Introduction, Feature and History

What is Java ???

Java is not just a programming language and computing platform, with a huge library, containing lots of reusable code.
Java is a high level, robust, secured and object-oriented programming language.

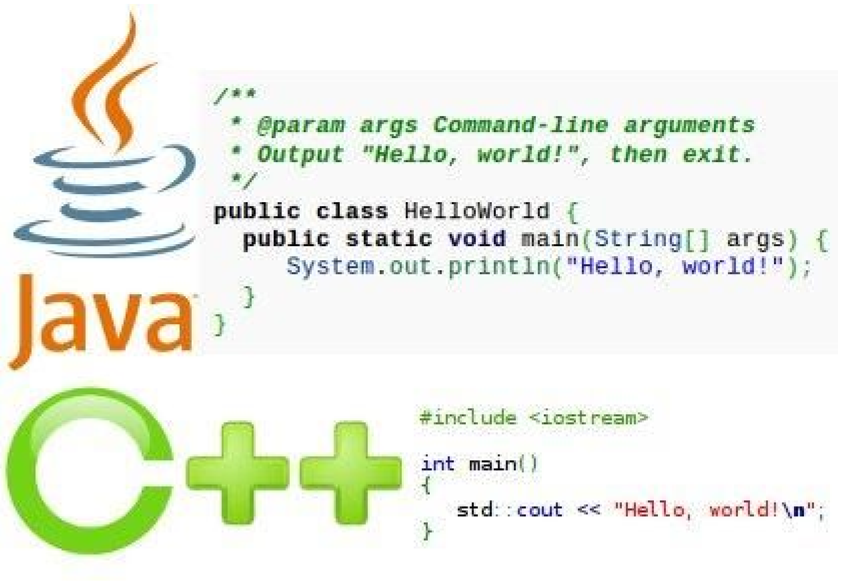
Platform: Any hardware or software environment in which a program runs, is known as a platform. Since Java has its own runtime environment (JRE) and API, it is called platform.

Java is intended to let application developers "**write once, run anywhere**" . Compiled Java code can run on all platforms that support Java without the need for recompilation. Java applications are typically compiled to bytecode that can run on any Java virtual machine (JVM) regardless of computer architecture .

Advantage of Java:

Platform Independence:
You can use the same code on Windows, Solaris, Linux, Macintosh, and so on.

Syntax:
Java has a syntax similar to that of C++, making it easy for C and C++ programmers to learn.



```
/**
 * @param args Command-line arguments
 * Output "Hello, world!", then exit.
 */
public class HelloWorld {
  public static void main(String[] args) {
     System.out.println("Hello, world!");
  }
}
```

```
#include <iostream>

int main()
{
   std::cout << "Hello, world!\n";
}
```

Introduction, Feature and History

Java is also fully object oriented—even more so than C++. Everything in Java, except for a few basic types like numbers, is an object.

Bug Free: It is far easier to turn out bug-free code using Java than using C++.

1. Eliminated manual memory allocation and deallocation. Memory in Java is automatically garbage collected. You never have to worry about memory corruption.
2. Eliminated multiple inheritance, replacing it with a new notion of interface (Interfaces give you most of what you want from multiple inheritance, without the complexity that comes with managing multiple inheritance hierarchies.)

Where it is used?

According to Sun, 3 billion devices run java. There are many devices where java is currently used. Some of them are as follows:

1. Desktop Applications such as acrobat reader, media player, antivirus etc.
2. Web Applications such as irctc.co.in, javatpoint.com etc.
3. Enterprise Applications such as banking applications.
4. Mobile
5. Embedded System
6. Smart Card
7. Robotics
8. Games etc.

Java SE stands for Java standard edition and is normally for developing desktop applications, forms the core/base API.(Banking software)

Java EE stands for Java enterprise edition for applications which run on servers, for example web sites. (Twitter)

Introduction, Feature and History

Java ME stands for Java micro edition for applications which run on resource constrained devices (small scale devices) like cell phones, for example mobile games (Nokia), micro-oven.

