**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.

**Code:**

public class Logger{

  private static Logger instance;

  private Logger(){

  }

  public static Logger getinstance(){

    if(instance==null){

      instance=new Logger();

      System.out.println("New log created");

    }

    else{

      System.out.println("Instance already exists");

    }

    return instance;

  }

  public void log(String message){

    System.out.println(message);

  }

}

public class SingletonPatternExample {

  public static void main(String[] args) {

    Logger log1=Logger.getinstance();

    Logger log2=Logger.getinstance();

    log1.log("First");

    log2.log("Second");

    if(log1==log2){

      System.out.println("Singleton Pattern Success");

    }

    else{

      System.out.println("Failure");

    }

  }

}

**Output:**



**s**