

Java Crash Course

Part I

Institut für Wirtschaftsinformatik
HU-Berlin WS 2004

Sebastian Kolbe
skolbe@wiwi.hu-berlin.de

Overview

- ◆ (Short) introduction to the environment
 - ◆ Linux in the computer lab
- ◆ Getting started with Java
 - ◆ Your first Java-application
 - ◆ Programming elements
 - ◆ Variables
 - ◆ Operators
 - ◆ Predefined functions

Environment

- ◆ Linux or MS-Windows
- ◆ In Room 025 only Linux!
 - ◆ All exercises and homework in this course must be runnable under Linux
- ◆ For the exercises you will need
 - ◆ Editor (e.g. emacs)
 - ◆ System shell
 - ◆ JDK
 - ◆ Web browser (e.g. Konquerer or Firefox)
(->later)

Java

- ◆ Platform independent
- ◆ Widely known
- ◆ Industrial standard
- ◆ Easy to learn
- ◆ Object orientated

Your first Java application

- ◆ Simple displays “Hallo Welt!”
- ◆ To create the program, you have to
 - ◆ Write a java sourcefile
e.g. `HelloWorld.java`
 - ◆ Compile this sourcefile to a bytecode file
e.g. `HelloWorld.class`
 - ◆ Run the program with the Java-interpreter

```
/*
 * The HelloWorld class implements an application that
 * simply displays "Hello World!" to the standard output.
 */

class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hallo Welt!");    // Display the string
    }
}
```

HelloWorld in detail

Class declaration:

Everything in Java is organized in classes.
(Small logical unit which defines a set of variables and methods(routines))

By convention you declare one class per file and name that file the same.

A comment:

Everything between `/*` and `*/` will be ignored by the compiler.

Comments are not important for computers but for humans!

Single line comment:

Everything after `//` will be ignored by the compiler

```
/*
 * The HelloWorld class implements an application that
 * simply displays "Hello World!" to the standard output.
 */

class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hallo Welt!"); // Display the string
    }
}
```

Output:

Write some characters *H a l l o W e l t !*
on the screen

The main method / routine:

Every program need one 'main' routine, where it starts to perform instructions.

At this time take the syntax as a fixed statement!

Brackets and parenthesis:

Every logical statement must be started and ended with curly brackets!

Arguments to methods/functions/routines always uses parenthesis. Arguments are separated with commas.

Be aware of case sensitiveness and the semicolon at the end of instructions!

Your first Java application

- ◆ The Java compiler **javac**
 - ◆ Translates the sourcecode into instructions that the Java Virtual Machine (Java VM) can understand (bytecode)
 - ◆ Produces **.class** files from **.java** files

```
sko@dusssel:/home/sko > javac HelloWorld.java  
sko@dusssel:/home/sko >
```

Your first Java application

- ◆ The Java Virtual Machine (Java VM)
 - ◆ Is implemented by the Java interpreter **java**
 - ◆ Can understand bytecode files (**.class**) and executes them in a way your local computer can understand

```
sko@dussel:/home/sko > java HelloWorld
Hallo Welt!
sko@dussel:/home/sko >
```


Variables

- ◆ The smallest item of a program
- ◆ Can store data
- ◆ Each variable has:
 - ◆ Type (what kind of data, e.g. *int* or *String*)
 - ◆ Identifier (a name)
 - ◆ Scope (characteristic that regularize access)
(see lectures)

Variables

◆ Declaration:

```
int i;  
double epsilon;  
String greeting;
```

◆ Initialization:

```
i = 3;  
epsilon = 0.02;  
greeting = "Hallo Welt!";  
int j = 4;
```

// combined!

◆ Using:

```
i = i + j;
```

Primitive data types

Name	Description	Value
byte	Byte-length integer	-128 – 127
short	Short integer	-32768 – 32767
int	Integer	$-2^{32} - 2^{32}-1$
long	Long integer	$-2^{64} - 2^{64}-1$
float	Single-precision floating point	$2^{-149} - (2-2^{23})*2^{127}$
double	Double-precision floating point	$2^{-1074} - (2-2^{52})*2^{1023}$
char	One character	1 unicode char.
boolean	Logical value (true or false)	true/false
String	Text (a string of characters)	

- * String is not really a primitive data type, but can be used as one

Operators

Operator	Use	Description
+	Op1 + Op2	Add Op1 and Op2
-	Op1 - Op2	Subtract Op1 from Op2
*	Op1 * Op2	Multiply Op1 with Op2
/	Op1 / Op2	Divides Op1 by Op2
++	Op1 ++	Adds 1 to Op1
--	Op1 --	Subtracts 1 from Op1
%	Op1 % Op2	Computes div remainder
<	Op1 < Op2	Op1 is less than Op2
>	Op1 > Op2	Op1 is greater than Op2
>=	Op1 >= Op2	Op1 is greater than or equal Op2
<=	Op1 <= Op2	Op1 is less than or equal Op2
==	Op1 == Op2	Op1 is equal Op2
!=	Op1 != Op2	Op1 is not equal Op2
	Op1 Op2	Op1 or Op2
&&	Op1 && Op2	Op1 and Op2
!	!Op1	Negates Op1

Predefined functions

- ◆ In Java exists a huge database of predefined routines and functions organized in hierarchic libraries
- ◆ Simple Output
 - ◆ Use single routines with full path

```
java.lang.System.out.println("Hallo Welt!");
```

- ◆ Prefix "java.lang" is omittable

Example

```
{  
    int i = 7;  
    double pi = 3.14159;  
    System.out.println( "Hello World!" );  
    System.out.println( "PI=" + pi );  
    System.out.println( "i+pi=" + (pi + i) );  
}
```

Output:

```
Hello World!  
PI=3.14159  
i+pi=10.14159
```