# Week-1

# Design Principles And Patterns

## Exercise 1- Implementing the Singleton Pattern

Problem Statement: You need to ensure that a logging utility class in your application has only one instance throughout the application lifecycle to ensure consistent logging.

Code:

#### Logger.java

public class Logger {

private static Logger instance;

private Logger() {

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

}

public static Logger getInstance() {

if (instance == null) {

instance = new Logger();

}

return instance;

}

public void log(String message) {

System.out.println("LOG: " + message);

}

}

#### Main.java

public class Main {

public static void main(String[] args) {

Logger first = Logger.getInstance();

first.log("Logging started...");

Logger second = Logger.getInstance();

second.log("Another log entry.");

if (first == second) {

System.out.println("Same Logger instance reused.");

} else {

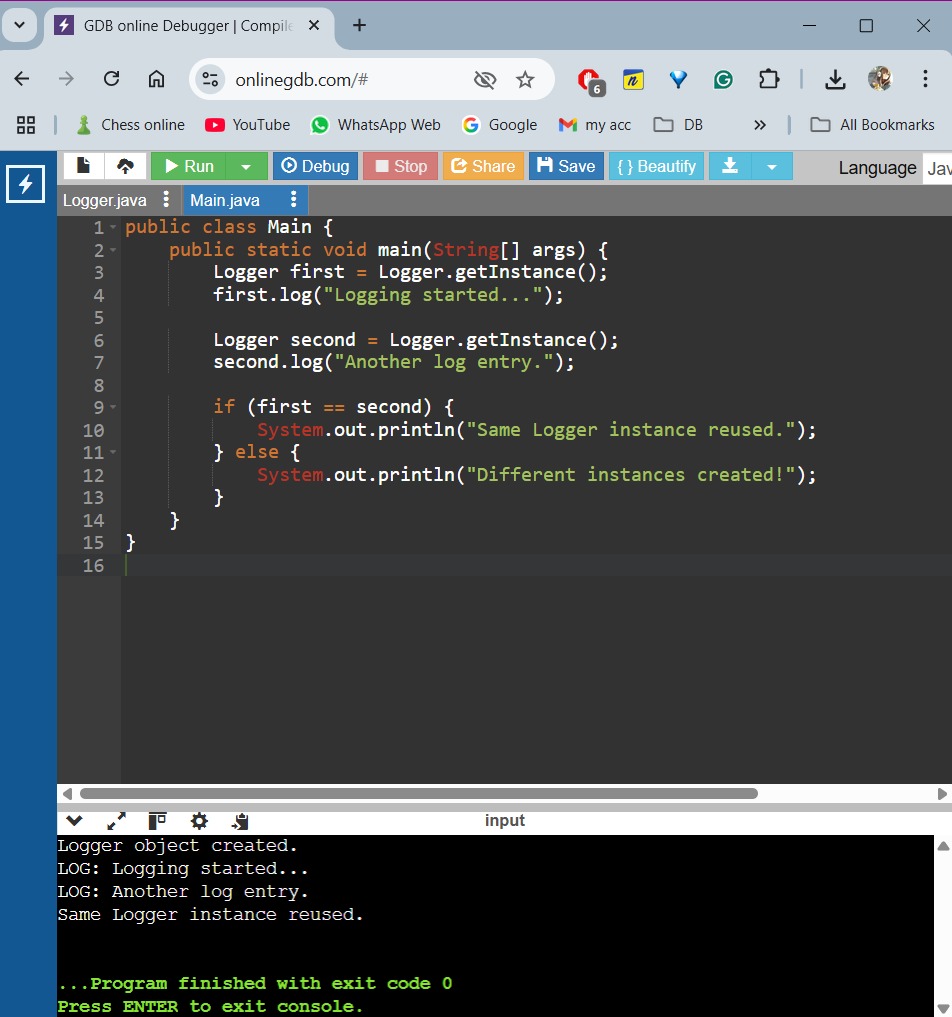
System.out.println("Different instances created!");

}

}

}

Output Screenshot:



Output is:

Logger object created.

LOG: Loggin started...

LOG: Another log entry.

Same Logger instance reused.

## Exercise 2: Implementing the Factory Method Pattern

Problem Statement- You are developing a document management system that needs to create different types of documents (Word, PDF, Excel) using the Factory Method Pattern.

