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.

ANSWER:

Logger.java:

```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 message) {
        System.out.println("LOG MESSAGE " + message);
    }
}
```

LoggerTest.java:

```java
public class LoggerTest {
    public static void main(String[] args) {
        Logger logger1 = Logger.getInstance();
        logger1.log("Message 1");
        Logger logger2 = Logger.getInstance();
        logger2.log("Message 2");
        if (logger1 == logger2) {
            System.out.println("Both logger instances are the same, Hence Singleton confirmed.");
        } else {
            System.out.println("Different instances, Singleton failed.");
        }
    }
}
```

Output:

```
Logger instance created.
LOG MESSAGE Message 1
LOG MESSAGE Message 2
Both logger instances are the same, Hence Singleton confirmed.
```