**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.

**//Logger.java**

```
public class Logger {
    private static Logger instance;

    // Private constructor
    private Logger() {
        System.out.println("Logger created!");
    }

    // Public method
    public static Logger getInstance() {
        if (instance == null) {
            instance = new Logger();
        }
        return instance;
    }

    public void log(String message) {
        System.out.println("LOG: " + message);
    }
}
```

**//Main.java**

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

```
        logger1.log("First log message.");

        Logger logger2 = Logger.getInstance();
        logger2.log("Second log message.");

        if (logger1 == logger2) {
            System.out.println("Both logger instances are
the same (Singleton works!)");
        } else {
            System.out.println("Different logger
instances (Singleton FAILED)");
        }
    }
}
```

**Output:**