

The Java Environment



Version 1.3 April 2009

Java Timeline

- 1991: SUN develops a programming language for cable TV set-top boxes
- Simple, OO, platform independent
- 1994: Java-based web browser (HotJava), the idea of “applet” comes out
- 1996: first version of Java (1.0)



Java timeline (cont'd)

- 1996: Netscape supports Java
 - Popularity grows
 - Java 1.02 released, followed by many updated releases in close rounds
- 1997: Java 1.1 released, major leap over for the language
- 1998: Java 2 platform (v. 1.2) released (libraries)
- 2005: Java 5 (language enhancements)
 - New features marked with
- 2007: Java 6 (Faster Graphics)



3

OO language features

- OO language provides constructs to:
 - ♦ Define classes (types) in a hierarchic way (inheritance)
 - ♦ Create/destroy objects dynamically
 - ♦ Send messages (w/ dynamic binding)
- No procedural constructs (pure OO language)
 - ♦ no functions, class methods only
 - ♦ no global vars, class attributes only



4

Java features

- Platform independence (portability)
 - ♦ Write once, run everywhere
 - ♦ Translated to intermediate language (bytecode)
 - ♦ Interpreted (with optimizations, e.g. JIT)
- High dynamicity
 - ♦ Run time loading and linking
 - ♦ Dynamic array sizes
- Automatic garbage collection



5

Java features (cont'd)

- Robust language, i.e. less error prone
 - ♦ Strong type model and no explicit pointers
 - Compile-time checks
 - ♦ Run-time checks
 - No array overflow
 - ♦ Garbage collection
 - No memory leaks
 - ♦ Exceptions as a pervasive mechanism to check errors



6

Java features (cont'd)

- Shares many syntax elements w/ C++
 - ♦ Learning curve is less steep for C/C++ programmers
- Quasi-pure OO language
 - ♦ Only classes and objects (no functions, pointers, and so on)
- Basic types deviates from pure OO...
- Easy to use

Java features – Classes

- There is one first level concepts: the class

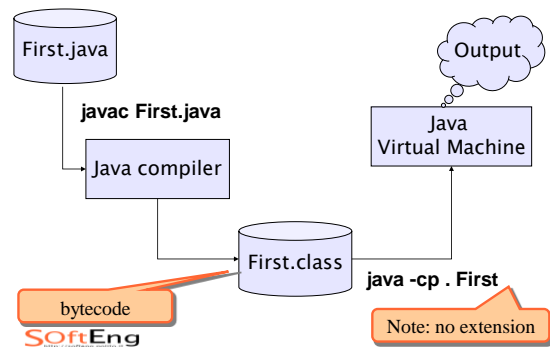

```
public class First {
}
```
- The source code of a class sits in a *.java* file having the same name
 - ♦ Rule: one file per class
 - ♦ Enforced automatically by IDEs

Java features – Methods

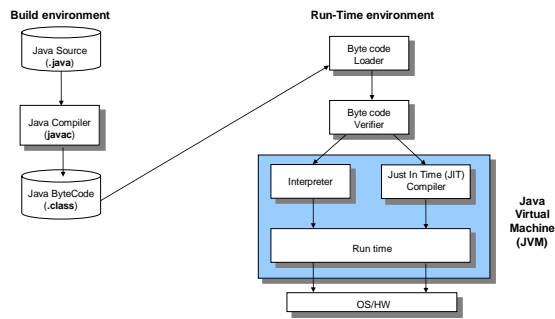
- In Java there are no functions, but only methods within classes
- The execution of a Java program starts from a special method:


```
public static void main(String[] args)
```
- Note
 - ♦ return type is void
 - ♦ args[0] is the first argument on the command line (after the program name)

Build and run



Building and running (simple)



Dynamic class loading

- JVM loading is based on the **classpath**:
 - ♦ list of locations whence classes can be loaded
- When class X is required, for each location in the classpath:
 - ♦ Look for file X.class
 - ♦ If present load the class
 - ♦ Otherwise move to next location

