Java

Introduction

Moritz Pflügner, Yannick Spörl

24. Oktober 2018

Java-Kurs

Overview

Proceeding

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 excercises

Some resources

- You can ask your tutor
- StackOverflow, FAQs, Online-tutorials, ...
- Official documentation
  `https://docs.oracle.com/javase/8/`
- Excercises-Repository
  `http://fsr.github.io/java-lessons/`
- Slides, Solutions,... -Repository
  `https://github.com/morpfl/javakurs`

Ersatztermin nächste Woche

- Montag 5. DS
- Dienstag 4. DS
- Donnerstag 2. DS
- Donnerstag 5. DS
- Freitag 2. DS
- Freitag 3. DS
- Ansonsten Ausweichen auf anderen Kurs :-)
- Abstimmen unter:
  `https://doodle.com/poll/unu8sk7edffshs3e`

How to publish your code

- Paste your code
  `https://pastebin.com/`
- Link your code
  `https://piratenpad.de/p/javakurs`

About Java

Pros:

- Syntax like C++
- Strongly encourages OOP
- Platform-independent (JVM)
- Very few external libraries

  $->$ Easy to use and very little to worry about

About Java

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

```
mkdir myProgram
cd myProgram
touch Hello.java
vim Hello.java

```

Hello World!

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

```
    public class Hello {

    }

```

Hello World!

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

```java
public class Hello {
    public static void main(String[] args) {
        System.out.println("Hello World!");
    }
}

```

How to run your program

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

```
        javac Hello.java
        java Hello

```

Hello World in an IDE

DEMO

Receive a copy of IntelliJ IDEA

IntelliJ IDEA is a powerful IDE[1], 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

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

[1]Integrated Development Environment

Basics

Comments

```java
public class Hello {
    // prints a "Hello World!" on your console
    public static void main(String[] args) {
        System.out.println("Hello World!");
    }
}

```

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

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

Code concepts

```java
public class Hello {
    // Calculates some stuff and outputs everything on the console
    public static void main(String[] args) {
        int x;
        x = 9;
        int y = 23;
        int z;
        z = x * y;

        System.out.println(z);
    }
}

```

Code concepts

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

```

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

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

```

Semicolons conclude all statements.

Blocks do not need a semicolon.

Blocks

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

```

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.

```java
    public class Calc {
        public static void main(String[] args) {
            int a = 0; // not very meaningful
            float myFloat = 5.3f; // also not meaningfull
            int count = 7; // quite a good name

            int rotationCount = 7; // there you go
        }
    }

```

Calculating with int I

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

```

After the first assignment the variable is initialized.

Calculating with int II

```java
public class Calc {
    public static void main(String[] args) {
        int a = -9; // declaration and assignment of a
        int b; // declaration of b
        b = a; // assignment of b
        System.out.println(a); // prints: -9
        System.out.println(b); // prints: -9
        a++; // increments a
        System.out.println(a); // prints: -8
    }
}
```

Calculating with int III

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

```
public class Calc {
    public static void main(String[] args) {
        float a = 9;
        float b = 7.5f;
        System.out.println(a); // prints: 9.0
        System.out.println(b); // prints: 7.5
        System.out.println(a + b); // prints: 16.5
    }
}

```

Calculating with float II

```java
public class Calc {
    public static void main(String[] args) {
        float a =       8.9f;
        float b = 3054062.5f;
        System.out.println(a); // prints: 8.9
        System.out.println(b); // prints: 3054062.5
        System.out.println(a + b); // prints: 3054071.5
    }
}

```

Float has a limited precision.

This might lead to unexpected results!

Mixing int and float

```java
    public class Calc {
        public static void main(String[] args) {
            float a = 9.3f;
            int b = 3;
            System.out.println(a + b); // prints: 12.3
            float c = a + b;
            System.out.println(c); // prints: 12.3
        }
    }

```

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.

```java
    public class Calc {
        public static void main(String[] args) {
            String hello = "Hello World!";
            System.out.println(hello); // print: Hello World
    !
        }
    }

```

Concatenation

```java
public class Calc {
    public static void main(String[] args) {
        String hello = "Hello";
        String world = " World!";
        String sentence = hello + world;
        System.out.println(sentence);
        System.out.println(hello + " World!");
    }
}

```

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

Strings and Numbers

```
public class Calc {
    public static void main(String[] args) {
        int factorA = 3;
        int factorB = 7;
        int product = factorA * factorB;
        String answer =
            factorA + " * " + factorB + " = " + product;
        System.out.println(answer); // prints: 3 * 7 =
21
    }
}

```

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