Step 1: Create a New Java Project

Create a new Java project named SingletonPatternExample.

Step 2: Define a Singleton Class

Create a class named Logger that has a private static instance of itself:

public class Logger {

private static Logger instance;

private Logger() {} // private constructor

public static Logger getInstance() {

if (instance == null) {

instance = new Logger();

}

return instance;

}

}

Step 3: Implement the Singleton Pattern

Implement the Singleton pattern by ensuring that the Logger class follows the Singleton design pattern:

public class Logger {

private static Logger instance;

private Logger() {} // private constructor

public static Logger getInstance() {

if (instance == null) {

synchronized (Logger.class) {

if (instance == null) {

instance = new Logger();

}

}

}

return instance;

}

public void log(String message) {

System.out.println("Logging: " + message);

}

}

Step 4: Test the Singleton Implementation

Create a test class to verify that only one instance of Logger is created and used across the application:

public class LoggerTest {

public static void main(String[] args) {

Logger logger1 = Logger.getInstance();

Logger logger2 = Logger.getInstance();

System.out.println("Logger 1: " + logger1);

System.out.println("Logger 2: " + logger2);

logger1.log("Hello, World!");

logger2.log("This is a test log message.");

// Verify that both loggers are the same instance

System.out.println("Are loggers the same instance? " + (logger1 == logger2));

}

}

OUTPUT:

Logger 1: Logger@12345678

Logger 2: Logger@12345678

Logging: Hello, World!

Logging: This is a test log message.

Are loggers the same instance? true