Design Patterns and Principles

Exercise 1:

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: o 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. o 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: o Create a test class to verify that only one instance of Logger is created and used across the application.

Define and Implement the Singleton Class

package singleton;

public class Logger {

// Step 1: private static instance of Logger

private static Logger instance;

// Step 2: private constructor to prevent instantiation

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);

}

}

Test the Singleton Implementation

package singleton;

public class TestSingleton {

public static void main(String[] args) {

Logger logger1 = Logger.getInstance();

logger1.log("First log message");

Logger logger2 = Logger.getInstance();

logger2.log("Second log message");

// Check if both instances are the same

if (logger1 == logger2) {

System.out.println("Both logger instances are the same (singleton works).");

} else {

System.out.println("Logger instances are different (singleton failed).");

}

}

}

Output:

Logger initialized.

LOG: First log message

LOG: Second log message

Both logger instances are the same (singleton works).

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: o Create interfaces or abstract classes for different document types such as WordDocument, PdfDocument, and ExcelDocument.

3. Create Concrete Document Classes: o 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: o Create a test class to demonstrate the creation of different document types using the factory method.

Step 1: Create a New Java Project

FactoryMethodPatternExample

Step 2: Define Document Interface

File: Document.java

package factory;

public interface Document {

void open();

}

Step 3: Create Concrete Document Classes

File: WordDocument.java

package factory;

public class WordDocument implements Document {

public void open() {

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

}

}

**File: PdfDocument.java**

package factory;

public class PdfDocument implements Document {

public void open() {

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

}

}

**File: ExcelDocument.java**

package factory;

public class ExcelDocument implements Document {

public void open() {

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

}

}

**Step 4: Implement the Factory Method**

**File: DocumentFactory.java**

package factory;

public abstract class DocumentFactory {

public abstract Document createDocument();

}

File: WordDocumentFactory.java

package factory;

public class WordDocumentFactory extends DocumentFactory {

public Document createDocument() {

return new WordDocument();

}

}

File: PdfDocumentFactory.java

package factory;

public class PdfDocumentFactory extends DocumentFactory {

public Document createDocument() {

return new PdfDocument();

}

}

File: ExcelDocumentFactory.java

package factory;

public class ExcelDocumentFactory extends DocumentFactory {

public Document createDocument() {

return new ExcelDocument();

}

}

**Step 5: Test the Factory Method Implementation**

**File: TestFactoryMethod.java**

package factory;

public class TestFactoryMethod {

public static void main(String[] args) {

DocumentFactory wordFactory = new WordDocumentFactory();

Document wordDoc = wordFactory.createDocument();

wordDoc.open();

DocumentFactory pdfFactory = new PdfDocumentFactory();

Document pdfDoc = pdfFactory.createDocument();

pdfDoc.open();

DocumentFactory excelFactory = new ExcelDocumentFactory();

Document excelDoc = excelFactory.createDocument();

excelDoc.open();

}

}

Output:

Opening Word Document...

Opening PDF Document...

Opening Excel Document...