Programmierpraktikum

Claudius Gros, SS2012

Institut für theoretische Physik
Goethe-University Frankfurt a.M.

Java - A First Glance

programming languages

compiled

- Algol
- C
- C++
- Fortran
- (Latex)
- Lisp
- Java
- Pascal

scripting

- Phyton
- PHP
- Unix shell scripts
- (Word)

computing environments

- Mathlab
- Mathematica

development environments

- Eclipse

compilation and execution

javac – Java programming language compiler

- **`javac`** *`Program.java`*:
  compile the source code *Program.java*
  and create the executable *Program.class*
- the executable *.class* file is independent of the
  operating system (Linux, Windows, ...)

java – the Java application launcher

- **`java`** *`Program`*: launch (execute) the Java executable
  Program.class
- **`java -jar`** *`file.jar`*: launches the Java program in the package

file.jar

- abort with Ctrl + C

javadoc – creating html documentation

- **javadoc** *Program*: scans the Java code *Program.java* for comments
  formatted in **javadoc** style and creates html-doc pages
- all official Java docs are creates this way

a first program

trivial Java source example

- create file *HelloWorld.java*
- copy/paste complete source code
- compile source code using **javac** *HelloWorld.java*
- run program using **java** *HelloWorld*

```
/** Class HelloWorld prints out a standard string.
 * This is a comment in javadoc format.
 * @date 11.11.2011
 * @author Hebert Fasching
 * @version 11.11
 */
public class HelloWorld {
  public static void main(String[] args){
// --- this is another comment, explaining the command
// --- print the line (println) 'Hello World!'
// --- the standard (default) output, the console
    System.out.println("Hello World!");
  }
```

```
}
```

entry point and scope

- **`class`**: the fundamental building block;
  nothing exists outside a **`class`** (object)
- **`main()`**: entry point; here does the execution starts,
  stopping automatically at the end of **`main(){}`**
- **`main()`** is a member function of the **`class`** *`HelloWorld`*;
  there is a strict naming convention
- **`{ ... }`**: scope
  - *defines begining and end of a class, function, control loop, ...*
  - *everything defined within a given scope is not known outside*
  - *if available use keyboards with english layout*
- a program is a sequence of commands, any command ends with a
  semicolon **`;`**

```
class HelloWorld {
  public static void main(String[] args){
    System.out.println("Hello World!");
  }
}
```

example: calculation

- example programs, systematics of commands and definitions later
- the format specfiers like *%3d* used for formated printing with **prinf** will be explained later

```
/** Evaluates sum of integers.
  */
public class ExampleCalc {
  public static void main(String[] args)
    {

// --- define an integer variable and set its value to zero
    int sum = 0;

// --- loop example: do a calculation many times
// --- increase 'index' by one as long as 'index' is smaller than 10
    for (int index=0; index<10; index=index+1)
      {
      sum = sum + index;
      System.out.printf("index and sum: %2d %3d\n",index, sum);
      }  // the loop over 'index' ends here
    }  // the main() routine of the class ExampleCalc ends here
} // the class ExampleCalc ends here
```

example: functions

- `.` : dot operator for access to member functions and variables
- **`return`** the result of a function
- predefined mathematical functions in **`class Math`**

```
/** Evaluates the area of a circle.
  */
public class ExampleFunction {

/** Caclulates the area of a circle with of a given input radius.
  * Both input and return value are real (double precison) numbers.
  */
  static public double area(double radius)
    {
// --- the predefined (real) constant 'PI' of the predefined
// --- class Math is accessed via the dot operator
    return Math.PI*radius*radius;
    }

  public static void main(String[] args)
    {

// --- define real(double) precision variables
    double result;
```

```
    double inputRadius = 10.0;

// --- the member function area() of the class ExampleFunction
// --- is accessed via the dot operator
    result = ExampleFunction.area(inputRadius);

// --- a command end with a semicolon, can wrap an
// --- arbitrary number of lines
    System.out.printf("radius and area: %10.4f %10.4f\n",
                      inputRadius, result);

// --- another time
    inputRadius = Math.sqrt(10.0);
    result = ExampleFunction.area(inputRadius);
    System.out.printf("radius and area: %10.4f %10.4f\n",
                      inputRadius, result);

    }
}
```

