Java Features

Java Features (1)

- **Simple**
  - **It is very easy to learn and its syntax is simple and easy to understand**
  - Removed many complicated features like :
    - pointers and operator overloading
  - rich pre-defined class library
  - No need to remove unreferenced objects, because it has automatic garbage collection

- **Object oriented**
  - focus on the data (objects) and methods manipulating the data
  - Everything in java is an object.

Java Features (2)

- **Robust :**
  - It uses strong memory management.
  - There is a lack of pointers that avoids security problems.
  - There is automatic garbage collection in java which runs on the Java Virtual Machine to get rid of objects which are not being used by a Java application anymore.
  - There are exception handling and the type checking mechanism in Java. All these points make Java robust.
- **Portable:**
  - **Java is portable because it provides you to carry the bytecode to any platform.**
  - java compiler generate byte-codes, not native machine code
  - The compiled byte-codes are platform-independent
  - **same application runs on all platforms**

Java Features (3)

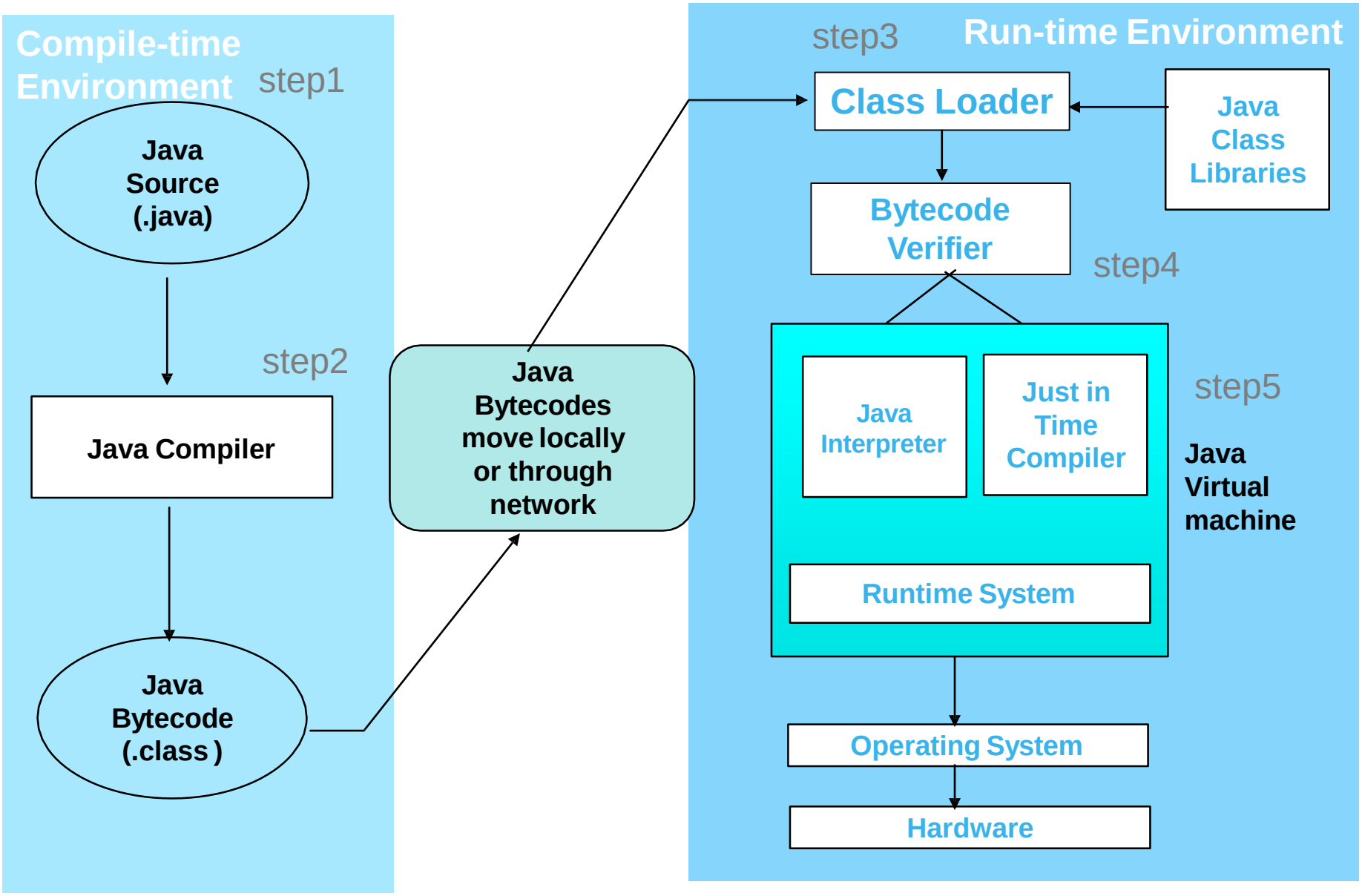
- **Platform Independent:**
  - H/W or S/W environment in which a program runs.
  - Java code is compiled by the compiler and converted into byte code. This byte code is Platform Independent because it will run n any platform.
- **Secure**
  - No explicit pointers
  - memory allocation model is a major defense
  - Java program run inside virtual machine
  - access restrictions are forced (private, public)
  - Class loader :
  - Byte code verifier:
  - Security manager

