24084052000

Assignment No: 03

Q.1) Explain the components of JDK.

→ a) JDK (Java Development kit) includes:

⇒ JRE (Java Runtime Environment)
↳ Tools to run Java programs

⇒ Java compiler (javac)
↳ Converts Java code into bytecode.

⇒ Tools & Libraries
↳ Debuggers, documentation tools, etc.

⇒ key components:
↳ javac, java, javadoc, jar & JRE:

Q.2) Differentiate between JDK, JVM & JRE

→ a) JDK (Java Development-kit)
- Development environment for building Java applications (JVM + JRE + development tools).

b) JRE (Java Runtime Environment)
- Provides libraries & resources needed to run Java applications (no development-tools)

c) JVM (Java Virtual Machine)
- Executes Java bytecode (part of JRE)

Q.3) What is the role of JVM in Java? How does the JVM execute Java code?

→ 1) Java is responsible for executing Java bytecode on any machine (translates Java bytecode into machine code for execution)
2) It handles execution, memory management
* 3) Considering all the points, it proves Java platform-independent.

• Execution flow:

a) Java Source Code (.java)
↳ Compiled to bytecode (.class) by javac.

b) JVM reads & interprets the bytecode for the host machine.

Q.4) Explain the memory management system of JVM.

→ *) JVM divides memory into:
1) Heap: where objects are stored.
2) Stack: where method calls & local variables are stored.
3) Method area: stores class data.
4) Garbage Collector
↳ Frees up memory by removing objects no longer in use.

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

→ 1) JIT (Just-In-Time compiler):
↳ Converts bytecode to native machine code during runtime to improve performance.
2) Bytcode
↳ An intermediate code generated after compilation, (platform-independence)

Q.6) Describe the architecture of JVM?

→ a) class loader: loads class files into memory
b) Execution Engine: Executes bytecode (JIT compiler & interpreter)
c) Garbage collector (Frees unused memory)

Q.7) How does Java achieve platform independence through the JVM?

→ Platform Independence is achieved through JVM, which interprets bytecode in a way that the same Java program can run on any platform w/o modification.

Q.8) a) What is the significance of the class loader in Java?
b) What is the process of garbage collection in Java?

→ A] Class loader

→ 1) The class loader is responsible for dynamically loading Java classes into JVM at runtime.
2) W/o it, JVM cannot find & load classes neccessary for program execution.
3) Types of class loaders:
a) Bootstrap class loader: (loads core Java libs rt.jar
b) Extension class loader
(loads classes from ext directories (ext folder)
c) Application class loader:
(loads classes from the class path specified by user).

4) Class loading process:
a) Loading: Locates & loads the bytecode of a class
b) Linking: Verifies bytecode, prepares class variables
c) Initialization: Executes static initializers & static blocks

B] Garbage collector in Java
1) Java uses automatic garbage collection to manage memory.
2) The GC frees memory by removing objects that are no longer referenced ensuring efficient memory

3) Process of garbage collection:
a) Marking: Identifies objects that are still in use & no longer needed
b) Sweeping: Removes unmarked objects that are.
c) Compacting: Reorganizes the heap to eliminate fragmentation & improve memory allocation

Q.9) What are the four access modifiers in Java, how do they differs from each other?
→ • Private: Accessible only within the class.
• Default (Package private): Accessible within package.
• Protected: Accessible within same 'p' & subclasses
• Public: Accessible from anywhere.

Q.10) Difference between public, protected & default access modifiers?
→ • public: accessible from anywhere
• protected: accessible by the classes of same package & subclasses.
• Default (no modifier specified)
↳ accessible by classes of same package

Q.11) a) Can you override a method with a different access modifiers in a subclass?
→ • we cannot override a method & reduce it's visibility.

Q.11) b) Can a protected method in a superclass be overridden with a private method in a subclass?
→ Protected method cannot be overridden with a private one

Q.12) What is the difference between protected & default (package-private) access?

→ Protected allows access in subclasses outside the package, while default access is restricted to the package

Q.13) Is it possible to make class private in Java? If yes where can it be done & what are the limitations?

→ A top level class cannot be private, but inner classes can be private, limiting their visibility to the outerclass.

Q.14) Can a toplevel class in Java be declared as private in class & try to access it from another class within the same package?

→ Java does not allow top level classes to be protected or private because they need to be accessible to the JVM & other classes.

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

→ If a variable / method is private, it is inaccessible outside the class, even from classes in the same package

Q.16) Explain the concept of "package private" or "default" access. How does it affect the visibility of class members?

→ • Default access allows visibility of classes/methods/fields within the same package but not from outside the package.
• This is the default if no access modifier is specified.