Example

- File: First.java:

```
public class First {
    public static void main(String[] args) {
        int a;
        a = 3;
        System.out.println(a);
    }
}
```

SoftEng

Java features (cont'd)

- Supports “programming in the large”
 - ♦ JavaDoc
 - ♦ Class libraries (Packages)
- Lots of standard utilities included
 - ♦ Concurrency (thread)
 - ♦ Graphics (GUI) (library)
 - ♦ Network programming (library)
 - socket, RMI
 - applet (client side programming)

SoftEng

14

Types of Java programs

- **Application**
 - ♦ It's a common program, similarly to C executable programs
 - ♦ Runs through the Java interpreter (java) of the installed Java Virtual Machine

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

SoftEng

15

Types of Java programs

- **Applet** (client browser)
 - ♦ Java code dynamically downloaded
 - ♦ Execution is limited by “sandbox”
- **Servlet** (web server)
 - ♦ In J2EE (Java 2 Enterprise Edition)
- **Midlet** (mobile devices, e.g. smartphone and PDA)
 - ♦ In J2ME (Java 2 Micro Edition)

SoftEng

16

Java development environment

- JSE 1.6.x
 - ♦ javac compiler
 - ♦ jdb debugger
 - ♦ JRE (Java Run Time Environment)
 - Interpreter
 - Native packages (awt, swing, system, etc)
- Docs
 - ♦ <http://java.sun.com/j2se/1.6.0/docs/>
- Eclipse integrated development environment
 - ♦ <http://www.eclipse.org/>

SoftEng

17

Coding conventions

- Use **camelBackCapitalization** for compound names, not underscore
- Class name must be capitalized
- Method name, object instance name, attributes, method variables must all start in lowercase
- Constants must be all uppercases (w/ underscore)
- Indent properly

SoftEng

18

Coding conventions (example)

```
class ClassName {  
  
    final static double PI = 3.14;  
  
    private int attributeName;  
  
        public void methodName {  
            int var;  
            if ( var==0 ) {  
                }  
        }  
    }  
}
```

SoftEng

19

Deployment – Jar

- Java programs are packaged and deployed in **jar** files.
- Jar files are essentially compressed archives (like zip files)
- It is possible to directly execute the contents of a jar file from a JVM

SoftEng

Jar command

- A jar file can be created using:
`jar cvf my.jar *.class`
- The contents can be seen with:
`jar tf my.jar`
- To run a class included in a jar:
`java -cp my.jar Prova`
 - ♦ The “-cp my.jar” option adds the jar to the JVM classpath

SoftEng

Jar Main class

- When a main class for a jar is defined, it can be executed simply by:
`java -jar my.jar`
- To define a main class, a manifest file must be added to the jar with:
`jar cvfm my.jar manifest.txt`



Main-Class: Prova

SoftEng

FAQ

- Which is more “powerfull”: Java or C?
 - ♦ Performance: C is better though non that much better (JIT)
 - ♦ Ease of use: Java
 - ♦ Error containment: Java
- How can I generate an “.exe” file?
 - ♦ You cannot. Use an installed JVM to execute the program
 - ♦ GCJ: <http://gcc.gnu.org/java/>

SoftEng

FAQ

- I downloaded Java on my PC but I cannot compile Java programs:
 - ♦ Check you downloaded Java SDK (including the compiler) not Java RTE or JRE (just the JVM)
 - ♦ Check that the shell path include pathToJava/bin
- Note: Eclipse uses a different compiler than javac

SoftEng

FAQ

- Java cannot find a class (ClassNotFoundException)
 - ♦ The name of the class must not include the extension .class:
 - Es. java First
 - ♦ Check you are in the right place in your file system
 - java looks for classes starting from the current working directory

Wrap-up session

- Java is a quasi-pure OO language
- Java is interpreted
- Java is robust (no explicit pointers, static/dynamic checks, garbage collection)
- Java provides many utilities (data types, threads, networking, graphics)
- Java can be used for different types of programs
- Coding conventions are not “just aesthetic”