**Programming Lab 2**

Name: Harshvardhan Anil Bamane

Class: SY BTech CSE

PRN: 22510112

Batch: S-7

**Que 1) Write steps for installation of jdk package. Write procedure to set environment variable path.**

Installation of JDK:

For Windows:

1. Download JDK:

- Visit the official Oracle or OpenJDK website.

- Download the latest version of JDK for Windows.

2. Run Installer:

- Double-click the downloaded installer.

- Follow the on-screen instructions to install JDK.

- Choose the installation directory.

3. Verify Installation:

- Open Command Prompt.

- Type `java -version` and press Enter.

- Verify that the installed Java version is displayed.

Setting Environment Variable Path:

For Windows:

1. Find JDK Installation Path:

- Locate the folder where JDK is installed (e.g., `C:\Program Files\Java\jdk-<version>`).

2. Set JAVA\_HOME Variable:

- Right-click on "This PC" or "Computer" on the desktop or in the File Explorer.

- Select "Properties" -> "Advanced system settings" -> "Environment Variables."

- Under "System variables," click "New" and set the variable name as `JAVA\_HOME` and the value as the path to the JDK installation directory.

3. Update Path Variable:

- Find the "Path" variable in the "System variables" section.

- Click "Edit" and add `%JAVA\_HOME%\bin` at the end of the existing values.

- Click "OK" to save the changes.

4. Verify Configuration:

- Open a new Command Prompt.

- Type `java -version` and `javac -version` to verify that Java and the Java Compiler are recognized.

**Que 2) Describe JDK, JRE, JVM.**

The terms JDK (Java Development Kit), JRE (Java Runtime Environment), and JVM (Java Virtual Machine) are closely related components in the Java programming language ecosystem. Each serves a specific purpose in the development and execution of Java applications.

JDK (Java Development Kit):

The JDK is a software development kit that provides the tools, executables, and binaries necessary for Java development. It includes the following components:

1. Compiler (`javac`): Converts Java source code (`.java` files) into bytecode (`.class` files).

2. Java Runtime Environment (JRE): A complete runtime environment that includes the Java Virtual Machine (JVM) and libraries, allowing Java applications to run.

3. Java API Libraries: A set of libraries and APIs (Application Programming Interfaces) that provide pre-built functionality for common tasks, making it easier for developers to write Java programs.

4. Development Tools: Various tools for debugging, profiling, monitoring, and packaging Java applications.

Developers use the JDK to create, compile, and package Java applications. It's essential for anyone involved in Java software development.

JRE (Java Runtime Environment):

The JRE is a subset of the JDK and is used for running Java applications. It includes the following components:

1. Java Virtual Machine (JVM): The runtime engine that executes Java bytecode. It translates the platform-independent bytecode into machine code specific to the host system.

2. Java Class Libraries: A set of pre-compiled classes and packages that provide essential functionality for Java applications.

3. Java Launcher: The executable (`java` command) that starts the Java Virtual Machine and launches Java applications.

The JRE is sufficient for end-users who only need to run Java applications but are not involved in the development process. It lacks the development tools present in the JDK.

JVM (Java Virtual Machine):

The JVM is the virtualized execution environment in which Java bytecode is executed. It abstracts the underlying hardware and provides a consistent runtime environment across different platforms. Key characteristics of the JVM include:

1. Platform Independence: Java programs are compiled into bytecode, which can run on any device with a compatible JVM.

2. Memory Management: The JVM handles memory allocation and garbage collection, ensuring efficient use of memory resources.

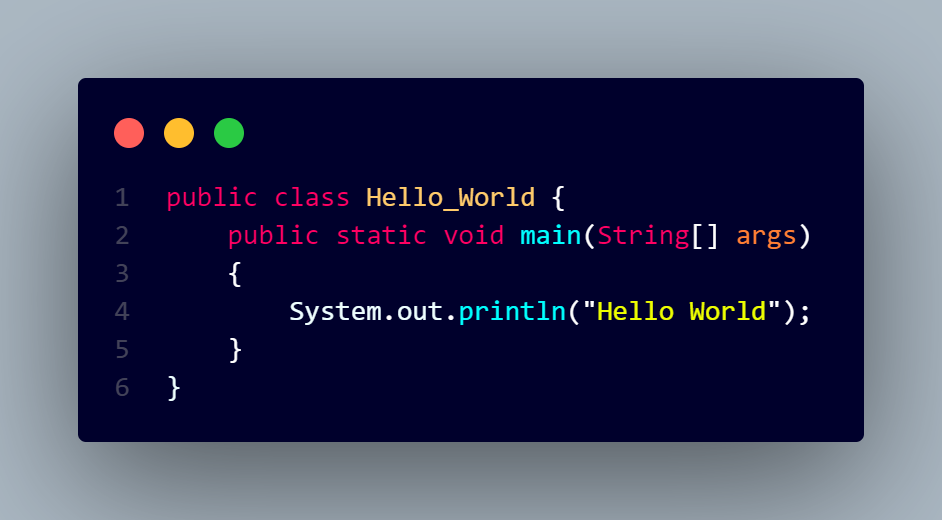
3. Security: The JVM includes a security manager that enforces access controls to prevent malicious activities.

