Introduction to Java programming

Java is an **object-oriented programming language** developed by Sun Microsystems.

It was conceived by **James Gosling and Patrick Naughton**

James Gosling **Patrick Naughton.**

It helps to create **modular programs** and **reusable code in 1995**

Object Oriented: In Java, everything is an Object. Java can be easily extended since it is based on the Object model.

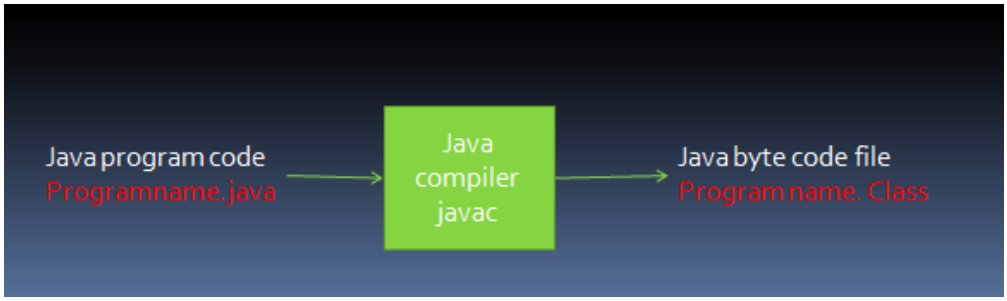
Why Java is not a purely Object-Oriented Language?

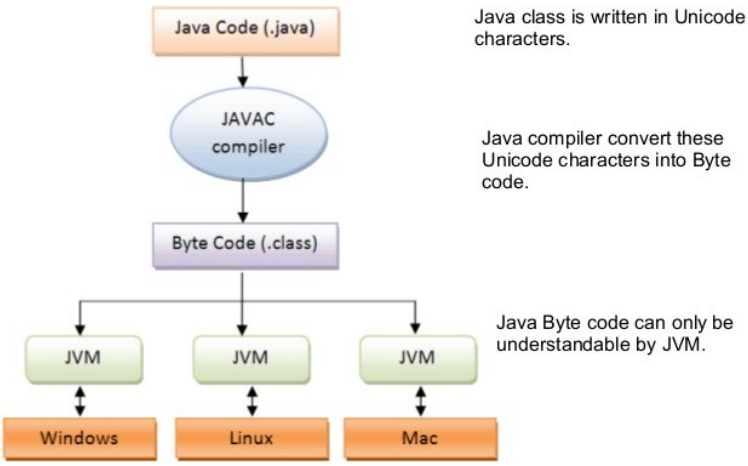
Pure Object Oriented Language/ complete/fully object oriented programming language satisfies the following qualities

1. **Encapsulation/Data Hiding**
2. **Inheritance**
3. **Polymorphism**
4. **Abstraction**
5. **All predefined types are objects**
6. **All user defined types are objects**
7. **All operations performed on objects must be only through methods exposed at the objects.**

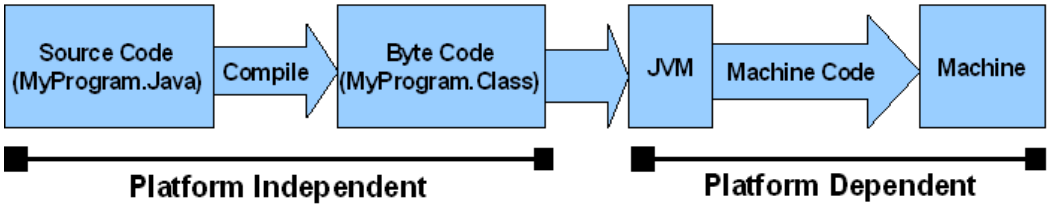
But 5 and 7 are not supported by Java

Platform independent: Unlike many other programming languages including C and C++, when Java is compiled, it is **not compiled into platform specific machine,** rather into **platform independent byte code**. This byte code is distributed over the web and interpreted by virtual Machine (JVM) on whichever platform it is being run. ***Platform-independence*** is a program's capability of **moving easily from one Computer system to another**. Java is platform-independent at both the source and the binary level





JVM is native code and specific to OS



Simple: Java is designed to be easy to learn. If you understand the **basic concept of OOP** Java would be easy to master.

Secure: With Java's secure feature it enables to develop **virus-free, tamper-free systems**. Authentication techniques are based on public-key encryption.

Architectural-neutral: Java compiler generates an **architecture-neutral object file format** which makes the compiled c**ode to be executable on many processors**, **with the presence of Java runtime system.** The primary motivation of Java was the need for a platform-independent (that is, architecture-neutral) language that could be used to create software to be embedded in various consumer electronic devices, such as microwave ovens and remote controls.

Why Java does not support pointers ?

Java does not support pointers because of pointers need **so much of memory space** at the runtime. In order to reduce the usage of memory space, java does not support pointers and also **pointers take more time at the run time**.
Java **does not use pointers because using pointer the memory area can be directly accessed,** which is **a security issue.** In this way Java has fixed a much debated issue with C/C++ Programming.

