**QUESTION:**

**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:**

class Logger{

    private static Logger log;

    private Logger(){};

    public static Logger getinstance(){

        if(log == null){

            log = new Logger();

            System.out.println("log object is null, creating new log instance.");

        }else{

            System.out.println("log object already exists, returning existing log instance.");

        }

        return log;

    }

}

public class  SingletonPatternExample{

    public static void main(String[] args) {

        System.out.println("Testing Singleton Pattern Example");

        System.out.println("Creating first object...");

        Logger log1 = Logger.getinstance();

        System.out.println("Creating second object...");

        Logger log2 = Logger.getinstance();

    }

}

**OUTPUT:**

A screenshot of a computer

AI-generated content may be incorrect.