4. Just-In-Time (JIT) Compilation: The JVM can dynamically translate bytecode into native machine code at runtime, improving execution performance.

**Que 3) Differentiate between C, C++, Java**

|  |  |  |  |
| --- | --- | --- | --- |
| **Point** | **C** | **C++** | **Java** |
| Paradigm | Procedural Programming | Extension of C with support of Object-Oriented Programming | Purely Object-Oriented Programming |
| Memory Management | Manual | Both manual and automatic | Automatic through garbage collection |
| OOP Support | Does not support | Full support | Purely Object-Oriented with strong support |
| Platform Independence | Platform Dependent | Platform Dependent | Platform Independent |
| Standard Library | Limited | Extensive STL for generic programming with pre-built data structures and algorithms | Comprehensive standard library with a wide range of packages and classes. |
| Memory Safety | No-built in Memory Safety | Provides some Memory Safety Features | Emphasize automatic memory management for enhanced Memory Safety |

**Que 4) Explain Simple Hello World Program in Java. Explain compilation and execution of Java Program.**



Explanation =>

- public class Hello\_World: This declares a class named `Hello\_World`. In Java, every executable program is defined within a class.

- public static void main(String[] args): This line declares the main method, which serves as the entry point for the Java program. The program execution begins from here.

- System.out.println("Hello, World!");: This line prints the "Hello World" message to the console. `System.out` is an object that represents the standard output (console), and `println` is a method to print a line of text.

Compilation =>

1. Source Code: We start with a Java source code file (`.java` extension) containing your program written in the Java programming language.

2. Compilation: The Java compiler (`javac`) is used to compile the source code. The compiler translates the high-level Java source code into an intermediate form called bytecode i.e. file named `Hello\_World.class` in this case.

3. Bytecode: Bytecode is a platform-independent, intermediate representation of the program. It is not machine code but a set of instructions that can be interpreted by the Java Virtual Machine (JVM).

Execution =>

1. Java Virtual Machine (JVM): The JVM is responsible for executing Java bytecode. It is a part of the Java Runtime Environment (JRE), which needs to be installed on the system.

2. Running the Java Program: The `java` command is used to run the Java program, specifying the name of the main class (the class containing the `main` method). The JVM loads the bytecode (`HelloWorld.class`) and starts executing the program.

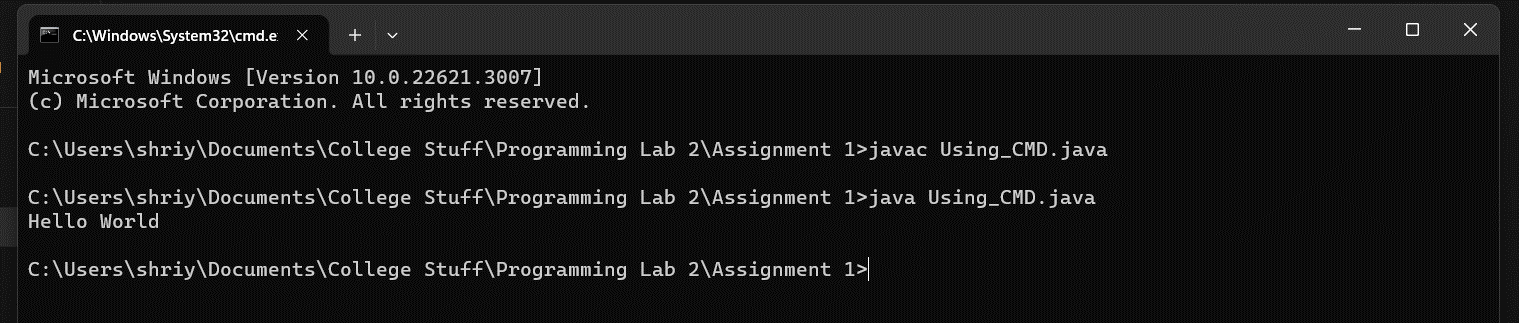
3. Class Loading: The JVM performs class loading, which involves loading the compiled classes into memory. In this case, it loads the `Hello\_World` class.

