Program No 1

Objective:

To use Java compiler and Eclipse platform to write and execute Java programs.

Theory:

Introduction to Java

Java is a high-level, **object-oriented programming language** designed to have as few implementation dependencies as possible. It was initially developed by **James Gosling** and his team at **Sun Microsystems** in 1991 as part of the **Green Project**, which aimed to create software for embedded consumer electronics. The language was originally called **Oak**, named after an oak tree outside Gosling's office. Later, it was renamed to **Java** in 1995 when it was officially released to the public. The name "Java" was inspired by **Java coffee**, reflecting the language's simplicity, productivity, and dynamic nature. Java was later acquired by **Oracle Corporation** when it bought Sun Microsystems in 2010.

Java gained popularity due to its **platform independence**, made possible by the **Java Virtual Machine (JVM)**. When a Java program is compiled, it generates an intermediate form of code known as **bytecode** which can run on any device that has the JVM installed. This gives Java its famous slogan: **"Write Once, Run Anywhere (WORA)"**. Over the years, Java has evolved into one of the most widely used programming languages, powering everything from **Android mobile apps** and **web servers** to **enterprise-level applications** and **IoT systems**. Its features like **automatic memory management (garbage collection)**, **robust exception handling**, **multi-threading**, and **strong security** mechanisms make it a preferred choice for developers around the world.

Key Features of Java

- Object-Oriented: Everything in Java is treated as an object.
- Platform Independent: Java code is compiled into bytecode which can run on any system with a JVM.
- Robust and Secure: Java handles memory management and provides built-in security features.
- Multithreaded: Java supports multithreaded programming for performing multiple tasks simultaneously.
- Rich API and Libraries: Java provides a wide range of classes and packages.

Java Compiler (javac)

The **Java compiler**, known as **javac**, is a fundamental tool in the Java Development Kit (JDK). It plays a crucial role in the **Java development lifecycle** by converting human-readable **.java source code** into **bytecode** files with a .class extension. This **bytecode** is a platform-independent,

intermediate code that can be executed on any machine equipped with a **Java Virtual Machine (JVM)**, making Java a **platform-independent** language.

Unlike traditional compilers that generate machine-specific code, the javac compiler ensures that Java programs are **write-once, run-anywhere**. When a Java program is compiled using the command javac FileName.java, the compiler performs **syntax checking**, **error detection**, and **code translation**. If there are no errors, it creates a .class file that contains the bytecode.

The bytecode is **optimized**, **secure**, and **portable**, and the JVM interprets or just-in-time (JIT) compiles this bytecode into machine-specific instructions during runtime. This layered compilation approach not only adds flexibility and security but also allows **cross-platform compatibility** and **robust performance**, which are essential for modern-day software development. The javac tool can also be integrated into **IDEs** like **Eclipse**, **IntelliJ IDEA**, or used in **automated build tools** like **Maven** and **Gradle**.

Eclipse IDE

Eclipse is a popular, open-source Integrated Development Environment (IDE) for Java development. It simplifies the process of writing, compiling, debugging, and running Java programs.

Steps to Write and Execute a Java Program using Eclipse:

- Install Eclipse IDE from https://www.eclipse.org.
- Create a Java Project: File → New → Java Project.
- Create a Java Class inside `src`: New → Class.
- Write Java code in the editor window.
- Run the program: Right-click → Run As → Java Application.

Advantages of Using Eclipse Over Command-Line Compiler:

Feature	javac (CLI)	Eclipse IDE
Compilation	Manual via terminal	Automatic and real-time
Debugging	Not user-friendly	Powerful debugger
Error Highlighting	After compilation	Real-time syntax highlighting
Project Management	Manual folder setup	GUI-based management
Code Suggestions	Not available	Available (IntelliSense)

Conclusion:

Using the Java compiler and Eclipse IDE enhances the programming experience by providing a user-friendly environment with features like auto-completion, debugging tools, and project organization. It is a valuable skill for Java developers and helps in efficient development and testing of object-oriented programs.

Java Program:

```
package cscorner;

public class first_program {

    public static void main(String[] args) {
        System.out.println("Hello! Ashutosh Kumar Singh #Bihar");

    }

}
```

Output:

```
Hello! Ashutosh Kumar Singh #Bihar
```

Java Program Execution in Eclipse: