Java Crash Course
Part I

School of Business and Economics
Institute of Information Systems
HU-Berlin WS 2005

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
    - Simple predefined functions

Purpose of this course

- You need/take advantage of this course when:
  - You don't know about using the computer lab
  - You don't know about Java

- 1. week: First introduction
- 2. week: Programming elements
- 3. week: Object orientated programming with Java

Environment

- Linux or MS-Windows
- In Room 025 only Linux!
  - All exercises and homework in this course and ISI must be runnable under Linux in the Room 025 (This is reference!)

- For the exercises you will need
  - Editor (e.g. xemacs, kwrite, nEdit, UltraEdit or similar)
  - System shell (Konsole/xterm, command shell)
  - JDK (Version 1.5, download at www.java.sun.com)
  - Web browser (e.g. Konquerer or Firefox) (->later)

Java

- Platform independent
  (works on MS-Windows, Linux, MacOS,...)
- Widely known
- Industrial standard
- Easy to learn
- Object orientated
- Java Development Kit *(JDK)* free available

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 with the classname + .java.

A comment:

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

Comments are not important for computers but for humans!

```
/*
 * 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
   }
}
```

Single line comment:

Everything after // will be ignored by the compiler

Output:

Write some characters *`Hallo Welt!`* on the screen

The main method / routine:

Every program need one 'main' routine, where it starts to perform instructions. Command line arguments are stored in 'args'.

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 by 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@dussel:/home/sko > javac HelloWorld.java
sko@dussel:/home/sko >
```

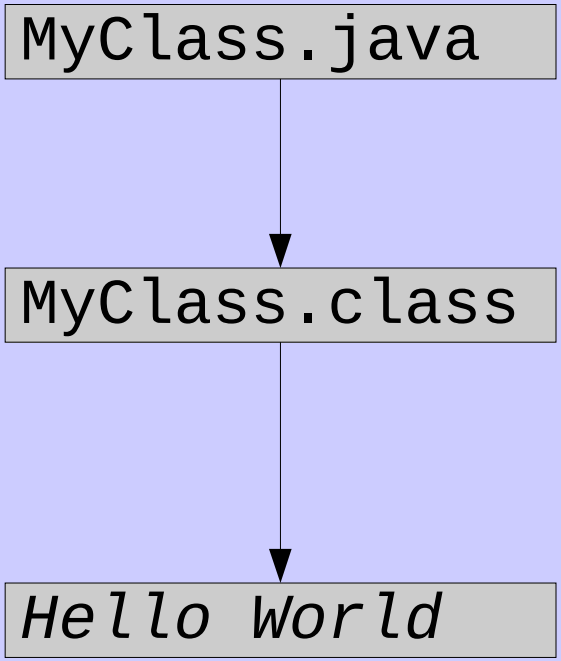
Your first Java application

- 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 computer can understand

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

From .java to a running program

```
kwrite MyClass.java &
```




```
javac MyClass.java
```

```
java MyClass
```

Variables

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

- Several variables of the same type can be stored in an array (see lectures)

Variables

- Declaration:

```
int i;
double epsilon;
String greeting;
String[] sliste;
```

- Initialization:

```
i = 3;
epsilon = 0.02;
greeting = “Hallo Welt!”;
int j = 4;                                  // combined!
sliste = new String[10];
```

- Using:

```
i = i + j;
i += j;          // same as above
j = j + 1;
j += 1;          // same as above
j++;             // same as above
sliste[0] = “Hallo”;
```

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)	

For every primitive data type exists an equivalent class definition. (e.g. int -> Integer)

* 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	Substract Op1 from Op2
*	Op1 * Op2	Multiply Op1 with Op2
/	Op1 / Op2	Divides Op1 by Op2
++	Op1 ++	Adds 1 to Op1
--	Op1 --	Substracts 1 from Op1
%	Op1 % Op2	Computes div remainder
<	Op1 > Op2	Op1 is less than Op2
>	Op2 > 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

Op1 = Operand 1, Op2 = Operand 2

Predefined functions

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

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

  - Prefix “java.lang” is omittable (= default package)
- Calculation in `java.lang.Math`
  - Constants and functions

Example

```
{
 int i = 7;
 double pi = 3.14159;
 String[] argListe = { “Hola”, “Mundo!” };
 System.out.println( "Hello World!" );
 System.out.println( "PI=" + pi );
 System.out.println( "i+pi=" + (pi + i) );
 System.out.println( argListe[0] + “ “ + argListe[1] );
}
```

Output:

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

Homework

- Write an application that:
  - Outputs the first command line parameter on the console

```
> java MyClass   Hello
Hello
>
```

  - Calculates the area of a circle. The radius is given as parameter on command line

```
> java MyCircle   1
The area of a circle with radius 1 is
3.141592653589793
>
```