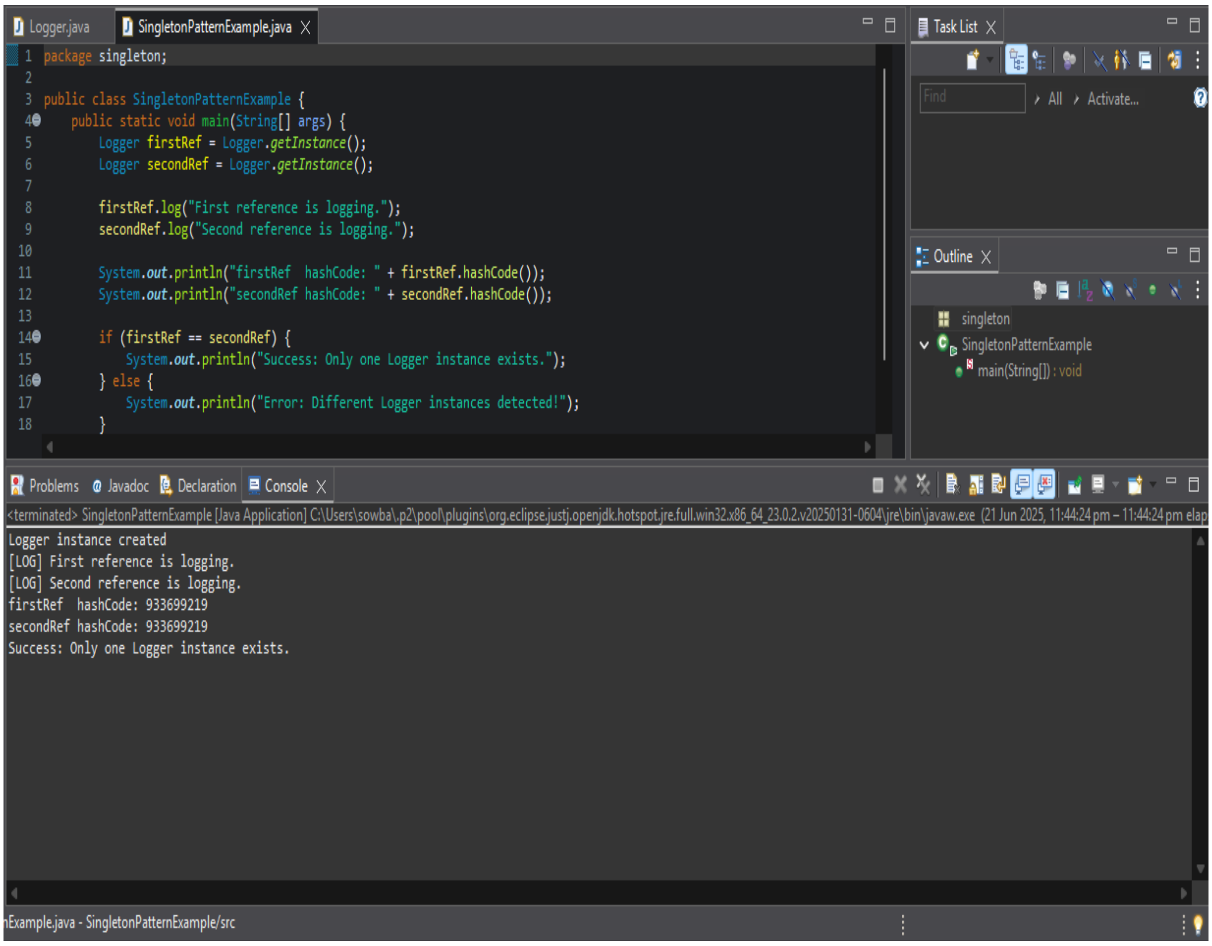
Exercise 1: Implementing the Singleton Pattern

OUTPUT:



```java
package singleton;

public class SingletonPatternExample {
    public static void main(String[] args) {
        Logger firstRef = Logger.getInstance();
        Logger secondRef = Logger.getInstance();

        firstRef.log("First reference is logging.");
        secondRef.log("Second reference is logging.");

        System.out.println("firstRef  hashCode: " + firstRef.hashCode());
        System.out.println("secondRef hashCode: " + secondRef.hashCode());

        if (firstRef == secondRef) {
            System.out.println("Success: Only one Logger instance exists.");
        } else {
            System.out.println("Error: Different Logger instances detected!");
        }
```

```
<terminated> SingletonPatternExample [Java Application] C:\Users\sowba\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_23.0.2.v20250131-0604\jre\bin\javaw.exe (21 Jun 2025, 11:44:24 pm – 11:44:24 pm elap
Logger instance created
[LOG] First reference is logging.
[LOG] Second reference is logging.
firstRef  hashCode: 933699219
secondRef hashCode: 933699219
Success: Only one Logger instance exists.
```

The Singleton Pattern was successfully implemented using a Logger class to ensure only one instance is created and shared across the application. This approach ensures consistent logging, saves memory, and provides a global access point to the logger. The test confirmed the Singleton works correctly.