**Exercise 1: Implementing the Singleton Pattern**

***Code*:**

**Logger.java**

package SingletonPatternExample;

public class Logger {

// Step 1: Private static instance

private static Logger instance;

// Step 2: Private constructor to prevent instantiation

private Logger() {

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

}

// Step 3: Public method to provide access to the instance

public static Logger getInstance() {

if (instance == null) {

instance = new Logger();

}

return instance;

}

// Example logging method

public void log(String message) {

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

}

}

**SingletonTest.java**

package SingletonPatternExample;

public class SingletonTest {

public static void main(String[] args) {

// Get Logger instance multiple times

Logger logger1 = Logger.getInstance();

Logger logger2 = Logger.getInstance();

// Log messages

logger1.log("First message by Logger1");

logger2.log("Second message by Logger2");

// Check if both instances are the same

if (logger1 == logger2) {

System.out.println("Only one Logger instance exists (Singleton verified).");

} else {

System.out.println("Different Logger instances exist (Singleton failed).");

}

// Check if both instances are the same by hash code

Logger logger3 = Logger.getInstance();

logger3.log("Third message by Logger3");

System.out.println("HashCode logger1: " + logger1.hashCode());

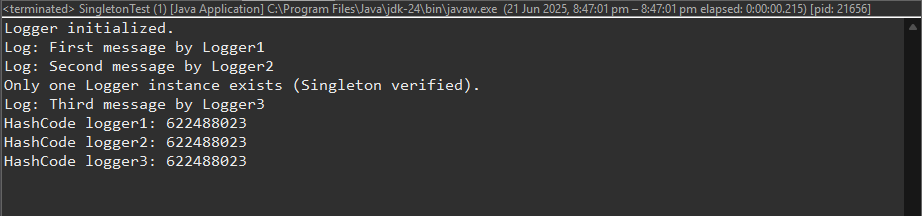
System.out.println("HashCode logger2: " + logger2.hashCode());

System.out.println("HashCode logger3: " + logger3.hashCode());

}

}

***Output*:**

****

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

***Code*:**

**Document.java**

// Document interface defines the basic operations for all document types

public interface Document {

// Open the document

void open();

// Save the document

void save();

// Close the document

void close();

}

**ExcelDocument.java**

// ExcelDocument implements the Document interface for Excel files

public class ExcelDocument implements Document {

*@Override*

public void open() {

// opening method

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

}

*@Override*

public void save() {

// saving method

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

}

*@Override*

public void close() {

// closing method

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

}

}

**PdfDocument.java**

// PdfDocument implements the Document interface for PDF files

public class PdfDocument implements Document {

*@Override*

public void open() {

// opening method

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

}

*@Override*

public void save() {

// saving method

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

}

*@Override*

public void close() {

// closing method

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

}

}

**WordDocument.java**

// WordDocument implements the Document interface for Word files

public class WordDocument implements Document {

*@Override*

public void open() {

// opening method

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

}

*@Override*

public void save() {

// saving method

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

}

*@Override*

public void close() {

// closing method

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

}

}

**DocumentFactory.java**

// Abstract factory for creating documents

public abstract class DocumentFactory {

// Factory method to create a document

public abstract Document createDocument();

}

**ExcelDocumentFactory.java**

// Factory for creating ExcelDocument instances

public class ExcelDocumentFactory extends DocumentFactory {

*@Override*

public Document createDocument() {

// Returns a new ExcelDocument

return new ExcelDocument();

}

}

**PdfDocumentFactory.java**

// Factory for creating PdfDocument instances

public class PdfDocumentFactory extends DocumentFactory {

*@Override*

public Document createDocument() {

// Returns a new PdfDocument

return new PdfDocument();

}

}

**WordDocumentFactory.java**

// Factory for creating WordDocument instances

public class WordDocumentFactory extends DocumentFactory {

*@Override*

public Document createDocument() {

// Returns a new WordDocument

return new WordDocument();

}

}

**FactoryMethodPatternDemo.java**

public class FactoryMethodPatternDemo {

public static void main(String[] args) {

DocumentFactory wordFactory = new WordDocumentFactory();

Document wordDoc = wordFactory.createDocument();

wordDoc.open();

wordDoc.save();

wordDoc.close();

DocumentFactory pdfFactory = new PdfDocumentFactory();

Document pdfDoc = pdfFactory.createDocument();

pdfDoc.open();

pdfDoc.save();

pdfDoc.close();

DocumentFactory excelFactory = new ExcelDocumentFactory();

Document excelDoc = excelFactory.createDocument();

excelDoc.open();

excelDoc.save();

excelDoc.close();

}

}

***Output:***

A screen shot of a computer

AI-generated content may be incorrect.