PAGE No.
DATE / /

Sumit Deshmukh _ KH

OOJP - Assignment No 3

Q1. Explain the components of the JDK.

→ - The Java Development Kit (JDK) is a cross platformed software development enviornment that offers a collection of tools and libraries necessary for developing Java - Based software applications and applets.

- Components of JDK are:

1) Javac

- Java compiler converts source code into Java Bytecode.

2) java
- The loader of Java apps

3) javap
- Class file dissambler

4) javadoc
- Documentation generator

5) jar
- Java Archiver helps manage JAR files.

6) appletviewer
- Debugging of Java applets without a web Browser.

7) xjc
- Accepts an XML Schema and generates Java classes.

8) apt
- Annotation - processing tool

9) jdb
- Debugger

10) jmc
- Java Mission control

11) Jconsole
- Monitoring and Management console

12) pack200
- JAR compression tool

13) extcheck
- Utility tool to detect JAR file conflicts

14) idlj
- IDL-to-java compiler

15) Keytool
- The keystore manipulating tool.

16) jstatd
- jstat daemon (experimental)

17) jstat
- JVM statistics monitoring tool

18) jshell
- jshell introduced in Java 9

19) jstack
- Prints Java stack traces (experimental)

20) jrunscript
- Java command-line script shell.

21) jhat
- Java Heap Analysis Tool (experimental)

22) jpackage
- Generate self-contained application bundles.

23) javaws
- Web Start launcher for JNLP applications.

24) javah
- C header and stub generator

25) jarsigner
- jar signing and verification tool

26) jinfo
- configuration information (experimental)

27) javafxpackager
- Package and Sign JavaFX applications.

PAGE No.
DATE / /

Q2- Differentiate between JDK, JVM and JRE.

→ - JDK (Java Development kit)

The JDK is a complete software development kit used to develop Java applications. It includes tools like compilers (Javac), debuggers and other utilities necessary for developing, compiling and packaging java code.

- JVM (Java Virtual Machine)

The JVM is an abstract computing machine that enables a computer to run Java programs. It is platform independent and provides an environment in which java Bytecode can be executed.

- JRE (Java Runtime environment)

The JRE provides the libraries, Java Virtual Machine and other components necessary to run applications written in Java. It is essentially a subset of the JDK, containing everything needed to run Java applications but not to develop them.

- The JRE does not include development tools like compilers and debuggers, which are part of JDK.

PAGE No.
DATE / /

Q3. What is the role of the JVM in Java? How does the JVM execute code?

→ Role of JVM in Java:

1) Platform independence
- The JVM allows Java applications to be platform independent. Java code is compiled into Bytecode, which can be executed on any device or operating system that has a JVM, making Java a "Write once, run anywhere" language.

2) Memory Management:
- The JVM handles memory allocation and deallocation automatically through its garbage collector. This helps manage memory more efficiently and reduces the chances of memory leaks.

3) Security:
- The JVM provides a secure environment for running Java applications. It includes a class loader and Bytecode verifier that ensure code follows Java's safety rules, such as preventing unauthorized memory access.

4) Exception Handling
- The JVM manages exceptions and errors during the execution of Java programs, providing a Robust framework for handling runtime errors.

PAGE No.
DATE / /

* How the JVM executes Java code.

1) Compilation to Bytecode
2) Class Loading
3) Bytecode Verification
4) Execution
5) Runtime environment management
6) Garbage collection

- JVM ensures that Java applications run efficiently, securely and consistently across different environments.

Q4. Explain the memory management system of JVM.

→ The Memory management system of the Java virtual Machine (JVM) is designed to efficiently manage memory allocation, usage, and garbage collection during the execution of Java applications.

Divisions of JVM:

1) Heap Memory
   - This is where all the Java objects are allocated. The heap is shared among all threads of a Java application

2) Stack Memory
   - Each thread in a Java application has its own stack memory, which stores local variables, method call information and partial results.

