1. THE JAVA PROGRAMMING LANGUAGE

What is Java?

Java programming language is a high-level language that can be characterized by all of the following buzzwords:

- Compiled and Interpreted
- Architecture neutral
- Distributed
- Object oriented
- Robust
- Secure
- Multithreaded
- Dynamic
- Extensible
- High performance
- Easy to learn

Java features elaborated:

1. Compiled and Interpreted

Usually a computer language is either compiled or interpreted. Java combines both these features and thus there are two stages involved in a Java program. First, Java translates the source code into ***bytecode*** instructions. Bytecodes are not machine instructions. So in the next stage, the Java interpreter generates the machine code that can be directly executed by the machine that is running the Java program. This interpretation is done by the Java virtual machine.

2. Platform-independent and portable

Java programs can be easily moved from one computer system to another. If there are any changes in the OS, processor, and other system resources, there is no need to change the Java program. This portability is achieved in two ways: first Java generates the bytecode instructions that can be run on *any* machine. Also, the size of the primitive data types are machine independent. This portability is the reason behind the huge success of Java in the networking domain, especially the Internet. We can download an applet from a remote computer to our computer system and execute it locally.

3. Distributed

Java was designed with networking in mind. Java programs can share data across networks. Java programs can open and access objects on network devices.

4. Object-oriented

Java is object oriented. No coding is permitted outside class definitions. An extensive class library can be used by the programmer

5. Robust

Java is a robust language. All local variables must be initialized. There is strong type-checking. That is, all data must be declared explicitly. It has garbage-collection features which means that memory management is no longer something the programmer needs to

be worried about. Java also does exception handling – serious errors are captured and the risk of crashing the system is minimized.

6. Secure

Since Java has been designed for networked environments, security is an important feature of this language. Java systems verify all memory accesses. The absence of pointers in Java ensures that programs do not have access to all memory locations.

7. Multithreaded

Multithreading means handling many (multiple) tasks simultaneously. Java handles multithreaded programs. It means that we need not wait for one task to be completed before starting the next task. Such systems are very useful for graphical interactive environments.

8. Dynamic and extensible

The linking of data and methods to where they are located, is done at run-time. New classes can be loaded while a program is running. Linking is done on the fly. Even if libraries are recompiled, there is no need to recompile code that uses classes in those libraries. This differs from C++, which uses static binding.

By *extensible* we mean that Java programs can support functions written in other languages such as C or C++. These functions are called *native methods*. Such functions are linked dynamically at runtime.

9. High performance

Even though Java is interpreted (in the second stage), its speed is impressive compared to other interpreted languages. This is due to the use of the intermediate bytecode. Also, the multithreading feature further increases the speed of execution of Java programs.

10. Simple to learn

Java is a simple and small language. Many features of C / C++ have been removed from Java. For example, pointers, preprocessor header files (e.g., #include <stdio.h>), goto statement, operator overloading, multiple inheritance, etc have been eliminated from Java.

Differences between Java and C:

Although Java syntax is in many ways including operators and statements, there are many significant differences in these languages.

No preprocessor

Java does not include a preprocessor(C and C++ both have the `#include` preprocessor directive). Also `#define`, `#ifdef` are missing from Java.

No macro definitions

Java does not support macros (#define in C).

No global variables

There are no global variables in Java. Packages contain classes, classes contain fields and methods, and methods contain local variables.

Well-defined primitive data types

All primitive data types in Java have well-defined sizes. In C, the size of `short`, `int`, and `long` data types is platform-dependent. This hampers portability.

No pointers

Java does not support the concept of pointers. There is no address-of operator like `&`, dereference operator like `*` or `->`, or `sizeof` operator. Pointers are a source of bugs.

No `goto` statement

Java doesn't support a `goto` statement. Use of `goto` is regarded as poor programming practice. Java adds labeled `break` and `continue` statements to the flow-control statements offered by C. These are a good substitute for `goto`.

Garbage collection

The Java Virtual Machine performs garbage collection so that Java programmers do not have to explicitly manage the memory used by all objects and arrays. This feature eliminates another entire category of common bugs. It also eliminates memory leaks from Java programs.

Variable declarations anywhere

C requires local variable declarations to be made at the beginning of a method or block, while Java allows them anywhere in a method or block.

Forward Reference

In C, functions must be declared in a header file before defining them. The Java compiler allows methods (functions) to be invoked before they are defined.

