

PAGE NO.
 DATE //

CDAC Mumbai PG-DAC Aug 24,
Assignment no 3

Note:-

(1) - Explain the Components of JDK
→ The Java Development Kit (JDK) is a comprehensive suite of tools and resources necessary for developing, compiling, debugging & running Java applications.

(1) Java Compiler

- Function: Converts Java source (Java files) into bytecode (.class files). Bytecode is an intermediate representation of the code that can be executed by JVM.
- Purpose: Ensures that the code is syntactically correct and translates it into a form that JVM can interpret & run.

(2) Java Runtime Environment (JRE)

- Function: Provides the necessary libraries & components to run Java applications.

• Component

- Java Virtual Machine (JVM): Executes bytecode generated by the compiler. It provides independence, allowing Java programs to run on any device that has a JVM.

- Class Libraries: A collection of pre-written classes and interfaces that provide standard functionality like data structures, networking, file I/O, and more.

- Purpose: While the JRE is part of the JDK, it can also be installed separately to run Java applications without the need to compile them.

3. Java Debugger (jdb)

- Function: A command-line tool used to find & fix bugs in Java programs.

- Features: Allows developers to set breakpoints, step through code, & inspect variables & objects at runtime.

~~4. Java Debugger (jdb)~~

- ~~• Function: A command-line tool used to find & fix bugs in Java programs.~~

4. Java Archiver (jar)

- Function: packages Java class files, resources & metadata into a single compressed archive file with a jar extension.

- Purpose: facilitates the distribution & deployment of Java applications or libraries, making it easier to share & execute the code.

5. Java Documentation Generator (javadoc)

- Function: Generates HTML documents from Java Source Comments.
- purpose: Helps developers create readable & well-organized ~~doc~~ documentation for their code, which is essential for understanding & maintaining large code bases.

6. Java Command (java)

- Function: Used to run Java Applications. It launches JVM & loads necessary classes & resources to execute the app.
- purpose: It is entry point for executing Java programs, either from compiled bytecode (.class files) or directly from .jar file.

7. Additional Tools

- Java P: A class file disassembler useful for inspecting the bytecode of compiled classes.

- Javah: Generates C headers & source files from a Java. It is useful when interfacing Java with native code.

- jconsole, jvisualvm: monitoring & performance analysis tools for Java applications.

- jshell: An interactive command-line tool introduced in JDK 9 for quick testing & ~~running~~ snippets of

- Java code without writing a full program.

12) Differentiate betn JDK, JVM & JRE.

JDK	JRE	JVM.
(1) It stands for Java Development kit.	(1) It stands for Java Runtime Environment.	(1) It stands for Java Virtual Machine.
(2) It is software Development kit used to develop app in Java.	It's a Software bundle that provides Java class libraries with necessary components to run ^{Java} code.	It's an abstract machine that provides an environment to run & execute Java byte code.
(3) It contains tools for developing, debugging, & monitoring Java code.	It contains class libraries & other supporting files that JVM requires to execute the program.	Software development tools are not included in JVM.
(4) The JDK enables developers to create Java programs that can be executed & run by the JRE & JVM.	The JRE is the part of the JDK that creates the JVM.	It is the Java component platform component that executes source code.
(5) Super Set	Subset of JDK.	Subset of JRE.

(3) What is role of JVM in Java & how does JVM execute Java code?
 → The JVM's role is to execute Java byte code. It provides an environment where Java applications can run. The JVM converts byte code into machine code using an interpreter or JIT compiler for execution on host machine.

(4) Explain memory management system of JVM.

→ JVM is divided into several areas:

- (a) Heap: Stores objects & their instances.
- (b) Stack: Stores local variables & methods call frames.
- (c) Method Area: Contains class-level information like class definitions & static variables.
- (d) PC Registers: Holds the address of currently executing JVM instruction.
- (e) Native method stack: manages native method calls.
- (f) Garbage Collection (GC): Automatically reclaims memory by removing objects that no longer exist.

(5) What are the JIT Compiler & its role in JVM? What is the byte code & why it is important for Java?

→ JIT Compiler: Converts bytecode into native machine code at runtime to improve performance. It optimizes code execution after it has been interpreted multiple times.

Byte code: An intermediate code generated by the Java Compiler. It is important because it allows Java to be platform-independent, enabling the same code to run on any system with JRE.

(6) Describe architecture of JVM.

→ The JVM architecture includes:
 Class loader: Loads class files.
 Memory Area: manages different memory regions (heap, stack, method area).
 Execution Engine: Executes the bytecode using the interpreter or JIT Compiler.
 Native method Interface: Calls native methods written in other languages.
 Native method Libraries: Contains libraries for native methods.

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

→ Java achieves platform independence by compiling Java code into bytecode which is platform-agnostic. The JVM specific to each OS interprets or compiles this bytecode into native machine code, enabling Java Applet to run on any device.

(8) What is the significance of the class loader in Java? What is the process of garbage collection in Java?

→ **Class Loader**: Responsible for dynamically loading JAVA classes into ~~Java~~ JVM at runtime. It loads classes on demand & ensures that classes are loaded in a hierarchical manner (bootstrap, extension, & application, class loader).
Garbage Collection: The JVM automatically handles memory management by identifying & disposing of objects that are no longer referenced or used, freeing up memory for other objects.

(9) What are 4 access modifiers in Java & how do they differ from each other?

→ **public**: The class, method or field is accessible from any other class.

protected: Accessible within its own package & by subclassing the default: Accessible only within its own package.

private: Accessible only within the class it is defined in.

(10) What is difference betn public, protected & default access modifiers.

→ Same as Q9.

(11) Can you provide a method with a different access modifier is a subclass? For example, can a protected method in a superclass be ~~over~~ overridden with a private method in a subclass? explain

→ No we cannot override a method with a more ~~rest~~ restrictive access modifier for the instance. A protected method as it would reduce the visibility of the method in the subclass.

(12) Difference betn protected & default access?

→ Same as Q9.

(13) Is it possible to make a class private in Java? If yes, where it can be done & what are limitations?

→ A top-level class cannot be private only nested (inner) classes can be private. A private nested class is only accessible within the outer class it is defined in.

(14) Can a top-level class in Java be declared as protected or private? Why or why not?

→ No, a top-level ~~can~~ class cannot be declared as protected or private, because it would make the class

inaccessible. don't overclass
defeating the purpose of having
a public or package-private
class.

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

→ If a variable or method is declared
as private, it cannot be accessed
from another class, even if it's
within same package. Access is
restricted to the class in which it is
defined.

(16) Explain the concept of "package-
private" or "default" access.
How does it affect the visibility

→ Package-private (default) access.
When no access modifier is specified,
the default access is package-
private. Members with this access
level are accessible only within
the same package & meaning class
outside the package cannot access
them.