Java Features (4)

- **Multithreaded**
  - multiple concurrent threads of executions can run simultaneously.
  - Multiple tasks are executed at a same time.
- **Architecture-neutral:**
  - Java is architecture neutral because there are no implementation dependent features, for example, the size of primitive types is fixed.
  - In C programming,- Eg : int

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)

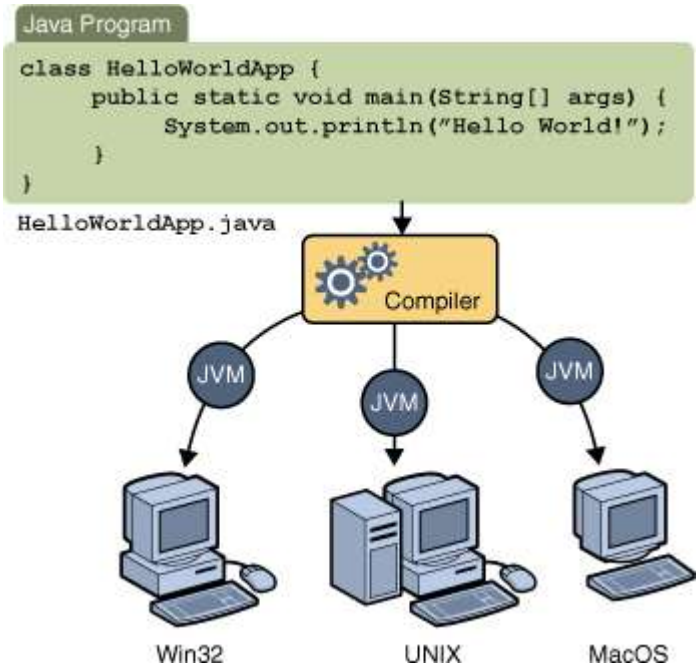
The 5 phases of Java Programs (Contd.).

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
- Only the Java Virtual Machine (JVM) needs to be implemented for each platform
- The JVM will differ from platform to platform, and is, platform-specific.
- Executable code always generated to a CPU-OS Combination.

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 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 Java Architecture - The Java .class file

