Java

tutorialspoint

SIMPLY EASY LEARNING

www.tutorialspoint.com

https://www.facebook.com/tutorialspointindia

https://twitter.com/tutorialspoint

About the Tutorial

This **Java tutorial** has been written for beginners to advanced programmers who are striving to learn Java programming. We have provided numerous practical examples to explain the concepts in simple and easy steps. This tutorial has been prepared and reviewed by experienced Java programmers at Tutorials Point, and the best effort has been put into making it useful for the students and Java developers.

After completing this tutorial, you will find yourself at a moderate level of expertise in Java programming, from where you can elevate yourself to the next levels.

What is Java?

Java is a popular high-level, object-oriented programming language that was originally developed by Sun Microsystems and released in 1995. Currently, Java is owned by Oracle, and more than 3 billion devices run Java. Java runs on a variety of platforms, such as Windows, Mac OS, and the various versions of UNIX. Today Java is being used to develop numerous types of software applications, including desktop apps, mobile apps, web apps, games, and much more.

*Java is a general-purpose programming language intended to let programmers **Write Once, Run Anywhere (WORA)**. This means that compiled Java code can run on all platforms that support Java without the need to recompile.*

In this tutorial, you will learn everything about Java, starting from basics to advanced concepts such as overview, history, installations, basic input/output, conditional & control statements, arrays, classes, inheritance, method overloading & overriding, exception handling, and many more.

Java First Example

The first example in Java is to print "Hello, World!" on the screen. Let's have a quick look at the first example in Java programming:

```
public class MyFirstJavaProgram {


   /* This is my first java program.
    * This will print 'Hello, World!' as the output
    */


   public static void main(String []args) {
      System.out.println("Hello, World!"); // prints Hello, World!
   }
}
```

Getting Started with Java

Explore the following chapters to get started learning Java programming:

- Java - Home
- Java - Overview
- Java - History
- Java - Features
- Java Vs. C++
- JVM - Java Virtual Machine
- Java - JDK vs JRE vs JVM
- Java - Hello World Program
- Java - Environment Setup
- Java - Basic Syntax

Java Basics

Explore the basic topics such as data types, operators, user input, etc.:

- Java - Variable Types
- Java - Data Types
- Java - Type Casting
- Java - Unicode System
- Java - Basic Operators
- Java - Comments
- Java - User Input
- Java - Date & Time

Java Conditional Statements

Furthermore, in Java, explore the conditional statements to implement the logic and to make decisions based on the conditions:

- Java - Decision Making
- Java - If-else
- Java - Switch

Java Loops and Control Statements

Explore the loops and control statements to learn how to control the execution of the programming logics:

- Java - Loop Control
- Java - For Loops
- Java - For-Each Loops
- Java - While Loops
- Java - do-while Loops
- Java - Break

- Java - Continue

Java String and Arrays

Explore the following strings and array-related chapters:

- Java Strings and String Class
- Java - Strings

Java OOPs

Explore the following chapters to learn the object-oriented programming approach in Java:

- Java - OOPs Concepts
- Java - Object & Classes
- Java - Class Attributes
- Java - Class Methods
- Java - Methods
- Java - Variables Scope
- Java - Constructors
- Java - Access Modifiers
- Java - Inheritance
- Java - Aggregation
- Java - Polymorphism
- Java - Overriding
- Java - Method Overloading
- Java - Dynamic Binding
- Java - Static Binding
- Java - Instance Initializer Block
- Java - Abstraction
- Java - Encapsulation

Java Collections and Interfaces

Explore Java interfaces and collections-related chapters:

- Java - Interfaces
- Java - List Interface
- Java - Queue Interface
- Java - Map Interface
- Java - SortedMap Interface
- Java - Set Interface
- Java - SortedSet Interface

- Java - Iterators
- Java - Comparators
- Java - Comparable Interface in Java
- Java - Collections
- Java - Collection Interface

