1.Difference Between JDK, JRE, and JVM

|  |  |  |  |
| --- | --- | --- | --- |
| **Parameter** | **JDK** | **JRE** | **JVM** |
| Full-Form | The JDK is an abbreviation for Java Development Kit. | The JRE is an abbreviation for Java Runtime Environment. | The JVM is an abbreviation for Java Virtual Machine. |
| Definition | The JDK (Java Development Kit) is a software development kit that develops applications in Java. Along with JRE, the JDK also consists of various development tools (Java Debugger, JavaDoc, compilers, etc.) | The Java Runtime Environment (JRE) is an implementation of JVM. It is a type of software package that provides class libraries of Java, JVM, and various other components for running the applications written in Java programming. | The Java Virtual Machine (JVM) is a platform-independent abstract machine that has three notions in the form of specifications. This document describes the requirement of JVM implementation. |
| Functionality | The JDK primarily assists in executing codes. It primarily functions in development. | JRE has a major responsibility for creating an environment for the execution of code. | JVM specifies all of the implementations. It is responsible for providing all of these implementations to the JRE. |
| Platform Dependency | The JDK is platform-dependent. It means that for every different platform, you require a different JDK. | JRE, just like JDK, is also platform-dependent. It means that for every different platform, you require a different JRE. | The JVM is platform-independent. It means that you won’t require a different JVM for every different platform. |
| Tools | Since JDK is primarily responsible for the development, it consists of various tools for debugging, monitoring, and developing java applications. | JRE, on the other hand, does not consist of any tool- like a debugger, compiler, etc. It rather contains various supporting files for JVM, and the class libraries that help JVM in running the program. | JVM does not consist of any tools for software development. |
| Implementation | **JDK** = Development Tools + JRE (Java Runtime Environment) | **JRE** = Libraries for running the application + JVM (Java Virtual Machine) | **JVM** = Only the runtime environment that helps in executing the Java bytecode. |
| Why Use It? | Why use JDK?  Some crucial reasons to use JDK are:   * It consists of various tools required for writing Java programs. * JDK also contains JRE for executing Java programs. * It includes an Appletviewer, Java application launcher, compiler, etc. * The compiler helps in converting the code written in Java into bytecodes. * The Java application launcher helps in opening a JRE. It then loads all of the necessary details and then executes all of its main methods. | Why use JRE?  Some crucial reasons to use JRE are:   * If a user wants to run the Java applets, then they must install JRE on their system. * The JRE consists of class libraries along with JVM and its supporting files. It has no other tools like a compiler or a debugger for Java development. * JRE uses crucial package classes like util, math, awt, lang, and various runtime libraries. | Why use JVM?  Some crucial reasons to use JVM are:   * It provides its users with a platform-independent way for executing the Java source code. * JVM consists of various tools, libraries, and multiple frameworks. * The JVM also comes with a Just-in-Time (JIT) compiler for converting the Java source code into a low-level machine language. Thus, it ultimately runs faster than any regular application. * Once you run the Java program, you can run JVM on any given platform to save your time. |
| Features | Features of JDK   * Here are a few crucial features of JDK: * It has all the features that JRE does. * JDK enables a user to handle multiple extensions in only one catch block. * It basically provides an environment for developing and executing the Java source code. * It has various development tools like the debugger, compiler, etc. * One can use the Diamond operator to specify a generic interface in place of writing the exact one. * Any user can easily install JDK on Unix, Mac, and Windows OS (Operating Systems). | Features of JRE   * Here are a few crucial features of JRE: * It is a set of tools that actually helps the JVM to run. * The JRE also consists of deployment technology. It includes Java Plug-in and Java Web Start as well. * A developer can easily run a source code in JRE. But it does not allow them to write and compile the concerned Java program. * JRE also contains various integration libraries like the JDBC (Java Database Connectivity), JNDI (Java Naming and Directory Interface), RMI (Remote Method Invocation), and many more. * It consists of the JVM and virtual machine client for Java HotSpot. | Features of JVM  Here are a few crucial features of JVM:   * The JVM enables a user to run applications on their device or in a cloud environment. * It helps in converting the bytecode into machine-specific code. * JVM also provides some basic Java functions, such as garbage collection, security, memory management, and many more. * It uses a library along with the files given by JRE (Java Runtime Environment) for running the program. * Both JRE and JDK contain JVM. * It is easily customizable. For instance, a user can feasibly allocate a maximum and minimum memory to it. * JVM can also execute a Java program line by line. It is thus also known as an interpreter. * JVM is also independent of the OS and hardware. It means that once a user writes a Java program, they can easily run it anywhere. |

2 . What is JIT Compiler?(just in time compiler)

**When we compile our Java program (e.g., using the javac command), we'll end up with our source code compiled into the binary representation of our code – a JVM bytecode**. This bytecode is simpler and more compact than our source code, but conventional processors in our computers cannot execute it.