4. Execution of `main` Method: The JVM looks for the `public static void main(String[] args)` method in the loaded class (`Hello\_World`) and starts executing it. This is the entry point for the program.

5. Output: The program executes the statements within the `main` method. In our example, it prints "Hello World" to the console using `System.out.println()`.

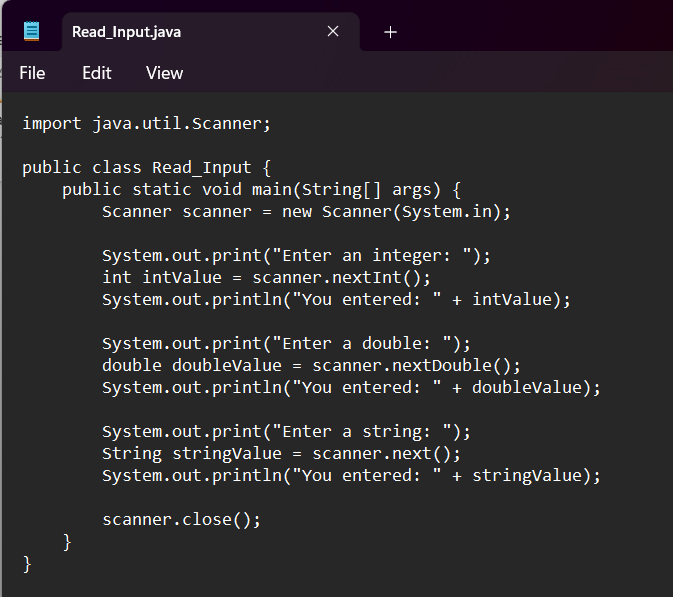
6. Termination: The program continues to execute until it reaches the end of the `main` method or encounters an explicit exit statement. Afterward, the program terminates.

