Java

Introduction

FSR Informatik

October 11, 2016

Overview

About this course

Requirements

- You know how to use a computer
- Maybe already knowledge in programming languages?

Proceeding

- There will be 9 lessons
- Each covers a topic and comes with excercises

Some resources

- You can ask your tutor
- Join the Auditorium group
  `http://auditorium.inf.tu-dresden.de`
- StackOverflow, FAQs, Online-tutorials, ...
- Official documentation
  `http://docs.oracle.com/javase/tutorial/java/nutsandbolts/index.html`

Some data about Java

- Volcanic island
- Part of Indonesia
- **Population:** 143 Million
  - $>$ 57% of indonesian population
- **Area:** 138,794 km^2
  - $>$ 13th largest island of the world

... the other Java

Pros:

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

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

... the other 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

Code concepts

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

Code concepts

```
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!");
5           }
6       }
7
```

Semicolons conclude all statements.
Blocks do not need a semicolon.

Blocks

```
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
```

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

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 = 21
9           }
10      }
11
```

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