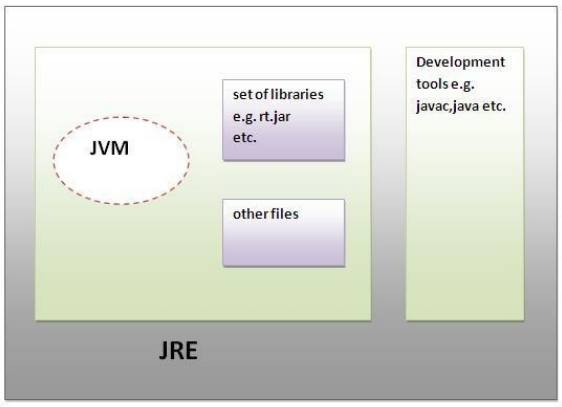
Difference between Java and C++

Comparison Index	C++	Java
Platform-independent	C++ is **platform-dependent.**	Java is **platform-independent.**
Mainly used for	C++ is mainly used for **system programming.**	Java is mainly used for **application programming**. It is widely used in **window, web-based, enterprise and mobile applications.**
Goto	C++ supports goto statement.	Java doesn't support goto statement.
Multiple inheritance	C++ supports **multiple inheritance.**	Java **doesn't support multiple inheritance** through class. It can be achieved by interfaces in java.
Operator Overloading	C++ supports **operator overloading.**	Java doesn't support **operator overloading**.
Pointers	C++ **supports pointers.** You can write pointer program in C++.	Java supports pointer internally. But you **can't write the pointer program** in java. It means java **has restricted pointer support** in java.
Compiler and Interpreter	**C++ uses compiler only.**	Java uses **compiler and interpreter both.**
Call by Value and Call by reference	**C++ supports both call by value and call by reference.**	Java supports **call by value only. There is no call by reference in java.**
Structure and Union	C++ supports **structures and unions.**	Java **doesn't support structures and unions.**
Thread Support	C++ doesn't have built-in support for threads. It relies on third-party libraries for thread support.	Java has built-in **thread support.**

JDK is an acronym for Java Development Kit.

It physically exists.

It contains JRE + development tools.



Portable: Being architectural-neutral, Java also provides for portable programming with **applets**.

Robust: Java makes a**n effort to eliminate error prone situations by emphasizing mainly on compile time error checking and runtime checking.**

Multithreaded: With Java's multithreaded feature it is possible to write programs tha**t can do many tasks simultaneously.** This design feature allows developers to construct smoothly running interactive applications.

Interpreted: Java byte code is translated on the fly to native machine instructions and is not stored anywhere. The development process is more rapid and analytical since the linking is an incremental and light weight process.

High Performance: With the use of Just-In-Time compilers, Java enables high performance.

Distributed: Java is designed for the distributed environment of the internet.

Dynamic: Java is considered to be more dynamic than C or C++ since it is designed to adapt to an evolving environment. Java programs can carry extensive amount of run-time information that can be used to verify and resolve accesses to objects on run-time.

Sun released the first public implementation as **Java 1.0 in 1995.** It promised **Write Once, Run Anywhere**(WORA), providing no-cost run-times on popular platforms.

Java Applets

An ***applet*** is a special kind of Java program that is designed to be transmitted over the Internet and automatically executed by a Java-compatible web browser.

Applets appear in a **Web page much in the same way as images do,** but **unlike images, applets are dynamic and interactive.** Applets can be used to create animations, figures, or areas that can respond to input from the reader, games, or other interactive effects on the same Web pages among the text and graphics.

FIRST PROGRAM IN JAVA

// java program starts with class class_name which should be the name of the java file

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

To write your Java programs, you will need a text editor. There are even more sophisticated IDEs available in the market. But for now, you can consider one of the following:

- **Notepad:** On Windows machine you can use any simple text editor like Notepad (Recommended for this tutorial), TextPad.
- **Netbeans:** is a Java IDE that is **open-source** and free which can be downloaded from http://www.netbeans.org/index.html.
- **Eclipse:** is also a Java IDE developed by the eclipse **open-source** community and can be downloaded from http://www.eclipse.org

Class Names - For all class names the **first letter should be in Upper Case.**
If several words are used to form a name of the class, each inner word's first letter should be in Upper Case.

Example ***class MyFirstJavaClass***

Method Names - All method names should start with a Lower Case letter.
If several words are used to form the name of the method, then each inner word's first letter should be in Upper Case.

public static void main(String args[]) - Java program processing starts from the main() method which is a mandatory part of every Java program.

```
public class Simple{
                public static void main(String args[]){
                        System.out.println("hello java");
                                                }
        }
```

Structure of Java Source code

1) It **essentially** consists of a **main() method**
2) **The controlling class of every Java application usually contain a main method**
3) This method is public and thus can be called by any object
4) This method is also **static** and **so can be called without instantiating the object of the class**
5) It d**oes not return any value** (therefore **void** keyword is used)
6) **Other methods** can **subsequently** be **called** in main()