example: input and output

- **Scanner**: a predefined Java class to access console input
- information about Java classes available on the web, → search
- with **new** a new object is created (instantiated), more details lateron

```
import java.util.Scanner;

public class ExampleIO {

/** A simple subroutine function multiplying two integer numbers
  * and returning the integer (int) result.
  */
  public static int area(int length, int width) {
    int result = length * width;                          // make calculation
    return result;                                        // return result
  }

  public static void main(String[] args) {

// --- creating an object of type Scanner named scanner
// --- we will learn lateron what this does exactly means
    Scanner scanner = new Scanner(System.in).useDelimiter("\n");
```

```
    System.out.print("Enter length: ");                    // input
    int a = scanner.nextInt();                             // length
    System.out.print("Enter width: ");
    int b = scanner.nextInt();                             // width

// -- call the (static) function area() of the class ExampleIO
    int calculatedValue = ExampleIO.area(a, b);
    System.out.println("calculation complete");
    System.out.printf("area with length %d and width %d is %d\n",
                      a, b, calculatedValue);              // formatted output
  }   // end of ExampleIO.main()
}   // end of class ExampleIO
```

output basics

- formated ouput is easier to read and
  cuts down debugging time substantially
- **`printf()`** takes exactly the same formating options as in C
  - *%b: boolean*
  - *%c: single char*
  - *%d: integer of arbitrary length*
  - *%s: string of arbitrary length*
  - *%10s: allocat 10 spaces for the string*
  - *%12.4f: real number with 4 decimals*
  - *\n: end of line (linebreak)*

```
public class OutputBasics
{
public static void main(String[] args)
```

```
  {           // non-standard bracket positioning
  final int nArgs = 1;
  if (args.length != nArgs)
    {         // non-standard bracket positioning
    return;
    }

  String[] sentence = {"This","is","a","sentence"};
  System.out.println("  ");
  for (int ss=0; ss<sentence.length; ss++)
    System.out.printf("%s ",sentence[ss]);
  System.out.printf("\n");

  double number = Double.parseDouble(args[0]);
  System.out.println("  ");
  System.out.printf("floating-point %10.3f number\n",number);
  System.out.printf("   exponential %10.3e number\n",number);
  System.out.printf("   fixed-width %10s string  \n","short");
  System.out.format("   fixed-width %10d integer \n",11);
            // printf (from C) and format are equivalent
  }
}
```

```
user@pc:~$ java OutputBasics 13.445

This is a sentence

floating-point     13.445 number
   exponential  1.345e+01 number
   fixed-width      short string
   fixed-width         11 integer
```

language and locale

- the language setting is always via the **locale**
  browser, desktop, Java, ...
- **attention**
  german *3,141*, US *3.141*

```
import java.util.*;      // for Locale, Scanner

public class TestLocale {

public static void main(String[] args) {

// --- in- and ouput with German numbers "3 Komma 141"
  Locale.setDefault(Locale.GERMAN);
  Scanner scanner = new Scanner(System.in).useDelimiter("\n");
  System.out.print("enter German floating point number: ");
  double rr = scanner.nextDouble();
  System.out.printf("you have entered (German): %8.3f\n\n",rr);

// --- in- and ouput with US numbers "3 dot 141"
  Locale.setDefault(Locale.US);
  scanner = new Scanner(System.in).useDelimiter("\n");
  System.out.print("enter US floating point number: ");
  rr = scanner.nextDouble();
```

```
  System.out.printf("you have entered (US): %8.3f\n\n",rr);

  }  // end of TestLocale.main()
}  // end of TestLocale
```

```
user@pc:~$ java TestLocale
enter German floating point number: 3,345
you have entered (German):    3,345

enter US floating point number: 7.145
you have entered (US):    7.145
```