Week 1: Design Principles and Patterns:

Exercise 1: Implementing the Singleton Pattern

Singleton Pattern Example: JAVA Project

The Project consists of two java files: Main.java, Logger.java

- Main.java – Has main function that tests the singleton pattern
- Logger.java – Consists of logger class which follows singleton pattern (singleton class)

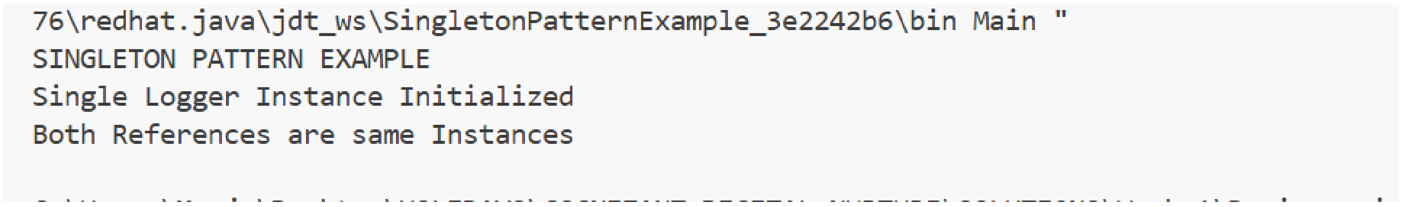
Logger.java

```
public class Logger {
    //PRIVATE STATIC INSTANCE OF ITSELF (Declared final so that there can only be one)
    private static final Logger instance = new Logger();

    //PRIVATE CONSTRUCTOR TO CREATE INSTANCE (Private to ensure no external code can
create another instance)
    private Logger() {
        System.out.println("Single Logger Instance Initialized");
    }

    //METHOD TO RETURN THE ONE AND ONLY CREATED FACTORY INSTANCE
    public static Logger getInstance() {
        return instance;
    }

    //CHECK IF THE REFERENCES POINT TO SAME INSTANCE (For Testing)
    public static void check(Logger a, Logger b) {
        if(a.equals(b))
            System.out.println("Both References are same Instances");
        else
            System.out.println("Both References are different Instances (SINGLETON PATTERN
FAILED)");
    }
}
```

Main.java

```
public class Main {
    //TESTING SINGLETON LOGGER CLASS
    public static void main(String[] args) {
        System.out.println("SINGLETON PATTERN EXAMPLE");
        Logger l1 = Logger.getInstance(); //Reference l1
        Logger l2 = Logger.getInstance(); //Reference l2
        Logger.check(l1, l2); //Checks if both are same instance
    }
}
```

Output



```
76\redhat.java\jdt_ws\SingletonPatternExample_3e2242b6\bin Main "
SINGLETON PATTERN EXAMPLE
Single Logger Instance Initialized
Both References are same Instances
```