1.

|  |  |
| --- | --- |
| 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. |

2. JVM: JVM (Java Virtual Machine) is an abstract machine. It is called a virtual machine because it doesn't physically exist. It is a specification that provides a runtime environment in which Java bytecode can be executed. It can also run those programs which are written in other languages and compiled to Java bytecode.

JRE: JRE is an acronym for Java Runtime Environment. It is also written as Java RTE. 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. The Java Development Kit (JDK) 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

3. Heap − Runtime storage allocation for objects (reference types).

Stack − Storage for local variables and partial results. A stack contains frames and allocates one for each thread. Once a thread gets completed, this frame also gets destroyed. It also plays roles in method invocation and return.

PC Registers − Program Counter Registers contains the address of an instruction that JVM is currently executing.

Execution Engine − It has a virtual processor, interpreter to interpret bytecode instructions one by one and a JIT, just in time compiler.

Native method stacks − It contains all the native methods used by the application.

4. **JIT** is a part of the JVM that optimizes the performance of the application. **JIT** stands for **Java-In-Time Compiler**. The **JIT compilation** is also known as dynamic compilation. It accelerates execution performance many times over the previous level. In other words, it is a long-running, computer-intensive program that provides the best performance environment. It optimizes the performance of the Java application at compile or run time.

5. There are four types of Java access modifiers:

1. **Private**: The access level of a private modifier is only within the class. It cannot be accessed from outside the class.
2. **Default**: The access level of a default modifier is only within the package. It cannot be accessed from outside the package. If you do not specify any access level, it will be the default.
3. **Protected**: The access level of a protected modifier is within the package and outside the package through child class. If you do not make the child class, it cannot be accessed from outside the package.
4. **Public**: The access level of a public modifier is everywhere. It can be accessed from within the class, outside the class, within the package and outside the package.

6. A compiler in Java is a computer program that is used for compiling Java programs. It is platform-independent. It converts (translates) source code (.java file) into bytecode (.class file).

In other words, the compiler (javac.exe) generates bytecode during the compilation process.

7. There are three types of variables in Java:

* local variable
* instance variable
* static variable

a) Local Variable: A variable declared inside the body of the method is called local variable. You can use this variable only within that method and the other methods in the class aren't even aware that the variable exists. A local variable cannot be defined with "static" keyword.

b) Instance Variable: A variable declared inside the class but outside the body of the method, is called an instance variable. It is not declared as static. It is called an instance variable because its value is instance-specific and is not shared among instances.

c) Static Variable: A variable that is declared as static is called a static variable. It cannot be local. You can create a single copy of the static variable and share it among all the instances of the class. Memory allocation for static variables happens only once when the class is loaded in the memory.

8. Data types are divided into two groups:

a) Primitive data types includes byte, short, int, long, float, double, Boolean, char.

b) Non-Primitive data types includes strings, arrays and classes.

9. Identifiers in Java are symbolic names used for identification. They can be a class name, variable name, method name, package name, constant name, and more. However, In Java, There are some reserved words that cannot be used as an identifier.

10. Classloader − Loads the class file into the JVM.

Class Area − Storage areas for a class elements structure like fields, method data, code of method etc.

Heap − Runtime storage allocation for objects.

Stack − Storage for local variables and partial results. A stack contains frames and allocates one for each thread. Once a thread gets completed, this frame also gets destroyed. It also plays roles in method invocation and returns.

PC Registers − Program Counter Registers contains the address of an instruction that JVM is currently executing.

Execution Engine − It has a virtual processor, interpreter to interpret bytecode instructions one by one and a JIT, just in time compiler.

Native method stack − It contains all the native methods used by the application.