**Exercise 1: Implementing the Singleton Pattern**

**Code:**

public class SingleObject {

// 1. Private static instance of the class

private static SingleObject instance;

// 2. Private constructor to prevent direct instantiation

private SingleObject() {

// Optional: Initialize resources here, though often done in the getInstance method

}

// 3. Public static method to provide the global access point

// This method is thread-safe using synchronized block

public static SingleObject getInstance() {

if (instance == null) { // Check if the instance is null

synchronized (SingleObject.class) {

// Synchronize on the class object to prevent race conditions

if (instance == null) { // Double-check inside the synchronized block

instance = new SingleObject(); // Create the instance only if it's still null

}

}

}

return instance;

}

// 4. Example method to demonstrate the Singleton's functionality

public void showMessage() {

System.out.println("Hello from the Singleton instance!");

}

public static void main(String[] args) {

// Get the only object available

SingleObject object1 = SingleObject.getInstance();

SingleObject object2 = SingleObject.getInstance();

// Show the message from both objects - they should be the same instance

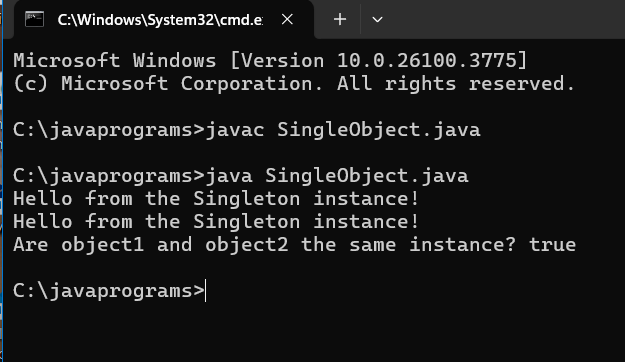
object1.showMessage();

object2.showMessage();

// Verify that both references point to the same object

System.out.println("Are object1 and object2 the same instance? " + (object1 == object2));

}

}**Output: **