Introduction to Java

The objectives of this chapter are:

- To describe the key aspects of Java
- To describe the Java software development kit (SDK)
- To explain the function of the Java Virtual Machine
- To explain the difference between the Java language and its class library (API)

What is Java?

- It is an object-oriented language developed by Sun in the mid 1990s.
  - Original language called Oak
  - Intended for embedded systems

- Unlike C++, it was developed from scratch.
  - The syntax is very similar to C.

- Sun describes it as

"A simple, object-oriented, distributed, interpreted, robust, secure, architecture neutral, portable, high-performance, multi-threaded and dynamic language."

What is Java? (cont'd)

- Object-Oriented
  - Designed to support Object-Oriented concepts
  - However, does contain non-Object-Oriented primitive data types
- Distributed
  - Applications are constructed using objects. Objects can be distributed in multiple locations within a network environment.
  - Extensive integration with TCP/IP
- Interpreted
  - Java compiles to byte-code (not machine code). Byte code is interpreted.
  - Most Java versions after 1.2 include a JIT (Just-In-Time) compiler which compiles byte code to machine code.

What is Java? (cont)

- Robust
  - **Memory management is done automatically**
  - Use of pointers is limited
- Secure
  - All Java code subject to security model.
- Architecture-Neutral/Portable
  - Compiled Java (byte code) will run on any platform which has a Java Virtual Machine
  - The Java Virtual Machine is available for almost all platforms...
    - Even mainframes. (powerful computers used by large organization for critical application)
      - IBM OS z/OS, z/VM

What is Java? (cont)

- High-Performance
  - Originally, Java's performance was poor.
  - Now, Java's performance **contest / rivals C++.**

- Multi-Threaded
  - Processes contain multiple threads of execution.
  - Similar to **multi-tasking but all threads share the same memory space.**

- Dynamic
  - Makes heavy use **of dynamic memory allocation**.
  - Classes can be dynamically loaded at any time.

Platform Independence. How does Java do it?

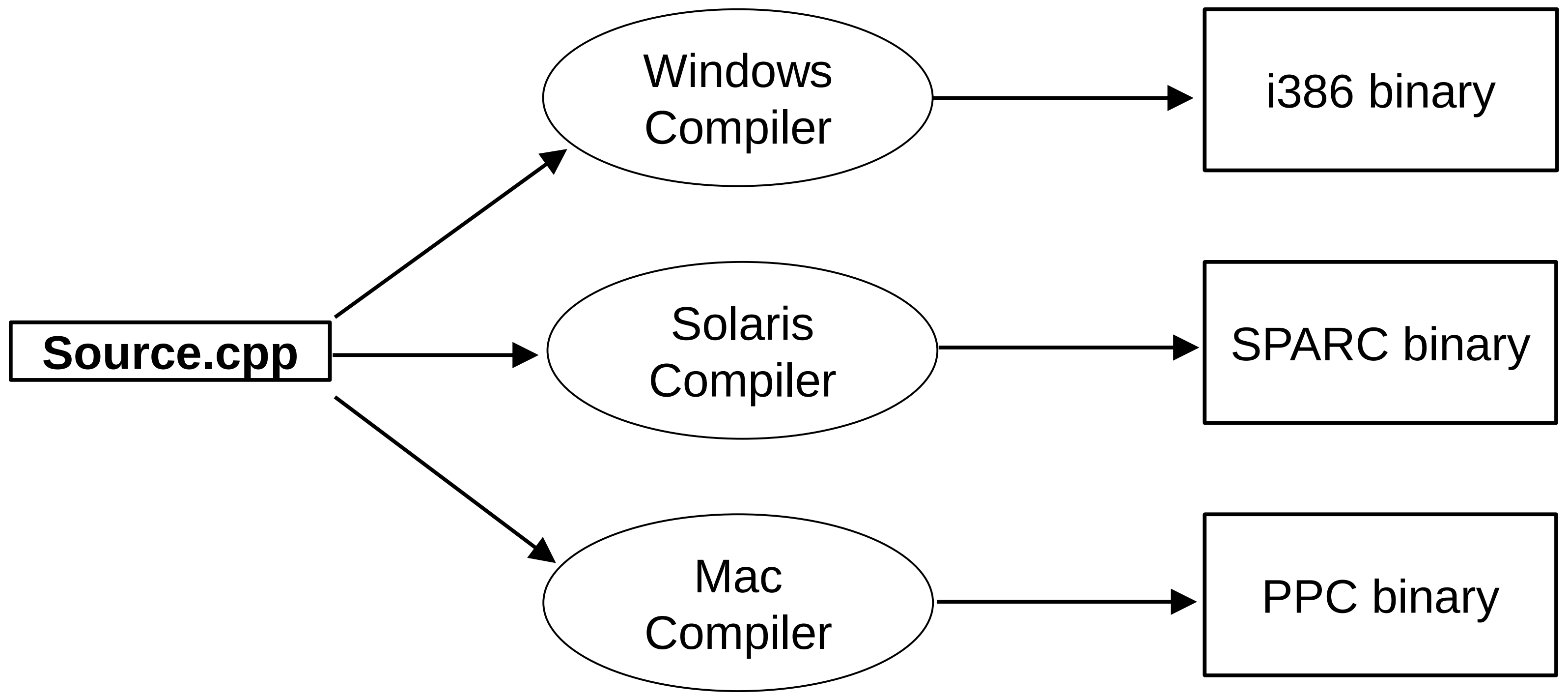
- Java has been described as **WORA** (Write once, Run Anywhere)
  - In most cases, this is true.
  - *Not always true with GUI*.

