Shraddha Sunil Jadhav.
Kharghar.

Assignment 3.

Interview Questions

1. Explain the components of jdk.

Ans. There are several components of jdk such as :- i) JRE
ii) Java interpreter / loader
iii) java compiler (javac).
iv) Java archive (Jar)
v) Java debugger (jdb)
vi) Applet viewer
vii) Libraries.
viii) Javadoc.

i) JRE :- This component allows you to run your program by acting as abstract class.
- It includes JVM, core classes & supporting files.

ii) Java interpreters / loader :- This component read & executes the program.

iii) Jar :- This component is similar to zip file.

iv) Java debugger :- This component is allows to debug your java program in command line interface.

v) Libraries :- It includes all the libraries needed to run programs.

2. Differentiate between JDK, JVM & JRE

Ans. • JDK :- JDK is the software development kit that contains JRE & tools for developing java applications.
- JDK is used for development & it includes java debugger & javaDa.

JVM :- JVM is the foundation of java programming language.
- Basically ~~java~~ running java programs are depend on JVM.
- JVM included in JDK & JRE.
- JVM is platform independent.

JRE :- JRE provides the minimum requirements of running Java programs.
- JRE needed libraries for execution.

8. What is the role of JVM in java & How does JVM execute java code?

Ans:- All language compilers translate source code into machine code for a specific comp.
- Java compiler produces an intermediate code known as bytecode for machine that does not exist.
- So that machine is called Java virtual machine & exists only inside comp. memory.

• Steps for execute java code :-
- When we write a java code it's basically high level lang. So JVM doesn't understand this lang.
- So, the java compiler compiles the source code into bytecode.
- Then JVM uses a component which is ClassLoader to load compiled .class file into memory.
- The classloader required classes dynamically a runtime & organizes them in a hirarchical structure.

- After that bytecode will verified. that ensure the security rules & doesn't perform illegal operations. This process protects the JVM from running unsafe bytecode.
- JVM is responsible for executing bytecode is the execution engine.
- JVM automatically manages memory using a process called garbage collectn.
- Once the program finishes execn, the JVM terminates, performing any necessary cleanup.

4. Explain the memory management of JVM.

Ans. - In JVM memory there are method area, heap, JVM lang stacks, PC registers & native method stacks are included.
- In method area, all class level infor" like class name, parent class name, methods & variable info. are stored, including static variable.
- In heap area, informatn of all objects is stored in heap area.
- Stack area, create a run time stack.
- PC registers can store address of currer execution instructn of a thread.
- In native method stack, for every thread, a speperate native stack is created. It stores native info.

5. What are the JIT compiler & its role in JVM? What is the bytecode & why is it important?

Ans. - JIT is the ~~java~~ just in time which is part of ~~compiler~~ JVM, it compiles byte code into execut^n code in real time, on a piece-by-piece demand basis.
- It is important to understand that it is not possible to compile an entire java program into executable code all at once.
- Because java performs various run time checks that can be done only at run time.
- Instead, the JIT compiles code as it is needed during execution.

6. Describe the architecture of JVM.
- The JVM architecture consists of several key components, including the class loader, runtime data areas, execution engine & native method interface (JNI).
- This components work together to execute java bytecode & manage runtime operations.

7. How does java achieve platform independent through the JVM.

Ans. - The JVM achieve platform independen by interpreting java bytecode, which i generated by compiling java source cod

- Since bytecode is platform neutral, the same bytecode can run on any system with a compatible JVM implemtatn regardless of the underlying h/w or OS.

8. What is significance of the class loader in Java & ~~how do they~~ What is the process of garbage collection ?

Ans. - A classloader in java is a subsystem of java virtual machine & dedicated to loading ~~time~~ class files when a program is executed.

- Class loader is first to load the executable file.

• Garbage collection

- The JVM garbage collection, collector automatically identifies & removes unreferenced objects, preventing memory leaks & optimizing memory usage.

9. What are the 4 access modifier in java & how do they differ from each other?

Ans. 1. public : Allows access to elements from any other class in the appl^n, regardless of the package

2. private : Restricts access to elements to only within the class they are declared in.

3. protected: Allows access within the same package or in subclasses, which might be in different packages.

4. default: Limits the visibility to access within the same package. If no access modifier is specified, the default automatically apply.

10. What is the difference bet? public, protected & default access modifier.

Ans. • Public: public members are visible anywhere. So, we can access it anywhere within or outside the package.

• protected :- protected members of a class are visible within the package. - Therefore, we can only access within the package but can be accessed to the subclasses outside the package through the inheritance only.

• default :- Default members of a class are accessible within the same package due to visible only within the package. - They can't be accessed from outside the package.

11. Can you override a method with a different access modifier in subclass? For example, can a protected method in a superclass be overridden with a private method in a subclass? Explain.

Ans. - Yes protected method in a superclass be overridden. - If the superclass method is protected, the subclass overridden method can have protected or public (but no default) which means the subclass overridden method can't have a weaker to access specifier.

12. What is the difference betⁿ protected & default (package private) access?

Ans. • protected : accessible by the classes of the same package & the subclasses residing in any package --
• Default : accessible by the classes of the same package..

13. Is it possible to make a class private in java? If yes, where can it be done, and what are the limitations.

Ans. Making a class private does not make any sense as we can not access the code of its class from the outside.

14. Can we a top level class in java be declared as protected or private? why or why not.

Ans. Top level class can't be private because private & protected classes are allowed but only as inner or nested classes.

15. What happens if you declare a variable or method as private in a class & try to access it from another class within the same package.

Ans. At this condition, any other class of the same package will not be able to access these members.

16. Explain the concept of package-private or default access. How does it affect the visibility of class member?

Ans:
- Private restricts access to the elements only within the class they are declared.
- Protected allows access within the same package or in subclasses, which might be in different packages.
- Last the default access limits the visibility to classes with the same package.