3) Method Area
   - This region stores class level data such as class definitions, method data, static variables and runtime

4) Native method Stack
- This stack is used for executing native method (methods written in languages other than Java, C/C++).

5) Program counter Register
- Each thread has its own PC register that holds the address of the current instruction being executed.

6) Garbage collection (GC):
- The GC is responsible for automatically detecting and reclaiming memory that is no longer in use, thus preventing memory leaks and optimizing the performance of the application.

7) Memory Management Algorithms:

8) Memory Leaks

PAGE No.
DATE / /

Q5- What are the JIT compiler and its role in the JVM? What is the Bytecode and why is it important for Java?

→ - Just-in-Time (JIT) compiler is a crucial component of Java Virtual Machine (JVM) that plays a significant role in enhancing the performance of Java applications.

- Role of JIT compiler -

1) Execution Optimization
2) Selective compilation
3) Performance enhancements
4) Adaptive Optimization

The JIT compiler is essential for improving the runtime performance of Java applications by converting bytecode into native machine code as the application runs, with a focus on optimizing frequently used code paths.

PAGE No.
DATE / /

Q6. Describe the architecture of JVM.

→ The JVM architecture consist of classloader, memory area, execution engine etc.

1) ClassLoader is a subsystem of JVM which is used to load class files.
Whenever we run the Java program, it is loaded first by the classloader.
There are three built-in classloaders in Java:
i) Bootstrap class loader
ii) Extension class loader
iii) System / Application Class Loader

2) Class (Method) Area
- Class (Method) Area stores per-class structures such as the runtime constant pool, field and method data, the code for methods.

3) Heap
- It is the runtime data area in which objects are allocated.

4) Stack
- Java stack stores frames. It holds local variables and partial results, and plays a part in method invocation and return.

5) Program counter Register
- PC (Program counter) register contains the address of the Java virtual machine instruction currently being used.

6) Native method stack and execution engine

PAGE No.
DATE / /

Q7. How does Java achieve platform independence through the JVM?

→ - Java achieves platform independence by compiling source code into Bytecode, which can be executed on any machine with a JVM.
- The JVM interprets or compiles the Bytecode into native machine code, making the program runnable on different platform without modification.

Q8. What is the significance of the class loader in Java? What is process of Garbage collection in Java.

→ Class loader - The class loader is responsible for dynamically loading classes into the JVM at runtime.

Garbage collection - It is an automatic memory management process that identifies and removes objects that are no longer in use freeing up memory and preventing memory leaks.

Q9. What are the four access modifiers in Java? How do they differ from each other?

→ Public: The class, method or variable is accessible from any other class.
Protected: Accessible within the same package and by Subclass.
Default: (package private) Accessible only within the same package.
Private: Accessible only within the class where it is defined.

PAGE No.
DATE / /

Q10. What is the difference between public, protected and default access modifier?

→ Same as Q9

Q11. Can you override a method with a different access modifier in a Subclass? For example can a protected method in a Superclass be overridden with a private method in Subclass?

→ NO. We cannot override a method with a more restrictive access modifier, for instance. A protected method cannot be overridden with a private method as it would reduce the visibility of the method in the Subclass.

Q12. What is the difference between protected and default access?

→ Same as Q9.

Q13. Is it possible to make a class private in Java? If yes, where can it be done?

→ Yes, an inner class (a class within another class) can be made private. However, top-level classes cannot be declared as Private, meaning they must have public or package-private access.

PAGE No.
DATE / /

Q14. Can a top-level class in Java be declared as protected or private? Why or Why not?

→ No, a top-level class in Java cannot be declared as protected or private. Top-level classes can only be public or package-private because they need to be accessible by JVM and other classes in the package.

Q15. What happens if you declare a variable or method as private in a class and try to access it from another class within the same package?

→ If we declare a variable or method as private it is not accessible from any other class, even within the same package. Attempting to access it from another class will result in a compilation error.

Q16. Explain the concept of 'package-private' access?

→ When no access modifier is specified, the class method or variable is accessible only within its own package. It is not accessible from classes in other packages restricting its visibility to the package level.