Java Architecture

Agenda

1. Evolution of Java
2. Java Architecture

Evolution of Java

Key Founders

- Java was the brainchild of:
  - James Gosling
  - Patrick Naughton
  - Chris Warth
  - Ed Frank &
  - Frank Sheridan
- The origin of Java can be traced back to the fall of 1992, and was initially called **Oak**
- Oak was renamed as **Java** in **1995**

Design Goal

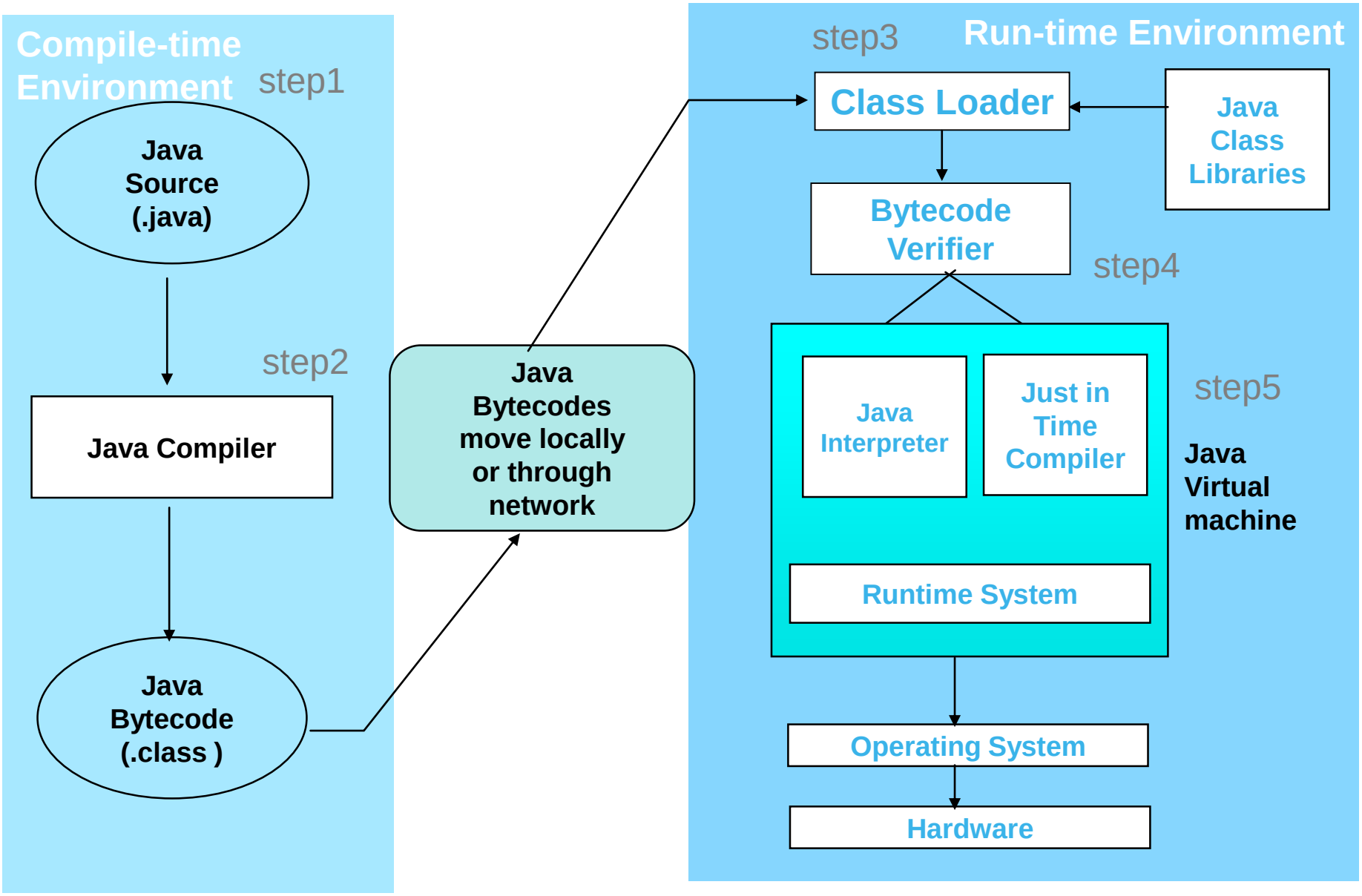
- Java was originally meant to be a **platform-neutral language** for embedded software in devices

- The goal was to move away from platform and OS-specific compilers that would compile source for a particular target platform to a language that would be portable, and platform-independent

- The language could be used to produce platform-neutral code

Java Architecture

Java Architecture



Java Architecture (Contd.).

Step1:

Create a java source code with .java extension

Step2:

Compile the source code using java compiler, which will create bytecode file with .class extension

Step3:

Class loader reads both the user defined and library classes into the memory for execution

Java Architecture (Contd.).

Step4:

Bytecode verifier validates all the bytecodes are valid and do not violate Java's security restrictions

Step5:

JVM reads bytecodes and translates into machine code for execution. While execution of the program the code will interact to the operating system and hardware

The 5 phases of Java Programs

Java programs can typically be developed in five stages:

1. Edit

   Use an editor to type Java program (**Welcome.java**)

2. Compile

   - Use a compiler to translate Java program into an intermediate language called bytecodes, understood by Java interpreter (**javac Welcome.java)**
   - Use a compiler to create **.class** file, containing bytecodes (**Welcome.class**)

3. Loading

   Use a class loader to read bytecodes from **.class** file into memory

The 5 phases of Java Programs (Contd.).

