

# Java

## Introduction

---

Hannes Ueck, Jakob Krude

16. November 2020

Java-Kurs

## 1. Proceeding

About this course

## 2. What you need

## 3. Basics

Some definitions

Calculating

Text with Strings

# Proceeding

---

# About this course

## Requirements

- null

## Proceeding

- There will be 10 lessons
- Each covers a topic and comes with exercises

# Resources

- You can find the materials here  
<https://github.com/jkrude/java-begginer-lesson>
- Official documentation <https://docs.oracle.com/javase/14/>
- SoloLearn <https://www.sololearn.com/Course/Java/>
- Reddit <https://www.reddit.com/r/learnjava/>
- Questions?
  - Ask us!
  - StackOverflow <https://stackoverflow.com>

# Why Java?

- Free to use
- Readable code
- Errors recognized early by compiler
- Platform-independent (JVM)<sup>1</sup>
- Important Frameworks (Spring, Android)
  - > Easy to use and very little to worry about

---

<sup>1</sup>Java Virtual Machine

# Cons of Java

- A lot of unnecessary features in the JDK
- Slower than assembly or C
- Mediocre support of mutable and immutable types
- Mediocre support for other programming paradigms
  - > Neither fast, small nor geeky

## What you need

---



IntelliJ IDEA is a powerful IDE<sup>2</sup>, e.g. for Java.

- You can download and install<sup>3</sup> IntelliJ IDEA at  
<https://www.jetbrains.com/idea/>
- Get an Ultimate-License at  
<https://www.jetbrains.com/student/>
- Install JetBrains IDEs for Java or other programming languages with  
toolbox  
<https://www.jetbrains.com/toolbox-app>

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

---

<sup>2</sup>Integrated Development Environment

<sup>3</sup><https://www.jetbrains.com/help/idea/installation-guide.html>

## And ...

- Git (Version Control)  
<https://git-scm.com/downloads>
- Java SDK <sup>4</sup>  
<https://www.oracle.com/de/java/technologies/javase-downloads.html>
- "Code with me" (Plugin for IntelliJ) <https://plugins.jetbrains.com/plugin/14896-code-with-me>

---

<sup>4</sup>Software Development Kit, can also be downloaded within IntelliJ IDEA

# 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  
3     the 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  
3     the 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}{\texttt{;}}@  
5     }  
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      `a + b;`

Subtraction    `a - b;`

Multiplication `a * b;`

Division        `a / b;`

Modulo         `a % b;`

Increment      `a++;`

Decrement     `a--;`

# 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 =  
9         21  
10    }  
11 }
```

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