1) What is Java?

[Java](https://www.javatpoint.com/java-tutorial) is the high-level, [object-oriented](https://www.javatpoint.com/java-oops-concepts) programming language. It is used for:

Develop Mobile applications (specially Android apps) Desktop applications. Web applications. Web servers and application servers. Games.

### 2) What are the differences between C++ and Java?

C++ is platform-dependent.

Java is platform-independent.

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

C++ supports multiple inheritance.

Java doesn't support multiple inheritance through class. It can be achieved by [interfaces in java](https://www.javatpoint.com/interface-in-java).

C++ supports [operator overloading](https://www.javatpoint.com/cpp-overloading).

Java doesn't support operator overloading.

### 3) List the features of Java Programming language.

* **Simple:** Java is easy to learn. The syntax of Java is based on C++ which makes easier to write the program in it.
* **Object-Oriented:** Java follows the object-oriented paradigm which allows us to maintain our code as the combination of different type of objects that incorporates both data and behavior.
* **Portable:** Java supports read-once-write-anywhere approach. We can execute the Java program on every machine. Java program (.java) is converted to bytecode (.class) which can be easily run on every machine.
* **Interpreted:** Java uses the Just-in-time (JIT) interpreter along with the compiler for the program execution.
* **High Performance:** Java is faster than other traditional interpreted programming languages because Java bytecode is "close" to native code. It is still a little bit slower than a compiled language (e.g., C++).

