Java

Introduction

Julius Felchow (Mail - julius.felchow@mailbox.tu-dresden.de),
Benjamin Weller (Mail - benjamin.weller@tu-dresden.de)

13. November 2019

Java-Kurs

Overview

1. Proceeding

2. Your first program
   - Hello World!
   - Setting up IntelliJ IDEA

3. Basics
   - Some definitions
   - Calculating
   - Text with Strings

Proceeding

About you

- What are you studying (and what semester)?
- Have you programmed before?
- Do you have any experience with Java? (I hope not :))
- What are your Expectations?

About this course

Requirements

- None
- If you can, bring your own computer

Proceeding

- There will be about 14 lessons
- We cover some topics and have excercises prepared
- We are gonna try to adjust to the progress of the course

Some resources

- Ask us
- 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`
- Material-Repository
  `https://github.com/pibebtol/java-lessons`

About Java

Pros:

- Platform-independent (JVM)
- “Easy” to use and very little to worry about
- Broad appliance in many areas

  – > Good Place to start

Cons:

- Considered slow
- No multi-inheritance
- Mediocre support for more complex programming paradigms

  – > Neither fast, small nor geeky

Your first program

Setup

- Install JDK (V13.x (not that relevant))
- To test - open terminal and enter
  `java -version`
- Problems? - Path-Variable (ask me or Google)

Creating your Working Environment

Open the Terminal (to compile and run your Javacode from the command line)

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

Hello World!

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

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

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    }
```

How to run your program

Save your program and then on the command line:

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

Hello World in an IDE

DEMO

Receive a copy of IntelliJ IDEA

IntelliJ IDEA is a powerful IDE[1]

- Get an Ultimate-License at
  - >`https://www.jetbrains.com/student/`
- Available for multiple programming languages

Ecplipse is a free and open-source IDE

- Downloadable at
  - >`https://www.eclipse.org/`
- Supports multiple programming languages, but especially java

[1] 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  }
```

You should always document your code.

Code is read more often than it is written.

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

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!");
5         }
6     }
```

Statements and Blocks:

Statements are *always* concluded by Semicolons.

Everything between { and } is a *block*.

Blocks may be nested.

Blocks do not need to be closed by a semicolon.

Naming of Variables

- Names of classes are always written uppercase.
- The names of variables and functions 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  }
```

Declaration of Variables

- Variables have to be declared before being able to initialize ("set values") it.
- Variables have to be initialized, before being able to use them.

```
1              int a; // declaration of a
2              a = 8; // initialization of a
3              int b = 5; // this works
4
5              b = a + c; // c is not assigned -> error
6
7              String name = "anna"; // string initialization
```

Calculating with int i

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

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; // initialization 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  }
```

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

```
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    }
```

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    }
```

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    }
```

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    }
```

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    }
```

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  }
```

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