- Because Java source code is compiled **to byte code** and the byte code is **interpreted**, Java code can be executed anywhere an interpreter is available.

- The "**Interpreter**" is call the Java Virtual Machine **(JVM)**.

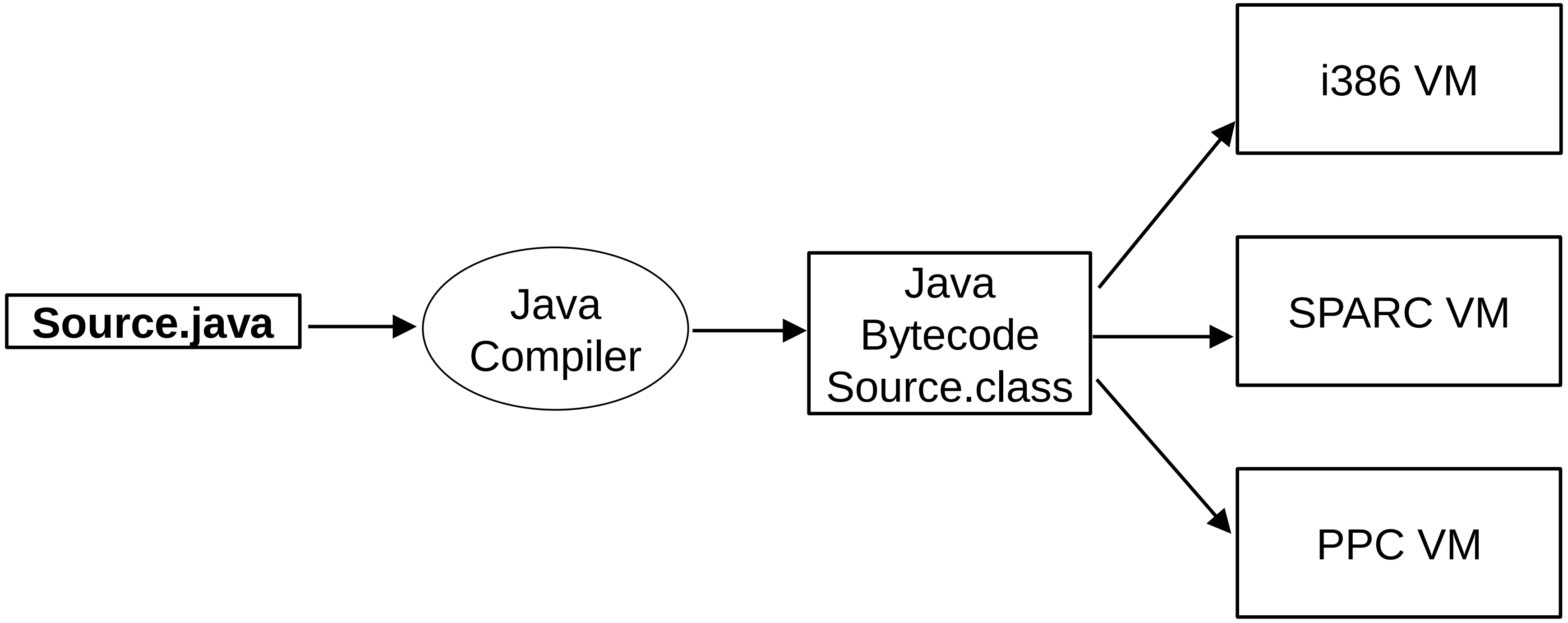
The Java Virtual Machine.

- **Traditionally**, source code had to be compiled for the target hardware and OS platform:



The Java Virtual Machine.

