Java Programming
Manuel Oriol, March 22nd, 2007
Goal

• Teach Java to proficient programmers
Roadmap

• Java Basics
• Eclipse
• Java GUI
• Threads and synchronization
• Class loading and reflection
• Java Virtual Machines
• Byte-Code and Just-In-Time compilation
• Java Middleware
• Java Components

and more...
Modus Operandi

• Assistants: Andreas Leitner, Christoph Angerer, Marco Terzer, and?

• Lectures:
  - 2 hours lectures on Thursday
  - 1 hour exercise on Wednesday

• Exercises will be corrected in the exercise lectures, not graded

• Written exam: 50% of the grade

• Project: 50% of the grade
Documents

- Slides - Web
- Course Abstract - Web
- Articles/Reading material - Web links
Java?

• One of the most popular OO programming language out there.
• class-based OO imperative language
• Inspired by C syntax (without pointers, only references).
• “Compile once run anywhere”.
• Broad support.
• Made by SUN Microsystems.
Java History

• Back to 1995
• First named Oak and developed to integrate devices
• Trivia:
  
A very simple program

```java
/**
 * Class used as an example for the students
 * @author manueloriol
 */

public class HelloWorld {

/**
 * Entry point of the program.
 * @param args the arguments of the program
 */

public static void main(String[] args) {
    // prints with a carriage return at the end
    System.out.println("Hello World");
}
```
A simple method

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

- **class method**: `main`
- **return type**: `void`
- **method name**: `main`
- **argument name**: `args`
- **argument type**: `String[]`
- **class**: `System`
- **access to field**: `System.out`
- **call to method**: `println("Hello World")`
A not so simple example

```java
public class HelloWorld {
    String helloString;
    public HelloWorld()
    {
        helloString = "Hello World";
    }

    public void printHello()
    {
        System.out.println(helloString);
    }

    public static void main(String[] args) {
        HelloWorld hl;
        hl = new HelloWorld();
        hl.printHello();
    }
}
```

Field definition

constructor

Field access

temporary variable initialization

assignment

"HelloWorld.java"
Compiling

macmini:~/Documents/workspace/HelloWorld manueloriol$ ls
HelloWorld.java
macmini:~/Documents/workspace/HelloWorld manueloriol$ javac HelloWorld.java
macmini:~/Documents/workspace/HelloWorld manueloriol$ ls
HelloWorld.class HelloWorld.java
macmini:~/Documents/workspace/HelloWorld manueloriol$ .
Executing

macmini:~/Documents/workspace/HelloWorld manueloriol$ ls
HelloWorld.class HelloWorld.java
macmini:~/Documents/workspace/HelloWorld manueloriol$ java HelloWorld
Hello World
macmini:~/Documents/workspace/HelloWorld manueloriol$ 

Getting the JDK

http://java.sun.com/javase/downloads/index.jsp
Forming Groups

- Please form 4 groups by signing up to one of the sheets
Java Basics:
Part 1 - Java Tools

Manuel Oriol, April 5th, 2006
The Big Picture

Source Code

Compiler

Documentation Generator

Byte Code, Program

Virtual Machine

Documentation

Virtual Machine

Compiler
Environment Variables

PATH

CLASSPATH
Java Compiler

javac *.java

• arguments are files with extension .java

• each class has a .class file

• options:
  - `g` for debugging
  - `verbose`
  - `cp` for classpath
  - `d` for output directory
Java Virtual Machine (JVM)

```
java HelloWorld
```

- arguments is a class name containing the main method
- each class loaded on-demand
- options:
  - `-verbose`
  - `-cp for classpath`
  - `-d for output directory`
Java Documentation

javadoc *.java

• comments /** */ and //

• options:
  - -verbose
  - -cp for classpath
  - -d for output directory
Java Archive

```
jar cvf classes.jar *.class
```

- first argument is the target is option f
- options:
  - c create the archive
  - v verbose mode
  - f first argument is target file name
  - x extract archive
Java Debugger

```java
jdb MyMain

• commands during execution:
  
  stop at MyMain:25
  stop in MyMain.myMehtod
  next
  step
  run
  print
```
Java Basics:
Part 2 - Language

Manuel Oriol, March 22nd, 2007
Back to the example

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

"HelloWorld.java"
Packages

- Package Names: general.lessGeneral.precise
- Defines a directory infrastructure
- Fully Qualified Name

```
package general.particular;
```
Primitive Types

- Primitive types: int, byte, char, long, short, boolean
- Comparisons: <, >, ==, >=, <=, !=
- Operators: +, -, *, /
- Equivalent classes: Integer, Byte, Character, Long, Short, Boolean
Reference Types

• Classes

• Generic classes

• Interfaces
Class

- abstract
- final
- public

```java
public abstract class MyClass{
    ...
}
```
Interfaces

• Only signature of methods
• has to be implemented in classes

```java
public interface MyInterface{
    ...
}
```
Inheritance

• Single inheritance

• implementation of interfaces to simulate multiple inheritance

```java
public class MyClass extends Object implements MyInterface {
    ...
}
```
Arrays

- Type []
- Declared when using variables

```java
public static void main(String[] args){
    int[] a = new int[5];
    a[2] = 2;
}
```
Generic Classes

- Classes dependent on another class
- (Will come back to this)

```java
public class MyClass < E >{
    ...
}
```
Variables

- Local Variables
- Instance Variables
- Class Variables
Local Variable

• Declared in the code (no matter the location)
• Local to the current block ({} )
• Must be initialized before using them

```java
int a;
int b=3;
MyClass c;
String s="Hello World!";
```
Instance Variables

- Declared the same way as local variables but outside any method
- final, transient, volatile
- Visibility:
  - public
  - protected
  - default
  - private

```java
public int a;
private int b = 3;
MyClass c;
protected String s = "Hello World!";
```
Class Variables

- As instance variables but with static
- shared by all instances

```java
static int b=3;
```
this

• refers to the current object
Methods

- Constructors
- Instance methods
- Class methods
Constructors

• have the same name as the class
• do not return anything
• begin with a call to the constructor from the parent class (super(...))

```java
public class MyClass{
    public MyClass(){}
    ...
    MyClass mc = new MyClass();
}
```
Instance Method

- are called on a reference to an instance
- can return values
- C-style declaration and value return

```java
public class MyClass{
    public boolean test(){
        return true;
    }
}
...
MyClass mc=new MyClass();
mc.test();
```
Class Method

- Are called on the class name, instance names
- declared static

```java
public class MyClass{
    public static boolean test(){
        return true;
    }
    ...
    MyClass.test();
}
```
The main method

• static, returns nothing, public, String[] as parameters

```java
public class MyClass{
    public static void main(String[] args){...}
}
```
Inheritance

• Methods and fields are inherited
• They can be used in a child class without any redeclaration
• They can be overriden (contravariant redef. return types, novariance of arguments)
• Overloading is ok
Exercises

- final & volatile?
- static and multithreaded?
- local variable usable from another class?
- declare a String that would be visible from outside the package, shared by all instances and can be accessed concurrently.
Why doesn’t it compile?

```java
public class HelloWorld {
    public String message = "Hello World!";
    public HelloWorld(String s) {
        super();
        message = s;
    }
}

class HelloWorld2 extends HelloWorld {
    public HelloWorld2(String s) {
        message = s;
    }
}
```