Output =>

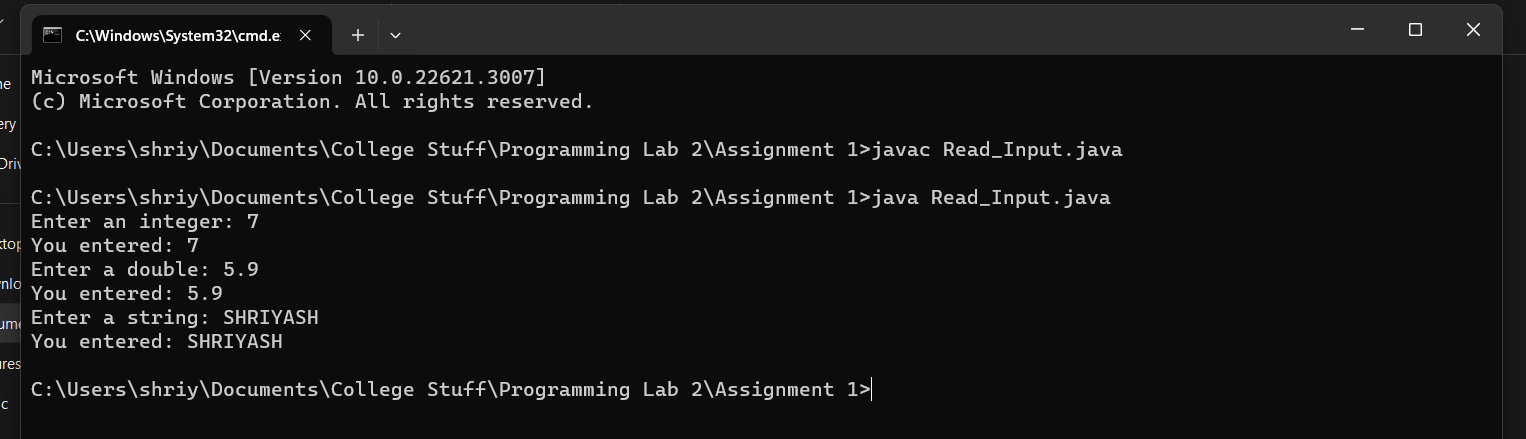


**Que 5) Write java program for reading inputs from user using Scanner class.**

**Code =>**

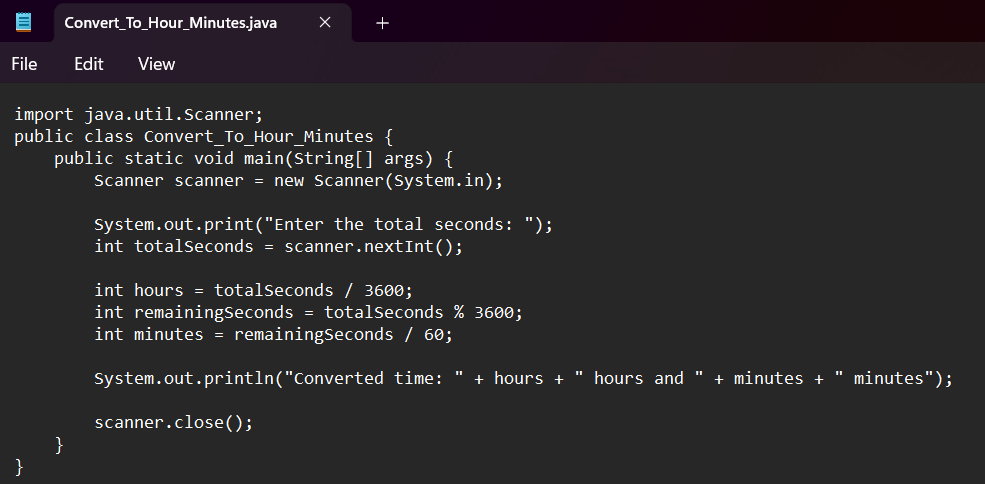


