WEEK 1

DESIGN PATTERNS AND PRINCIPLES

Nipuna Amanapu

[namanapu@gitam.in](mailto:namanapu@gitam.in)

superset id: 6432842

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.

**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.

Solution:

Project name: SingletonPatternExample

Logger.java:

public class Logger {

   private static Logger instance;

   private Logger() {

      System.out.println("Logger instance created.");

   }

   public static Logger getInstance() {

      if (instance == null) {

         instance = new Logger();

      }

      return instance;

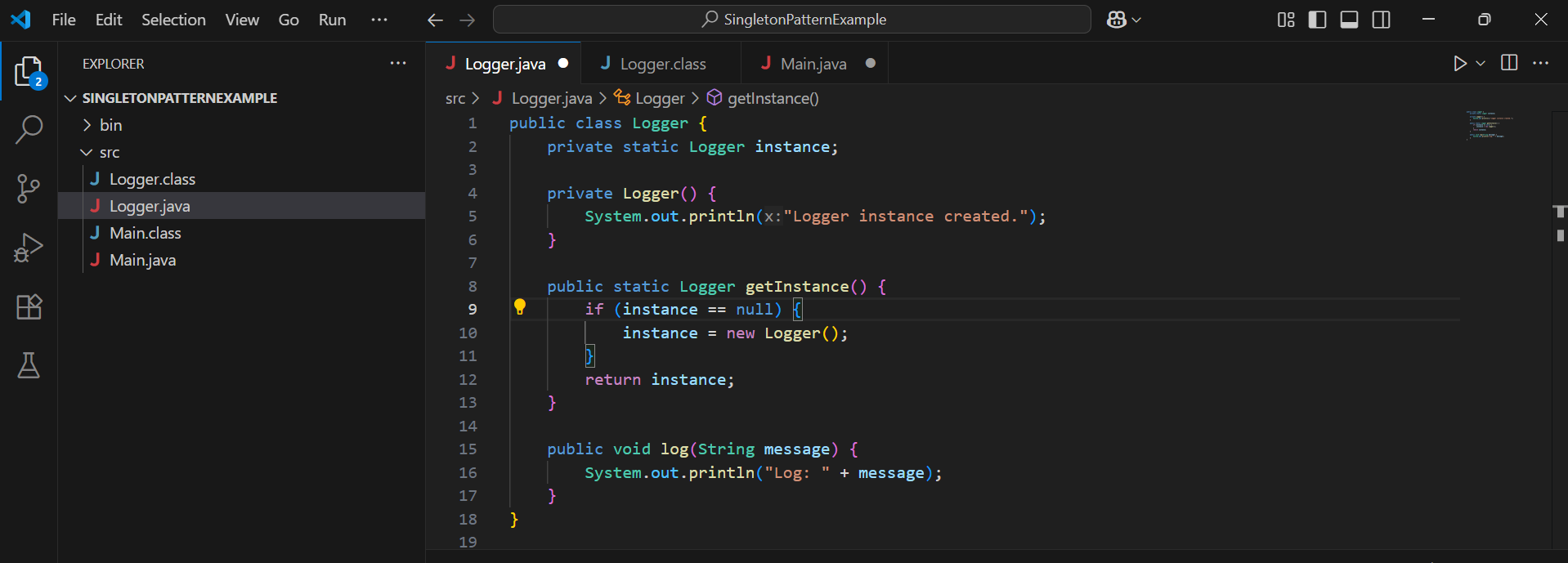
   }

   public void log(String var1) {

      System.out.println("Log: " + var1);

   }

}



Main.java:

public class Main {

    public static void main(String[] args) {

        Logger logger1 = Logger.getInstance();

        logger1.log("this is the first log message");

        Logger logger2 = Logger.getInstance();

        logger2.log("this is the second log message");

        if (logger1 == logger2) {

            System.out.println("both logger1 and logger2 are the same instance");

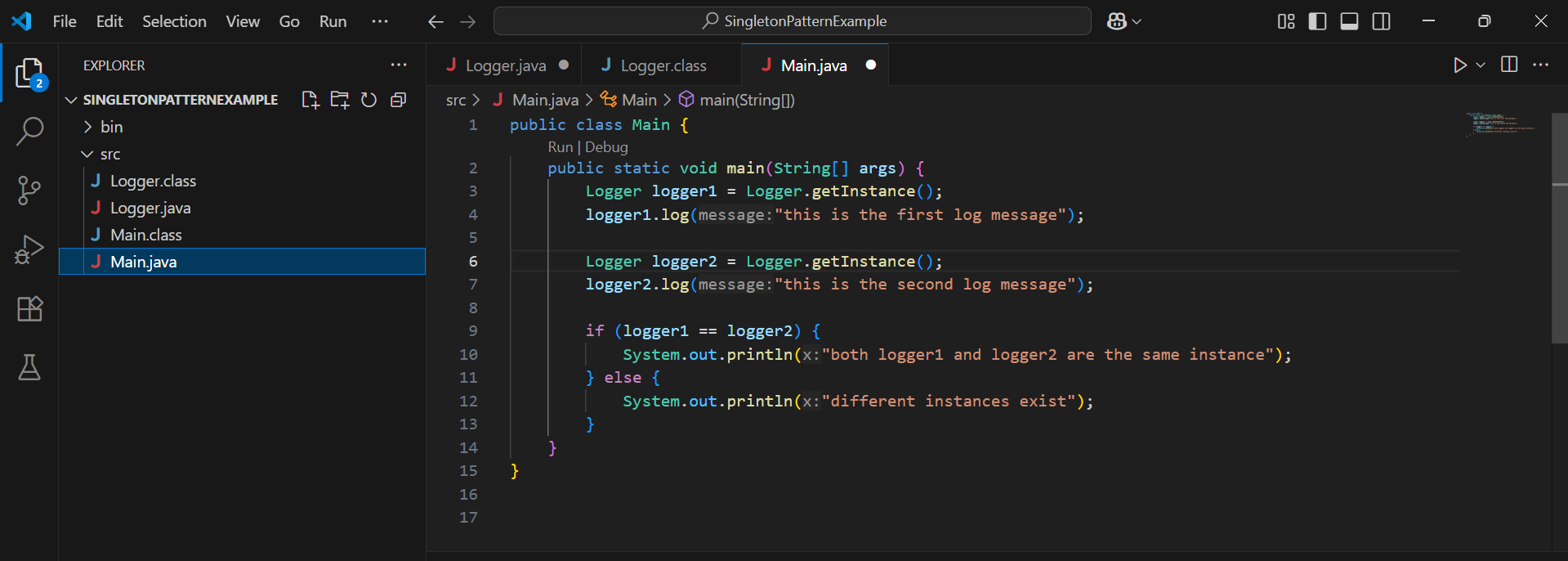
        } else {

            System.out.println("different instances exist");

        }

    }

}



Logger.class and main.class are the respective classes for logger.java and main.java

Output:

