Singleton Patterns

Note: created the class according to provided wordfile of hands-on file in the git Repository

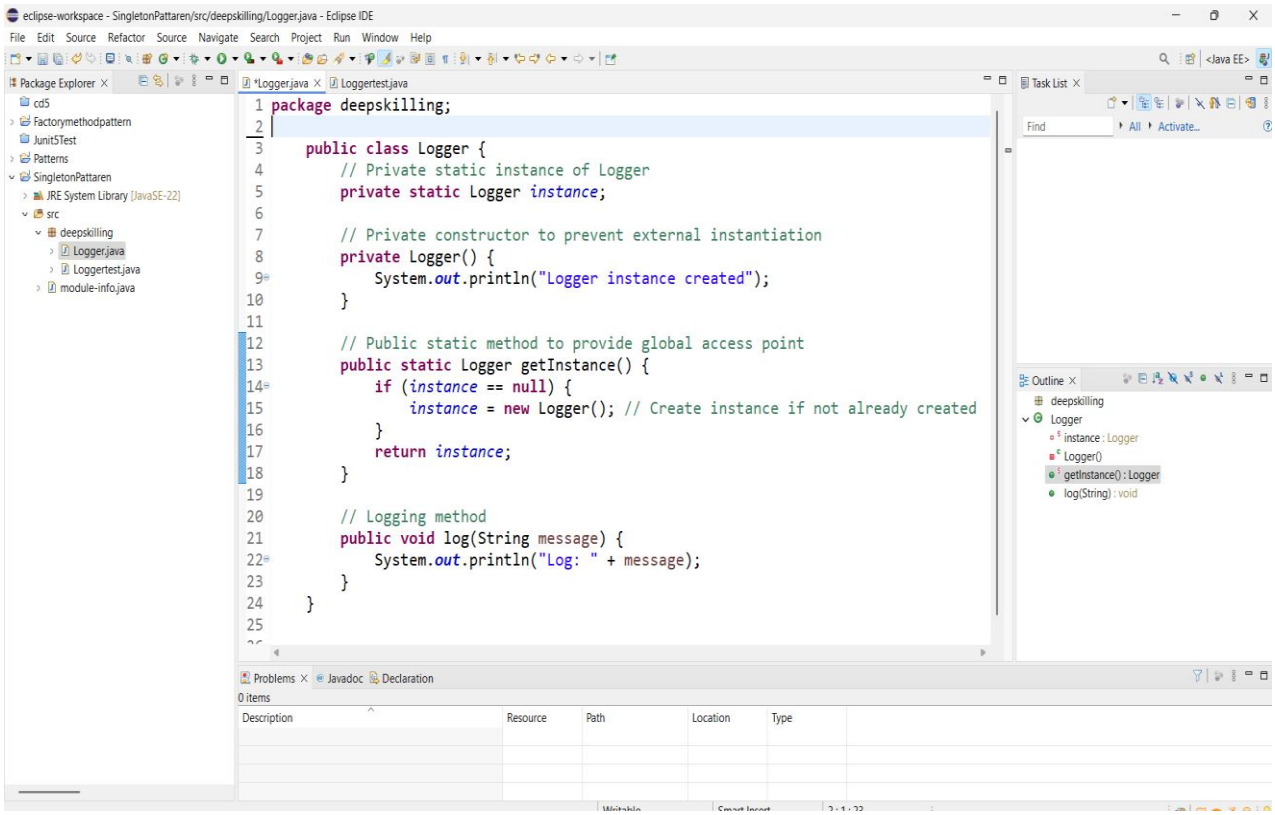
Loggerclass.java

Code:

```
package deepskilling;
        public class Logger {
            // Private static instance of Logger
            private static Logger instance;

            // Private constructor to prevent external instantiation
            private Logger() {
                System.out.println("Logger instance created");
            }

            // Public static method to provide global access point
            public static Logger getInstance() {
if (instance == null) {
```

```
            instance = new Logger(); // Create instance if not
already created
        }
        return instance;
    }

    // Logging method
    public void log(String message) {
        System.out.println("Log: " + message);
    }
}
```

