Name: Sangamithra U

SupersetID: 6372485

Design Principles and Patterns

Exercise 1: Implementing the Singleton Pattern

Singleton Pattern is a creational design pattern that restricts the instantiation of a class to one single object. It's commonly used for configuration classes, loggers, and resource managers.

Code:

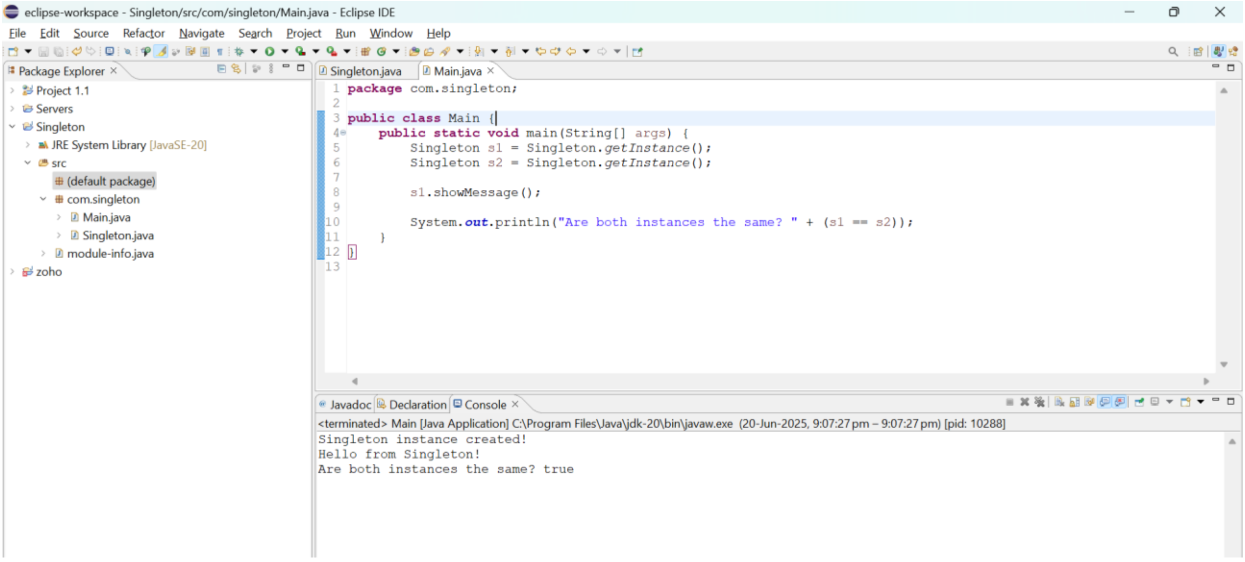
Singleton.java

```
public class Singleton {
    private static Singleton singleInstance = null;
    private Singleton() {
        System.out.println("Singleton instance created!");
    }
    public static Singleton getInstance() {
        if (singleInstance == null) {
            singleInstance = new Singleton();
        }
        return singleInstance;
    }
    public void showMessage() {
        System.out.println("Hello from Singleton!");
    }
}
```

Main.java

```
public class Main {
    public static void main(String[] args) {
        Singleton s1 = Singleton.getInstance();
        Singleton s2 = Singleton.getInstance();
        s1.showMessage();
        System.out.println("Are both instances the same? " + (s1 == s2));
    }
}
```

Output:



```
package com.singleton;

public class Main {
    public static void main(String[] args) {
        Singleton s1 = Singleton.getInstance();
        Singleton s2 = Singleton.getInstance();

        s1.showMessage();

        System.out.println("Are both instances the same? " + (s1 == s2));
    }
}
```

```
<terminated> Main [Java Application] C:\Program Files\Java\jdk-20\bin\javaw.exe  (20-Jun-2025, 9:07:27 pm – 9:07:27 pm) [pid: 10288]
Singleton instance created!
Hello from Singleton!
Are both instances the same? true
```