JavaFX, java developed for flash application (Flash Games)

Introduction, Feature and History

Types of Java Applications

There are mainly 4 types of applications that can be created using java programming:

Standalone Application:

It is also known as a desktop application or window-based application. An application that we need to install on every machine such as media player, antivirus etc. AWT and Swing are used in java for creating standalone applications.

Web Application:

An application that runs on the server side and creates dynamic page, is called web application. Currently, servlet, jsp, struts, jsf etc. technologies are used for creating web applications in java.

Enterprise Application:

An application that is distributed in nature, such as banking applications etc. It has the advantage of high level security, load balancing and clustering. In java, EJB is used for creating enterprise applications.

Mobile Application:

An application that is created for mobile devices. Currently Android and Java ME are used for creating mobile applications.

History of Java

1. Brief history of Java
2. Java Version History

Introduction, Feature and History

Java history is interesting to know. The history of java starts from **Green Team**. Java team members (also known as the **Green Team**), initiated a revolutionary task to develop a language for digital devices such as set-top boxes, televisions etc.

For the green team members, it was an advanced concept at that time. But, it was suited for internet programming. Later, Java technology as incorporated by Netscape.

Currently, Java is used in internet programming, mobile devices, games, e-business solutions etc. There are given the major points that describes the history of java.

1. James Gosling, Mike Sheridan, and Patrick Naughton initiated the Java language project in June 1991. The small team of sun engineers called Green Team.
2. Originally designed for small, embedded systems in electronic appliances like set-top boxes.
3. Firstly, it was called **"Greentalk"** by James Gosling and file extension was **.gt**.
4. After that, it was called **Oak** and was developed as a part of the Green project.

Why Oak name for java language?

5. Why Oak? Oak is a symbol of strength and chosen as the national tree of many countries like U.S.A., France, Germany, Romania, etc.
6. In 1995, Oak was renamed as "Java" because it was already a trademark by **Oak Technologies**.

Why Java name for java language?

7. Why they chosen java name for java language? The team gathered to choose a new name. The suggested words were "dynamic", "revolutionary", "Silk", "jolt", "DNA" etc. They wanted something that reflected the essence of the technology: revolutionary, dynamic, lively, cool, unique, and easy to spell and fun to say.

According to James Gosling "Java was one of the top choices along with Silk". Since java was so unique, most of the team members preferred java.

8. Java is an island of Indonesia where first coffee was produced (called java coffee).
9. Notice that Java is just a name not an acronym.
10. Originally developed by James Gosling at Sun Microsystems (which is now a subsidiary of Oracle Corporation) and released in 1995.
11. In 1995, Time magazine called Java **one of the Ten Best Products of 1995**.
12. JDK 1.0 released in January 23, 1996.

Java Version History

There are many java versions that has been released. Current stable release of Java is Java SE 8.

1. JDK Alpha and Beta (1995)
2. JDK 1.0 (23rd Jan, 1996)
3. JDK 1.1 (19th Feb, 1997)
4. J2SE 1.2 (8th Dec, 1998)
5. J2SE 1.3 (8th May, 2000)
6. J2SE 1.4 (6th Feb, 2002)
7. J2SE 5.0 (30th Sep, 2004)
8. Java SE 6 (11th Dec, 2006)
9. Java SE 7 (28th July, 2011)
10. Java SE 8 LTS (18th March, 2014)
11. Java SE 9 (September 2017)
12. Java SE 10 (March 2018)
13. Java SE 11 LTS (September 2018)
14. Java SE 12 (March 2019)
15. Java SE 13 (September 2019)

Introduction, Feature and History

JAVA Major Features:

Following are the notable features of Java:

Object Oriented

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

Basic concepts of OOPs are:

1. Object
2. Class
3. Inheritance
4. Polymorphism
5. Abstraction
6. Encapsulation

Platform Independent

A platform is the hardware or software environment in which a program runs. There are two types of platforms software-based and hardware-based. Java provides software-based platform. The Java platform differs from most other platforms in the sense that it's a software-based platform that runs on top of other hardware-based platforms.It has two components:

1. Runtime Environment
2. API(Application Programming Interface)

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 bytecode. This bytecode is distributed over the web and interpreted by the Virtual Machine (JVM) on whichever platform it is being run on.