**Output =>**

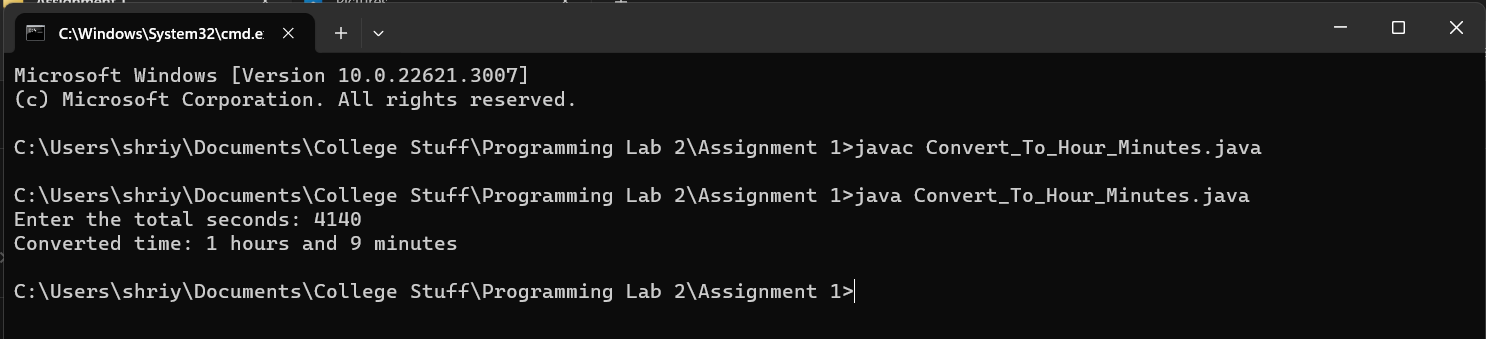


**Que 6) Write java program to convert seconds to hour and minute.**

**Code =>**

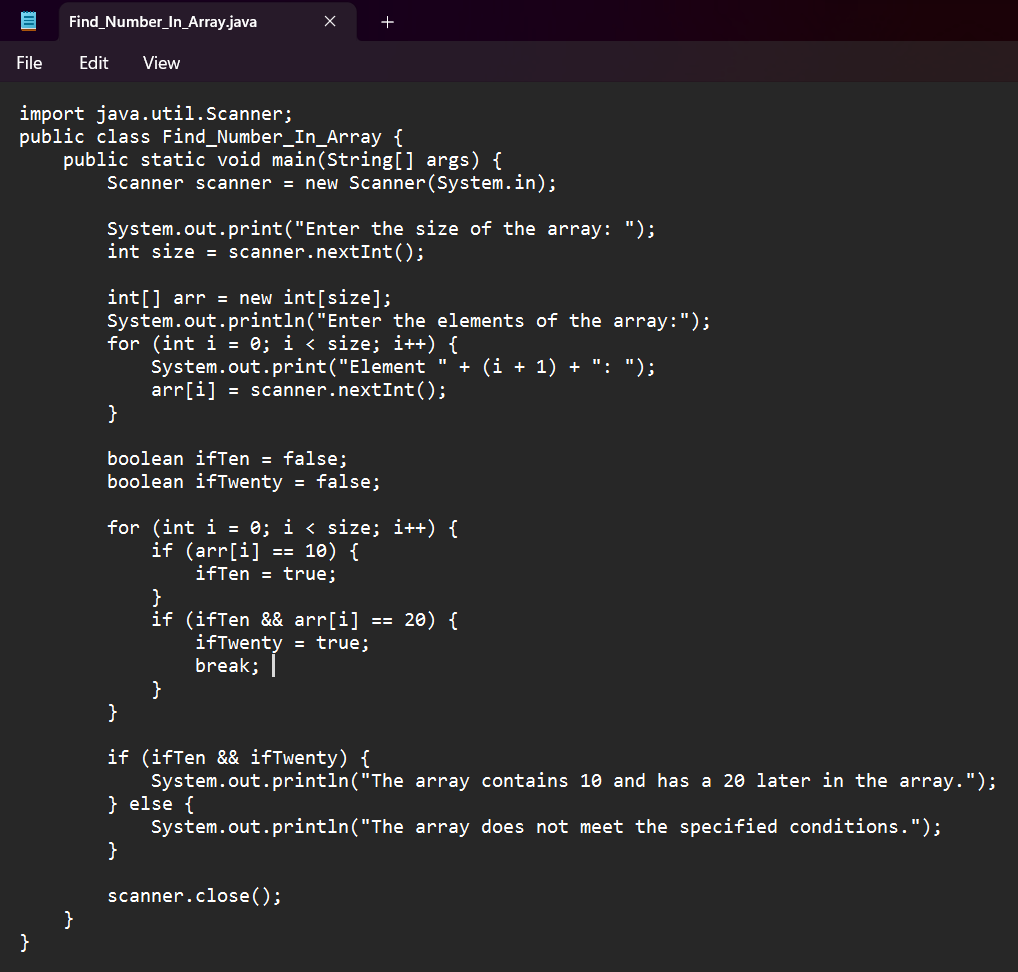


**Output =>**

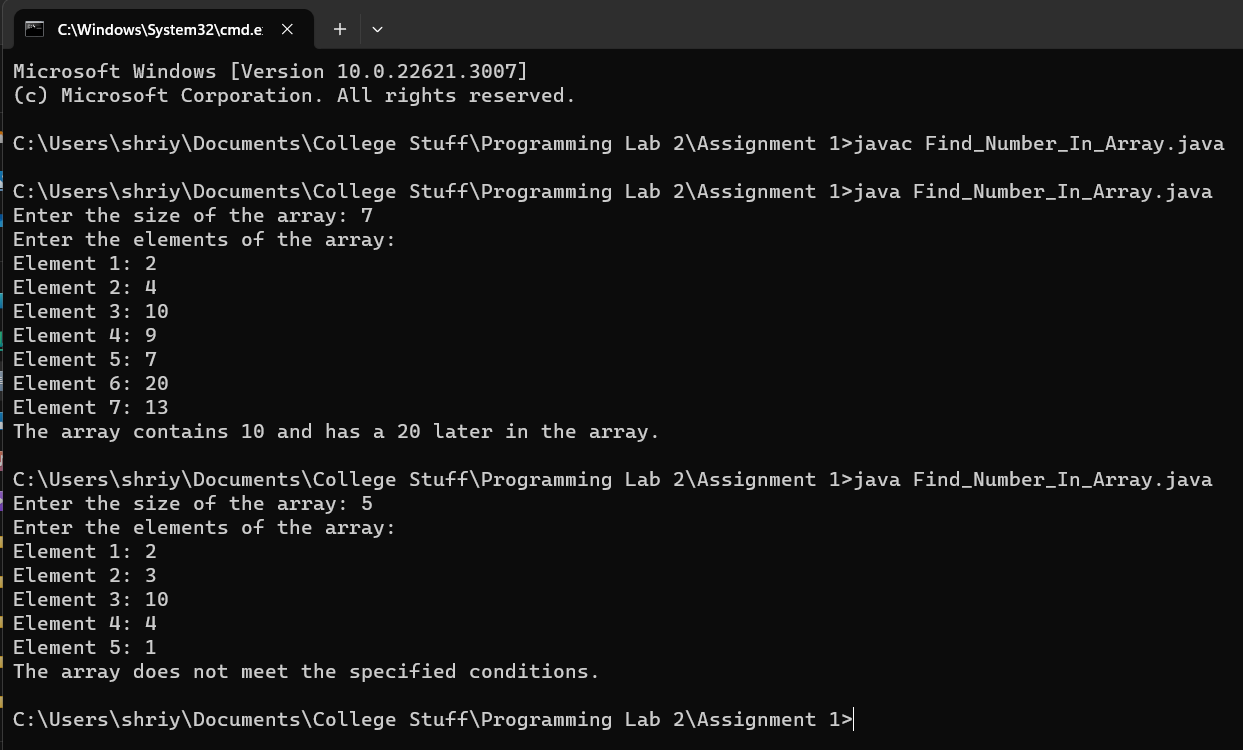