Java Exceptional Handling

Explore the file handling chapters to learn how to create, write, read, and manipulate the files and directories:

- Java - Files
- Java - Create a File
- Java - Write to File
- Java - Read Files
- Java - Delete Files
- Java - Directories
- Java - I/O Streams

Java Multithreading

Explore the threading-related chapters to learn to manage multiple threads in Java:

- Java - Multithreading
- Java - Thread Life Cycle
- Java - Creating a Thread
- Java - Starting a Thread
- Java - Joining Threads
- Java - Naming Thread
- Java - Thread Scheduler
- Java - Thread Pools
- Java - Main Thread
- Java - Thread Priority
- Java - Daemon Threads
- Java - Thread Group
- Java - Shutdown Hook

Java Practices

- Java Quick Guide
- Java Interview Questions
- Java 8 Interview Questions
- Java Online Test

- Java Mock Test

Java References

Here, you can find the package, class, and method references –

- Java Scanner Class
- Java Arrays Class
- Java Strings Class
- Java Date Class
- Java ArrayList Class
- Java Vector Class
- Java Stack Class
- Java PriorityQueue Class
- Java LinkedList Class
- Java ArrayDeque Class
- Java HashMap Class
- Java LinkedHashMap Class
- Java WeakHashMap Class
- Java EnumMap Class
- Java TreeMap Class
- Java IdentityHashMap Class
- Java HashSet Class
- Java EnumSet Class
- Java LinkedHashSet Class
- Java TreeSet Class
- Java BitSet Class
- Java Dictionary Class
- Java Hashtable Class
- Java Properties Class
- Java Collection Class
- Java Array Class

Online Java Compiler

Our Java programming tutorial provides various examples to explain the concepts. To compile and execute the given Java programming examples in your browser itself, we have provided Online Java Compiler. You can edit and run almost all the examples directly from your browser without the need to set up your development environment.

Try to click the icon  to run the following Java code to print conventional "Hello, World!" using Java Programming.

Below code box allows you to change the value of the code. So, please try to change the value inside **println()** and run it again to verify the result.

```
public class MyFirstJavaProgram {


   /* This is my first java program.
    * This will print 'Hello, World!' as the output
    */


   public static void main(String []args) {
      System.out.println("Hello, World!"); // prints Hello, World!
   }
}
```

Java Features

Java is a feature-rich language. Java is evolving continuously with every update, and updates are coming every six months. Following are some of the main features of the Java language:

- **Object Oriented**: Java is a pure object-oriented language, and everything in Java is an object. Java supports OOPS principles like Inheritance, Encapsulation, Polymorphism, Classes , and so on. Java itself can be extended as well, being based on an object model.
- **Platform Independent**: Java code is platform independent. A Java code is not compiled into machine-specific code. It is compiled into a platform-neutral byte code. This byte code is executed by JVM which runs the code on the underlying platform. This capability makes Java a Write Once Run Anywhere language.
- **Easy to Learn**: Java inherits features from C and C++, and developers can easily learn Java if they know C or C++ languages. Even for someone new to computer languages, Java is very easy to learn from scratch.
- **Secure**: Java is secure by architecture. A developer is not required to directly interact with the underlying memory or operating system. Java provides automatic garbage collection, so developers are not required to worry about memory leaks, management, etc.
- **Architectural-Neutral**: Java byte code can be executed on any kind of processor. JRE automatically handles the code execution on different types of processors.
- **Portable**: A Java code written on a Windows machine can be executed without any code change on MacOS and vice versa. There is no need to make any operating system-specific code changes.
- **Robust**: Java is a very robust language with very strong compile-time error checks, strict type checking, and runtime exception handling.
- **Multithreading**: Java provides inbuilt support for multiprocessing and multithreading. Java provides thread handling, monitors, deadlock handling, racing conditions, etc.

- **High Performance**: Java, although being interpreted, is still very performant. The JIT (Just In Time) compiler helps in improving performance.
- **Distributed**: Java is designed for distributed systems and is the most popular language for developing internet-based applications as the internet is a distributed environment.

