**DESIGN PATTERNS AND PRINCIPLES**

**EXERCISE-1: IMPLEMENTING THE SINGLETON PATTERN**

**Logger.java**

public class Logger {

private static Logger *instance*;

public Logger() {

// TODO Auto-generated constructor stub

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

}

public static Logger getInstance() {

if(*instance* == null) {

*instance* = new Logger();

}

return *instance*;

}

public void testLog(int number1, int number2) {

System.*out*.println("The two numbers are: "+ number1 + " " + number2);

}

}

**Test.java**

public class Test {

public static void main(String[] args) {

Logger logger1 = Logger.*getInstance*();

Logger logger2 = Logger.*getInstance*();

logger1.testLog(1, 2);

logger2.testLog(3, 4);

if (logger1 == logger2) {

System.*out*.println("Only one instance is used for the entire application");

} else {

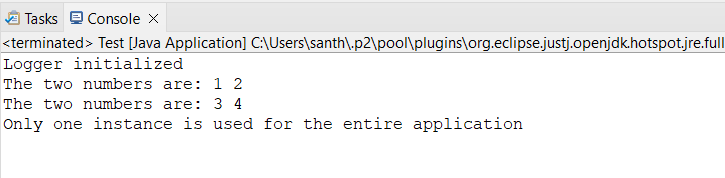
System.*out*.println("Different instances are used");

}

}

}

**OUTPUT:**



**EXERCISE-2: IMPLEMENTING THE FACTORY METHOD PATTERN**

**WordDocument.java**

package FactoryMethod;

public interface WordDocument {

void openDocuments();

}

PdfDocument.java

package FactoryMethod;

public interface PdfDocument {

void openDocuments();

}

**ExcelDocument.java**

package FactoryMethod;

public interface ExcelDocument {

void openDocuments();

}

**WordInterfaceImplement.java**

package FactoryMethod;

public class WordInterfaceImplement implements WordDocument {

@Override

public void openDocuments() {

System.*out*.println("Word is opened");

}

}

**PdfInterfaceImplements.java**

package FactoryMethod;

public class PdfInterfaceImplements implements PdfDocument{

@Override

public void openDocuments() {

System.*out*.println("Pdf is opened");

}

}

**ExcelInterfaceImplements.java**

package FactoryMethod;

public class ExcelInterfaceImplements implements ExcelDocument{

@Override

public void openDocuments() {

System.*out*.println("Excel is opened");

}

}

**DocumentFactory.java**

package FactoryMethod;

public abstract class DocumentFactory {

public abstract Object createDocument();

}

**WordFactory.java**

package FactoryMethod;

public class WordFactory extends DocumentFactory {

@Override

public WordDocument createDocument() {

return new WordInterfaceImplement();

}

}

**PdfFactory.java**

package FactoryMethod;

public class PdfFactory extends DocumentFactory {

@Override

public PdfDocument createDocument() {

return new PdfInterfaceImplements();

}

}

**ExcelFactory.java**

package FactoryMethod;

public class ExcelFactory extends DocumentFactory {

@Override

public ExcelDocument createDocument() {

return new ExcelInterfaceImplements();

}

}

**Test.java**

package FactoryMethod;

public class Test {

public static void main(String[] args) {

WordFactory wordFactory = new WordFactory();

WordDocument wordDoc = wordFactory.createDocument();

wordDoc.openDocuments();

PdfFactory pdfFactory = new PdfFactory();

PdfDocument pdfDoc = pdfFactory.createDocument();

pdfDoc.openDocuments();

ExcelFactory excelFactory = new ExcelFactory();

ExcelDocument excelDoc = excelFactory.createDocument();

excelDoc.openDocuments();

}

}

**OUTPUT:**

