Java

Controll Statements and Functions

Vincent Gerber, Tilman Hinnerichs

Java Kurs

17. Mai 2018

Object Oriented Programming

Class Student

```java
public class Student {

    // Attributes
    private String name;
    private int matriculationNumber;


    // Methods
    public void setName(String name) {
        this.name = name;
    }

    public int getMatriculationNumber() {
        return matriculationNumber;
    }

}
```

Creation

We learned how to declare and assign a primitive datatype.

```java
        int a; // declare a
        a = 273; // assign 273 to a

```

The creation of an object works similar.

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

```

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

Calling a Method

```java
public class Student {

    private String name;

    public String getName() {
        return name;
    }

    public void setName(String newName) {
        name = newName;
    }

}

```

The class *Student* has two methods: *void printTimetable()* and *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"
    }

}

```

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

Calling a Method

```
public class Student {

     private String name;

     public void setName(String newName) {
          name = newName;
          printName();    // Call own method
          this.printName(); // Or this way
     }

     public void printName() {
          System.out.println(name);
     }

}
```

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

Methods with Arguments

```java
public class Calc {

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

    public static void main(String[] args) {
        int summandA = 1;
        int summandB = 2;
        Calc calculator = new Calc();
        System.out.print("1 + 2 = ");
        calculator.add(summandA, summandB);
        // prints: 3
    }

}

```

Methods with Return Value

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

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

```

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

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

```

Calling Methods with a return value

```
public class Calc {

    public int add(int summand1, int summand2) {
        return summand1 + summand2;
    }

    public static void main(String[] args) {
        Calc calculator = new Calc();
        int sum = calculator.add(3, 8);
        System.out.print("3 + 8 = " + sum);
        // prints: 3 + 8 = 11
    }

}
```

Constructors

```java
public class Calc {

    private int summand1;
    private int summand2;

    public Calc() {
        summand1 = 0;
        summand2 = 0;
    }

}

```

A constructor gets called upon creation of the object

Constructors with Arguments

```java
public class Calc {

    private int summand1;
    private int summand2;

    public Calc(int x, int y) {
        summand1 = x;
        summand2 = y;
    }

}

```

```java
[...]
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
...	