Miscellaneous Features

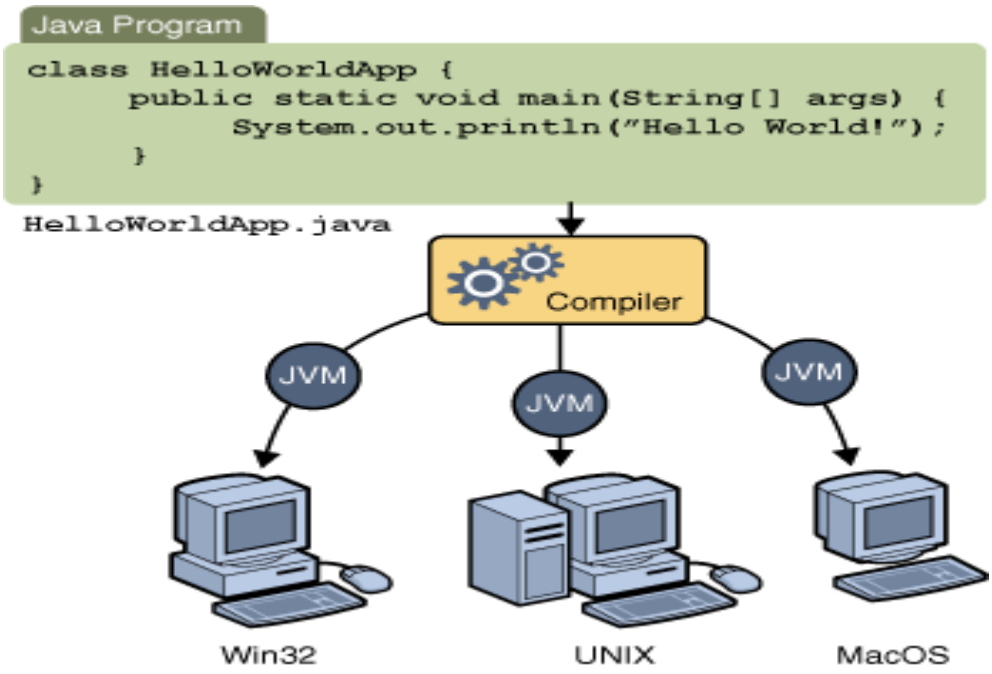
Java does not support the following features of C: typedef, variable-length argument lists (e.g., used in printf () statement), bitfields, enumerated types, structure and union types.

Java Environment

In Java programming language, all source code is first written in plain text files ending with the `.java` extension. Those source files are then compiled into `.class` files by the `javac` compiler. A `.class` file does not contain code that is native to your processor; it instead

contains *bytecodes* — the machine language of the Java Virtual Machine (Java VM). The `java` launcher tool then runs your application with an instance of the Java Virtual Machine.

Because the Java VM is available on many different operating systems, the same `.class` files are capable of running on Microsoft Windows, the Solaris ™ Operating System (Solaris OS), Linux, or Mac OS.



Java environment includes many tools and classes. The development tools are part the *Java Development Kit* (JDK) and the classes and methods are part of the *Java Standard Library* (JSL).

Java Development Kit

The JDK includes:

javac – Java compiler which translates the Java source code into bytecode.

java – Java interpreter which runs applications by reading the bytecode files.

jdb – Java debugger, which helps in debugging Java code

javap – Java disassembler, which converts bytecode files into a program description

appletviewer – for viewing Java applets without using a Java-enabled browser

The Java Platform

A *platform* is the hardware or software environment in which a program runs. Some of the most popular platforms are Microsoft Windows, Linux, Solaris OS, and Mac OS. Most platforms can be described as a combination of the operating system and underlying

hardware. The Java platform differs from most other platforms in that it's a software-only platform that runs on top of other hardware-based platforms.

The Java platform has two components:

- The *Java Virtual Machine*
- The *Java Application Programming Interface* (API)

Java Virtual Machine:

Java Virtual Machine is the base for the Java platform and is ported onto various hardware-based platforms. A Virtual Machine is a software that that creates a virtualized environment between the computer platform so that the end-user can operate software. Computer platform includes computer's hardware, operating system, and programming languages.

The Java Virtual Machine (JVM) is key to the independence of the underlying operating system and hardware - it is a platform that hides the underlying operating system from Java-powered applets and applications.

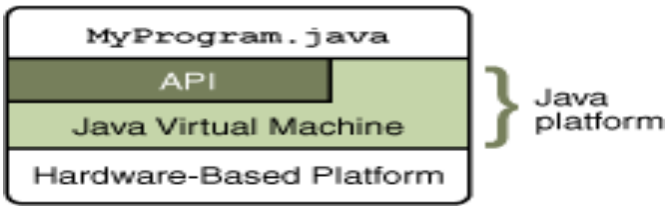
1. The Java Virtual Machine, or JVM, is an abstract computer that runs compiled Java programs.
2. The JVM is "virtual" because it is generally implemented in software on top of a "real" hardware platform and operating system.
3. All Java programs are compiled for the JVM. Therefore, the JVM must be implemented on a particular platform before compiled Java programs will run on that platform.

Compiled Java Programs
Java Virtual Machine
Hardware Platform and Operating System

4. Virtual Machine defines a machine-independent format for binary files called the class (.class) file format. This format includes instructions for a virtual computer in the form of bytecodes.

The API

The API is a large collection of ready-made software components that provide many useful capabilities. It is grouped into libraries of related classes and interfaces; these libraries are known as *packages*. The API and Java Virtual Machine insulate the program from the underlying hardware.



Advantages of Java

1. Easy to learn especially for those familiar with C or C++
2. Programs written in Java are usually smaller than in other languages.
3. Automatic garbage collection – avoids memory leaks.
4. Program development time is less in Java.
5. Platform dependencies can be avoided.
6. Applications written in Java are complied into machine-independent bytecode, and so they run consistently on any Java platform.

QUESTIONS

1. Why is Java known as platform-independent language?
2. Explain the term bytecode. Why is Java said to be compiled as well as interpreted?
3. In what way is Java a robust language?
4. Which features of C language are missing from Java? Give at least 5 features.
5. List the components of the JDK. State the function of each of these components.
6. Java is multithreaded, dynamic and extensible. What does this mean for the programmer?
7. In the context of programming languages, define the term “platform”. Elaborate on the two components of Java platform.