OOPS LAB ASSIGNMENT 09 Date – 17/10/2023

Roll No - 21BCS043

Title: Insights from JDK, JVM, and JRT

Answer to Problem

Problem 1:

2. Answered Below

3. The name of file should be the same as the name of the (main)class we want to define

Problem 2:

1. In java there is no destructor, besides it has a java garbage collector

Notes

Introduction: This short report encapsulates my recent learning journey in Java, focusing on the fundamental aspects of Java setup, the Java Runtime (JRT), Java Virtual Machine (JVM), Java Development Kit (JDK), and the usage of the 'instanceof' operator.

Java Development Kit (JDK):

- The JDK emerged as the cornerstone of Java development, providing a comprehensive toolkit for writing, compiling, and running Java applications. Its components, including the Java compiler (**javac**) and interpreter (**java**), equip developers with the tools needed for robust application development.

Java Virtual Machine (JVM):

- My exploration reaffirmed the pivotal role of the JVM, serving as a runtime environment for executing Java bytecode. It abstracts hardware specifics, ensuring the platform independence of Java applications. Understanding its function enhanced my grasp of how Java achieves "write once, run anywhere."

Java Runtime (JRT):

- The concept of JRT, introduced in Java 9, brought modularity and efficiency to the runtime environment. It encompasses the JVM along with libraries and resources, offering a streamlined approach to executing Java applications. This modular structure fosters better resource management and application scalability.

Verification and "Welcome to JMI!":

- Verifying the installation through terminal commands reassured the correctness of the setup. Writing a simple "Welcome to JMI!" program and executing it underscored the practical application of the installed JDK and validated the readiness of the development environment.

Instance of:

1. Applying the 'instanceof' operator in real-world scenarios enhanced my ability to write more flexible and dynamic code. It proved valuable when dealing with polymorphic structures and making runtime decisions based on object types.

Garbage Collector:

Java employs automatic memory management through a Garbage Collector, relieving developers from manual memory deallocation. Objects that are no longer reachable are identified and automatically reclaimed.