**Que 7) Write java program check if 10 is there in array of integers with a 20 somewhere later in array.**

**Code =>**

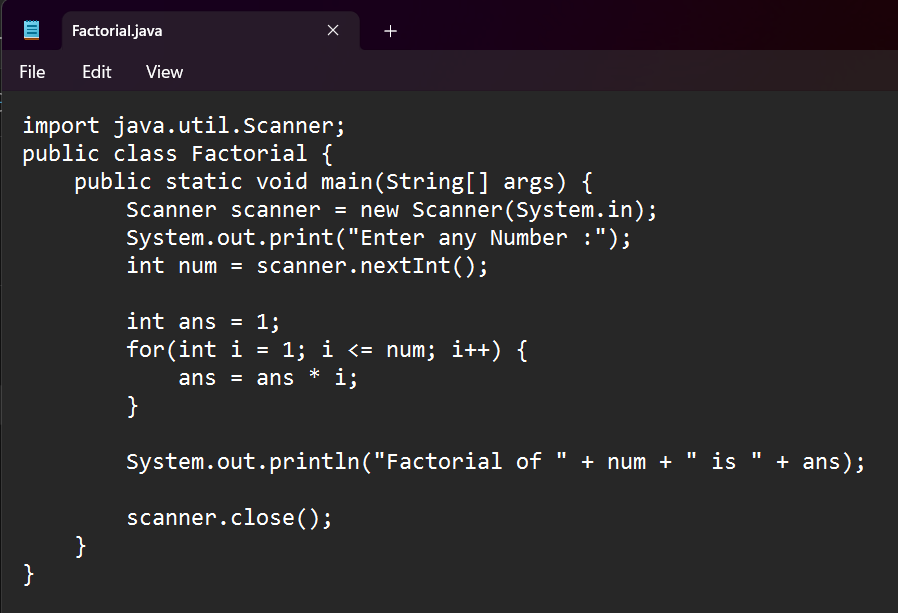


**Output =>**

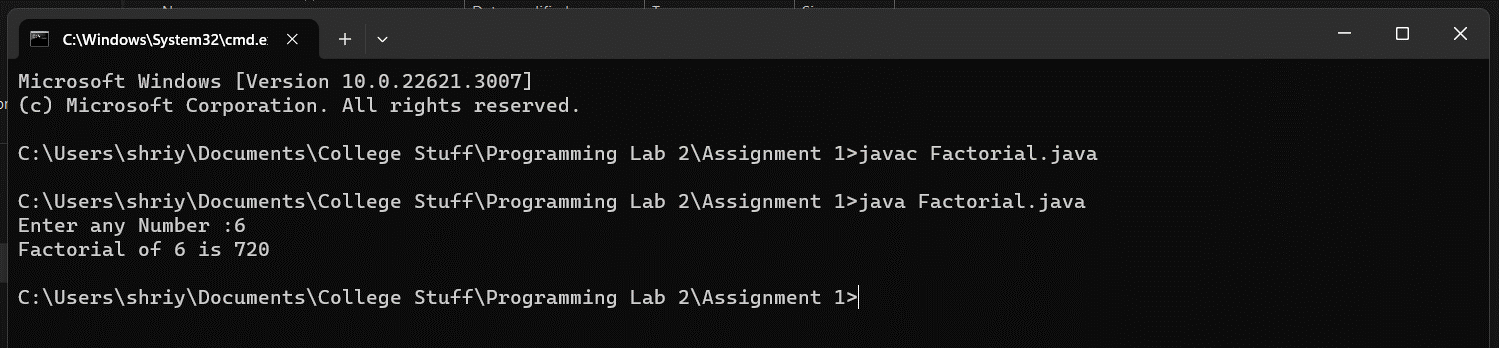


