Assignment 1

Q1.What is the difference between Compiler and Interpreter ?

Ans)

Compiler	Interpreter
1.It convert high level language(HLL) like java,python,c++ to low level language(LLL) like machine language once	2.It convert high level language(HLL) like javascript to low level language(LLL) like machine language in multiple stages
2.Scan the entire code and convert to machine code	2.Scan one statement(one line) and translate to machine code
3.Less memory efficient because object code is generate	3.Memory efficient because no object code is generate
4.Faster as compare to interpreter	4.Slower than compiler

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

Ans)JVM

- JVM stands for java virtual machine
- It is an abstract machine(virtual machine)

- It provides a runtime environment in which java bytecode can be executed.
- It also run other programs which are written in other language and compiled to java bytecode
- Main function
    - ❖ Load code
    - ❖ Varifies code
    - ❖ Executes code
    - ❖ Runtime environment provides

JRE (Java Runtime Environment):
The JRE is an environment that allows you to run Java applications. It includes the Java Virtual Machine (JVM) and the core class libraries required for executing Java programs. The JRE is what end-users need to have installed on their systems in order to run Java applications, but it doesn't include the development tools like the compiler.

JVM (Java Virtual Machine):
The JVM is a virtual machine that executes Java bytecode. It is an integral part of both the JDK and JRE. When you compile a Java source code file, it gets converted into bytecode, which is a platform-independent representation of the program. The JVM is responsible for interpreting or compiling the bytecode and executing the program. It provides features such as memory management, garbage collection, and security.

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

ans)

The JVM (Java Virtual Machine) allocates memory areas for different purposes during the execution of a Java program. Here are the main memory areas managed by the JVM:

1. Heap:
The heap is the runtime data area where objects are allocated. It is a shared memory region used by all threads in a Java application.
2. Stack:
Each thread in a Java program has its own stack. The stack is used to store local variables, method calls, and partial results. Each time a method is called, a new frame is pushed onto the stack, and it is popped off when the method returns.
3. Method Area:
The method area, also known as the "permanent generation" or "metaspace" (since Java 8), is where the JVM stores class structures, method code, and runtime constant pool. It contains information about classes and methods used by the program.

4. Program Counter (PC) Register:
Each thread in a Java program has a program counter that keeps track of the current executing instruction. It points to the next instruction to be executed.

5. Native Method Stacks:
The native method stacks are used to store native method information and data. Native methods are methods implemented in languages other than Java,

such as C or C++, and they are typically used to interact with the underlying operating system.

Q4.What is JIT compiler?

Ans)JIT (Just-In-Time) compiler is a component of the Java Virtual Machine (JVM) that dynamically compiles Java bytecode into native machine code at runtime. It optimises the performance of Java applications by analysing and selectively compiling parts of the bytecode that are frequently executed or deemed performance-critical. The JIT compiler takes advantage of runtime information to make informed decisions about code optimization, such as inlining methods, eliminating redundant computations, and performing loop unrolling. By converting bytecode into efficient native machine code, the JIT compiler significantly improves the execution speed of Java programs compared to interpreting bytecode line by line, resulting in better performance for the applications running on the JVM.

Q5.What are the various access specifiers in Java?

ans) In Java, access specifiers are keywords that define the accessibility or visibility of classes, methods, variables, and constructors within a program. There are four access specifiers in Java:

1. Public:
The public access specifier allows unrestricted access from anywhere in the program. Classes,

methods, variables, and constructors marked as public can be accessed by any other class or package.

2. Private:
The private access specifier restricts access to only within the same class. Private members cannot be accessed or modified by other classes or packages. They are typically used to encapsulate implementation details and provide data hiding.

3. Protected:
The protected access specifier allows access within the same class, subclasses (inheritance), and within the same package. Protected members are not accessible by unrelated classes in different packages.

4. Default (no specifier):
If no access specifier is specified, it is referred to as the default access specifier (also known as package-private). Default access allows access within the same package but restricts access from classes in different packages.

Q6.What is a compiler in Java?

ans)A compiler in Java is a software tool that translates human-readable Java source code into machine-readable bytecode. It performs the task of converting the high-level programming language code written by developers into a lower-level representation that can be executed by the Java Virtual Machine (JVM). The compiler checks the syntax, semantics, and structure of the code,

identifies errors or issues, and generates the corresponding bytecode instructions. The bytecode produced by the compiler is a platform-independent representation of the original code, allowing it to run on any system with a compatible JVM, providing the "write once, run anywhere" characteristic of Java.

Q7.Explain the types of variables in Java?

ans)In Java, there are three types of variables based on their scope and usage:

1. Local Variables:
Local variables are declared within a method, constructor, or block of code and are accessible only within that specific scope. They are created when the block is entered and cease to exist once the block is exited. Local variables must be initialised before they are used and cannot have default values.

2. Instance Variables:
Instance variables, also known as member variables or fields, are declared within a class but outside of any method, constructor, or block. Each instance of the class (object) has its own copy of instance variables. Instance variables have default values if not explicitly initialised and can be accessed and modified by methods of the class.

3. Class/Static Variables:
Class variables, also called static variables, are declared with the `static` keyword within a class but outside of any method, constructor, or block. Unlike

instance variables, class variables are shared among all instances (objects) of a class. They are stored in the memory for the entire duration of the program and can be accessed using the class name.

Q8.What are the Datatypes in Java?

ans)Java provides several built-in datatypes to represent different types of data. The datatypes in Java can be categorized into two categories: primitive datatypes and reference datatypes.

Primitive Datatypes:
1. boolean: Represents a boolean value (true or false).
2. byte: Represents a signed 8-bit integer.
3. short: Represents a signed 16-bit integer.
4. int: Represents a signed 32-bit integer.
5. long: Represents a signed 64-bit integer.
6. float: Represents a single-precision 32-bit floating-point number.
7. double: Represents a double-precision 64-bit floating-point number.
8. char: Represents a single Unicode character.

Reference Datatypes:
1. classes: Represents objects created from user-defined classes.
2. arrays: Represents a collection of elements of the same type.
3. interfaces: Represents interfaces, which define a contract for classes to implement.
4. enums: Represents enumerated types, which are special classes used to define a set of constants.

Q9.What are the identifiers in java?

ans)In Java, identifiers are names used to identify variables, methods, classes, packages, and other program elements. They are user-defined names given to entities in the code to make them recognizable and distinguishable.

Q10.Explain the architecture of JVM

ans)The Java Virtual Machine (JVM) is the runtime environment that executes Java bytecode. It provides a layer of abstraction between Java programs and the underlying hardware and operating system. The architecture of the JVM consists of several components:

1. Class Loader:
The Class Loader subsystem is responsible for loading Java class files into the JVM.

2. Runtime Data Areas:
The JVM manages different runtime data areas for storing data during program execution:

3. Execution Engine:
The Execution Engine interprets and executes Java bytecode. It consists of two components:

- Just-In-Time (JIT) Compiler: It dynamically compiles frequently executed bytecode into machine code for improved performance.
- Interpreter: It interprets bytecode line by line and executes the corresponding operations.

4. Garbage Collector:
The Garbage Collector is responsible for automatic memory management.