Java

Introduction

Hannes Ueck, Jakob Krude

12. November 2020

Java-Kurs

Overview

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/fsr/java-lessons`
- 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)[1]
- Important Frameworks (Spring, Android)
  - $-$ > Easy to use and very little to worry about

---

[1] 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 IDE

IntelliJ IDEA is a powerful IDE[2], e.g. for Java.

- You can download and install[3] 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

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

---

[2]Integrated Development Environment
[3]https://www.jetbrains.com/help/idea/installation-guide.html

And ...

- Git (Version Control)
  https://git-scm.com/downloads
- Java SDK [4]
  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

---

[4]Software Development Kit, can also be downloaded within Intellij IDEA

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);
    }
}
```

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

```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

```java
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.

```
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
```

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--;`

```java
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

```java
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*.