**WEEK - 1**

**SINGLETON PATTERN**

**QUESTION – 1 :**

**Scenario:**

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

**Steps:**

1. **Create a New Java Project:**
   * Create a new Java project named **SingletonPatternExample**.
2. **Define a Singleton Class:**
   * Create a class named Logger that has a private static instance of itself.
   * Ensure the constructor of Logger is private.
   * Provide a public static method to get the instance of the Logger class.
3. **Implement the Singleton Pattern:**
   * Write code to ensure that the Logger class follows the Singleton design pattern.
4. **Test the Singleton Implementation:**
   * Create a test class to verify that only one instance of Logger is created and used across the application.

**SOLUTION :**

**Logger.java :**

public class Logger {

private static Logger logger;

private Logger(){

System.out.println("Logger Constructor called");

}

public static Logger getInstance(){

if(logger == null){

logger = new Logger();

}

return logger;

}

public void log(String msg){

System.out.println("Log : " + msg);

}

}

**Main.java :**

public class Main {

public static void main(String[] args) {

Logger log1 = Logger.getInstance();

Logger log2 = Logger.getInstance();

log1.log("message from log 1");

log2.log("message from log 2");

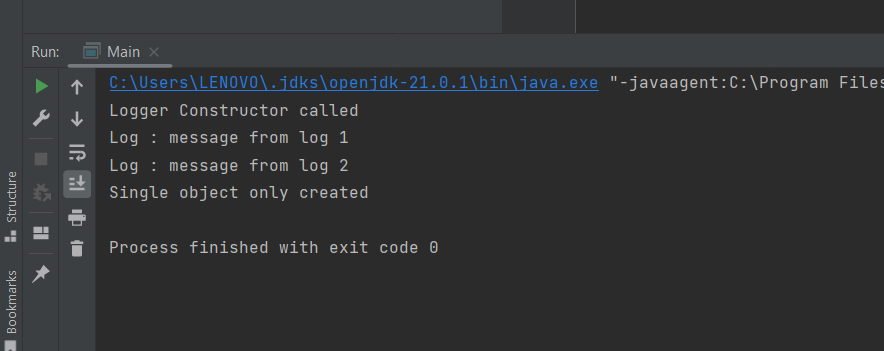
if(log1 == log2)

System.out.println("Single object only created");

}

}

**OUTPUT :**

****

**FACTORY METHOD PATTERN**

**QUESTION – 2 :**

Scenario:

You are developing a document management system that needs to create different types of documents (e.g., Word, PDF, Excel). Use the Factory Method Pattern to achieve this.

Steps:

1. Create a New Java Project:
   * Create a new Java project named FactoryMethodPatternExample.
2. Define Document Classes:
   * Create interfaces or abstract classes for different document types such as WordDocument, PdfDocument, and ExcelDocument.
3. Create Concrete Document Classes:
   * Implement concrete classes for each document type that implements or extends the above interfaces or abstract classes.
4. Implement the Factory Method:
   * Create an abstract class DocumentFactory with a method createDocument().
   * Create concrete factory classes for each document type that extends DocumentFactory and implements the createDocument() method.
5. Test the Factory Method Implementation:

Create a test class to demonstrate the creation of different document types using the factory method.

**SOLUTION :**

**Main.java :**

package FactoryPattern;

public class Main {

public static void main(String[] args) {

DocumentFactory wordFactory = new WordFactory();

Document wordDoc = wordFactory.createDocument();

wordDoc.open();

DocumentFactory pdfFactory = new PdfFactory();

Document pdfDoc = pdfFactory.createDocument();

pdfDoc.open();

DocumentFactory excelFactory = new ExcelFactory();

Document excelDoc = excelFactory.createDocument();

excelDoc.open();

}

}

**Document.java :**

package FactoryPattern;

public interface Document {

void open();

}

**DocumentFactory.java :**

package FactoryPattern;

public abstract class DocumentFactory {

public abstract Document createDocument();

}

**ExcelDocument.java :**

package FactoryPattern;

public class ExcelDocument implements Document {

public void open() {

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

}

}

**ExcelFactory.java :**

package FactoryPattern;

public class ExcelFactory extends DocumentFactory {

public Document createDocument() {

return new ExcelDocument();

}

}

**PdfDocument.java :**

package FactoryPattern;

public class PdfDocument implements Document {

public void open() {

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

}

}

**PdfFactory .java :**

package FactoryPattern;

public class PdfFactory extends DocumentFactory {

public Document createDocument() {

return new PdfDocument();

}

}

WordDocument.java :

package FactoryPattern;

public class WordDocument implements Document {

public void open() {

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

}

}

**WordFactory.java :**

package FactoryPattern;

public class WordFactory extends DocumentFactory {

public Document createDocument() {

return new WordDocument();

}

}

**OUTPUT :**

