

## The Java Environment



Version 1.2 Sep 2008

## Java timeline (cont'd)

- 1996: Netscape supports Java
  - ♦ Popularity grows
- 1996: 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 (1.2 ver) released (libraries)
- 2005: Java 5 (language enhancements)
  - ♦ New features marked with 

SoftEng

4

## Learning objectives

- Understand the basic features of Java
  - ♦ What are portability and robustness?
- Understand the concepts of bytecode and interpreter
  - ♦ What is the JVM?
- Learn few coding conventions
  - ♦ How shall I name identifiers?

SoftEng

2

## 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

SoftEng

5

## 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)

SoftEng

3

## 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

SoftEng

6

## Java features (cont'd)

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

## 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)

## 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

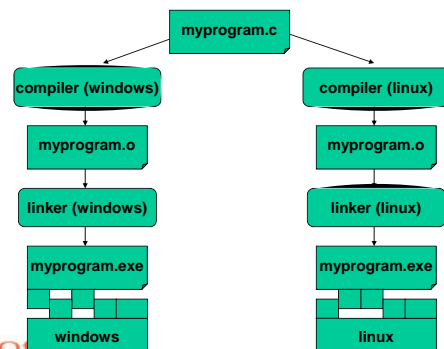
- Interpreted
- Dynamic linking

## Java features – Classes

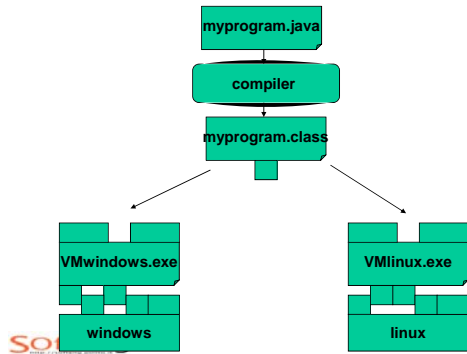
- There is one first level concept: the class

```
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

## C: compilation + execution



## Java: compilation + interpretation



SoftEng

## Example

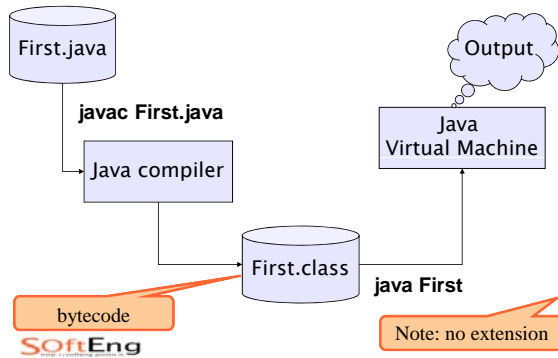
- File: First.java:

```

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

SoftEng

## Compile and run



SoftEng

## Java - compile and run

File HelloWorld.java

```

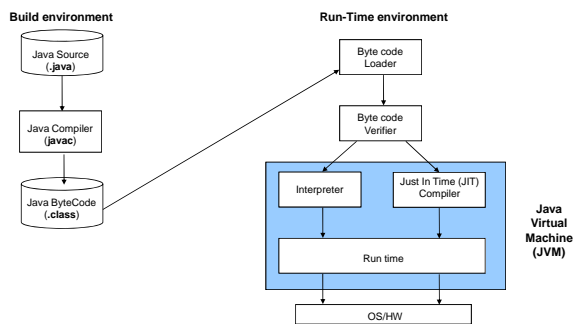
public class HelloWorld {
    public static void main(String args[]){
        System.out.println("Hello world!");
    }
}
  
```

```

> javac HelloWorld.java -- compile
> java HelloWorld -- run
HelloWorld!
>
  
```

SoftEng

## Building and running (simple)



SoftEng

15

## Dynamic linking

- Byte code for a class is loaded when needed at run time
- (static linking - all executable code is prepared *before* run time)

SoftEng

## Ex.

File HelloWorld2.java

```
public class HelloWorld2 {
    public static void main(String args[]){
        System.out.println("Hello world!");
        Bidone b = new Bidone();
        b.print(); } }
```

File Bidone.java

```
public class Bidone {
    public Bidone(){
        public print(){
            System.out.println("Hello Bidone;")
        }
    }
}
```

## Where to look for files

- file myclass.class
  - ♦ in directory <dir> defined by
  - ♦ SET CLASSPATH = <dir>
    - one folder for all programs
  - ♦ java -classpath <dir> myclass.class
    - One folder per each program

SoftEng

## Ex. compilation

```
> javac HelloWorld2.java → Produces file HelloWorld2.class
> javac Bidone.java → Produces file Bidone.class
```

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

23

## Ex. dyn link

```
> java HelloWorld2
JVM searches for and loads file HelloWorld2.class
executes its byte code

public class HelloWorld2 {
    public static void main(String args[]){
        System.out.println("Hello world!");
        Bidone b = new Bidone();
        b.print(); } }

Prints Hello World!
```

JVM searches for and loads file Bidone.class  
(dynamic linking)

SoftEng

## 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

24

## 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)

## Coding conventions (example)

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

## Java development environment

- JSE 1.6.0\_03
  - ♦ 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 editor
  - ♦ <http://www.eclipse.org/>

## Wrap-up session

- Java is a quasi-pure OO language
- Java is interpreted
- Java is robust (no 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”

## 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

## FAQ

- Which is more “powerful”: 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 don't do it. Use an installed JVM to execute the program
  - ♦ GCJ: <http://gcc.gnu.org/java/>

## 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 Prova**
  - ♦ Check you are in the right place in your file system
    - java looks for classes starting from the current working directory

SoftEng

---