Java Applications

Since Java supports object-oriented features and is platform-independent, it is extensively used in various fields. Listed below are a few areas where Java is used -

- Enterprise solutions
- Game development
- Secured web development
- Embedded systems
- Mobile application development
- Big Data Applications, and many more.

Java Platforms (Editions)

Platforms of Java are divided into four Java editions, which are –

- **Java SE (Java Standard Edition)**: It is a standard edition that is used to develop and deploy portable code for desktop and server environments.
- **Java EE (Java Enterprise Edition)**: It is an enterprise edition that is used to develop web applications.
- **Java ME (Java Micro Edition)**: J2ME is used to develop mobile applications. It is a micro edition of Java.
- **JavaFx:** It is used to develop lightweight user interfaces for rich internet applications.

Java Jobs & Opportunities

Java is in demand, and all the major companies are recruiting Java programmers to develop their desktop, web, and mobile applications.

Today, a Java programmer with 3-5 years of experience is asking for around $120,000 in as an annual package, and this is the most demanding programming language in America. Though it can vary depending on the location of the job. Following are the great companies that are using Java and they need good Java programmers:

- Google
- Microsoft
- Facebook
- IBM
- Amazon
- Netflix
- Pinterest

- Uber
- JetBrains
- Many more...

So, you could be the next potential employee for any of these major companies. We have developed great learning material for Java that will help you prepare for the technical interviews and certification exams based on Java. So, start learning Java using this simple and effective tutorial from anywhere and anytime, absolutely at your pace.

Why Learn Java?

Java is a MUST to learn programming language for students and working professionals to become a great software engineer, especially when they are working in the software development domain. If you will conduct a survey about the best programming language around, Java is sure to come up.

Java is a fairly easy programming language to learn, so if you are starting to learn any programming language, then Java could be your great choice. There's also plenty of Java tools that make it easy for developers and beginners to learn Java and develop applications. There are many other good reasons that make Java the first choice of any programmer:

- Java is open source, which means it's available free of cost.
- Java is simple and so easy to learn.
- Java is much in demand and ensures a high salary.
- Java has a large, vibrant community.
- Java has powerful development tools.
- Java is platform-independent.

Who should Learn Java

This **Java tutorial** will help both students as well as working professionals who want to develop applications using Java technologies like banking systems, support systems, information systems, websites, mobile apps, personal blogs, etc. We recommend reading this tutorial in the sequence listed in the index.

Today, Java is one of the most demanding programming languages, and so it has become an essential language to learn for anyone involved in the software application development process, including software developers, software designers, project managers, etc.

Prerequisites to Learn Java

Though we have tried our best to present the Core Java concepts in a simple and easy way, still, before you start learning Java, it is assumed that the readers have a reasonable exposure to any programming environment and knowledge of basic concepts such as variables, commands, syntax, etc.

Learn Java by Examples

This tutorial provides a set of Java examples. Practice these examples to learn the Java concepts better: **Java Examples**

Java Online Quizzes

This Java tutorial helps you prepare for technical interviews and certification exams. We have provided various quizzes and assignments to check your learning level. Given quizzes have multiple-choice types of questions and their answers with short explanations.

Following is a sample quiz; try to attempt any of the given answers:

Q 1 - The Java programming language was developed by which of the following :

A - Google in 1990s

B - Micorsoft in 1980s

C - Sun Microsystems in 1995

D - None of the Above

Start your online quiz **Start Java Quiz**.

Download Java

Java's latest version can be downloaded from Oracle's official website: Java Downloads

Java Certification

Get certified in Java to boost your skills and career Get certified

Frequently Asked Questions about Java Tutorial

There are some important frequently asked questions (FAQs) about Java Programming tutorial. This section lists them down along with their answers briefly - –

1. What are the 4 important concepts in Java?

Java supports abstraction, encapsulation, polymorphism, and inheritance. These are 4 major theoretical principles of object-oriented programming. But Java also works with three further OOP concepts: association, aggregation, and composition.

