|  |  |
| --- | --- |
| A compiler translates the entire source code in a single run. | An interpreter translates the entire source code line by line. |
| It consumes less time i.e., it is faster than an interpreter. | It consumes much more time than the compiler i.e., it is slower than the compiler. |
| It is more efficient. | It is less efficient. |
| CPU utilization is more. | CPU utilization is less as compared to the compiler. |
| Both syntactic and semantic errors can be checked simultaneously. | Only syntactic errors are checked. |
| The compiler is larger. | Interpreters are often smaller than compilers. |
| It is not flexible. | It is flexible. |
| The localization of errors is difficult. | The localization of error is easier than the compiler. |
| A presence of an error can cause the whole program to be re-organized. | A presence of an error causes only a part of the program to be re-organized. |
| The compiler is used by the language such as C, C++. | An interpreter is used by languages such as Java. |

Q1. What is the difference between Compiler and Interpreter?

Q2. What is the difference between JDK, JRE, and JVM?

Ans : 1. JDK (Java Development Kit) is a Kit that provides the environment to develop and execute(run) the Java program. JDK is a kit(or package) that includes two things

* Development Tools(to provide an environment to develop your java programs)
* JRE (to execute your java program).

2. JRE (Java Runtime Environment) is an installation package that provides an environment to only run(not develop) the java program(or application)onto your machine. JRE is only used by those who only want to run Java programs that are end-users of your system.

3. [JVM (Java Virtual Machine)](https://www.geeksforgeeks.org/jvm-works-jvm-architecture/) is a very important part of both JDK and JRE because it is contained or inbuilt in both. Whatever Java program you run using JRE or JDK goes into JVM and JVM is responsible for executing the java program line by line, hence it is also known as an.

Q3. How many types of memory areas are allocated by JVM?

Ans :

1. Class (Method) Area
2. Heap
3. Stack
4. 4.Program Counter Register
5. Native Method Stack

Let’s see about them in brief:

1. Class (Method) Area

The class method area is the memory block that stores the class code, variable code(static variable, runtime constant), method code, and the constructor of a Java program. (Here method means the function which is written inside the class). It stores class-level data of every class such as the runtime constant pool, field and method data, the code for methods.

2. Heap

The Heap area is the memory block where objects are created or objects are stored. Heap memory allocates memory for class interfaces and arrays (an array is an object). It is used to allocate memory to objects at run time

3. Stack

Each thread has a private JVM stack, created at the same time as the thread. It is used to store data and partial results which will be needed while returning value for method and performing dynamic linking.

Java Stack stores frames and a new frame is created each time at every invocation of the method. A frame is destroyed when its method invocation completes

4. Program Counter Register:

Each JVM thread that carries out the task of a specific method has a program counter register associated with it. The non-native method has a PC that stores the address of the available JVM instruction whereas, in a native method, the value of the program counter is undefined. PC register is capable of storing the return address or a native pointer on some specific platform.

5. Native method Stacks:

Also called C stacks, native method stacks are not written in Java language. This memory is allocated for each thread when it’s created And it can be of a fixed or dynamic nature.

Q4. What is JIT compiler?

Ans: The JIT compiler helps improve the performance of Java programs by compiling bytecodes into native machine code at run time. The JIT compiler is enabled by default. When a method has been compiled, the JVM calls the compiled code of that method directly instead of interpreting it.

Q5. What are the various access specifiers in Java?

Ans : The access specifiers also determine which data members (methods or fields) of a class can be accessed by other data members of classes or packages etc. To ensure encapsulation and reusability, these access specifiers/modifiers are an integral part of object-oriented programming.

Modifiers in Java are of two types:

#1) Access Modifiers

Access modifiers in Java allow us to set the scope or accessibility or visibility of a data member be it a field, constructor, class, or method.

#2) Non-access Modifiers

Java also provides non-access specifiers that are used with classes, variables, methods, constructors, etc. The non-access specifiers/modifiers define the behavior of the entities to the JVM.

Some of the non-access specifiers/modifiers in Java are:

* static
* final
* abstract
* transient
* volatile
* synchronized
* native

We have covered static, synchronized, and volatile keywords in our earlier tutorials. We will cover the other non-access modifiers in our future tutorials as they are beyond the scope of this tutorial.

Types Of Access Modifiers In Java

Java provides four types of access specifiers that we can use with classes and other entities.

These are:

#1) Default: Whenever a specific access level is not specified, then it is assumed to be ‘default’. The scope of the default level is within the package.

#2) Public: This is the most common access level and whenever the public access specifier is used with an entity, that particular entity is accessible throughout from within or outside the class, within or outside the package, etc.

#3) Protected: The protected access level has a scope that is within the package. A protected entity is also accessible outside the package through inherited class or child class.

#4) Private: When an entity is private, then this entity cannot be accessed outside the class. A private entity can only be accessible from within the class.

Q6. What is a compiler in Java?

Ans : A Java compiler is a program that takes the text file work of a developer and compiles it into a platform-independent Java file. Java compilers include the Java Programming Language Compiler (javac), the GNU Compiler for Java (GCJ), the Eclipse Compiler for Java (ECJ) and Jikes.

Q7. Explain the types of variables in Java?

Ans : Based on scope, variables can be of four different types - Class variables, Instance variables, Local variables and Parameters. Scope of a variable is determined based on where it is declared within a java class.

Q8. What are the Data types in Java?

Ans : Data types are divided into two groups:

* Primitive data types - includes byte , short , int , long , float , double , boolean and char.
* Non-primitive data types - such as String , Arrays and Classes .

Q9. What are the identifiers in java?

Ans : In programming languages, identifiers are used for identification purposes. In Java, an identifier can be a class name, method name, variable name, or label. For

example :

public class Test {

public static void main(String[] args) {

int a = 20;

}

}

Q10. Explain the architecture of JVM?

Ans : JVM(Java Virtual Machine) acts as a run-time engine to run Java applications. JVM is the one that actually calls the main method present in a java code. JVM is a part of JRE(Java Runtime Environment).

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.

When we compile a *.java* file, *.class* files(contains byte-code) with the same class names present in *.java* file are generated by the Java compiler. This *.class* file goes into various steps when we run it. These steps together describe the whole JVM.