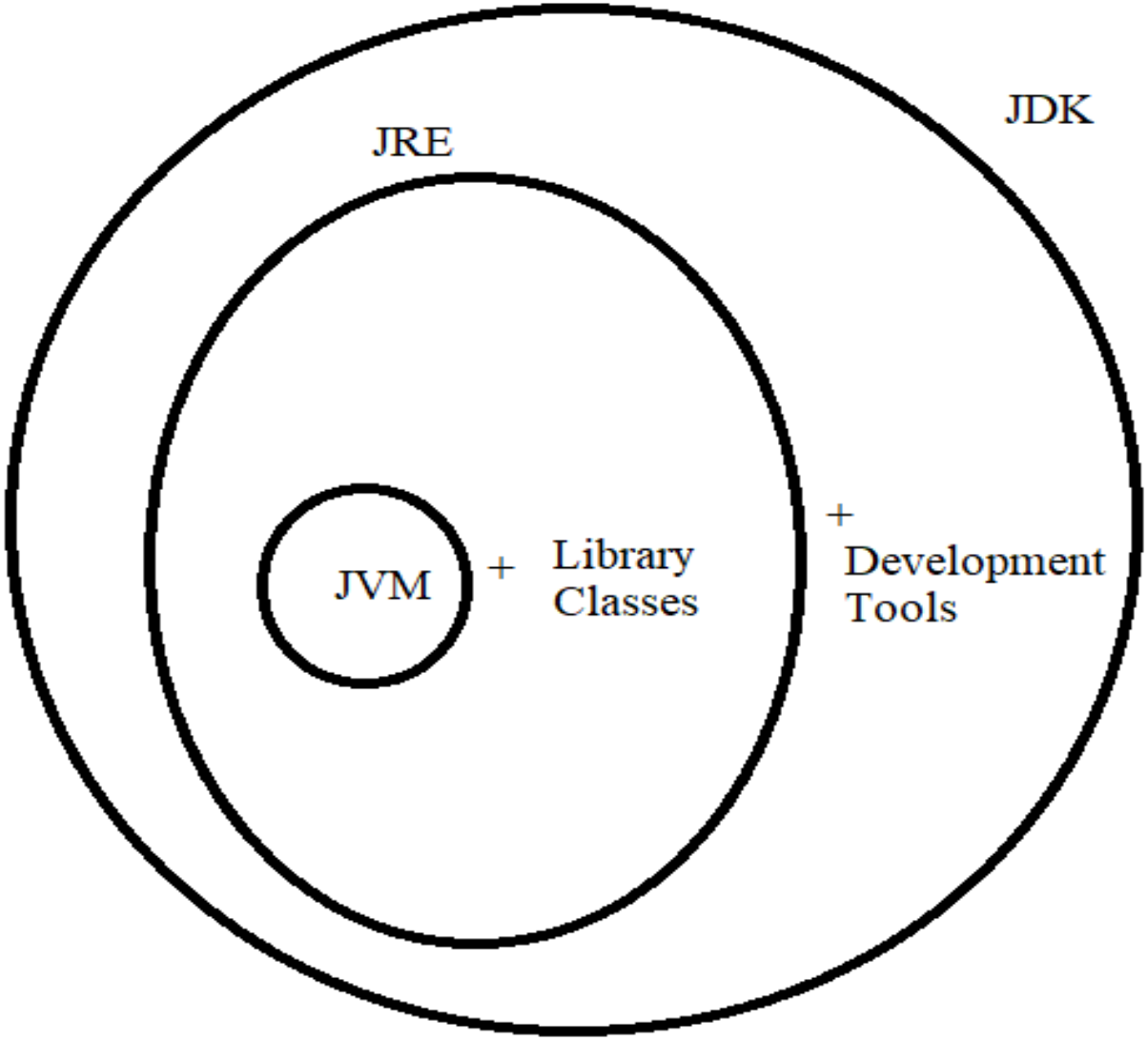
- The Java class file is designed for
  - platform independence
  - network mobility

- 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

JDK, JRE and JVM

JDK, JRE, JVM



JDK – Java Development Kit

- It provides the environment to **develop and execute(run)** the Java program.
- JDK is a kit(or package) which includes two things:
  - Development Tools(to provide an environment to develop your java programs. Like compiler, de-compiler, debugger, Java Document editor)
  - JRE (to execute your java program).
- **Note :** JDK is only used by Java Developers.

JRE – Java Runtime Environment

- JRE is only used by them who only wants to run the Java Programs i.e. end users of your system.
- By using JRE we are able to execute only already available or existing programs.
- Unable to develop new application and modification on old appication.

- JRE = JVM + Library files.

JVM – Java Virtual machine

- It provides a platform to run our java applications line by line.
- It has Interpreter and Just in time compiler (JIT).
- JVM is responsible for **executing the java program line by line** hence it is also known as interpreter.
- JIT:

It will increase the performance or execution speed of JVM