Overview

- Why Java?
  - Interoperability
  - Many and well-documented libraries
  - Widespread
  - JIT makes the performance acceptable for most applications
- Goals of this Introduction
  - Understanding the basic concepts
  - Understanding and extending existing Java code
  - Complete overview of all Java language features?
  - Writing whole applications in Java?

### 1 – Hello World

Has to be saved in HelloWorld.java

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

Starting point of any Java application

Formal parameter of Type String Array

System output stream. Appears on the console (shell or Eclipse Console)

Not restricted to Strings!

### 2 – Fibonacci

```java
public class Fibonacci {
    public static void main(String[] args) {
        System.out.println("1. Fibonacci Number: " + oldNumber);
        System.out.println("2. Fibonacci Number: " + newNumber);
        // generate the remaining numbers
        for (int i=3; i<=counter; i++) {
            int temp = oldNumber + newNumber;
            oldNumber = newNumber;
            newNumber = temp;
            System.out.println("i \(+\)th Fibonacci Number: " + newNumber);
        }
    }
}
```

Comments like in C:

```java
// ... or /* ... */
```

Primitive Types: char, byte, short, int, long, float, double, boolean, void

Plus Boxed Types: Character, Byte, Short, Integer, Long, Float, Double, Boolean, Void

Mixing of different types is possible!
3 – Fibonacci extended

```java
public class Fibonacci {

    public static void main(String[] args) {
        // declaration of variables
        int counter = 10;
        int oldNumber = 0;
        int newNumber = 1;

        // parse counter value
        counter = Integer.parseInt(args[0]);
        System.out.println("Printing the first " + counter + " numbers: ");

        // the first two Fibonacci Numbers are predefined
        System.out.println("1. Fibonacci Number: " + oldNumber);
        System.out.println("2. Fibonacci Number: " + newNumber);

        // generate the remaining numbers
        for (int i=3; i<=counter; i++) {
            int temp = oldNumber + newNumber;
            oldNumber = newNumber;
            newNumber = temp;
            System.out.println(i + ". Fibonacci Number: " + newNumber);
        }
    }
}
```

Distributed Computing Group
Java Tutorial
5

3 – Fibonacci extended

```java
... // parse counter value
try {
    counter = Integer.parseInt(args[0]);
    System.out.println("Printing the first " + counter + " numbers: ");
}

// the first two Fibonacci Numbers are predefined
System.out.println("1. Fibonacci Number: " + oldNumber);
System.out.println("2. Fibonacci Number: " + newNumber);

// generate the remaining numbers
for (int i=3; i<=counter; i++) {
    int temp = oldNumber + newNumber;
    oldNumber = newNumber;
    newNumber = temp;
    System.out.println(i + ". Fibonacci Number: " + newNumber);
}

catch (NumberFormatException e) {
    System.out.println("Sorry, first argument must be a number");
    return;
}

Exceptions from the try-Block
Distributed Computing Group
Java Tutorial
6

4 – Fibonacci even more extended

```java
... // parse counter value
System.out.println("How many Fibonacci Numbers ... ? ");
counter = readInteger();

// the first two Fibonacci Numbers are predefined
System.out.println("1. Fibonacci Number: " + oldNumber);
System.out.println("2. Fibonacci Number: " + newNumber);

// generate the remaining numbers
for (int i=3; i<=counter; i++) {
    int temp = oldNumber + newNumber;
    oldNumber = newNumber;
    newNumber = temp;
    System.out.println("" + i + ". Fibonacci Number: " + newNumber);
}

static int readInteger() {
    String line;
    BufferedReader input = new BufferedReader(
        new InputStreamReader(System.in));
    try {
        line = input.readLine();
        return Integer.parseInt(line);
    }
    catch (Exception e) {
        return 0;
    }
}
```

Distributed Computing Group
Java Tutorial
7

5 – Classes in Java

• Instance versus static fields / methods

```java
System.out.println("Hello World!");
new Integer(5).toString();
```

Distributed Computing Group
Java Tutorial
8

we have never created an instance of „System“. out is a static field: It’s always there (exactly once).

we have created an instance of Integer of value 5. toString() is a method of the instance, it thus returns the value of the instance that we have created.
5 – Classes in Java

public class A {
    public static int j;
    public int k;
    public static void print_j() {
        System.out.println("value j: " + j);
    }
    public void print_k() {
        System.out.println("value k: " + k);
    }
}

A.j = 22;
A.print_j();
A one = new A();
one.k = 4
one.print_k();
A two = new A();
two.k = 10;
two.print_k();

5 – Classes in Java

• Inheritance
  – A inherits from B: B is a specialization of A
  – No multiple inheritance in Java!

• Interfaces
  – A implements interface I: A has the facet I

• Casts
  – if A is a subclass of B, then A can be casted to B
  – If A implements the interface I, then A can be casted to I

6 – Bookmark Management

Bookmarks:
• Contains a list of web sites
• addBookmark() adds a web site entry
• deleteBookmark() removes a web site entry
• toString() for printing an entry

Website:
• name
• url
• description
(all Strings)

Allows the output of the content of an Object o by calling
System.out.print(o)
7 – Threads

Program:

8 – Threads

Examples:

Computation in the background, without blocking the GUI.

Worker thread: Every request is served by a separate thread.

One thread for incoming messages, a second for sending new messages.

Coordination of Threads can be complex and is a potential source for bugs.

9 – Threads

```java
public class MyThread extends Thread {
    public void run() {
        try {
            for (int i=0; i<10; i++) {
                System.out.println("Thread counts "+i);
                sleep(1000);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

public class MyMain {
    public static void main(String[] args) {
        new MyThread().start();
        try {
            for (int i=0; i<10; i++) {
                System.out.println("Main counts "+i);
                Thread.sleep(700);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Thread’s „main method“

Sleep for 1000 milliseconds

Instantiate and start the thread

Inherits from the Thread class