JAVA AND ECLIPSE TUTORIAL

Distributed
Computing
Group

Computer Networks
SS 07

Overview

- Why Java?
  - Interoperability
  - Many and well-documented libraries
  - Widespread
  - JIT makes the performance acceptable for most applications
  - Syntax derived from C
  - Experience: People prefer Java to Oberon / Eiffel / C
- 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

Starting point of any Java application

Formal parameter of Type String *Array*

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

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

Not restricted to Strings!

2 – Fibonacci

Comments like in C: // ... or /* ... */

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

                // 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);
                }
        }
}
```

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

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

known from C: for, while, if-else, switch

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);
                }
        }
}
```

This can easily fail!!

3 – Fibonacci extended

```java
...
// parse counter value
try {
        counter = Integer.parseInt(args[0]);
} catch (NumberFormatException e) {
        System.out.println("Sorry, first argument must be a number");
        return;
} catch (ArrayIndexOutOfBoundsException e) {
        System.out.println("Usage: java Fibonacci <number>");
        return;
}
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);

...
```

Exceptions from the try-Block

4 – Fibonacci even more extended

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

                // the first two Fibonacci Numbers are predefined
                ...
        }

        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;
                }
        }
}
```

additional Method readInteger() reads Integers from stdin

Creation of a new Instance of the class BufferedReader

InputStream: Character-based reading from a source (File, Network, ...)

Counterpart of System.out

Supertype of all Exceptions

5 – Classes in Java

- Instance versus static fields / methods

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

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);
        }
}
```

```
class A
   public static int j;
   public static void print_j();
```

5 – Classes in Java

```
…
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();
…
```

one.print_j() returns 22
two.print_j() returns 22

```
class A
   public static int j; // = 22
   public static void print_j();
```

```
Instance „one"
   public int k; // = 4
   public void print_k();
```

```
Instance „two"
   public int j; // = 10;
   public void 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

```
Class A {
   …
}
```

```
Class B extends A {
   …
}
```

```
Interface I {
   …
}
```

```
Class B implements I {
   …
}
```

A a = new A();
B b = (B) a;

6 – Bookmark Management

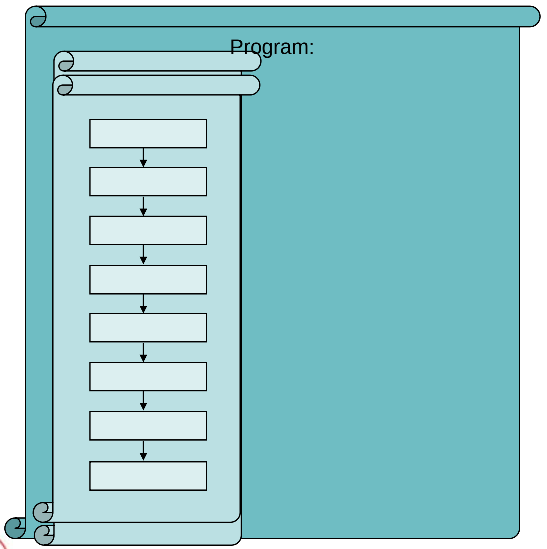
Bookmarks:
- Contains a list of web sites
- addBookmark() adds a web site entry
- deleteBookmark() removes a web site entry
- toString() for printing an entry

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

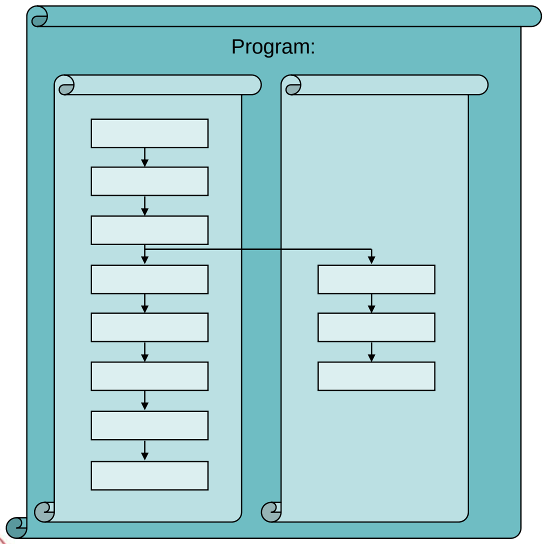
Website:
- name
- url
- description

(all Strings)

7 – Threads



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

```
public class MyThread   extends Thread {
    public void run() {
        try {
                for (int i=0; i<10; i++) {
                    System.out.println(“\t\tThread 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();
        }
    }
}
```

Inherits from the Thread class

Thread‘s „main method“

Sleep for 1000 milliseconds

Instantiate and start the thread