Assignment No:- 3
Interview Questions

1) explain the component of the JDK
⇒ JDK (Java Development Kit) is a important components that provides the environment to develop and execute java program.

JDK components
1) JRE (Java runtime environment)
2) JVM (Java virtual machine)
3) javac (Java compiler)
4) ~~jaab~~ Jar (Java archiver)
5) jab (Java degugger)
6) javadoc (Java Document generator)

2) Differentiate bet^n JDK, JVM & JRE
⇒ JDK (Java development Kit) is a tool for development, testing & running java application.

JVM (Java virtual machine) is a runtime environment that execute java byte code on any platform it is most important part of java platform.

JRE (Java Runtime environment) is a subset of JDK. it provides libraries, JVM and Other components necessary to run java program.

Q3) Role of JVM

⇒ . JVM has important platform independent environment for executing java.
. it manages memory allocation and garbage collection for java programs.

How JVM execute java code.
compilation : (.java files) it compiles into an intermediate
loading : load the byte code into memory
verification : it verify the byte code.
execution : jvm execute the bytecode interpretly.
Runtime environment : it provides Run time environment to the java code.

Q4) Memory management system in JVM.

⇒ # responsible for allocating, managing and deallocating memory for java program.

Q5) JIT (Just-in-time) compiler.
it a component of the jvm that dynamically compiles java byte code into native machine code at runtime.

Q6) Role of JIT
It compiles frequently executed byte code into native code, reducing interpretation overhead.
Optimized native code for better performance..

Bytecode
- an intermediate from of java code that platform independent and can be executed by the JVM.
- it generate by javac compiler from java source code (.java file)
- stored in .class files.

Bytecode importance.
it allows java code to be platform independent making it "write once run anywhere".

Q6) architecture of JVM.
three main components
1) class loader subsystem
2) Runtime data areas.
3) execute engine.

class loader subsystem.
it loads, links and initialize java classes.

Runtime data areas: provides memory area for storing and managing data native memory).

execute engine: execute java bytecode interpreter JIT compiler and

Q7)

⇒ Java achieves platform independence through the JVM

- by compiling Java code into platform agnostic bytecode (.class files).
- using the JVM to interpret and execute the bytecode on any platform that has a JVM without the needs for recompilation.

Q8)

Significance of class loader

- it loads Java classes into memory
- verifies the correctness of the loaded classes
- ensure that a class is loaded only once.
- provides a mechanism for custom class loading and dynamic loading for classes.

process of garbage collection.

It involves

1) marking : it identifying objects that are no longer referenced by the program.
2) sweeping : reclaiming memory occupied by the marked objects.
3) compacting : reorganizing memory to reduce fragmentation.

Q 9) access modifiers in java.

i) public: accessible from anywhere, both within and outside the class, package or project.

ii) private: accessible only within the same class, not from outside the class.

iii) protected: accessible within the same class, not from outside the class.

iv) Default (no modifier) - accessible within the same class and package but not from outside of the package.

Q 10) Refer que (9) answer.

Q 11) yes, we can override a method with a different access modifier in a subclass but with some restrictions.

you can override a method with a more permissive access modifier (eg. from protected to public) but not with a more restrictive access modifier (eg. from protected to private).

Q 12) The main difference between protected and default (Package - private) access is that protected allows access from subclasses in other packages while default (package private) access does not.

Q13) Yes, it is possible to make a class private in java but only as a nested class (i.e class within another class) a top-level class cannot declared private.

Q14) No, a top level class in java cannot be declared as protected or private this is because access modifiers apply to members of a class not to the class itself. top-level classes can only declared as public or default (Package - private).

Q15) yes, will not able to access the private variables or methods from another class where the same package. private members can only accessible within the same class, not from other classes even if they are in the same package.

Q 16) When a class member (variable or function) is declared without any access modifier i.e. public, private, protected) it is said to have "package-private" or default access. This means the member is accessible within the same package but not the ~~public~~ member but classes from other packages cannot.