**Week1\_DesignPrinciplesAndPatterns\_HandsOn**

**Exercise 1: Implementing the Singleton Pattern**

**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.

**CODE:**

**Logger.java**

public class Main {

public static void main(String[] args) {

Logger logger1 = Logger.getInstance();

Logger logger2 = Logger.getInstance();

System.out.println("Are logger1 and logger2 the same instance? " + (logger1 == logger2));

logger1.log("Hey guys!!");

logger2.log("This is a test message.");

System.out.println("Hash code of logger1: " + logger1.hashCode());

System.out.println("Hash code of logger2: " + logger2.hashCode());

}

}

class Logger {

private static Logger instance;

private Logger() {}

public static Logger getInstance() {

if (instance == null) {

synchronized (Logger.class) {

if (instance == null) {

instance = new Logger();

}

}}

return instance;

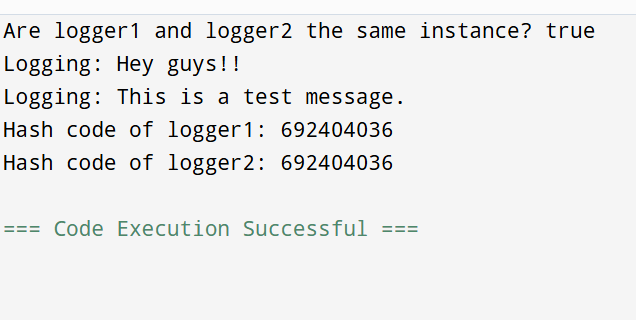
}

public void log(String message) {

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

}}

**OUTPUT:**



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

**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.

**CODE:**

**Document.java**

public interface Document {

void open();

void save();

void close();

}

**WordDocument.java**

public class WordDocument implements Document {

@Override

public void open() {

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

}

@Override

public void save() {

System.out.println("Saving Word document...");

}

@Override

public void close() {

System.out.println("Closing Word document...");

}

}

**PdfDocument.java**

public class PdfDocument implements Document {

@Override

public void open() {

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

}

@Override

public void save() {

System.out.println("Saving PDF document...");

}

@Override

public void close() {

System.out.println("Closing PDF document...");

}

}

**ExcelDocument.java**

public class ExcelDocument implements Document {

@Override

public void open() {

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

}

@Override

public void save() {

System.out.println("Saving Excel document...");

}

@Override

public void close() {

System.out.println("Closing Excel document...");

}

}

**DocumentFactory.java**

public abstract class DocumentFactory {

public abstract Document createDocument();

public void processDocument() {

Document document = createDocument();

document.open();

document.save();

document.close();

}}

**WordDocumentFactory.java**

public class WordDocumentFactory extends DocumentFactory {

@Override

public Document createDocument() {

return new WordDocument();

}

}

**PdfDocumentFactory.java**

public class PdfDocumentFactory extends DocumentFactory {

@Override

public Document createDocument() {

return new PdfDocument();

}

}

**ExcelDocumentFactory.java**

public class ExcelDocumentFactory extends DocumentFactory {

@Override

public Document createDocument() {

return new ExcelDocument();

}

}

**DocumentFactoryTest.java**

public class DocumentFactoryTest {

public static void main(String[] args) {

DocumentFactory wordFactory = new WordDocumentFactory();

wordFactory.processDocument();

public class Main {

public static void main(String[] args) {

DocumentFactory wordFactory = new WordDocumentFactory();

wordFactory.processDocument();

DocumentFactory pdfFactory = new PdfDocumentFactory();

pdfFactory.processDocument();

DocumentFactory excelFactory = new ExcelDocumentFactory();

excelFactory.processDocument();

}

}

interface Document {

void open();

void save();

void close();

}

// --- Concrete Document Implementations ---

class WordDocument implements Document {

@Override

public void open() {

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

}

@Override

public void save() {

System.out.println("Saving Word document...");

}

@Override

public void close() {

System.out.println("Closing Word document...");

}

}

class PdfDocument implements Document {

@Override

public void open() {

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

}

@Override

public void save() {

System.out.println("Saving PDF document...");

}

@Override

public void close() {

System.out.println("Closing PDF document...");

}

}

class ExcelDocument implements Document {

@Override

public void open() {

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

}

@Override

public void save() {

System.out.println("Saving Excel document...");

}

@Override

public void close() {

System.out.println("Closing Excel document...");

}

}

// --- Abstract Factory ---

abstract class DocumentFactory {

public abstract Document createDocument();

public void processDocument() {

Document document = createDocument();

document.open();

document.save();

document.close();

}

}

// --- Concrete Factories ---

class WordDocumentFactory extends DocumentFactory {

@Override

public Document createDocument() {

return new WordDocument();

}

}

class PdfDocumentFactory extends DocumentFactory {

@Override

public Document createDocument() {

return new PdfDocument();

}

}

class ExcelDocumentFactory extends DocumentFactory {

@Override

public Document createDocument() {

return new ExcelDocument();

}

}

DocumentFactory pdfFactory = new PdfDocumentFactory();

pdfFactory.processDocument();

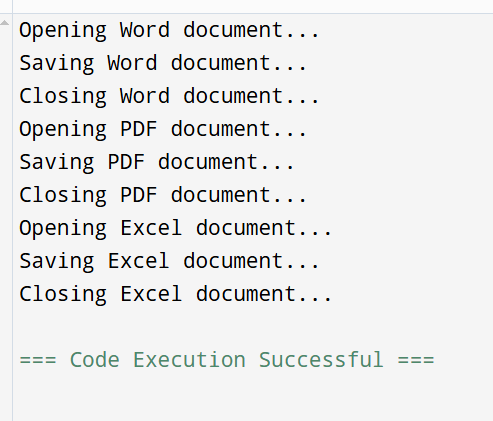
DocumentFactory excelFactory = new ExcelDocumentFactory();

excelFactory.processDocument();

}

}

**OUTPUT:**

****