The Java Environment

SOftEng
http://softeng.polito.it

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?

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” appeared
- 1996: first version of Java (1.0)

Java timeline (cont'd)

- 1996: Netscape supports Java
  - Popularity grows
  - Java 1.02, followed by many updates in close rounds
- 1997: Java 1.1, major leap over for the language
- 1998: Java 2 platform, v. 1.2 (libraries)
- …
- 2005: J2SE 5 (language enhancements)
- 2007: Java SE 6 (faster graphics)
- 2010: acquisition by Oracle
- 2011: Java SE 7 (I/O improvements)

Java timeline (cont'd)

- 2014: Java SE 8 (language evolutions)
  - Lambda expressions
  - Functional paradigm
- 2017: Java SE 9 (start of a 6-month release plan)
- 2018: Java SE 10, Java SE 11
- 2019: Java SE 12, Java SE 13
- 2020: Java SE 14, Java SE 15
- 2021: Java SE 16, Java SE 17
- … (expected March 2022, September 2022)

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

Java features

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

Java features

- 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

Java features

- 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

- Supports "programming in the large"
  - JavaDoc
  - Class libraries (packages)
- Lots of standard utilities included
  - Concurrency (**`Thread`**)
  - Graphics, GUI (e.g., **`Swing`** library)
  - Network programming
    - **`Socket`**, RMI
    - **`Applet`** (client side programming)