2. What are the benefits of Java?

- Java is free and open source
- Java is community driven and has expert leadership
- Java is fast and high-performance
- Java is easy to learn
- Java is statically typed
- Java is object-oriented
- Java supports functional programming

3. How much time will it take to learn Java?

Learning style and dedication to the amount of time you can spend each day affect a lot. However, on average, it takes around 6 to 12 months to learn Java programming.

4. Why is Java a so popular programming language?

The Java language is easily extensible because it is based on an object model. Unlike many other programming languages, Java is compiled, not into a platform-dependent machine but into platform-independent byte code.

5. What are the advantages of Java over Python?

Python and Java are two most popular programming languages among software programmers. Java is generally faster and more efficient than Python because it is a compiled language, whereas Python is an interpreted language and has simpler, more concise syntax than Java.

6. How do I start learning Java?

Here is the summarized list of tips that you can follow to start learning Java:

- First and most important is to make your mind to learn Java.
- Install Java Virtual Machine and Java Compilers on your computer system.
- Follow our tutorial step by step, starting from the very beginning.
- Read more articles, watch online courses, or buy a book on Java to enhance your knowledge in Java.
- Try to develop small software using Java and other technologies like MySQL if you want to make use of a database.

7. What are Java Technologies for Web Applications?

Java provides the following technologies to help web development:

- Java Servlet API.
- JavaServer Pages Standard Tag Library.
- JavaServer Faces Technology.
- Java Message Service API.
- JDBC API.
- Java Persistence API.
- Java Naming and Directory Interface.
- NetBeans IDE.

8. Which is the best place to learn Java?

You can use our simple and the best Java tutorial to learn Core Java and Advanced Java. We have removed all the unnecessary complexity while teaching you Java concepts. You can start learning it now Start Learning Java.

Copyright & Disclaimer

any contents or a part of contents of this e-book in any manner without written consent of the publisher.

We strive to update the contents of our website and tutorials as timely and as precisely as possible, however, the contents may contain inaccuracies or errors. Tutorials Point (I) Pvt. Ltd. provides no guarantee regarding the accuracy, timeliness or completeness of our website or its contents including this tutorial. If you discover any errors on our website or in this tutorial, please notify us at contact@tutorialspoint.com

Table of Contents

Java Tutorial

1. Java - Overview

Java programming language was originally developed by Sun Microsystems, which was initiated by James Gosling and released in 1995 as a core component of Sun Microsystems' Java platform (Java 1.0 [J2SE]).

The latest release of the Java Standard Edition is Java SE 23. With the advancement of Java and its widespread popularity, multiple configurations were built to suit various types of platforms. For example: J2EE for Enterprise Applications, J2ME for Mobile Applications.

The new J2 versions were renamed as Java SE, Java EE, and Java ME. Java is guaranteed to be **Write Once, Run Anywhere.**

Java is –

- **Object-Oriented:** In Java, everything is an object. Java can be easily extended since it is based on the object model.
- **Platform Independent:** Unlike many other programming languages, including C and C++, when Java is compiled, it is not compiled into a platform-specific machine but rather into platform-independent byte code. This byte code is distributed over the web and interpreted by the virtual machine (JVM) on whichever platform it is being run on.
- **Simple:** Java is designed to be easy to learn. If you understand the basic concept of OOP Java, it would be easy to master.
- **Secure:** With Java's secure feature, it enables the development of virus-free, tamper-free systems. Authentication techniques are based on public-key encryption.
- **Architecture-neutral:** The Java compiler generates an architecture-neutral object file format, which makes the compiled code executable on many processors with the presence of the Java runtime system.
- **Portable:** Being architecture-neutral and having no implementation-dependent aspects of the specification makes Java portable. Compiler in Java is written in ANSI C with a clean portability boundary, which is a POSIX subset.
- **Robust:** Java makes an effort to eliminate error-prone situations by emphasizing mainly compile-time error checking and runtime checking.
- **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.
- **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 an extensive amount of run-time information that can be used to verify and resolve accesses to objects in run-time.

