Java

Controll Statements & OOP

Max Langer, Leonard Follner, Alexander Hesse

20. Oktober 2016

Java-Kurs

Overview

1. Recalling last session
2. Controll Statements
    - Ite
    - for
    - while
3. OOP in Java
    - General information
    - Methods
    - Return Value
    - Constructor
4. Conclusion
    - An Example

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

```
if(condition) {
    //do something if condition is true
} else if(another condition){
    //do if "else if" condition is true
} else {
    //otherwise do this
}
```

If Then Else example

```
1  public class IteExample {
2      public static void main(String[] args) {
3          int myNumber = 5;
4
5          if(myNumber == 3) {
6              System.out.println("Strange number");
7          } else if(myNumber == 2) {
8              System.out.println("Unreachable code");
9          } else {
10             System.out.println("Will be printed");
11         }
12     }
13 }
```

for

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

for example

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

while

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

while example

```
1 public class WhileExample {
2     public static void main(String[] args) {
3         int a = 0;
4         while(a <= 10) {
5             System.out.println(a);
6         }
7     }
8 }
```

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

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

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

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

Methods with Arguments

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

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
```

Constructors

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

A constructor gets called upon creation of the object

Constructors with Arguments

```
1     public class Calc {
2         private int summand1;
3         private int summand2;
4
5         public Calc(int x, int y) {
6             summand1 = x;
7             summand2 = x;
8         }
9     }
10    [...]
11    Calc myCalc = new Calc(7, 9);
12
```

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
...	

Class *Student*

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