**To be able to run a Java program, the JVM interprets the bytecode**. Since interpreters are usually a lot slower than native code executing on a real processor, the **JVM can run another compiler which will now compile our bytecode into the machine code that can be run by the processor**. This so-called just-in-time compiler is much more sophisticated than the javac compiler, and it runs complex optimizations to generate high-quality machine code.

## More Detailed Look into the JIT Compiler

The JDK implementation by Oracle is based on the open-source OpenJDK project. This includes the **HotSpot virtual machine**, available since Java version 1.3. It **contains two conventional JIT-compilers: the client compiler, also called C1 and the server compiler, called opto or C2**.

C1 is designed to run faster and produce less optimized code, while C2, on the other hand, takes a little more time to run but produces a better-optimized code. The client compiler is a better fit for desktop applications since we don't want to have long pauses for the JIT-compilation. The server compiler is better for long-running server applications that can spend more time on the compilation.

3. Class Loader

The **Java ClassLoader** is a part of the [**Java Runtime Environment**](https://www.geeksforgeeks.org/differences-jdk-jre-jvm/) that dynamically loads Java classes into the [**Java Virtual Machine**](https://www.geeksforgeeks.org/jvm-works-jvm-architecture/). The Java run time system does not need to know about files and file systems because of classloaders. [Java classes](https://www.geeksforgeeks.org/classes-objects-java/) aren’t loaded into memory all at once, but when required by an application. At this point, the **Java ClassLoader** is called by the **JRE** and these ClassLoaders load classes into memory dynamically.

4. What gives java WORA (write once run anywhere nature)

Java applications are called WORA (Write Once Run Anywhere). This means a programmer can develop Java code on one system and can expect it to run on any other Java-enabled system without any adjustment. This is all possible because of JVM.

5. What was original name of java? Why it was renamed?

The language was initially called Oak after an oak tree that stood outside Gosling's office but it was already trademarked by oak technologies. Later the project went by the name Green and was finally renamed Java, from Java coffee, a type of coffee from Indonesia.

6. List Features of java

### Simple

Java is very easy to learn, and its syntax is simple, clean and easy to understand. According to Sun Microsystem, Java language is a simple programming language because:

* Java syntax is based on C++ (so easier for programmers to learn it after C++).
* Java has removed many complicated and rarely-used features, for example, explicit pointers, operator overloading, etc.

### Platform Independent



Java is platform independent because it is different from other languages like [C](https://www.javatpoint.com/c-programming-language-tutorial), [C++](https://www.javatpoint.com/cpp-tutorial), etc. which are compiled into platform specific machines while Java is a write once, run anywhere language. 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 a software-based platform.

The Java platform differs from most other platforms in the sense that it is 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)

Java code can be executed on multiple platforms, for example, 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 best known for its security. With Java, we can develop virus-free systems. Java is secured because:

* **No explicit pointer**
* **Java Programs run inside a virtual machine sandbox**



* **Classloader:** Classloader in Java is a part of the Java Runtime Environment (JRE) which is used to load Java classes into the Java Virtual Machine dynamically. It adds security by separating the package for the classes of the local file system from those that are imported from network sources.
* **Bytecode Verifier:** It checks the code fragments for illegal code that can violate access rights to objects.
* **Security Manager:** It determines what resources a class can access such as reading and writing to the local disk.

Java language provides these securities by default. Some security can also be provided by an application developer explicitly through SSL, JAAS, Cryptography, etc.

### Robust

The English mining of Robust is strong. Java is robust because:

* It uses strong memory management.
* There is a lack of pointers that avoids security problems.
* Java provides automatic garbage collection which runs on the Java Virtual Machine to get rid of objects which are not being used by a Java application anymore.
* There are exception handling and the type checking mechanism in Java. All these points make Java robust.

### Architecture-neutral

Java is architecture neutral because there are no implementation dependent features, for example, the 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. However, it occupies 4 bytes of memory for both 32 and 64-bit architectures in Java.

### Portable

Java is portable because it facilitates you to carry the Java bytecode to any platform. It doesn't require any implementation.

### 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++). Java is an interpreted language that is why it is slower than compiled languages, e.g., C, C++, 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.

### 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.

### Dynamic

Java is a dynamic language. It supports the dynamic loading of classes. It means classes are loaded on demand. It also supports functions from its native languages, i.e., C and C++.

Java supports dynamic compilation and automatic memory management (garbage collection).

* There is no need to remove unreferenced objects because there is an Automatic Garbage Collection in Java.

### Object-oriented

Java is an [object-oriented](https://www.javatpoint.com/java-oops-concepts) programming language. Everything in Java is an object. Object-oriented means we organize our software as a combination of different types of objects that incorporate both data and behavior.

