Java

Controll Statements and Functions

Vincent Gerber, Tilman Hinnerichs

Java Kurs

16. 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.

```
1 int a; // declare a
2 a = 273; // assign 273 to a
3
```

The creation of an object works similar.

```
1 Student example = new Student();
2 // create an instance of Student
3
```

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
9       public void setName(String newName) {
10          name = newName;
11      }
12
13  }
14
```

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

```
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  }
16
```

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**:

```
1    public void add(int summand1, int summand2) {
2        System.out.println(summand1 + summand2);
3    }
4
```

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

```
1    public int add(int summand1, int summand2) {
2        return summand1 + summand2;
3    }
4
```

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  }
15
```

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  }
12
```

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 }
12
```

```
1  [...]
2  Calc myCalc = new Calc(7, 9);
3
```

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
...	