4. Verify

   Use a Bytecode verifier to make sure bytecodes are valid and do not violate security restrictions

5. Execute
   - Java Virtual Machine (JVM) uses a combination of interpretation and just-in-time compilation to translate bytecodes into machine language
   - Applications are run on user's machine, i.e. executed by interpreter with java command (java Welcome)

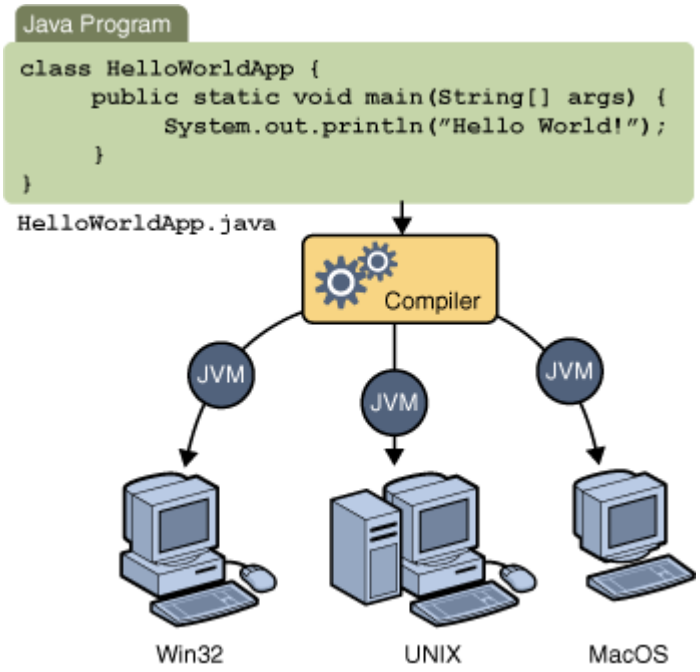
The 5 phases of Java Programs (Contd.).

4. Verify

   Use a Bytecode verifier to make sure bytecodes are valid and do not violate security restrictions

5. Execute
   - Java Virtual Machine (JVM) uses a combination of interpretation and just-in-time compilation to translate bytecodes into machine language
   - Applications are run on user's machine, i.e. executed by interpreter with java command (java Welcome)

Java Virtual Machine



- The output of the compiler is bytecode
- The bytecodes are executed by JVM
- It is an interpreter which converts the byte code to machine specific instructions and executes
- JVM is platform specific

The Java Architecture – The JVM (Contd.).

- Most modern languages are designed to be compiled
- Compilation is a one-time exercise and executes faster
- Execution of compiled code over the Internet an impossibility
- Executable code always generated to a CPU-OS combination
- Interpreting a Java program into byte code facilitates its execution in a wide variety of environments

The Java Architecture – The JVM (Contd.).

- Only the Java Virtual Machine (JVM) needs to be implemented for each platform
- Once the Java runtime package exists for a given system, any Java program can run on it
- The JVM will differ from platform to platform, and is, platform-specific
- All versions of JVM interpret the same Java byte code

The Java Architecture – The JVM (Contd.).

- Interpreted code runs much slower compared to executable code
- The use of bytecode enables the Java runtime system to execute programs much faster
- Java facilitates on-the-fly compilation of bytecode into native code

The Java Architecture – The Adaptive optimizer

- Another type of execution engine is an adaptive optimizer
- The virtual machine starts by interpreting bytecodes
- It also keeps a tab on the code that is running and identifies only the heavily used areas
- The JVM compiles these heavily used areas of code into native code
- The rest of the code, which is not heavily used continues to be interpreted and executed

The Java Architecture - The Class Loader

- The class loader is that part of the VM that is important from:
  - A security standpoint
  - Network mobility

- The class loader loads a compiled Java source file (**.class** files represented as bytecode) into the Java Virtual Machine (JVM)

- The bootstrap class loader is responsible for loading the classes, programmer defined classes as well as Java's API classes

The Java Architecture - The Java .class file

- The Java class file is designed for
  - platform independence
  - network mobility

- The class file is compiled to a target JVM, but independent of underlying host platforms

- **The Java class file is a binary file** that has the capability to run on any platform

Quiz

1. Write the correct order of the Java program execution

A. Class Loader
B. Interpretation
C. Compilation
D. Byte Code Verification
E. Java Source Code
F. Execution

2. Which of the following is used to load a .class file?

A. Class Loader
B. Byte Code Verifier
C. JIT Compiler
D. Interpreter

Quiz

3. When a java program is compiled, it creates a

A. an obj file
B. an exe file
C. a .class file
D. a .sh file

4. The JDK is a superset of the JRE, and contains everything that is in the JRE, plus tools such as the compilers and debuggers necessary for developing applets and applications.

A. TRUE
B. FALSE

Summary

In this session, you were able to :

- Learn about Evolution of Java and forces that shaped it
- Understand Java Architecture along with JVM Concepts