**Design Patterns and Principles**

**Exercise 1: Implementing the Singleton Pattern**

**Logger.java:**

package SingletonPatternExample;

public class Logger {

private static Logger *instance*;

private Logger() {

System.*out*.println("Logger initialized");

}

public static Logger getInstance() {

if (*instance* == null) {

*instance* = new Logger();

}

return *instance*;

}

public void log(String message) {

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

}

}

**Main.java:**

package SingletonPatternExample;

public class Main {

public static void main(String[] args) {

Logger logger1 = Logger.*getInstance*();

Logger logger2 = Logger.*getInstance*();

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

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

if (logger1 == logger2) {

System.*out*.println("Both logger1 and logger2 refer to the same instance!Singleton is followed.");

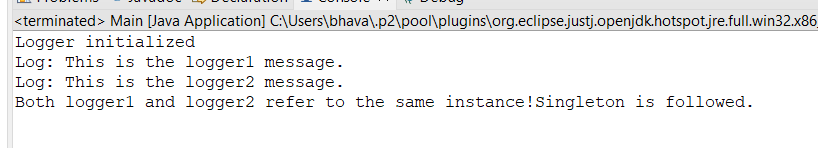
} else {

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

}

}

}

**Output:**

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

**wordDocument.java:**

package FactoryMethodPatternExample;

public interface WordDocument {

void openWord();

}

**pdfDocument.java:**

package FactoryMethodPatternExample;

public interface PdfDocument {

void openPdf();

}

**excelDocument.java:**

package FactoryMethodPatternExample;

public interface ExcelDocument {

void openExcel();

}

**wordDocumentImpl.java:**

package FactoryMethodPatternExample;

public class WordDocumentImpl implements WordDocument {

@Override

public void openWord() {

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

}

}

**PdfDocumentImpl.java:**

package FactoryMethodPatternExample;

public class PdfDocumentImpl implements PdfDocument {

@Override

public void openPdf() {

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

}

}

**excelDocumentImpl.java:**

package FactoryMethodPatternExample;

public class ExcelDocumentImpl implements ExcelDocument {

@Override

public void openExcel() {

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

}

}

**documentFactory.java:**

package FactoryMethodPatternExample;

public abstract class DocumentFactory {

public abstract Object createDocument();

}

**wordDocumentFactory.java:**

package FactoryMethodPatternExample;

public class WordDocumentFactory extends DocumentFactory {

@Override

public WordDocument createDocument() {

return new WordDocumentImpl();

}

}

**pdfDocumentFactory.java:**

package FactoryMethodPatternExample;

public class PdfDocumentFactory extends DocumentFactory {

@Override

public PdfDocument createDocument() {

return new PdfDocumentImpl();

}

}

**excelDocumentFactory.java:**

package FactoryMethodPatternExample;

public class ExcelDocumentFactory extends DocumentFactory {

@Override

public ExcelDocument createDocument() {

return new ExcelDocumentImpl();

}}

**Main.java:**

package FactoryMethodPatternExample;

public class Main {

public static void main(String[] args) {

WordDocumentFactory wordFactory = new WordDocumentFactory();

WordDocument wordDoc = wordFactory.createDocument();

wordDoc.openWord();

PdfDocumentFactory pdfFactory = new PdfDocumentFactory();

PdfDocument pdfDoc = pdfFactory.createDocument();

pdfDoc.openPdf();

ExcelDocumentFactory excelFactory = new ExcelDocumentFactory();

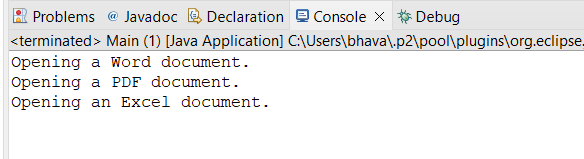
ExcelDocument excelDoc = excelFactory.createDocument();

excelDoc.openExcel();

}

}

**Output:**

****