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…
  - … with exceptions

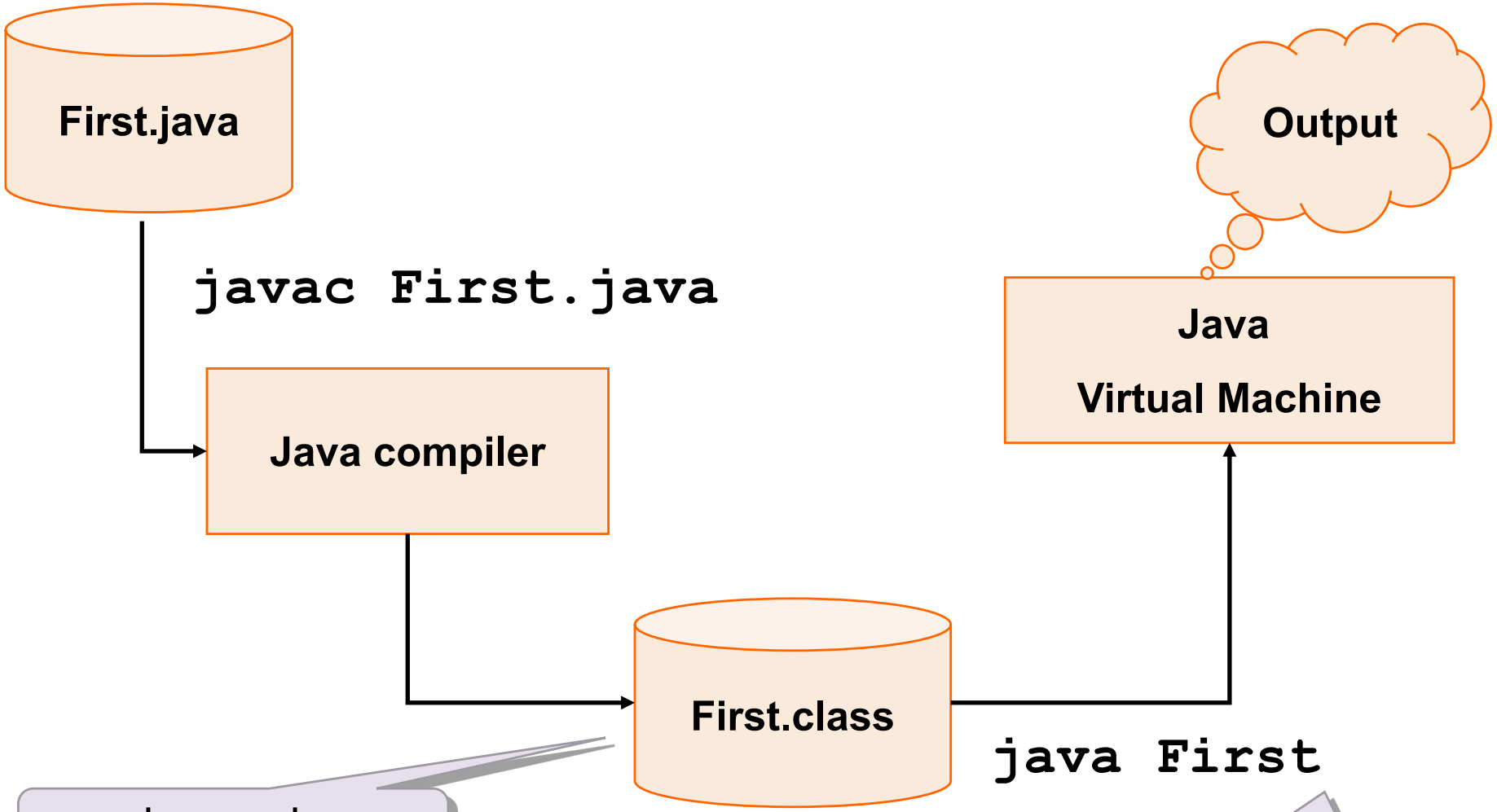
Java features – Methods

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

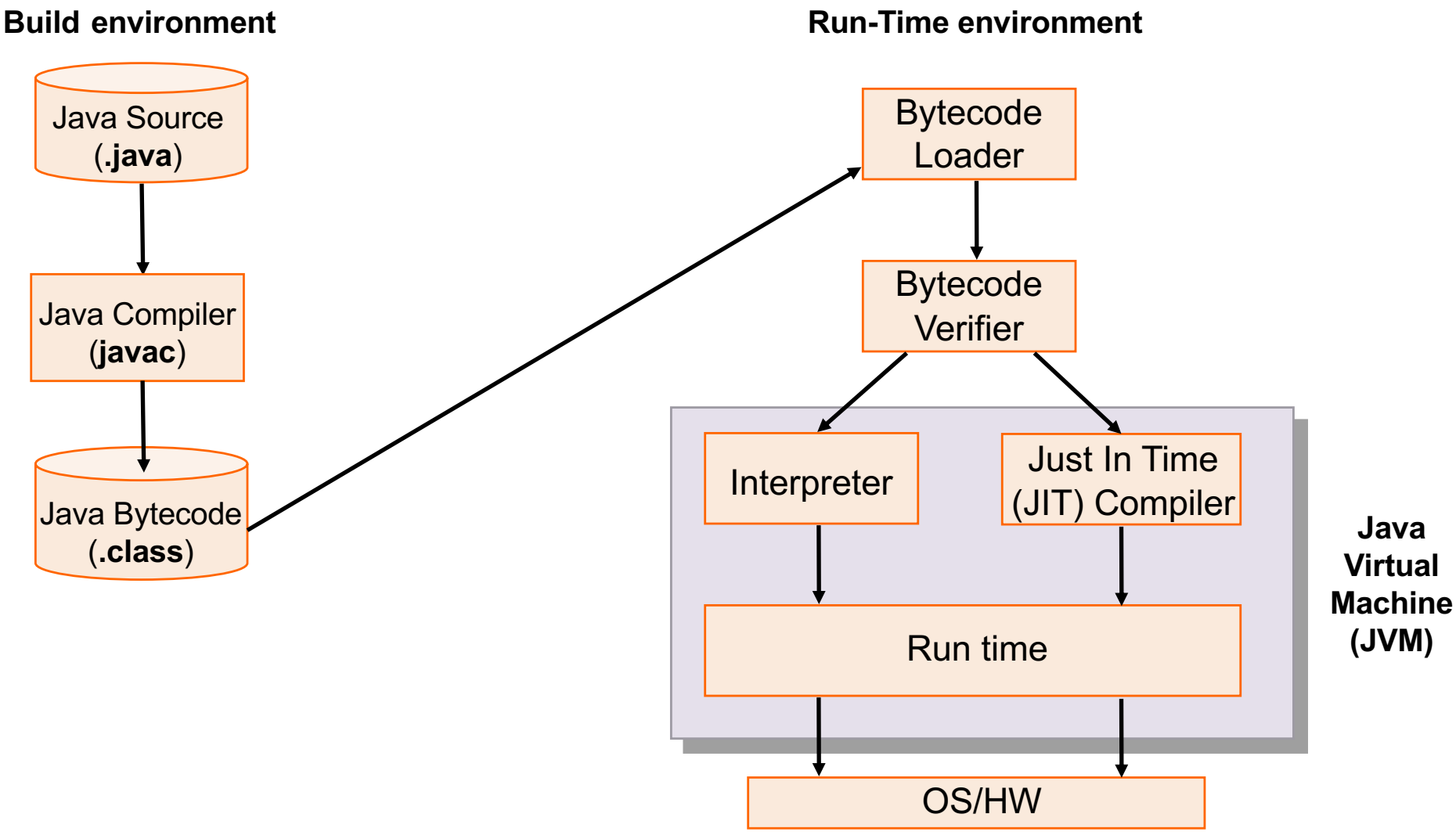
```
public static void main(String[] args)
```

- Note
  - return type is **`void`**
  - **`args[0]`** is the first argument on the command line (after the program name)

Build and run



Build and run



Java Ecosystem

- Java language
- Java platform
  - JVM
  - Class libraries (API)
  - SDK

Dynamic class loading

- JVM loading is based on the classpath:
  - List of locations classes can be loaded from
- 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

Types of Java programs

- Application
  - It is 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!");
 }
}
```

Types of Java programs

- Applet (client browser, deprecated)
  - Java code dynamically downloaded
  - Execution is limited by “sandbox”
- Servlet (web server)
  - In J2EE (Java 2 Enterprise Edition)
- Midlet (mobile devices)
  - In J2ME (Java 2 Micro Edition)
- Android App (Android devices)

Java Development Environment

- Java SE (e.g., 8 / 1.8)
  - **`javac`** compiler, **`jdb`** debugger
  - JRE (Java Run Time Environment)
    - JVM
    - Native packages (awt, swing, system, etc)
  - JDK (Java Development Kit)
- Docs
  - https://docs.oracle.com/javase/8/docs/
- Eclipse integrated development environment
  - http://www.eclipse.org/

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

Coding conventions (example)

```
class ClassName {

final static double PI = 3.14;

private int attributeName;

      public void methodName {
            int var;
            if ( var==0 ) {
            }
      }
}
```

Deployment – Jar

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

FAQ

- Which is more “poweful”: Java or C?
  - Performance: C is better though not so much better (JIT)
  - Ease of use: Java
  - Error containment: Java
- How can I generate an “.exe” file?
  - You cannot! Or you should not! Use an installed JVM to run the program, instead
  - 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 JRE (just the JVM)
  - Check that the path include pathToJavaInstFolder**`/bin`**
- Note: Eclipse uses a different compiler than **`javac`**

FAQ

- Java cannot find a class (**`ClassNotFoundException`**)
  - The name of the class must not include the extension **`.class`**:
    - E.g., **`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 used for different types of programs
- Coding conventions are not “just aesthetic”