = new Xzy();                     // Evt. nicht ganz so toll ;)
```