Java

Controll Statements & OOP

Max Langer, Leonard Follner, Alexander Hesse
24. Oktober 2016

Java-Kurs

Overview

Recalling last session

Conclusion

Datatypes

- int, long
- float, double
- String

Hello World example

Controll Statements

Controll Statements

- if, else, else if
- for
- while

If Then Else

```
1  if(condition) {
2      // do something if condition is true
3  } else if(another condition){
4      // do if "else if" condition is true
5  } else {
6      // otherwise do this
7  }
```

If Then Else example

```java
public class IteExample {

    public static void main(String[] args) {
        int myNumber = 5;

        if(myNumber == 3) {
            System.out.println("Strange number");
        } else if(myNumber == 2) {
            System.out.println("Unreachable code");
        } else {
            System.out.println("Will be printed");
        }
    }

}
```

Conditions?

How to compare things:

- `==` Equal
- `!=` Not Equal
- `>` Greater Than
- `>=` Greater or Equal than

Note: You can concatenate multiple conditions with `&&` (AND) or `||` (OR)

for

```
1 for(initial value, condition, change) {
2     // do code while condition is true
3 }
```

```
 1   public class ForExample {
 2
 3       public static void main(String[] args) {
 4           for(int i = 0; i <= 10; i++) {
 5               System.out.print("na ");
 6           }
 7           System.out.println("BATMAN!");
 8       }
 9
10   }
```

while

```
1  while(condition) {
2      // do code while condition is true
3  }
```

while example

```java
public class WhileExample {

    public static void main(String[] args) {
        int a = 0;
        while(a <= 10) {
            System.out.println(a);
            a++; // Otherwise you would get an endless loop ∞
        }
    }

}
```

OOP in Java

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

```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

```
1  public class Main {
2
3      public static void main(String[] args) {
4          Student example = new Student(); // creation
5          example.setName("Jane"); // method call
6          String name = example.getName();
7          System.out.println(name); // Prints "Jane"
8      }
9
10 }
11
```

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

Calling a Method

```java
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**:

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

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

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

Calling Methods with a return value

```java
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

```
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 = x;
9        }
10
11    }
12


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

A constructor can have Arguments aswell!

Conclusion

An Example

You want to program an enrollment system, for a programming course.

Your classes are:

student	who wants to attend the course
lesson	which is a part of the course
tutor	the guy with the bandshirt
room	where your lessons take place
...	

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