Interview Questions.

Assignment - 3

~~* Notes question.~~

1. Components of JDK.

JDK is comprehensive toolkit used for developing java apps
- Java compiler (javac) - converts java source code into byte code.
- Java virtual machine (JVM) - executes java ~~bytecode~~ and manages system sources
- Java runtime environment (JRE) - provides libraries, JVM and other components necessary for running Java applications
- Java API libraries - A set of standard class libraries for Java including utility classes, data structures, networking
- Development tools - utilities like javadoc for generating documentation, jar for packaging applications and javap for disassembling byte code

2. Differentiate between JDK, JVM, JRE

- JDK - a complete software development kit that includes the JRE, compiler and other tools for developing java applications
- JRE (Java runtime environment) - Provides libraries and the JVM necessary to run Java applications. It does not include development tools like compiler
- JVM (Java Virtual Machine) - The engine that executes Java byte code. It provides runtime environment and is part of JRE

3. Role of JVM and execution of java code.
- Role of JVM - The JVM is responsible for executing java bytecode. It provides platform independence by abstracting the underlying OS and hardware.
- Execution of java code.
- compilation - java source code (.java files) is compiled into bytecode (.class) by java compiler.
- class loading - The JVM loads the bytecode into memory using class loader.
- Bytecode verification - The JVM verifies the bytecode to ensure it adheres to Java's safety and security constraints.
- Execution - The bytecode is executed by JVM. This can involve interpretation (executing bytecode directly) or just in time (JIT) compilation (converting bytecode into native machine code).

4. Memory management System of the JVM.
- JVM manages memory through several key components.
- Heap - Stores objects and class instances. It is divided into the young Generation (where new objects are allocated) and the old Generations (where long-lived objects are eventually moved)
- Stack - Stores method call frames, including local variables, method parameters and return addresses
- Method area - Contains class level information such as methods and field data along with the runtime constant pool
- Garbage Collection - Automatically reclaims memory occupied by object that are no longer in use

5. JIT compiler and Bytecode

- JIT compiler - The just in time compiler improves performance by compiling bytecode into native machine code at runtime, which speeds up execution by reducing the need for repeated interpretation.
- Bytecode - Bytecode is the intermediate representation of Java code that the JVM understand. It is important because it allows Java to be platform-independent - bytecode can be executed on any JVM regardless of the underlying hardware & OS.

6. Architecture of the JVM -

- class loader Subsystem - loads, links and initializes classes and interfaces.
- Runtime data areas - includes the heap, stack, method area and PC registers.
- Execution engine - Executes byte code either via interpreter or JIT compilation.
- Native Interface - provides access to native lib and API.
- Garbage collector - Manages memory and performs GC to reclaim unused memory.

7. Platform Independence through JVM.

Java achieves platform independence through JVM by
- Compiling Java code into bytecode - Byte code is a platform independent intermediate representation.
- Executing byte code on the JVM - Diff. JVM implementations are provided for diff platforms (Win, Linux, macOS) JVM translates bytecode into machine code appropriate for host machine.

Jar - package file format use to aggregate java class files.
rt jar - rt-runtime has all compiled class files for core java. like java.lang
JIT - Just in time - helps improve performance of java programs by compiling byte code to native machine code at run time.

8. Significance of class loader and Garbage collection.
- class loader - The class loader dynamically loads classes into the JVM at runtime. It handles loading, linking and initializes classes.
- Garbage collection - process of automatically identifying and reclaiming memory occupied by objects that are no longer needed by application.

9. four access modifiers.

1. Public - Accessible from any other class
2. Protected - access by within its own package & sub classes.
3. Default (Package level private) - Accessible within its own package.
4. Private - within its own class

10. Difference - Public vs Protected vs Default.

Access modifiers	within class	within package	outside package by subclass only	outside package
Private	Y	N	N	N
Default	Y	Y	N	N
Protected	Y	Y	Y	N
Public	Y	Y	Y	Y

~~overriding methods with different access modifiers~~
- Yes we can override a method that has different access modifier but cannot lower its access scope

11. → The access specifier for an method can allow more, but not less access than overridden method. i.e if superclass method is protected the subclass overridden method can have protected or public. so it means subclass overridden method cannot have weaker access specifier

12. Difference between protected & default (package-private) access

- Default - The access level of default modifier is only within package. cannot be accessed from outside package. If not specified, it is default.
- protected - The access level of a protected modifier is within the package and outside the package through child class.

13 We can declare java class as private. but that class wont be accessible from its outer scope. main class cannot be private

14. No, we cannot declare a top class as private or protected It can be public or default.

15) what happens if you declare a variable or method as pri. and try to access it from another class within same package
→ we can do it if we want to with help of Reflection API. which are present in java.lang.reflect.package

16) Explain concept of private package or default access. How does it affect visibility of class membe
→ If class member doesnt have any access modifier specified it acts as default. The access is more restricted than public and protected but less restricted than private.
private < default < protected < public.

JIT - Just in time Singletasking & multitasking? final class
GC - Garbage collection, Remote method invocation ?
multiclass inheritance, JVM threads
memory leakage ? jcmd & jstack & jinfo for thread

* Byte code can be run in any arch, x64, CPU's
* Java doesn't support sizeof.
Imp → * How will u find size of variable in java?
* In Java all methods are virtual
Imp → * Java is truely dynamic.
* process is container for thread

* Java modifier - Keyword which is used to change the behaviour of variable, field, method, class
modifier which is used to control visibility of the members of the class / enum / interface
- 4 access modifiers in Java
  1. private.
  2. Package level private
  3. protected
  4. public

* Entry point method.

- main method is considered as entry point

* Print, println, Printf.

Printf("%-30s%-15d%-10.2f\n", "SRM", 100036, 125.005f);
↓
return type - PrintStream

return type of print, println - void.