Introduction, Feature and History

Simple

Java is designed to be easy to learn. If you understand the basic concept of OOP Java, it would be easy to master.

According to Sun, Java language is simple because:

- Syntax is based on C++ (so easier for programmers to learn it after C++).
- removed many confusing and/or rarely-used features e.g., explicit pointers, operator overloading etc.
- No need to remove unreferenced objects because there is Automatic Garbage Collection in java..

Secure

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

- Java run inside a virtual machine sandbox
- Classloader in JRE used to load java classes into JVM dynamically
- Bytecode verifier checks code fragment for illegal code that can violate access right to object

Architecture-neutral

Java compiler generates an architecture-neutral object file format, which makes the compiled code executable on many processors, with the presence of Java runtime system.(int always 32bit)

Portable

Being architecture-neutral and having no implementation dependent aspects of the specification makes Java portable.

Robust

Java makes an effort to eliminate error-prone situations by emphasizing mainly on compile time error checking and runtime checking.

- Strong memory management
- Automatic garbage collection
- Exception handling and type checking mechanism

Multithreaded

With Java's multithreaded feature, it is possible to write programs that can perform many tasks simultaneously. This design feature allows the developers to construct interactive applications that can run smoothly. The main advantage of multi-threading is that it shares the same memory. Threads are important for multimedia, Web applications etc.

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. Java is faster than traditional interpretation since byte code is "close" to native code still somewhat slower than a compiled language

Distributed

Java is designed for the distributed environment of the internet.

Introduction, Feature and History

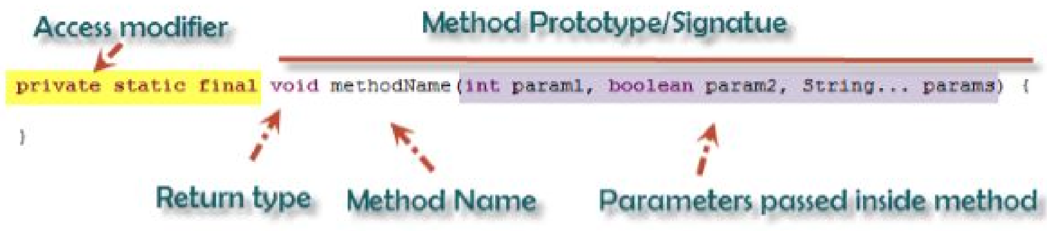
Understanding first java program

Let's see what is the meaning of class, public, static, void, main, String[], System.out.println().

- **class** keyword is used to declare a class in java.
- **public** keyword is an access modifier which represents visibility, it means it is visible to all.
- **static** is a keyword, if we declare any method as static, it is known as static method. The core advantage of static method is that there is no need to create an object to invoke the static method. The main method is executed by the JVM, so it doesn't require to create object to invoke the main method. So it saves memory.
- **void** is the return type of the method, it means it doesn't return any value.
- **main** represents startup of the program.
- **String[] args** is used for command line argument. We will learn it later.
- **System.out.println()** is used print statement. We will learn about the internal working of System.out.println statement later.

How many ways can we write a java program

There are many ways to write a java program. The modifications that can be done in a java program are given below:



```
private static final void methodName(int param1, boolean param2, String... params) {
}
```

- By changing the sequence of the modifiers, method prototype is not changed. **static public void main(String args[])**

Introduction, Feature and History

- Subscript notation in java array can be used after type, before variable or after variable.

  Let's see the different codes to write the main method.

  public static void main(String[] args)

  public static void main(String []args)

  public static void main(String args[])
- You can provide var-args support to main method by passing 3 ellipses (dots) We will learn about var-args later in Java New Features chapter.

  public static void main(String... args)
- Having semicolon at the end of class in java is optional.

