Naveen Anamaneni

Core Java Fundamentals

1. **What is Java**

Core Java Tutorial or Java Programming Tutorial is a widely used robust technology. Let's start learning Java from basic questions like what is Java tutorial, Core Java, where it is used, what type of applications are created in Java, why use java and Java platforms etc. Our Java tutorial helps you to learn Java with easy and simple examples.

**What is Java**

* Java is a programming language and a platform.
* 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 Example**

Let's have a quick look at java programming example. A detailed description of hello Java example is given in next page.

|  |
| --- |
| **public** **class** Test {  **public** **static** **void** main(String[] args) {    System.***out***.println("Hello Java");  }  } |
| Hello Java |

**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:

* Desktop Applications such as acrobat reader, media player, antivirus etc.
* Web Applications such as irctc.co.in, javatpoint.com etc.
* Enterprise Applications such as banking applications.
* Mobile
* Embedded System
* Smart Card
* Robotics
* Games etc.

**Types of Java Applications**

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

**1) Standalone Application**

It is also known as 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.

**2) 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.

**3) 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.

**4) Mobile Application**

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

**Java Platforms / Editions**

There are 4 platforms or editions of Java:

**1) Java SE (Java Standard Edition)**

It is a java programming platform. It includes Java programming APIs such as **java.lang, java.io, java.net, java.util, java.sql, java.math** etc. It includes core topics like OOPs, String, Regex, Exception, Inner classes, Multithreading, I/O Stream, Networking, AWT, Swing, Reflection, Collection etc.

**2) Java EE (Java Enterprise Edition)**

It is an enterprise platform which is mainly used to develop web and enterprise applications. It is built on the top of Java SE platform. It includes topics like Servlet, JSP, Web Services, EJB, JPA etc.

**3) Java ME (Java Micro Edition)**

It is a micro platform which is mainly used to develop mobile applications.

**4) JavaFx**

It is used to develop rich internet applications. It uses light-weight user interface API.

1. **History of Java**

Java history is interesting to know. The history of java starts from **Green Team**. Java team members (also known as **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 advance concept at that time. But, it was suited for internet programming. Later, Java technology as incorporated by Netscape.

**James Gosling - founder of java**

**James Gosling**

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.

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

**Why "Oak" name**

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

**Why "Java" name**

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

**Features of Java**

There is given many features of java. They are also known as java buzzwords. The Java Features given below are simple and easy to understand.

1. Simple
2. Object-Oriented
3. Portable
4. Platform independent
5. Secured
6. Robust
7. Architecture neutral
8. Dynamic
9. Interpreted
10. High Performance
11. Multithreaded
12. Distributed

Java Features

**Simple**

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.

**Object-oriented**

Object-oriented means we organize our software as a combination of different types of objects that incorporates both data and behaviour. Object-oriented programming(OOPs) is a methodology that simplify software development and maintenance by providing some rules.

Basic concepts of OOPs are:

* Object
* Class
* Inheritance
* Polymorphism
* Abstraction
* Encapsulation

**Platform Independent**

**java is 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 is a software-based platform that runs on the top of other hardware-based platforms. It has two components:

**Runtime Environment**

**API (Application Programming Interface)**

Java code can be run on multiple platforms e.g. Windows, Linux, Sun Solaris, Mac/OS etc. Java code is compiled by the compiler and converted into bytecode. This bytecode is a platform-independent code because it can be run on multiple platforms i.e. Write Once and Run Anywhere(WORA).

**Secured**

Java is secured because:

* No explicit pointer
* Java Programs run inside virtual machine sandbox
* **Classloader:** adds security by separating the package for the classes of the local file system from those that are imported from network sources.
* Bytecode Verifier: checks the code fragments for illegal code that can violate access right to objects.
* Security Manager: determines what resources a class can access such as reading and writing to the local disk.
* These securities are provided by java language. Some security can also be provided by application developer through SSL, JAAS, Cryptography etc.

**Robust**

Robust simply means strong. Java uses strong memory management. There is lack of pointers that avoids security problem. There is automatic garbage collection in java. There is exception handling and type checking mechanism in java. All these points makes java robust.

**Architecture-neutral**

There are no implementation dependent features e.g. size of primitive types is fixed.

In C programming, int data type occupies 2 bytes of memory for 32-bit architecture and 4 bytes of memory for 64-bit architecture. But in java, it occupies 4 bytes of memory for both 32 and 64-bit architectures.

**Portable**

We may carry the java bytecode to any platform.

**High-performance**

Java is faster than traditional interpretation since byte code is "close" to native code still somewhat slower than a compiled language (e.g., C++)

**Distributed**

We can create distributed applications in java. RMI and EJB are used for creating distributed applications. We may access files by calling the methods from any machine on the internet.

**Multi-threaded**

A thread is like a separate program, executing concurrently. We can write Java programs that deal with many tasks at once by defining multiple threads. The main advantage of multi-threading is that it doesn't occupy memory for each thread. It shares a common memory area. Threads are important for multi-media, Web applications etc.