**Que 8) Write a program to calculate factorial of a number.**

**Code =>**

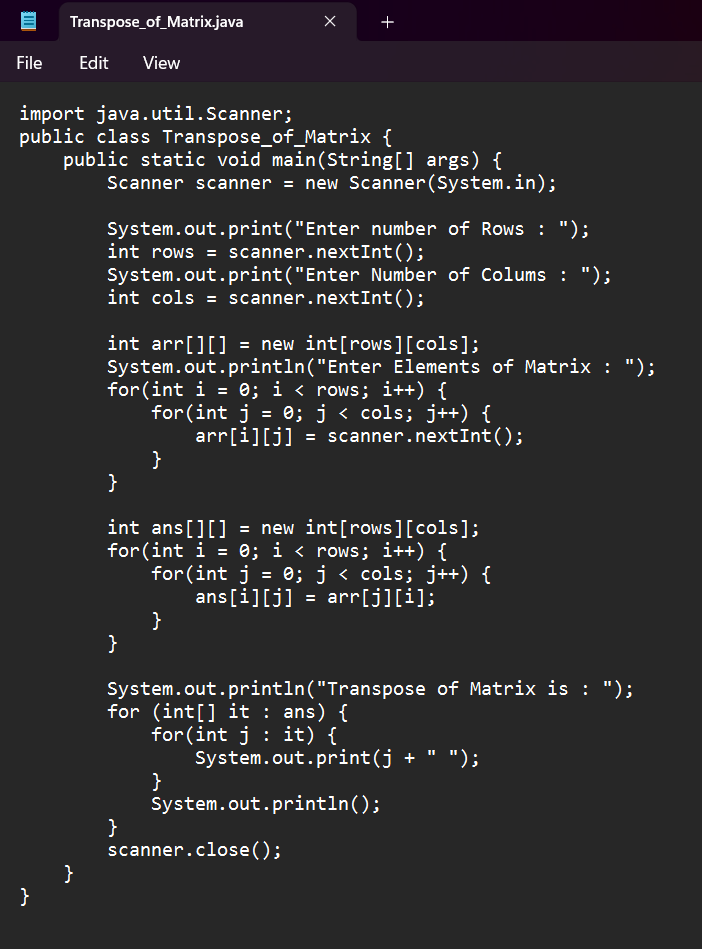


**Output =>**

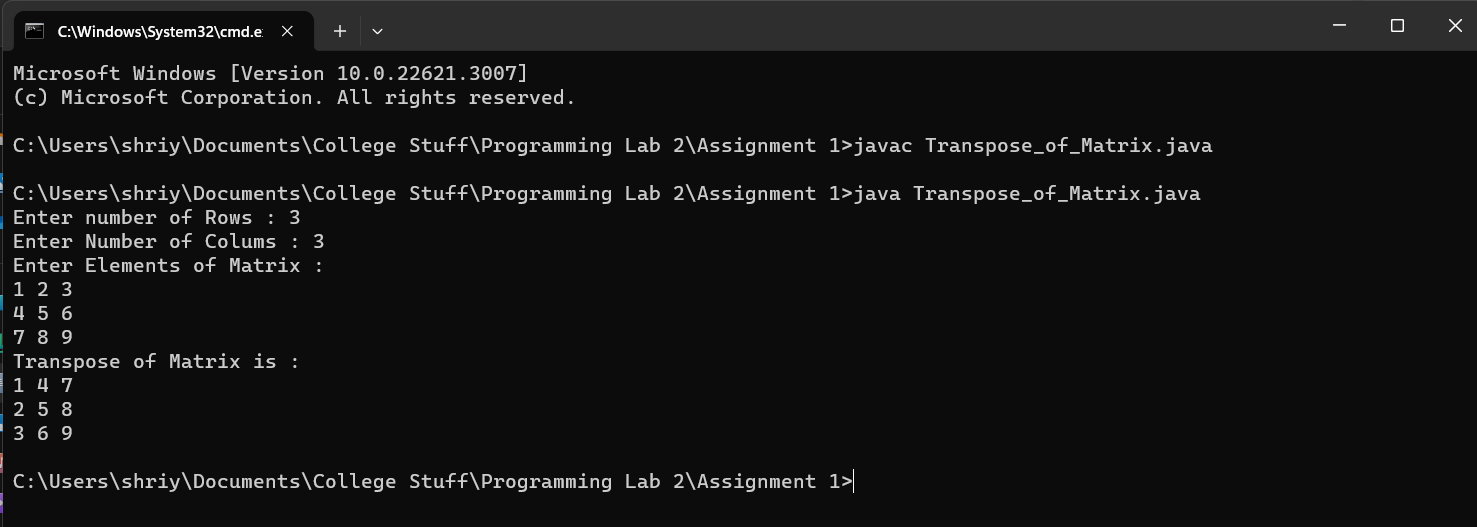


