Week 1  
Design principles and patterns  
  
**Exercise 1: Implementing the Singleton Pattern**

Solution:

Logger.class  
// Source code is decompiled from a .class file using FernFlower decompiler.

package singleton;

public class Logger {

private static Logger instance;

private Logger() {

System.out.println("Logger instance created");

}

public static Logger getInstance() {

if (instance == null) {

instance = new Logger();

}

return instance;

}

public void log(String var1) {

System.out.println("[LOG] " + var1);

}

}

Logger .java

package singleton;

public class Logger {

private static Logger instance;

private Logger() {

System.out.println("Logger instance created");

}

public static Logger getInstance() {

if (instance == null) {

instance = new Logger();

}

return instance;

}

public void log(String message) {

System.out.println("[LOG] " + message);

}

}

Main.class  
// Source code is decompiled from a .class file using FernFlower decompiler.

package singleton;

public class Main {

public Main() {

}

public static void main(String[] var0) {

Logger var1 = Logger.getInstance();

Logger var2 = Logger.getInstance();

var1.log("This is the first message.");

var2.log("This is the second message.");

if (var1 == var2) {

System.out.println("Both logger instances are the same (Singleton works).");

} else {

System.out.println("Logger instances are different (Singleton failed).");

}

}

}

Main.java  
package singleton;

public class Main {

public static void main(String[] args) {

Logger logger1 = Logger.getInstance();

Logger logger2 = Logger.getInstance();

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

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

// Verify if both logger1 and logger2 are the same

if (logger1 == logger2) {

System.out.println("Both logger instances are the same (Singleton works).");

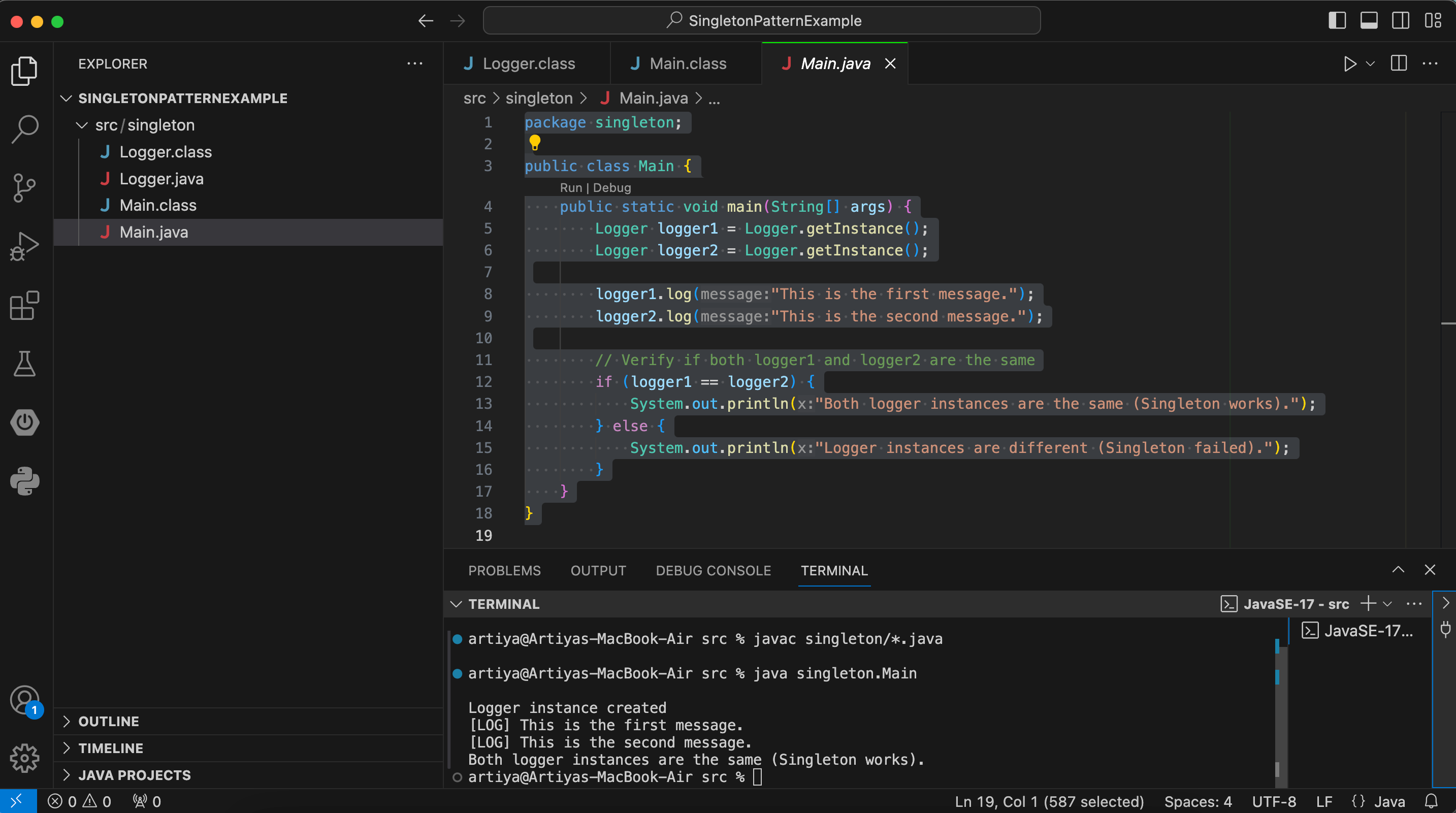
} else {

System.out.println("Logger instances are different (Singleton failed).");

}

}

}

Output  


**Exercise 2: Implementing the Factory Method Pattern**

Solution:

Document.java  
package factory;

public interface Document {

void open();

}

WordDocument.java  
package factory;

public class WordDocument implements Document {

public void open() {

System.out.println("Opening Word Document...");

}

}

PdfDocument.java  
package factory;

public class PdfDocument implements Document {

public void open() {

System.out.println("Opening PDF Document...");

}

}

ExcelDocument.java  
package factory;

public class ExcelDocument implements Document {

public void open() {

System.out.println("Opening Excel Document...");

}

}

DocumentFactory.java  
package factory;

public abstract class DocumentFactory {

public abstract Document createDocument();

}

WordDocumentFactory.java  
package factory;

public class WordDocumentFactory extends DocumentFactory {

public Document createDocument() {

return new WordDocument();

}

}

PdfDocumentFactory.java  
package factory;

public class PdfDocumentFactory extends DocumentFactory {

public Document createDocument() {

return new PdfDocument();

}

}

ExcelDocumentFactory.java  
package factory;

public class ExcelDocumentFactory extends DocumentFactory {

public Document createDocument() {

return new ExcelDocument();

}

}

Main.java  
package factory;

public class Main {

public static void main(String[] args) {

// Create a Word document

DocumentFactory wordFactory = new WordDocumentFactory();

Document wordDoc = wordFactory.createDocument();

wordDoc.open();

// Create a PDF document

DocumentFactory pdfFactory = new PdfDocumentFactory();

Document pdfDoc = pdfFactory.createDocument();

pdfDoc.open();

// Create an Excel document

DocumentFactory excelFactory = new ExcelDocumentFactory();

Document excelDoc = excelFactory.createDocument();

excelDoc.open();

}

}

Output  
