Object Oriented Programming

What is OOP?

We have a lot of objects around us

- They have a state and a behaviour
- All have a building plan

Classes $\rightarrow$ building plan
Objects $\rightarrow$ entity in real world build from building plan

How to build a class

```
1   public class MyClass {
2       //Constructor to create the Object
3       public MyClass(){
4       ...
5       }
6
7       //Attributes to describe state
8       private int myAttribute;
9       ...
10
11      //Methods to describe behaviour
12      public MyMethod(int myParameter, ...) {
13          this.myAttribute = myParameter;
14          ...
15      }
16  }
17
```

How to build a student?

Attributes describing a student?

How to build a student?

Attributes describing a student?

- Name
- Matriculation number
- Age

How to build a student?

Attributes describing a student?

- Name
- Matriculation number
- Age

Methods describing its behaviour?

How to build a student?

Attributes describing a student?

- Name
- Matriculation number
- Age

Methods describing its behaviour?

- Change name
- Increase age
- Calculate 1+1

Class Student

```
1   public class Student {
2   //Constructor
3   public Student (String name, int matriculationNumber) {
4   this.name = name;
5   this.matriculationNumber = matriculationNumber;
6   }
7
8   // Attributes
9   private String name;
10  private int matriculationNumber;
11
12
13  // Methods
14  public void setName(String name) {
15  this.name = name;
16  }
17
18  public int getMatriculationNumber() {
19  return matriculationNumber;
20  }
21
22  }
23
```

Creation

We learned how to declare and assign a primitive datatype.

```
1 int a; // declare a
2 a = 273; // assign 273 to a
```

The creation of an object works similar.

```
1 Student example = new Student();
2 // create an instance of Student
```

The **object** derived from a **class** is also called **instance**. The variable is called the **reference**.

Calling a Method

```
1  public class Student {
2
3  private String name;
4
5  public String getName() {
6  return name;
7  }
8  public void setName(String newName) {
9  name = newName;
10 }
11 public void printName(){
12 System.out.println(this.name);
13 }
14 }
```

The class *Student* has three methods: *void printName()*.

Calling a Method

```java
public class Main {

public static void main(String[] args) {
Student example = new Student(); // creation
example.setName("Jane"); // method call
String name = example.getName();
System.out.println(name); // Prints "Jane"

example.printName();
}

}
```

You can call a method of an object after its creation with **reference.methodName();**.

Calling a Method

```
1  public class Student {
2
3  private String name;
4
5  public void setName(String newName) {
6  name = newName;
7  printName();   // Call own method
8  this.printName(); // Or this way
9  }
10
11 public void printName() {
12 System.out.println(name);
13 }
14
15 }
```

You can call a method of the own object by simply writing **methodName();** or **this.methodName();**

Methods with Arguments

```
1  public class Calc {
2
3  public void add(int summand1, int summand2) {
4  System.out.println(summand1 + summand2);
5  }
6
7  public static void main(String[] args) {
8  int summandA = 1;
9  int summandB = 2;
10 Calc calculator = new Calc();
11 System.out.print("1 + 2 = ");
12 calculator.add(summandA, summandB);
13 // prints: 3
14 }
15
16 }
```

Methods with Return Value

A method without a return value is indicated by **void**:

```
1 public void add(int summand1, int summand2) {
2 System.out.println(summand1 + summand2);
3 }
```

A method with an **int** as return value:

```
1 public int add(int summand1, int summand2) {
2 return summand1 + summand2;
3 }
```

Calling Methods with a return value

```
1  public class Calc {
2
3  public int add(int summand1, int summand2) {
4  return summand1 + summand2;
5  }
6
7  public static void main(String[] args) {
8  Calc calculator = new Calc();
9  int sum = calculator.add(3, 8);
10 System.out.print("3 + 8 = " + sum);
11 // prints: 3 + 8 = 11
12 }
13
14 }
```

Constructors

```
1  public class Calc {
2
3  private int summand1;
4  private int summand2;
5
6  public Calc() {
7  summand1 = 0;
8  summand2 = 0;
9  }
10
11 }
```

A constructor gets called upon creation of the object

Constructors with Arguments

```
1  public class Calc {
2
3  private int summand1;
4  private int summand2;
5
6  public Calc(int x, int y) {
7  summand1 = x;
8  summand2 = y;
9  }
10
11 }
```

```
1 [...]
2 Calc myCalc = new Calc(7, 9);
```

A constructor can have arguments as well!

Let's build a car

Create a car with doors, wheels, gas, seats,...

Focus on:

ID	unique id for each car
Car	gas, speed
Doors	can open/close
Wheels	air pressure, size,...
Seats	free, quality
...	

Also implement refueling, open/close doors, decrease/increase air pressure,...

Class Student

```
1 public static void main(String[] args) {
2 Student peter = new Student();
3 peter.changeName("Peter");
4 }
```