  Let's look at the simple code.

```
class A{
    static public void main(String... args){
        System.out.println("hello world!");
        }
};
```

Valid java main method signature

- public static void main(String[] args)
- public static void main(String []args)
- public static void main(String args[])
- public static void main(String... args)
- static public void main(String[] args)
- public static final void main(String[] args)
- final public static void main(String[] args)

Invalid java main method signature

- **public void main(String[] args):** (signature not wrong but you can't run non static main)
- **static void main(String[] args):** (signature not wrong but you can't run private main)
- **public void static main(String[] args)** : Method signature modified
- **abstract public static void main(String[] args)** : (abstract method can't be static because abstract methods have no body. It's not a good idea to call method without body)

Internal Details of Hello Java Program

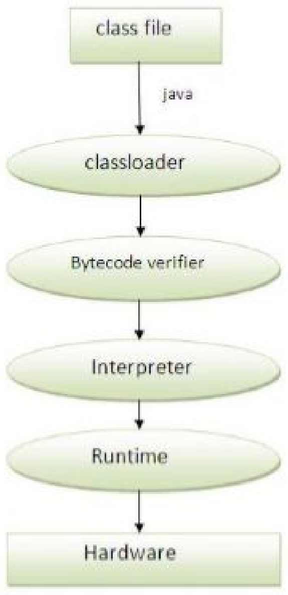
Here, we are going to learn, what happens while compiling and running the java program.

Moreover, we will see some questions based on the first program.

What happens at compile time?

At compile time, java file is compiled by Java Compiler (It does not interact with OS) and converts the java code into bytecode.

What happens at runtime?



At runtime, following steps are performed:

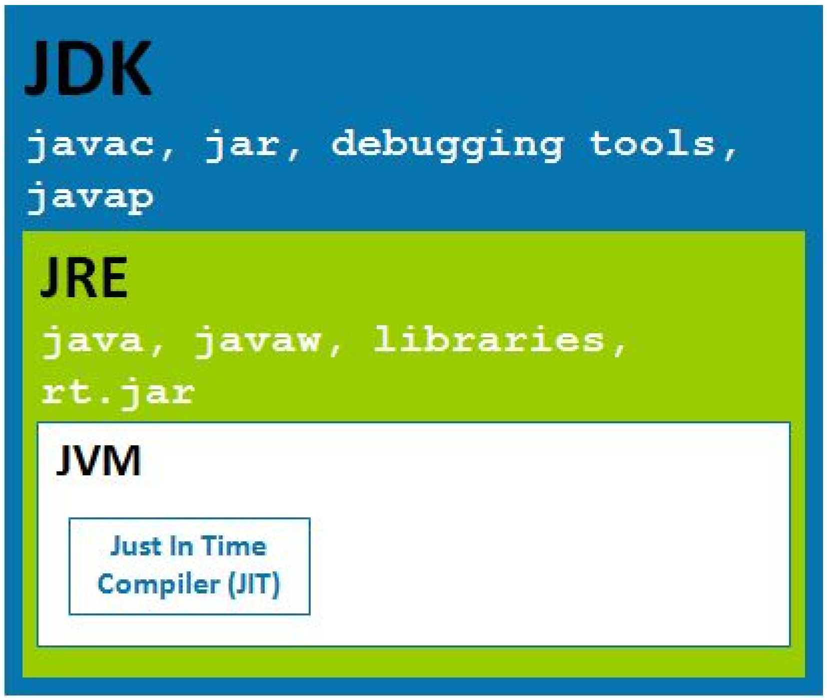
Classloader: is the subsystem of JVM that is used to load class files.

Bytecode Verifier: checks the code fragments for illegal code that can violate access rights to objects.

Interpreter: read bytecode stream then execute the instructions.

Introduction, Feature and History

JDK, JVM And JRE


JDK
javac, jar, debugging tools, javap
JRE
java, javaw, libraries, rt.jar
JVM
Just In Time Compiler (JIT)

JRE (Java Runtime Environment)

Java Runtime Environment contains JVM, class libraries, and other supporting files. It does not contain any development tools such as compiler, debugger, etc. Actually JVM runs the program, and it uses the class libraries, and other supporting files provided in JRE. If you want to run any java program, you need to have JRE installed in the system.