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 300 Core Java Interview Questions | Set 190% assurance of interview questions There is the list of 300 core Java interview questions. If there is any core Java interview question that has been asked to you, kindly post it in the ask question section. We assure that you will get here the 90% frequently asked interview questions and answers.  The answers to the Core Java interview questions are short and to the point. The core Java interview questions are categorized in Basics of Java interview questions, OOPs interview questions, String Handling interview questions, Multithreading interview questions, collection interview questions, JDBC interview questions, etc.  [**1**](https://www.javatpoint.com/corejava-interview-questions) [**2**](https://www.javatpoint.com/corejava-interview-questions-2) [**3**](https://www.javatpoint.com/java-multithreading-interview-questions) [**4**](https://www.javatpoint.com/java-collections-interview-questions) [**5**](https://www.javatpoint.com/jdbc-interview-questions) Core Java: Basics of Java Interview Questions1) What is Java? [Java](https://www.javatpoint.com/java-tutorial) is the high-level, [object-oriented](https://www.javatpoint.com/java-oops-concepts), robust, secure programming language, platform-independent, high performance, Multithreaded, and portable programming language. It was developed by [**James Gosling**](https://www.javatpoint.com/james-gosling-father-of-java) in June 1991. It can also be known as the platform as it provides its own JRE and API. 2) What are the differences between C++ and Java? The differences between [C++](https://www.javatpoint.com/cpp-tutorial) and Java are given in the following table.   |  |  |  | | --- | --- | --- | | **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. | | **Design Goal** | C++ was designed for systems and applications programming. It was an extension of [C programming language](https://www.javatpoint.com/c-programming-language-tutorial). | Java was designed and created as an interpreter for printing systems but later extended as a support network computing. It was designed with a goal of being easy to use and accessible to a broader audience. | | **Goto** | C++ supports the [goto](https://www.javatpoint.com/cpp-goto-statement) statement. | Java doesn't support the goto statement. | | **Multiple inheritance** | C++ supports multiple inheritance. | Java doesn't support multiple inheritance through class. It can be achieved by [interfaces in java](https://www.javatpoint.com/interface-in-java). | | **Operator Overloading** | C++ supports [operator overloading](https://www.javatpoint.com/cpp-overloading). | Java doesn't support operator overloading. | | **Pointers** | C++ supports [pointers](https://www.javatpoint.com/cpp-pointers). You can write pointer program in C++. | Java supports pointer internally. However, 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. C++ is compiled and run using the compiler which converts source code into machine code so, C++ is platform dependent. | Java uses compiler and interpreter both. Java source code is converted into bytecode at compilation time. The interpreter executes this bytecode at runtime and produces output. Java is interpreted that is why it is platform independent. | | **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](https://www.javatpoint.com/multithreading-in-java) 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 a single inheritance tree always because all classes are the child of Object class in java. The object class is the root of the [inheritance](https://www.javatpoint.com/inheritance-in-java) tree in java. | | **Hardware** | C++ is nearer to hardware. | Java is not so interactive with hardware. | | **Object-oriented** | C++ is an object-oriented language. However, in C language, single root hierarchy is not possible. | Java is also an [object-oriented](https://www.javatpoint.com/java-oops-concepts) language. However, everything (except fundamental types) is an object in Java. It is a single root hierarchy as everything gets derived from java.lang.Object. |  3) List the features of Java Programming language. There are the following features in Java Programming Language.   * **Simple:** Java is easy to learn. The syntax of Java is based on C++ which makes easier to write the program in it. * **Object-Oriented:** Java follows the object-oriented paradigm which allows us to maintain our code as the combination of different type of objects that incorporates both data and behavior. * **Portable:** Java supports read-once-write-anywhere approach. We can execute the Java program on every machine. Java program (.java) is converted to bytecode (.class) which can be easily run on every machine. * **Platform Independent:** Java is a platform independent programming language. It is different from other programming languages like C and C++ which needs a platform to be executed. Java comes with its platform on which its code is executed. Java doesn't depend upon the operating system to be executed. * **Secured:** Java is secured because it doesn't use explicit pointers. Java also provides the concept of ByteCode and Exception handling which makes it more secured. * **Robust:** Java is a strong programming language as it uses strong memory management. The concepts like Automatic garbage collection, Exception handling, etc. make it more robust. * **Architecture Neutral:** Java is architectural neutral as it is not dependent on the architecture. In C, the size of data types may vary according to the architecture (32 bit or 64 bit) which doesn't exist in Java. * **Interpreted:** Java uses the Just-in-time (JIT) interpreter along with the compiler for the program execution. * **High Performance:** Java is faster than other traditional interpreted programming languages because Java bytecode is "close" to native code. It is still a little bit slower than a compiled language (e.g., C++). * **Multithreaded:** 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. * **Distributed:** Java is distributed because it facilitates users to create distributed applications in Java. RMI and EJB are used for creating distributed applications. This feature of Java makes us able to access files by calling the methods from any machine on the internet. * **Dynamic:** Java is a dynamic language. It supports dynamic loading of classes. It means classes are loaded on demand. It also supports functions from its native languages, i.e., C and C++.  4) What do you understand by Java virtual machine? [Java Virtual Machine](https://www.javatpoint.com/jvm-java-virtual-machine) is a virtual machine that enables the computer to run the Java program. JVM acts like a run-time engine which calls the main method present in the Java code. JVM is the specification which must be implemented in the computer system. The Java code is compiled by JVM to be a Bytecode which is machine independent and close to the native code. 5) What is the difference between JDK, JRE, and JVM?JVM JVM is an acronym for Java Virtual Machine; it is an abstract machine which provides the runtime environment in which Java bytecode can be executed. It is a specification which specifies the working of Java Virtual Machine. Its implementation has been provided by Oracle and other companies. Its implementation is known as JRE.  JVMs are available for many hardware and software platforms (so JVM is platform dependent). It is a runtime instance which is created when we run the Java class. There are three notions of the JVM: specification, implementation, and instance. JRE JRE stands for Java Runtime Environment. It is the implementation of JVM. The Java Runtime Environment is a set of software tools which are used for developing Java applications. It is used to provide the runtime environment. It is the implementation of JVM. It physically exists. It contains a set of libraries + other files that JVM uses at runtime. JDK JDK is an acronym for Java Development Kit. It is a software development environment which is used to develop Java applications and applets. It physically exists. It contains JRE + development tools. JDK is an implementation of any one of the below given Java Platforms released by Oracle Corporation:   * Standard Edition Java Platform * Enterprise Edition Java Platform * Micro Edition Java Platform |