1. **C++ vs Java**

There are many differences and similarities between C++ programming language and Java. A list of top differences between C++ and Java are given below:

|  |  |  |
| --- | --- | --- |
| 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. |
| Documentation comment | C++ doesn't support documentation comment. | Java supports documentation comment (/\*\* ... \*/) to create documentation for java source code. |
| Virtual Keyword | C++ supports virtual keyword so that we can decide whether or not override a function. | Java has no virtual keyword. We can override all non-static methods by default. In other words, non-static methods are virtual by default. |
| unsigned right shift >>> | C++ doesn't support >>> operator. | Java supports unsigned right shift >>> operator that fills zero at the top for the negative numbers. For positive numbers, it works same like >> operator. |
| Inheritance Tree | C++ creates a new inheritance tree always. | Java uses single inheritance tree always because all classes are the child of Object class in java. Object class is the root of inheritance tree in java. |

1. **Simple Program of Java**

In this page, we will learn how to write the simple program of java. We can write a simple hello java program easily after installing the JDK.

To create a simple java program, you need to create a class that contains main method. Let's understand the requirement first.

Creating hello java example

Let's create the hello java program:

|  |
| --- |
| **package** com.anamaneni; //package Name  /\*\*  \* **@author** Naveen Anamaneni  \* Class Name : Test  \*/  **public** **class** Test {  **public** **static** **void** main(String[] args) {  System.***out***.println("Hello Java");  }  } |
| Hello Java |

**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 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.

**Internal Details of Hello Java Program**

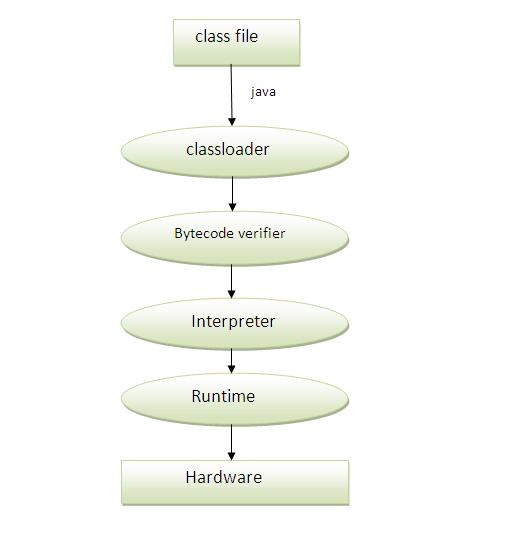
**What happens at compile time?**

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

compilation of simple java program

**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 right to objects.

**Interpreter:** read bytecode stream then execute the instructions.

1. **Difference between JDK, JRE and JVM**

**JVM**

**JVM (Java Virtual Machine)** is an abstract machine. It is a specification that provides runtime environment in which java bytecode can be executed.

JVMs are available for many hardware and software platforms. JVM, JRE and JDK are platform dependent because configuration of each OS differs. But, Java is platform independent.

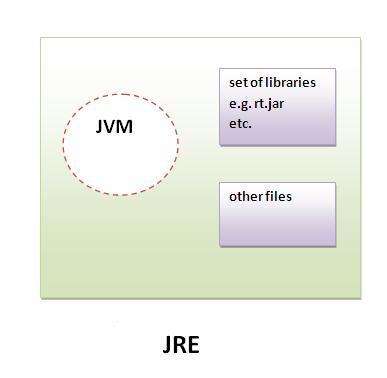
The JVM performs following main tasks:

* Loads code
* Verifies code
* Executes code
* Provides runtime environment

**JRE**

JRE is an acronym for Java Runtime Environment. It is used to provide runtime environment. It is the implementation of JVM. It physically exists. It contains set of libraries + other files that JVM uses at runtime.

Implementation of JVMs are also actively released by other companies besides Sun Micro Systems.



**JDK**

JDK is an acronym for Java Development Kit. It physically exists. It contains JRE + development tools.



1. **JVM (Java Virtual Machine)**

**JVM (Java Virtual Machine)** is an abstract machine. It is a specification that provides runtime environment in which java bytecode can be executed.

**JVMs** are available for many hardware and software platforms (i.e. JVM is platform dependent).

**What is JVM**

It is:

A specification where working of Java Virtual Machine is specified. But implementation provider is independent to choose the algorithm. Its implementation has been provided by Sun and other companies.

An implementation Its implementation is known as JRE (Java Runtime Environment).

Runtime Instance Whenever you write java command on the command prompt to run the java class, an instance of JVM is created.

**What it does**

**The JVM performs following operation:**

* Loads code
* Verifies code
* Executes code
* Provides runtime environment

**JVM provides definitions for the:**

* Memory area
* Class file format
* Register set
* Garbage-collected heap
* Fatal error reporting etc.

**Internal Architecture of JVM**

Let's understand the internal architecture of JVM. It contains class loader, memory area, execution engine etc.



**1) Classloader**

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

**2) Class(Method) Area**

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

**3) Heap**

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

**4) 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.

**5) Program Counter Register**

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

**6) Native Method Stack**

It contains all the native methods used in the application.

**7) Execution Engine**

It contains:

* A virtual processor
* Interpreter: Read bytecode stream then execute the instructions.
* 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.