JDK (Java Development Kit)

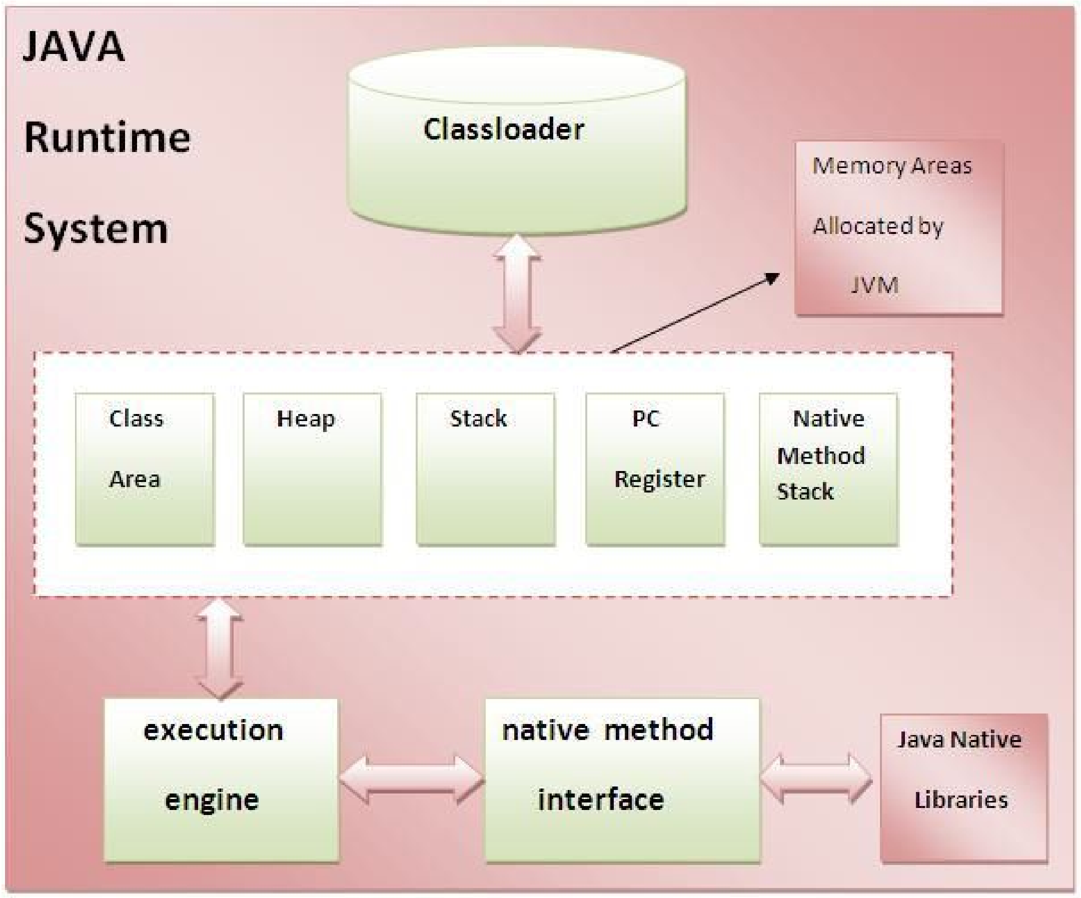
Java Development Kit (JDK) The JDK is a superset of the JRE, and contains everything that is in the JRE, plus tools such as compilers and debuggers necessary for developing applets and applications.

JVM (Java Virtual Machine)

As we all aware when we compile a Java file, output is not an ‘exe’ but it’s a ‘.class’ file. ‘.class’ file consists of Java byte codes which are understandable by JVM. Java Virtual Machine interprets the byte code into the machine code depending upon the underlying

operating system and hardware combination. It is responsible for all the things like garbage collection, array bounds checking, etc… JVM is platform dependent.
The JVM is called “virtual” because it provides a machine interface that does not depend on the underlying operating system and machine hardware architecture. This independence from hardware and operating system is a cornerstone of the write-once run-anywhere value of Java programs.
There are different JVM implementations are there. These may differ in things like performance, reliability, speed, etc. These implementations will differ in those areas where Java specification doesn’t mention how to implement the features, like how the garbage collection process works is JVM dependent, Java spec doesn’t define any specific way to do this.

Internal Architecture of JVM



Classloader:

Classloader is a subsystem of JVM that is used to load class files.

