**Exercise 1: 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.

**CODE FOR SINGLETON PATTERN**

package Assessment;

public class Logger

{

private static Logger *instance*;

private Logger()

{

System.*out*.println("Logger Initialized");

}

public static Logger getInstance()

{

if (*instance* == null)

*instance* = new Logger();

}

return *instance*;

}

public void log(String message)

{

System.*out*.println("Log: " + message);

}

public static void main(String[] args)

{

Logger l1 = Logger.*getInstance*();

l1.log("First log message");

Logger l2 = Logger.*getInstance*();

l2.log("Second log message");

if (l1 == l2)

{

System.*out*.println("Both logger instances are the same (Singleton confirmed).");

}

else

{

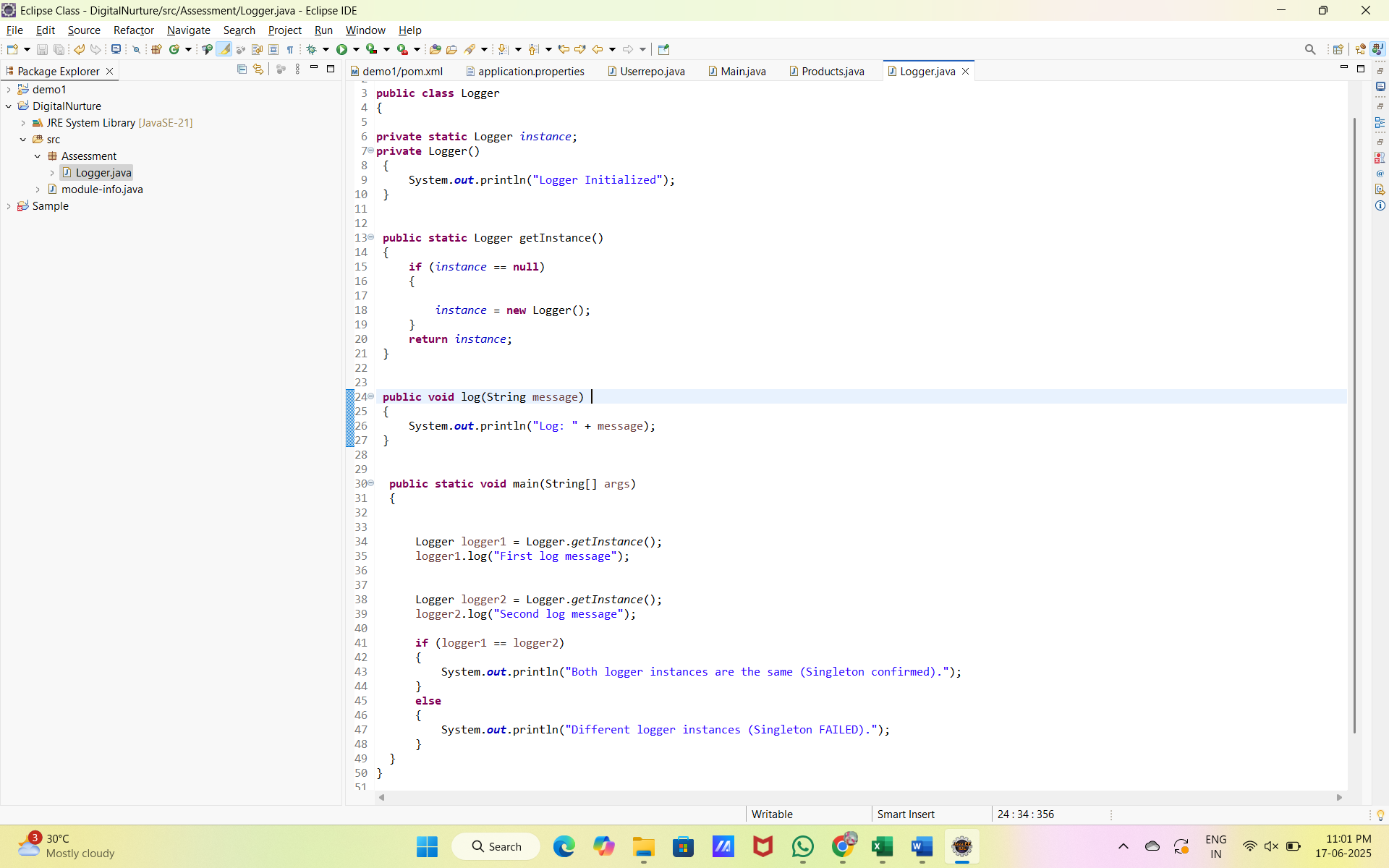
System.*out*.println("Different logger instances (Singleton FAILED).");

}

}

}

CODE SCREENSHOT



OUTPUT:

A screenshot of a computer

AI-generated content may be incorrect.