Code:

#### Document.java

public interface Document {

void open();

}

#### WordDocument.java

public class WordDocument implements Document {

public void open() {

System.out.println("Opening a Word file.");

}

}

#### PdfDocument.java

public class PdfDocument implements Document {

public void open() {

System.out.println("Opening a PDF file.");

}

}

#### ExcelDocument.java

public class ExcelDocument implements Document {

public void open() {

System.out.println("Opening an Excel file.");

}

}

#### DocumentFactory.java

public abstract class DocumentFactory {

public abstract Document createDocument();

}

#### WordDocumentFactory.java

public class WordDocumentFactory extends DocumentFactory {

public Document createDocument() {

return new WordDocument();

}

}

#### PdfDocumentFactory.java

public class PdfDocumentFactory extends DocumentFactory {

public Document createDocument() {

return new PdfDocument();

}

}

#### ExcelDocumentFactory.java

public class ExcelDocumentFactory extends DocumentFactory {

public Document createDocument() {

return new ExcelDocument();

}

}

#### Main.java

public class Main {

public static void main(String[] args) {

DocumentFactory word = new WordDocumentFactory();

word.createDocument().open();

DocumentFactory pdf = new PdfDocumentFactory();

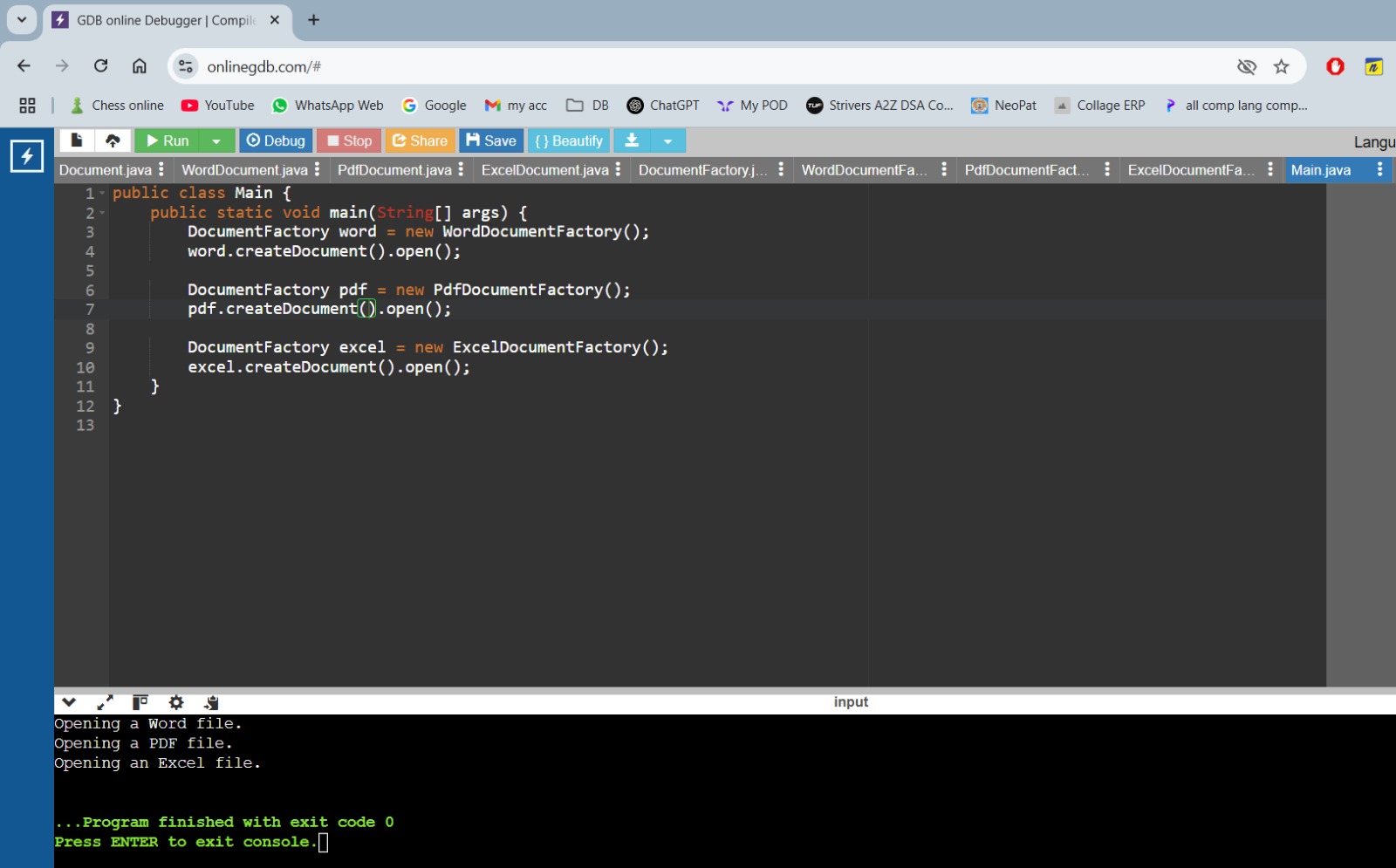
pdf.createDocument().open();

DocumentFactory excel = new ExcelDocumentFactory();

excel.createDocument().open();

}

}

Output Screenshot:

Output is:

Opening Word Document.

Opening PDF Document.

Opening Excel Document.