

# Organisation

---

# Java

## Introduction

---

Max Langer, Leonard Follner, Alexander Hesse

20. Oktober 2016

Java-Kurs

# Overview

## 1. Organisation

Proceeding

## 2. Your first program

Hello World!

Setting up IntelliJ IDEA

## 3. Basics

Some definitions

Calculating

Text with Strings

# About this course

## Requirements

- You know how to use a computer
- Please bring your computer with You
- Maybe already knowledge in programming languages?

## Proceeding

- There will be 14 lessons
- Each covers a topic and comes with exercises

## Some resources

- You can ask your tutor
- Join the Auditorium group  
<http://auditorium.inf.tu-dresden.de>
- StackOverflow, FAQs, Online-tutorials, ...
- Official documentation  
<https://docs.oracle.com/javase/8/>
- mailinglist [programmierung@ifsr.de](mailto:programmierung@ifsr.de)
- Cyberspace (wednesday 5./6. DS)
- Material-Repository  
<https://github.com/LeonardFollner/java-lessons>

## Pros:

- Syntax like C++
- Strongly encourages OOP
- Platform-independent (JVM)
- Very few external libraries
  - > Easy to use and very little to worry about

## Cons:

- A lot of unnecessary features in the JDK
- Slower than assembly
- No multi-inheritance
- Weak generics
- Mediocre support for other programming paradigms
  - > Neither fast, small nor geeky

# Your first program

---



# Hello World

DEMO

# Creating your Working Environment

Open the Terminal

```
1  mkdir myProgram
2  cd myProgram
3  touch Hello.java
4  vim Hello.java
```

# Hello World!

This is an empty JavaClass. Java Classes always start with a capital letter

```
1  public class Hello {  
2  
3  }  
4
```

# Hello World!

This is a small program printing *Hello World!* to the console:

```
1 public class Hello {  
2     public static void main(String[] args) {  
3         System.out.println("Hello World!");  
4     }  
5 }  
6
```

# How to run your program

save your program by pressing 'esc', then ':w' exit vim by typing ':q' (and hit return) then:

```
1   javac Hello.java
2   java Hello
3
```

# Hello World in an IDE

DEMO

# Receive a copy of IntelliJ IDEA

IntelliJ IDEA is a powerful IDE<sup>1</sup>, e.g. for Java.

- You can download IntelliJ IDEA at <https://www.jetbrains.com/idea/>
- Get an Ultimate-License at <https://www.jetbrains.com/student/>
- Use JetBrains IDEs for all programming languages

Eclipse is free and open-source, but less powerful.

---

<sup>1</sup>Integrated Development Environment

# Basics

---



# Comments

```
1 public class Hello {  
2     // prints a "Hello World!" on your console  
3     public static void main(String[] args) {  
4         System.out.println("Hello World!");  
5     }  
6 }  
7
```

You should always comment your code.  
Code is read more often than it is written.

- `//` single line comment
- `/*` comment spanning multiple lines `*/`

# Code concepts

```
1 public class Hello {  
2     // Calculates some stuff and outputs everything on the  
3     console  
4     public static void main(String[] args) {  
5         int x;  
6         x = 9;  
7         int y = 23;  
8         int z;  
9         z = x * y;  
10        System.out.println(z);  
11    }  
12 }  
13
```

# Code concepts

```
1 public class Hello {  
2     // Calculates some stuff and outputs everything on the  
3     console  
4     public static void main(String[] args) {  
5         System.out.println(9 * 23);  
6     }  
7 }
```

# Primitive data types

Java supports some primitive data types:

`boolean` a truth value (either **true** or **false**)

`int` a 32 bit integer

`long` a 64 bit integer

`float` a 32 bit floating point number

`double` a 64 bit floating point number

`char` an ascii character

`void` the empty type (needed in later topics)

# About the Semicolon

```
1 public class Hello {  
2     // prints a "Hello World!" on your console  
3     public static void main(String[] args) {  
4         System.out.println("Hello World!")@\\textcolor{red}{\\  
5         texttt{;}}@  
6     }  
7 }
```

Semicolons conclude all statements.

Blocks do not need a semicolon.

