Design Patterns and Principles

**1. Implementing the Singleton Pattern  
  
Code :   
  
Logger.java**package singleton;

public class Logger {

private static volatile Logger instance;

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("[" + System.currentTimeMillis() + "] " + message);

}

}

**TestLogger.java**package singleton;

public class TestLogger {

public static void main(String[] args) {

Logger logger1 = Logger.getInstance();

Logger logger2 = Logger.getInstance();

logger1.log("First message");

logger2.log("Second message");

System.out.println("logger1 hash: " + System.identityHashCode(logger1));

System.out.println("logger2 hash: " + System.identityHashCode(logger2));

if (logger1 == logger2) {

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

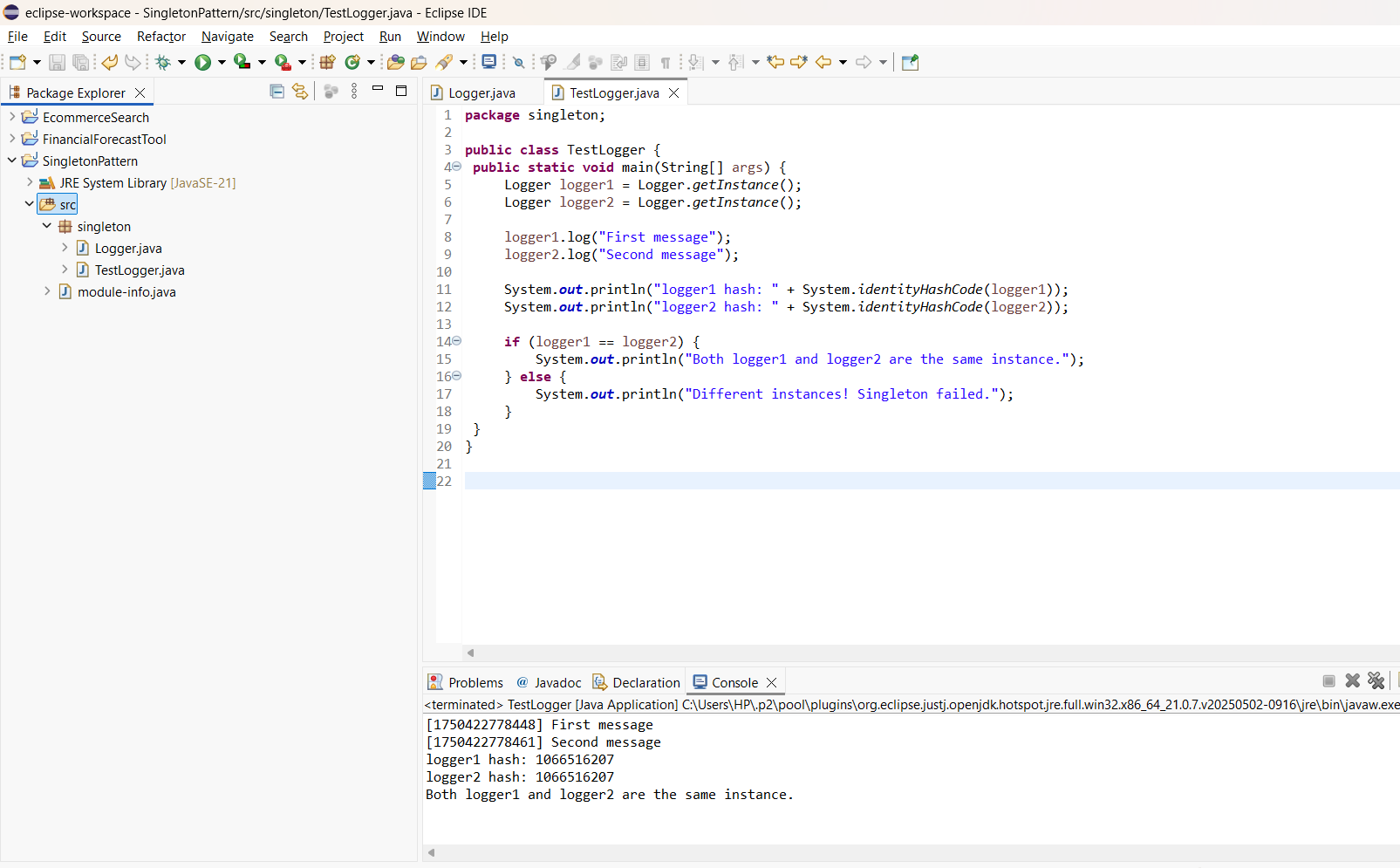
} else {

System.out.println("Different instances! Singleton failed.");

}

}

}

**Output :  
  
  
**

**2. Implementing the Factory Method Pattern  
  
Code :**

**Package : documents  
  
Document.java**package documents;

public interface Document {

void open();

}

**ExcelDocument.java**

package documents;

public class ExcelDocument implements Document {

@Override

public void open() {

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

}

} **PdfDocument.java**package documents;

public class PdfDocument implements Document {

@Override

public void open() {

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

}

}

**WordDocument.java**

package documents;

public class WordDocument implements Document {

@Override

public void open() {

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

}

}

**Package : factories  
  
DocumentFactory.java**

package factories;

import documents.Document;

public abstract class DocumentFactory {

public abstract Document createDocument();

}

**ExcelDocumentFactory.java**

package factories;

import documents.Document;

import documents.ExcelDocument;

public class ExcelDocumentFactory extends DocumentFactory {

@Override

public Document createDocument() {

return new ExcelDocument();

}

}

**PdfDocumentFactory.java**package factories;

import documents.Document;

import documents.PdfDocument;

public class PdfDocumentFactory extends DocumentFactory {

@Override

public Document createDocument() {

return new PdfDocument();

}

}

**WordDocumentFactory.java**package factories;

import documents.Document;

import documents.WordDocument;

public class WordDocumentFactory extends DocumentFactory {

@Override

public Document createDocument() {

return new WordDocument();

}

}

**Main.java**package FactoryMethodPatternExample;

import factories.\*;

import documents.\*;

public class Main {

public static void main(String[] args) {

DocumentFactory wordFactory = new WordDocumentFactory();

Document word = wordFactory.createDocument();

word.open();

DocumentFactory pdfFactory = new PdfDocumentFactory();

Document pdf = pdfFactory.createDocument();

pdf.open();

DocumentFactory excelFactory = new ExcelDocumentFactory();

Document excel = excelFactory.createDocument();

excel.open();

}

}  
  
  
**Output :**