Object-oriented programming (OOPs) is a methodology that simplifies software development and maintenance by providing some rules.

# 7.What are Datatypes in java?

# Data Types in Java

Data types specify the different sizes and values that can be stored in the variable. There are two types of data types in Java:

1. **Primitive data types:** The primitive data types include boolean, char, byte, short, int, long, float and double.
2. **Non-primitive data types:** The non-primitive data types include [Classes](https://www.javatpoint.com/object-and-class-in-java), [Interfaces](https://www.javatpoint.com/interface-in-java), and [Arrays](https://www.javatpoint.com/array-in-java).

## Java Primitive Data Types

In Java language, primitive data types are the building blocks of data manipulation. These are the most basic data types available in [Java language](https://www.javatpoint.com/java-tutorial).

Java is a statically-typed programming language. It means, all [variables](https://www.javatpoint.com/java-variables) must be declared before its use. That is why we need to declare variable's type and name.

There are 8 types of primitive data types:

* boolean data type
* byte data type
* char data type
* short data type
* int data type
* long data type
* float data type
* double data type



|  |  |  |
| --- | --- | --- |
| **Data Type** | **Default Value** | **Default size** |
| boolean | false | 1 bit |
| char | '\u0000' | 2 byte |
| byte | 0 | 1 byte |
| short | 0 | 2 byte |
| int | 0 | 4 byte |
| long | 0L | 8 byte |
| float | 0.0f | 4 byte |
| double | 0.0d | 8 byte |

## 8.What is a Platform, and What is a Platform-Independent Language?

The platform can be defined as a distinct combination of hardware, operating system, and software that provides an **environment to run programs**.

Java is called Platform Independent because programs written in Java can be run on multiple platforms without re-writing them individually for a particular platform, i.e., **Write Once Run Anywhere (WORA)**.

* The compilation process in Java  **generates a special format called byte code**. This generated file is also referred to as a .class file.

The Byte Code of Java is a set of machine instructions for a Java processor chip called Java Virtual Machine (JVM). Java Byte Code is very similar to machine language, but unlike machine language, Java byte code is absolutely the same on every platform. The byte code is not directly executable on any platform. Java code compiled into byte code still needs an interpreter to execute them on the platform.

Java Virtual Machine or JVM is a special Java Interpreter. **Java Virtual Machine takes byte code as input. It interprets and executes the byte code and provides the output of the program**.

## 9. What is Byte Code?

A byte code acts as an intermediate code present between a machine code and a source code.

A byte code is basically a low-level code that results from the compilation of source code that might be present in a high-level language.

A virtual machine such as a JVM (Java Virtual Machine) processes a byte code.

Machines cannot understand a byte code. It is a non-runnable type of code that becomes machine-understandable after an interpreter translates it into a machine code. One needs to compile it to run on a JVM. Thus, any system that already has JVM can easily run such a code irrespective of the OS (operating system). The Java platform is, thus, platform independent.

A byte code is also sometimes known as a portable code.

## What is Machine Code?

It basically refers to a set of various instructions that a machine can read and understand directly. The CPU (Central Processing Unit) can directly process the available machine code. The machine code is present in a binary format of 0s and 1s. Thus, it is completely different from the source code as well as the byte code.

| **Parameters** | **Byte Code** | **Machine Code** |
| --- | --- | --- |
| Definition and Meaning | A byte code acts as an intermediate code present between a machine code and a source code. | It basically refers to a set of various instructions that a machine can read and understand directly. |
| Level of Code | It is an intermediate-level code. | It is a low-level code. |
| Type of Instructions | It consists of hexadecimal, binary, and macro instructions such as swap, add, new, etc. | It consists of instructions in the binary language. Thus, the instructions are present in the codes of 0s and 1s. |
| CPU Understandable | A CPU cannot understand it directly. | Any CPU can directly understand as well as process this type of code. |
| Generation and Execution | We generate a byte code after source code compilation. But a CPU cannot directly run it. It entirely depends on an interpreter for its execution. | It is basically machine language. Thus the CPU can process it. It is present in binary format and does not need separate interpretation or compilation. |

10. Difference b/w jar file and Runnable jar file?

In simple terms, the difference between a JAR file and a Runnable JAR is that while a JAR file is a Java application which requires a command line to run, a runnable JAR file can be directly executed by double clicking it.

11. **Difference Between Path and ClassPath in Java**

The**ClassPath** is a parameter in the Java Virtual Machine(JVM) or the Java compiler that is used by a system or application ClassLoader to locate and load compiled Java bytecodes stored in the “.class” file.

On the other hand, The **Path** is also an environment variable path that behaves as a mediator between the operating system and developer to inform binary file path.