Hello World using Java Programming

Just to give you a little excitement about Java programming, I'm going to give you a small conventional Java Hello World program, You can try it using **Edit and Run**.

```
public class MyFirstJavaProgram {


   /* This is my first java program.
    * This will print 'Hello World' as the output
    */


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

History of Java

James Gosling initiated Java language project in June 1991 for use in one of his many set-top box projects. The language, initially called 'Oak' after an oak tree that stood outside Gosling's office, also went by the name 'Green' and ended up later being renamed as Java, from a list of random words.

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.

On 13 November, 2006, Sun released much of Java as free and open source software under the terms of the GNU General Public License (GPL).

On 8 May, 2007, Sun finished the process, making all of Java's core code free and open-source, aside from a small portion of code to which Sun did not hold the copyright.

Tools you will need

For performing the examples discussed in this tutorial, you will need a Pentium 200-MHz computer with a minimum of 64 MB of RAM (128 MB of RAM recommended).

You will also need the following software applications –

- Linux 7.1 (or higher) or Windows xp/7/8 operating system or higher
- Java JDK 8 or higher
- Microsoft Notepad or any other text editor

This tutorial will provide the necessary skills to create GUI, networking, and web applications using Java.

2. Java - History

History of Java

Java programming language was originally developed by Sun Microsystems which was initiated by James Gosling and released in 1995 as core component of Sun Microsystems' Java platform (Java 1.0 [J2SE]). History of even naming of the Java is very interesting. It went under many names.

Java Name History

GreenTalk

James Gosling was leading a team named as 'Green' team. Target of this team was to create a new project. Initially C++ was the original choice to develop the project. James Gosling wanted to enhance C++ to achieve the target but due to high memory usage, that idea was rejected and team started with a new language initially named as GreenTalk. The file extension used as .gt. Later this language was termed as Oak and finally to Java.

Oak

James Gosling renamed language to Oak. There was an Oak tree in front of his office. James Gosling used this name as Oak represents solidarity and Oak tree is the national tree of multiple countries like USA, France, Romania etc. But Oak technologies already had Oak as a trademark and James team had to brainstrom another title for the language.

Java

Team put multiple names like DNA, Silk, Ruby and Java. Java was finalized by the team. James Gosling tabled Java title based on type of espresso coffee bean. Java is an island in Indonesia where new coffee was discovered termed as Java coffee. As per James Gosling, Java was among the top choice along with Silk. Finally Java was selected as it was quite unique and represented the essence of being dynamic, revolutionary and fun to say.

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.

On 13 November, 2006, Sun released much of Java as free and open source software under the terms of the GNU General Public License (GPL).

On 8 May, 2007, Sun finished the process, making all of Java's core code free and open-source, aside from a small portion of code to which Sun did not hold the copyright.

The latest release of the Java Standard Edition is Java SE 21. With the advancement of Java and its widespread popularity, multiple configurations were built to suit various types of platforms. For example: J2EE for Enterprise Applications, J2ME for Mobile Applications.

Java Versions History

Over the period of nearly 30 years, Java has seen many minor and major versions. Following is a brief explanation of versions of Java till date.

Sr. No.	Version	Date	Description
1	JDK Beta	1995	Initial Draft version
2	JDK 1.0	23 Jan 1996	A stable variant JDK 1.0.2 was termed as JDK 1
3	JDK 1.1	19 Feb 1997	Major features like **JavaBeans**, **RMI**, **JDBC**, **inner classes** were added in this release.
4	JDK 1.2	8 Dec 1998	**Swing**, **JIT Compiler**, Java Modules, **Collections** were introduced to JAVA and this release was a great success.
5	JDK 1.3	8 May 2000	HotSpot **JVM**, JNDI, JPDA, JavaSound and support for Synthetic proxy classes were added.
6	JDK 1.4	6 Feb 2002	Image I/O API to create/read JPEG/PNG image were added. Integrated **XML parser** and XSLT processor (JAXP) and Preferences API were other important updates.
