WEEK-1 (6397530 – Nayudu Praveen Kumar)

Exercise 1: Implementing the Singleton Pattern

MAIN.JAVA:

public class Main {

public static void main(String[] args) {

Logger logger1 = Logger.getInstance();

Logger logger2 = Logger.getInstance();

if (logger1 == logger2) {

System.out.println("Both logger1 and logger2 reference the same instance.");

} else {

System.out.println("logger1 and logger2 reference different instances.");

}

}

}

LOGGER.JAVA:

public class Logger {

// Private static instance of the Logger class

private static Logger instance = null;

// Private constructor to prevent instantiation

private Logger() {

// Prevent instantiation from reflection

if (instance != null) {

throw new RuntimeException("Use getInstance() method to get the single instance of this class.");

}

}

// Public static method to get the single instance

public static Logger getInstance() {

// Lazy initialization

if (instance == null) {

synchronized (Logger.class) {

if (instance == null) {

instance = new Logger();

}

}

}

return instance;

}

// Method to log messages

public void log(String message) {

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

}

}

LOGGERTEST.JAVA:

public class LoggerTest {

public static void main(String[] args) {

// Get the Logger instance

Logger logger1 = Logger.getInstance();

logger1.log("This is the first log message.");

// Get another Logger instance

Logger logger2 = Logger.getInstance();

logger2.log("This is the second log message.");

// Verify that both references point to the same instance

if (logger1 == logger2) {

System.out.println("Singleton Pattern Verified: Both logger1 and logger2 are the same instance.");

} else {

System.out.println("Singleton Pattern Failed: Different instances detected.");

}

// Log another message to demonstrate consistent logging

logger1.log("Logging continues with the same instance.");

}

}

OUTPUT:





