**Assignment Questions 1**

**Q1. What is the difference between Compiler and Interpreter?**

**Compiler**

1. It is a software which takes sourcecode(HLL) as the input and generates MLL code as the ouput.
2. To convert the HLL code to MLL code compiler will scan the HLL code only once.
3. Compiler will speed up the process ,where as intrpretter will slow down the process.

**Interpreter**

1. It is a software which takes sourcecode(HLL) as the input and generates MLL code as the ouput.
2. To convert the HLL code to MLL code interpreter will scan the HLL code mulitple times(depends on the instrcutions).
3. Compiler in one Scan will identify all the problems in the code(if found),where as interpretter will do scanning line by line so it takes more time for identifying the problem.

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

**JDK:** It stands for Java Development Kit, which includes the tools needed to develop Java applications.

==> JDK :: JRE + JVM

**JRE:** It stands for Java Runtime Environment, which provides the necessary libraries and components to run Java applications.

**JVM:** It stands for Java Virtual Machine, which is responsible for interpreting Java bytecode and executing Java programs.

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

The JVM (Java Virtual Machine) allocates memory into several different areas to efficiently manage the execution of Java programs. The major memory areas allocated by the JVM are as follows:

1. **Heap:** For storing objects.
2. **Method Area:** For storing class-level information.
3. **JVM Stacks:** For storing method calls and local variables.
4. **PC Registers:** For keeping track of the current execution point.
5. **Native Method Stacks:** For executing native methods.

**Q4. What is JIT compiler?**

JIT (Just-In-Time) compiler is a part of the Java Virtual Machine (JVM) that compiles bytecode into native machine code at runtime, which improves the performance of the application. It works by analyzing the frequently executed bytecode and compiling it into native code to make it faster.

**Q5. What are the various access specifiers in Java?**

In Java, there are four access specifiers that control the visibility and accessibility of classes, methods, variables, and constructors. These access specifiers determine which parts of a Java program can access or modify certain members. The access specifiers are as follows:

**1. Public:** The "public" access specifier allows unrestricted access to a class, method, variable, or constructor. It can be accessed from any other class or package.

**2. Private:** The "private" access specifier restricts access to the members within the same class. Private members cannot be accessed from outside the class, including subclasses.

**3. Protected:** The "protected" access specifier allows access within the same class, subclasses, and other classes in the same package. It provides more accessibility than the "private" specifier but less accessibility than "public."

**4. Default (no specifier):** If no access specifier is specified, it is considered the default access specifier. It allows access within the same package but restricts access from outside the package.

These access specifiers provide control over the visibility and encapsulation of Java classes and members, enabling proper encapsulation and maintaining code security and integrity.

**Q6. What is a compiler in Java?**

In Java, a compiler is a software tool that converts human-readable Java code (HLL) into machine-readable bytecode (.class file), which can be executed by the Java Virtual Machine (JVM). It checks for syntax errors, verifies code logic, and generates optimized bytecode for execution.

**Q7. Explain the types of variables in Java?**

In Java, there are three types of variables:

**1. Local Variables:** Declared within a method or block of code. They have limited scope and exist only within the block where they are declared.

**2. Instance Variables:** Declared within a class but outside of any method. Each instance of the class has its own copy of instance variables.

**3. Static Variables:** Associated with the class itself rather than with individual instances. They are shared among all instances of the class.

****Q8. What are the Datatypes in Java?****

**In Java, there are two categories of data types: primitive data types and reference data types.**

****1. Primitive Data Types:** These are the basic data types provided by Java. They represent simple values and have a fixed size. Java has eight primitive data types:**

****boolean:** Represents either true or false.**

****byte:** Represents a signed 8-bit integer.**

****short:** Represents a signed 16-bit integer.**

****int:** Represents a signed 32-bit integer.**

****long:** Represents a signed 64-bit integer.**

****float:** Represents a single-precision 32-bit floating-point number.**

****double:** Represents a double-precision 64-bit floating-point number.**

****char:** Represents a single Unicode character.**

****2. Reference Data Types:** These data types are based on class and object references. They don't store the actual data, but rather refer to objects or memory locations. Reference data types include:**

****Classes:** User-defined types created using class definitions.**

****Arrays:** Objects that hold a fixed number of elements of the same type.**

****Interfaces:** Defines a set of methods that a class can implement.**

****Enums:** A special type used to define a set of constants.**

**Programmers can also create their own custom data types using classes and interfaces.**

**These data types provide the foundation for storing and manipulating different types of data in Java programs.**

****Q9. What are the identifiers in java?****

**It is a name in java program.**

**It can be a classname,methodname,variable name and label name.**

*****7 Rules(syntax for compiler + jvm) for writing an identifier:*****

****Rule1:** The only allowed characters in java identifiers are a to z, A to Z,0 to 9, \_(underscore),$**

****Rule2:** If we use any other characters it would result in compile time error**

**int ^\* = 10;(invalid)**

**int total = 10;(valid)**

**int total#= 35;(invalid)**

****Rule3:** Identifiers are not allowed to start with digits**

**int telusko1 =100;(valid)**

**int 1telusko = 100;(invalid)**

****Rule4:** Java identifiers are case sensitive,meaning number and Number is different.**

**class Demo{**

**int number=10;**

**int Number=20;**

**int nUmber= 30;**

**int NUMBER = 40;**

**}**

****Rule5:** There is no lenght limit on java identifiers, but still it is a good practise to keep the length of the identifier not more than 15characters.**

**int priorityOfThreadWithMinValue = 1;**

****Rule6:** We can't use reserve words as a identifers.**

**eg: int if = 10; //CE**

****Rule7:** Predefined class names can be used as identifiers like String,Runnable**

**eg#1**

**String Runnable = "sachin";**

**System.out.println(Runnable);//sachin**

**eg#2**

**int String = 10;**

**System.out.println(String);//10**

**Note: Even though predefined class names can be used as an indetifiers,it is not a good practise to keep.**

****Q10. Explain the architecture of JVM****

**The JVM architecture consists of the following components:**

****1.Class Loader:** Loads Java classes into the JVM.**

****2. Runtime Data Areas:****

**- Method Area: Stores class-level information.**

**- Heap: Allocates memory for objects.**

**- JVM Stacks: Stores method calls and local variables.**

**- PC Registers: Keeps track of the current execution point.**

**- Native Method Stacks: Stores native method information.**

****3. Execution Engine:****

**- Interpreter: Executes bytecode instructions.**

**- Just-In-Time (JIT) Compiler: Compiles bytecode into machine code for performance.**

**- Garbage Collector: Manages memory by reclaiming unused objects.**

****4. Native Method Interface (JNI):** Allows Java code to interact with native libraries.**

**The JVM architecture provides an environment for executing Java programs and ensures platform independence.**