7	JDK 1.5 or J2SE 5	30 Sep 2004	Various new features were added to the language like foreach, var-args, generics etc.
8	JAVA SE 6	11 Dec 2006	1. notation was dropped to SE and upgrades done to JAXB 2.0, JSR 269 support and JDBC 4.0 support added.
9	JAVA SE 7	7 Jul 2011	Support for dynamic languages added to JVM. Another enhancements included string in switch case, compressed 64 bit pointers etc.
10	JAVA SE 8	18 Mar 2014	Support for functional programming added. **Lambda expressions**, **streams**, **default methods**, new **date-time APIs** introduced.

11	JAVA SE 9	21 Sep 2017	**Module system** introduced which can be applied to JVM platform.
12	JAVA SE 10	20 Mar 2018	Unicode language-tag extensions added. Root certificates, threadlocal handshakes, support for heap allocation on alternate memory devices etc. were introduced.
13	JAVA SE 11	5 Sep 2018	Dynamic class-file constants, Epsilon a no-op garbage collector, local-variable support in lambda parameters, Low-overhead heap profiling support added.
14	JAVA SE 12	19 Mar 2019	Experimental Garbage Collector, Shenandoah: A Low-Pause-Time Garbage Collector, Microbenchmark Suite, JVM Constants API added.
15	JAVA SE 13	17 Sep 2019	Feature added - Text Blocks (Multiline strings), Enhanced Thread-local handshakes.
16	JAVA SE 14	17 Mar 2020	Feature added - Records, a new class type for modelling, **Pattern Matching for instanceof**, Intuitive **NullPointerException handling**.
17	JAVA SE 15	15 Sep 2020	Feature added - **Sealed Classes**, **Hidden Classes**, Foreign Function and Memory API (Incubator).
18	JAVA SE 16	16 Mar 2021	Feature added as preview - Records, Pattern Matching for switch, Unix Domain Socket Channel (Incubator) etc.
19	JAVA SE 17	14 Sep 2021	Feature added as finalized - Sealed Classes, Pattern Matching for instanceof, Strong encapsulation of JDK internals by default. New macOS rendering pipeline etc.
20	JAVA SE 18	22 Mar 2022	Feature added - UTF-8 by Default, Code Snippets in Java API Documentation, Vector API (Third incubator), Foreign Function, Memory API (Second Incubator) etc.

21	JAVA SE 19	20 Sep 2022	Feature added - Record pattern, Vector API (Fourth incubator), Structured Concurrency (Incubator) etc.
22	JAVA SE 20	21 Mar 2023	Feature added - Scoped Values (Incubator), Record Patterns (Second Preview), Pattern Matching for switch (Fourth Preview),Foreign Function & Memory API (Second Preview) etc.
22	JAVA SE 21	19 Sep 2023	Feature added - String Templates (Preview), Sequenced Collections, Generational ZGC, Record Patterns, Pattern Matching for switch etc.
23	Java SE 22	19 Mar 2024	Feature added - Region Pinning for G1 garbage collector, foreign functions and memory APIs , multi-file source code programs support, string templates, vector apis (seventh incubator), unnamed variables, patterns, stream gatherers (first preview) etc.
24	Java SE 23	17 Sep 2024	Feature added - Primitive types in patterns, class file APIs, vector APIs (Eighth incubator), stream gatherers (second preview), ZDC, generation mode by default etc.

3. Java - Features

Java programming language was initially developed to work on embedded systems, settop boxes, television. So by requirements, it was initially designed to work on varied platforms. Over the period of multiple years, Java evolved to become one of the most popular language used to develop internet based applications.

Java is a feature rich language and with every new version, it is continuously evolving. It is widely used across billions of devices. Following are the main features of the Java language -

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

Object Oriented

In Java, everything is an Object. Java can be easily extended since it is based on the Object model. As a language that has the Object-Oriented feature, Java supports the following fundamental concepts of OOPs –

- Polymorphism
- Inheritance
- Encapsulation
- Abstraction
- Classes
- Objects
- Instance
- Method
- Message Passing

Platform Independent

Unlike many other programming languages including C and C++, when Java is compiled, it is not compiled into platform specific machine. It is compiled into platform independent byte code. This byte code is distributed over the web and interpreted by the Java Virtual Machine (JVM) on whichever platform it is being run on.