# Blocks

```
1 public class Hello @\textcolor{red}{\texttt{\{\}}}\@  
2     // prints a "Hello World!" on your console  
3     public static void main(String[] args) {  
4         System.out.println("Hello World!");  
5     }  
6 @\textcolor{red}{\texttt{\}}}\@  
7
```

Everything between { and } is a *block*.

Blocks may be nested.

# Naming of Variables

- The names of variables can begin with any letter or underscore. Usually the name starts with small letter.
- Compound names should use CamelCase.
- Use meaningful names.

```
1 public class Calc {  
2     public static void main(String[] args) {  
3         int a = 0; // not very meaningful  
4         float myFloat = 5.3f; // also not meaningfull  
5         int count = 7; // quite a good name  
6  
7         int rotationCount = 7; // there you go  
8     }  
9 }  
10
```

# Calculating with *int* I

```
1 public class Calc {  
2     public static void main(String[] args) {  
3         int a; // declare variable a  
4         a = 7; // assign 7 to variable a  
5         System.out.println(a); // prints: 7  
6         a = 8;  
7         System.out.println(a); // prints: 8  
8         a = a + 2;  
9         System.out.println(a); // prints: 10  
10    }  
11 }  
12
```

After the first assignment the variable is initialized.



## Calculating with *int* II

```
1 public class Calc {  
2     public static void main(String[] args) {  
3         int a = -9; // declaration and assignment of a  
4         int b; // declaration of b  
5         b = a; // assignment of b  
6         System.out.println(a); // prints: -9  
7         System.out.println(b); // prints: -9  
8         a++; // increments a  
9         System.out.println(a); // prints: -8  
10    }  
11 }  
12
```

Some basic mathematical operations:

Addition	<code>a + b;</code>
Subtraction	<code>a - b;</code>
Multiplication	<code>a * b;</code>
Division	<code>a / b;</code>
Modulo	<code>a % b;</code>
Increment	<code>a++;</code>
Decrement	<code>a--;</code>

## Calculating with *float* I

```
1 public class Calc {  
2     public static void main(String[] args) {  
3         float a = 9;  
4         float b = 7.5f;  
5         System.out.println(a); // prints: 9.0  
6         System.out.println(b); // prints: 7.5  
7         System.out.println(a + b); // prints: 16.5  
8     }  
9 }  
10
```

## Calculating with *float* II

```
1 public class Calc {  
2     public static void main(String[] args) {  
3         float a = 8.9f;  
4         float b = 3054062.5f;  
5         System.out.println(a); // prints: 8.9  
6         System.out.println(b); // prints: 3054062.5  
7         System.out.println(a + b); // prints: 3054071.5  
8     }  
9 }  
10
```

Float has a limited precision.

*This might lead to unexpected results!*

## Mixing *int* and *float*

```
1 public class Calc {  
2     public static void main(String[] args) {  
3         float a = 9.3f;  
4         int b = 3;  
5         System.out.println(a + b); // prints: 12.3  
6         float c = a + b;  
7         System.out.println(c); // prints: 12.3  
8     }  
9 }  
10
```

Java converts from **int** to **float** by default, if necessary.  
But not vice versa.

# Strings

A String is not a primitive data type but an object.  
We discuss objects in detail in the next section.

```
1 public class Calc {  
2     public static void main(String[] args) {  
3         String hello = "Hello World!";  
4         System.out.println(hello); // print: Hello World!  
5     }  
6 }  
7
```

# Concatenation

```
1 public class Calc {  
2     public static void main(String[] args) {  
3         String hello = "Hello";  
4         String world = " World!";  
5         String sentence = hello + world;  
6         System.out.println(sentence);  
7         System.out.println(hello + " World!");  
8     }  
9 }  
10
```

You can concatenate Strings using the +. Both printed lines look the same.

# Strings and Numbers

```
1 public class Calc {  
2     public static void main(String[] args) {  
3         int factorA = 3;  
4         int factorB = 7;  
5         int product = factorA * factorB;  
6         String answer =  
7             factorA + " * " + factorB + " = " + product;  
8         System.out.println(answer); // prints: 3 * 7 = 21  
9     }  
10 }  
11
```

Upon concatenation, primitive types will be replaced by their current value as *String*.