- Java source files (.java) are compiled to **Java byte code (.class)**
- **Byte code** is interpreted on the target platform within a Java Virtual Machine



Java VM Responsibilities

- The Java VM does more than interpret byte code:
  - The class loader loads appropriate java classes.
  - All classes are verified to contain only legal byte codes and not permitted any illegal stack or register usage.
  - A Security Manager can limit access to resources such as the local file system or the network.
  - Any unreferenced memory (Objects) are returned to the system by the Garbage Collector thread.

- **Many database servers, application servers, web servers and browsers contain a Java virtual machine**
  - eg: Oracle, Tomcat (web server), WebSphere (app server), BEA Weblogic (app server), and Netscape and IE.

The Java Software Development Kit (SDK)

- The Java SDK comes in three versions:
  - J2ME - Micro Edition (for handheld and portable devices)
  - J2SE - Standard Edition (PC development)
  - J2EE - Enterprise Edition (Distributed and Enterprise Computing)

- The SDK is a set of command line tools for developing Java applications:
  - javac - Java Compiler
  - java - Java Interpreter (Java VM)
  - appletviewer - Run applets without a browser
  - javadoc - automated documentation generator
  - jdb - Java debugger

- *The SDK is NOT an IDE (Integrated Development Environment)*
  - *Command line only. No GUI.*

Integrated Development Environments (IDEs)

- There are many IDEs available. Some are public domain and some are commercial:
  - Symantic Visual Cafe
  - JBuilder
  - IBM Visual Age
  - Kawa
  - Forte for Java
  - **Eclipse**
  - Netbeans

- Most IDEs offer a "demo" mode so you can try before you buy.

Obtaining the Java SDK

- Download from Sun Web-site:
  - http://www.oracle.com/technetwork/java/javase/downloads/index.html
  - Java SE 7
  - Choose your version
  - Select your platform

- Download will be an installer file appropriate for your platform:
  - Installer .exe for windows
  - rpm or self extracting file for Linux
  - tar or self extracting file for SPARC

- To install, execute the installer program or extract from tar file.

Obtaining the Java Runtime Environment (JRE)

- You might notice, the full SDK is large
- If you only wish to run Java programs, you do not need to install the SDK. Instead, you can install the JRE:
  - Smaller installer file
  - Less time to download
  - No compiler or development tools. Just Java VM and support libraries for specified platform.
  - JRE = JVM + Java Packages Classes(like util, math, lang, awt,swing etc)+runtime libraries

Obtaining the Java API Documentation

- Accompanying the language is a Class library (API)
  - Contains core classes.
  - Contains extensions to Java.
  - The Java API takes a long time to learn.

The API Documentation is available for download.

Packages

- When you view the Java API Documentation, you'll note that the classes are grouped into logical units called "Packages".
- Because there are so many classes, packages provide a mechanism for classifying classes so that they are easier to learn and use.
- Developers can also make use of packages to classify their own classes. This will be discussed later in the course.

Commonly Used Packages

- While it should be our goal to learn as many packages as you can, there are some packages we will use more than others:

Language (general)	java.lang	Common classes used for all application development
GUI	java.awt java.awt.event javax.swing	Graphical User Interface, Windowing, Event processing
Misc. Utilities and Collections	java.util	Helper classes, collections
Input/Output	java.io	File and Stream I/O
Networking	java.net	Sockets, Datagrams

Java Version History

- Even though Java is not very old, there are several key versions to be aware of:
  - Java 1.0.2 - First stable version. Not very useful.
  - Java 1.1 (1997)
    - Security, Database connectivity (JDBC), Improved Performance
    - Most stable version 1.1.8
    - Unstable versions 1.1.4 and 1.1.5
  - Java 1.**2** (1998)   **MAJOR CHANGES**
    - Addition of Swing GUI (mostly replaces AWT)
    - Improved Security
    - Enterprise computing
  - Java 1.3 (2000)
    - Many extended APIs added
    - Improved performance
  - Java 1.4 (2002)
    - Improved performance
    - Bug Fixes

HelloWorld.java

- Here is Java's "HelloWorld" implementation:

In the file, HelloWorld.java:

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

Running HelloWorld

- To compile *HelloWorld.java*, use the compiler. If successful, it will produce a file called HelloWorld.class in the same directory.

```
> javac HelloWorld.java
  [ compiler output ]
```

errors and warnings

- To execute, run the Java VM and include the name of the class which contains the "main" method as the first command line parameter.

```
> java HelloWorld
Hello World
```

note: do not include the .class extension

output from program

Review

- What are the key features of Java?
- How does Java obtain platform independence?
- What is the Java Virtual Machine and what are its responsibilities?
- What is the Java SDK? What is the JRE?
- What is the Java API?
- What are packages?