Java is designed in **Write Once, Run Anywhere** (WORA) way. Code written in Java is not directly dependent on the type of machine it is running. A code in Java is compiled in ByteCode which is platform independent. Java Virtual Machine (JVM) can understand the byte code. Java provides platform specific JVMs. It is the responsibility of platform specific JVM to interpret the byte code correctly thus developers are free to write code without worrying about platforms like Windows, Linux, Unix, Mac etc. This feature makes Java a platform neutral language.

Byte code can be distributed over the web and interpreted by the Virtual Machine (JVM) on whichever platform it is being run on. It makes java code highly portable and useful for application running on multiple platforms.

Simple

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

Java is very easy to learn. It inherits many features from C, C++ and removes complex features like pointers, operator overloading, multiple inheritance, explicit memory allocation etc. It provides automatic garbage collection. With a rich set of libraries with thousands of useful functions, Java makes developers' life easy.

Secure

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

Java is by design highly secure as it is not asking developers to interact with underlying system memory or operation system. Bytecode is secure and several security flaws like buffer overflow, memory leak are very rare. Java exception handling mechanism allows developers to handle almost all type of error/exceptions which can happen during program execution. Automatic garbage collection helps in maintaining the system memory space utilization in check.

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.

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. With advancement in processor architectures or machine specific processors, Java code remains independent of any specific requirement of a processor. As java is an open standard, even a specific JVM can be prepared for a custom architecture. As in today's time, we've JVM available for almost all popular platforms, architectures, Java code is completely independent. For example, a Java program created in Windows machine can run on Linux machine without any code modification.

Portable

Being architecture-neutral and having no implementation dependent aspects of the specification makes Java portable. Compiler in Java is written in ANSI C with a clean portability boundary, which is a POSIX subset.

Due to this portability, Java was an instant hit since inception. It was particularly useful for internet based applications where platforms varied from place to place and same code base can be used across multiple platforms. So collaboration between developers was easy across multiple locations.

Robust

Java makes an effort to eliminate error prone situations by emphasizing mainly on compile time error checking and runtime checking. Automatic garbage collection, strong memory management, no pointers, no direct access to system memory, exception handling, error handling are some of the key features which makes Java a Robust, strong language to rely on.

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.

A multi-threaded program contains two or more parts that can run concurrently and each part can handle a different task at the same time making optimal use of the available resources specially when your computer has multiple CPUs.

By definition, multitasking refers to multiple processes that share common processing resources such as a CPU. Multithreading extends the idea of multitasking into applications where you can subdivide specific operations within a single application into individual threads. Each of the threads can run in parallel. The OS divides processing time not only among different applications, but also among each thread within an application.

Multi-threading enables you to write in a way where multiple activities can proceed concurrently in the same program.

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.

JVM sits in between the javac compiler and the underlying hardware. The javac (or any other compiler) compiler compiles Java code into the Bytecode, which is understood by a platform specific JVM. The JVM then compiles the Bytecode in binary using JIT (Just-in-time) compilation, as the code executes.

High Performance

With the use of Just-In-Time compilers, Java enables high performance. JVM uses JIT compiler to improve the execution time of the program. Below are some general optimizations that are done by the JIT compilers:

- Method inlining

- Dead code elimination
- Heuristics for optimizing call sites
- Constant folding

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.