**Que 9) Write java program to find transpose of matrix.**

**Code =>**

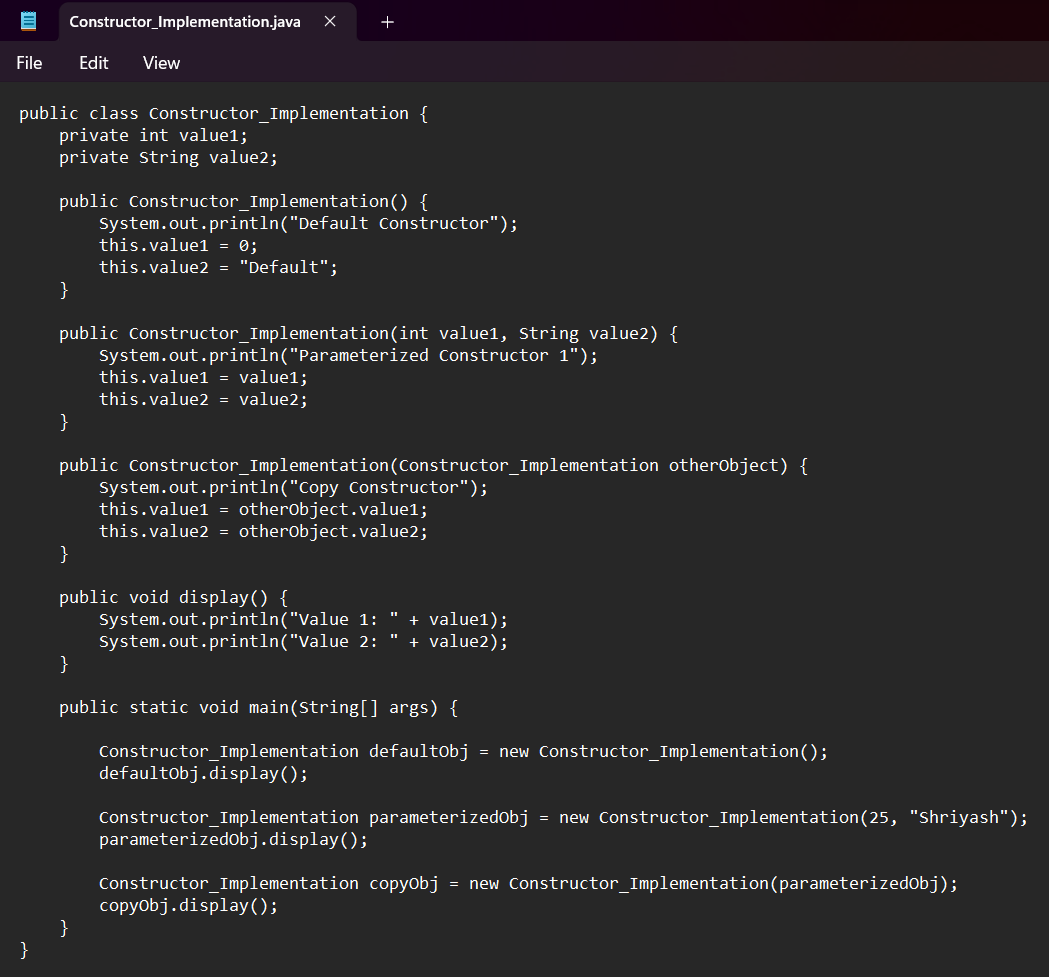


**Output =>**



**Que 10) Write java program to implement different types of constructors.**

**Code =>**



**Output =>**

