WEEK - 1

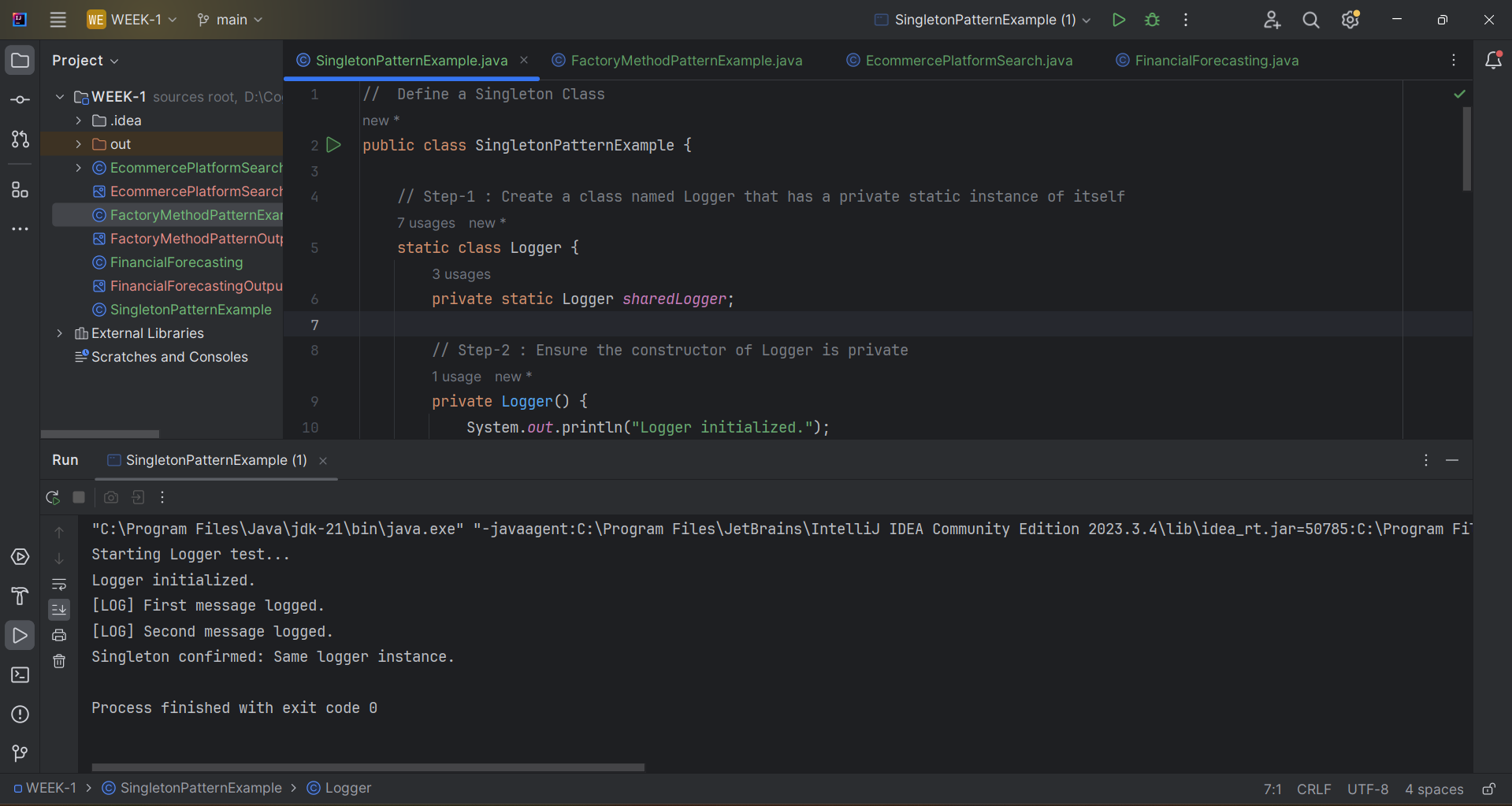
Design Patterns and Principles

Exercise – 1 : Implementing the Singleton Pattern

Code :

// Define a Singleton Class  
public class SingletonPatternExample {  
  
 // Step-1 : Create a class named Logger that has a private static instance of itself  
 static class Logger {  
 private static Logger *sharedLogger*;  
  
 // Step-2 : Ensure the constructor of Logger is private  
 private Logger() {  
 System.*out*.println("Logger initialized.");  
 }  
  
 // Step-3 : Provide a public static method to get the instance of the Logger class  
 public static Logger getInstance() {  
 if (*sharedLogger* == null) {  
 *sharedLogger* = new Logger();  
 }  
 return *sharedLogger*;  
 }  
  
 public void log(String message) {  
 System.*out*.println("[LOG] " + message);  
 }  
 }  
  
 public static void main(String[] args) {  
 System.*out*.println("Starting Logger test...");  
  
 Logger firstLogger = Logger.*getInstance*();  
 Logger secondLogger = Logger.*getInstance*();  
  
 firstLogger.log("First message logged.");  
 secondLogger.log("Second message logged.");  
  
 if (firstLogger == secondLogger) {  
 System.*out*.println("Singleton confirmed: Same logger instance.");  
 } else {  
 System.*out*.println("Singleton failed: Different logger instances.");  
 }  
 }  
}

Output :



Exercise – 2 : Implementing the Factory Method Pattern

Code :

// Factory Method Pattern  
public class FactoryMethodPatternExample {  
  
 // Create interfaces or abstract classes for different document types such as WordDocument, PdfDocument, and ExcelDocument.  
 interface Document {  
 void open();  
 }  
  
 // Concrete Word document  
 static class WordDocument implements Document {  
 @Override  
 public void open() {  
 System.*out*.println("Opening a Word Document");  
 }  
 }  
  
 // Concrete PDF document  
 static class PdfDocument implements Document {  
 @Override  
 public void open() {  
 System.*out*.println("Opening a PDF Document");  
 }  
 }  
  
 // Concrete Excel document  
 static class ExcelDocument implements Document {  
 @Override  
 public void open() {  
 System.*out*.println("Opening an Excel Spreadsheet");  
 }  
 }  
  
 // Abstract factory with a method to create a document  
 abstract static class DocumentFactory {  
 public abstract Document createDocument();  
 }  
  
 // Factory for Word documents  
 static class WordDocumentFactory extends DocumentFactory {  
 @Override  
 public Document createDocument() {  
 return new WordDocument();  
 }  
 }  
  
 // Factory for PDF documents  
 static class PdfDocumentFactory extends DocumentFactory {  
 @Override  
 public Document createDocument() {  
 return new PdfDocument();  
 }  
 }  
  
 // Factory for Excel documents  
 static class ExcelDocumentFactory extends DocumentFactory {  
 @Override  
 public Document createDocument() {  
 return new ExcelDocument();  
 }  
 }  
  
 // Entry point to test the Factory Method Pattern  
 public static void main(String[] args) {  
 // Create factories  
 DocumentFactory wordFactory = new WordDocumentFactory();  
 DocumentFactory pdfFactory = new PdfDocumentFactory();  
 DocumentFactory excelFactory = new ExcelDocumentFactory();  
  
 // Create documents  
 Document wordDoc = wordFactory.createDocument();  
 Document pdfDoc = pdfFactory.createDocument();  
 Document excelDoc = excelFactory.createDocument();  
  
 // Open documents  
 wordDoc.open();  
 pdfDoc.open();  
 excelDoc.open();  
 }  
}

Output :