Class(Method) Area:

Class(Method) Area stores per-class structures such as the runtime constant pool, field and method data, the code for methods.

Heap:

It is the runtime data area in which objects are allocated.

Stack:

Java Stack stores frames. It holds local variables and partial results, and plays a part in method invocation and return. Each thread has a private JVM stack, created at the same time as thread. A new frame is created each time a method is invoked. A frame is destroyed when its method invocation completes.

Program Counter Register:

PC (program counter) register. It contains the address of the Java virtual machine instruction currently being executed.

Native Method Stack:

It contains all the native methods used in the application.

Execution Engine:

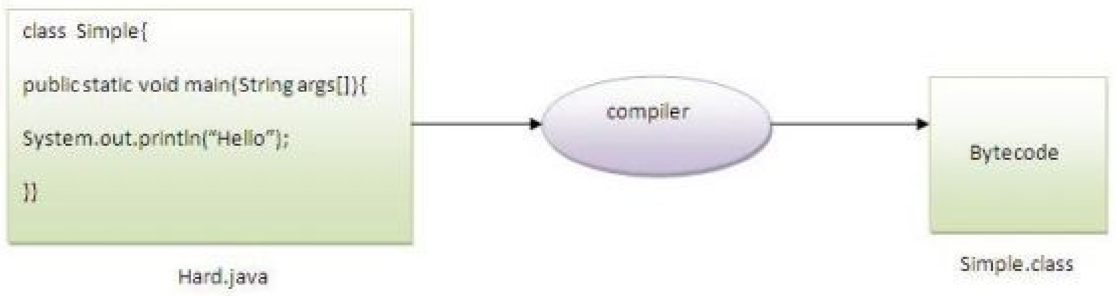
It contains:

1. **A virtual processor**
2. **Interpreter**: Read bytecode stream then execute the instructions.
3. **Just-In-Time(JIT) compiler**: It is used to improve the performance. JIT compiles parts of the byte code that have similar functionality at the same time, and hence reduces the amount of time needed for compilation. Here the term ?compiler? refers to a translator from the instruction set of a Java virtual machine (JVM) to the instruction set of a specific CPU.

Introduction, Feature and History

Q) Can you save a java source file by other name than the class name?

Yes, if the class is not public.



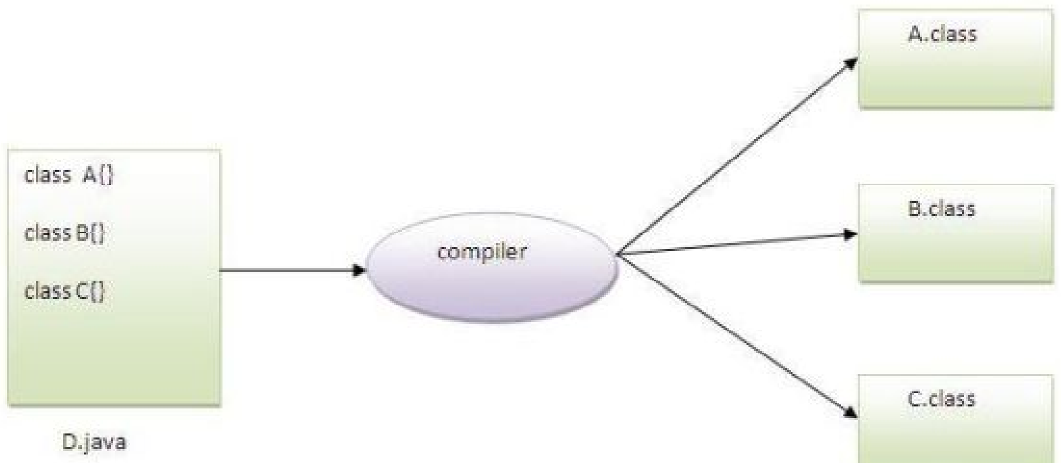
```
class Simple{
public static void main(String args[]){
System.out.println("Hello");
}}
```

To compile: javac Hard.java

To execute: java Simple

Q) Can you have multiple classes in a java source file?

Yes



```
class A{}
class B{}
class C{}
```

@Author Er. Prabeen Soti