Pic from Eclipse IDE: Logger.java



LoggerTest.java

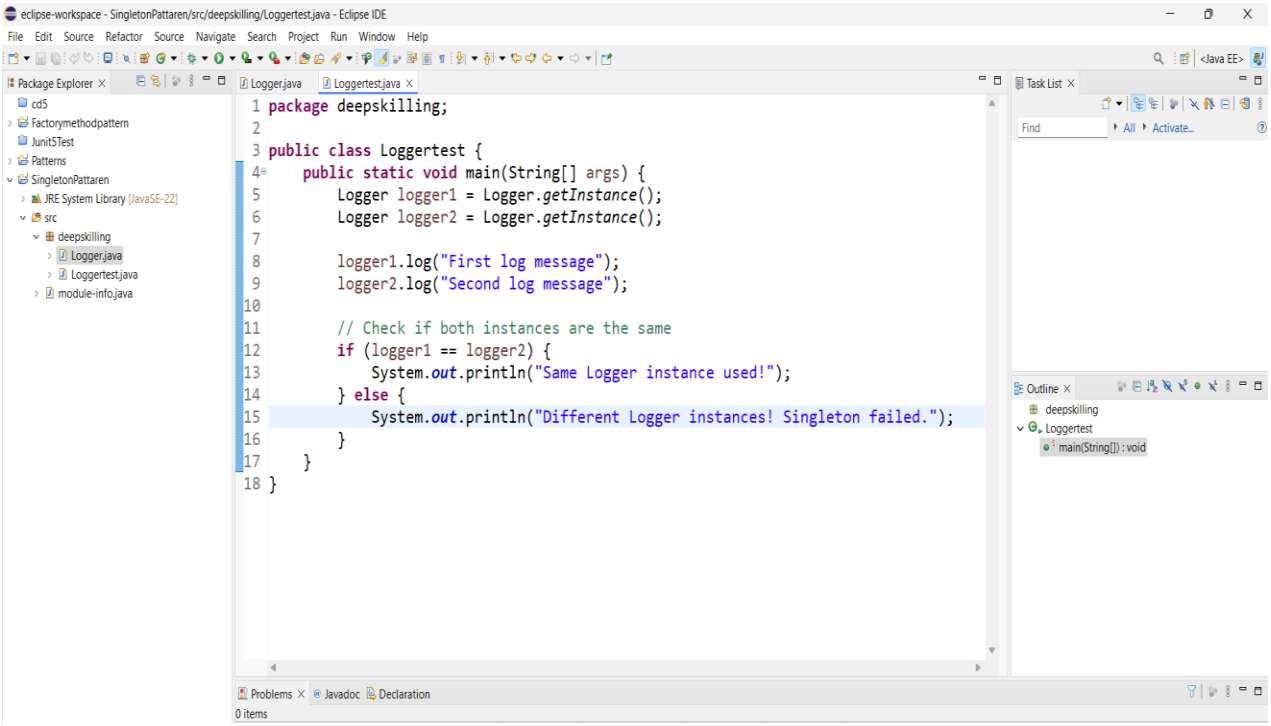
---

code:

```
package deepskilling;


public class Loggertest {
    public static void main(String[] args) {
        Logger logger1 = Logger.getInstance();
        Logger logger2 = Logger.getInstance();


        logger1.log("First log message");
        logger2.log("Second log message");


        // Check if both instances are the same
        if (logger1 == logger2) {
            System.out.println("Same Logger instance used!");
        } else {
            System.out.println("Different Logger instances!
Singleton failed.");
        }
```

```
    }
}
```

Pic from Eclipse IDE: LoggerTest.java



Result: OUTPUT