**CTS DIGITAL NURTURE - 4.0 JAVA FSE**

**WEEK 1 - DESIGN PATTERNS AND PRINCIPLES**

MANDATORY HANDS-ON 1:

QUESTION:

**Implementing the Singleton Pattern**

**Scenario:**

You need to ensure that a logging utility class in your application has only one instance throughout the application lifecycle to ensure consistent logging.

**Steps:**

1. **Create a New Java Project:**
   * Create a new Java project named **SingletonPatternExample**.
2. **Define a Singleton Class:**
   * Create a class named Logger that has a private static instance of itself.
   * Ensure the constructor of Logger is private.
   * Provide a public static method to get the instance of the Logger class.
3. **Implement the Singleton Pattern:**
   * Write code to ensure that the Logger class follows the Singleton design pattern.
4. **Test the Singleton Implementation:**
   * Create a test class to verify that only one instance of Logger is created and used across the application.

PROGRAM:

**Logger.java**

package designpatterns;

public class Logger {

private static Logger *logger*;

private Logger() {

}

public static Logger getInstance() {

if(*logger*==null) {

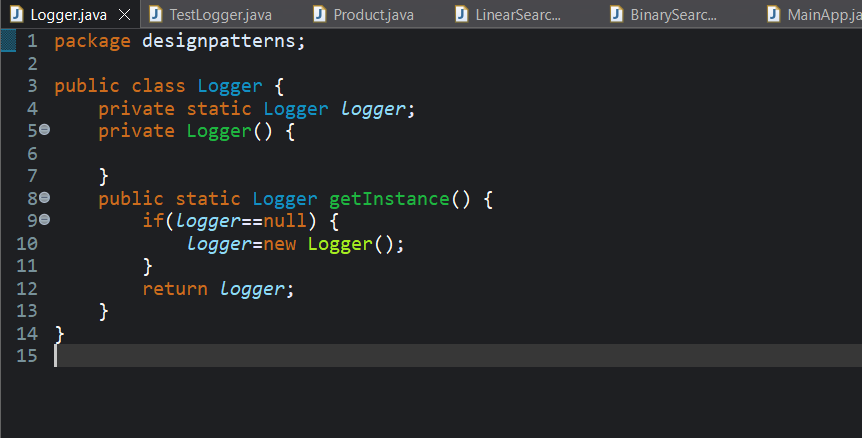
*logger*=new Logger();

}

return *logger*;

}

}



**TestLogger.java**

package designpatterns;

public class TestLogger {

public static void main(String[] args) {

Logger logger1=Logger.*getInstance*();

Logger logger2=Logger.*getInstance*();

if(logger1==logger2)

System.*out*.println("Singleton pattern exist");

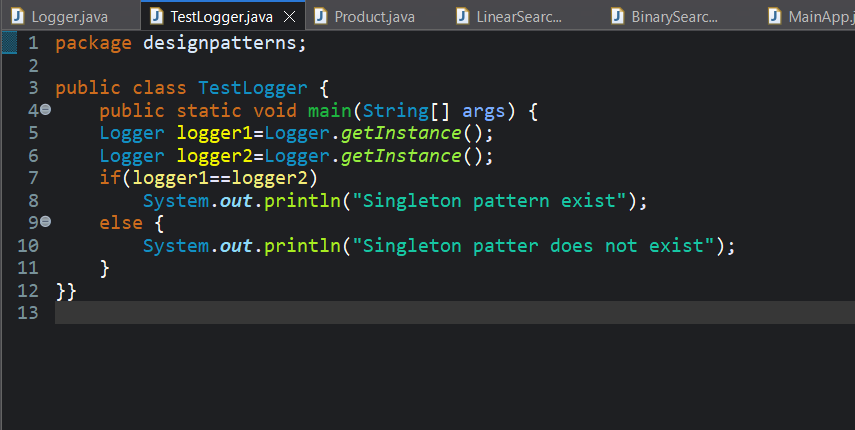
else {

System.*out*.println("Singleton patter does not exist");

}

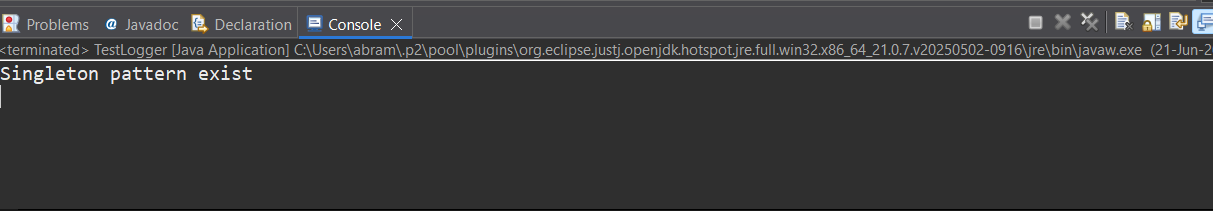
}

}



OUTPUT:

Singleton pattern exist

****