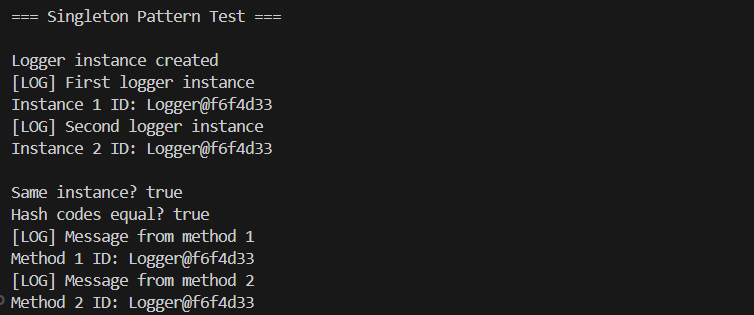
# 1. SingletonPatternExample.java

class Logger {  
 private static Logger inst = null;  
 private Logger() {  
 System.out.println("Logger instance created");  
 }  
   
 public static synchronized Logger getInstance() {  
 if (inst == null) {  
 inst = new Logger();  
 }  
 return inst;  
 }  
   
 public void log(String msg) {  
 System.out.println("[LOG] " + msg);  
 }  
   
 public String getInstanceId() {  
 return "Logger@" + Integer.toHexString(hashCode());  
 }  
}  
  
public class SingletonPatternExample {  
 public static void main(String[] args) {  
 System.out.println("=== Singleton Pattern Test ===\n");  
   
 Logger l1 = Logger.getInstance();  
 l1.log("First logger instance");  
 System.out.println("Instance 1 ID: " + l1.getInstanceId());  
   
 Logger l2 = Logger.getInstance();  
 l2.log("Second logger instance");  
 System.out.println("Instance 2 ID: " + l2.getInstanceId());  
   
 System.out.println("\nSame instance? " + (l1 == l2));  
 System.out.println("Hash codes equal? " + (l1.hashCode() == l2.hashCode()));  
   
 testFromMethod1();  
 testFromMethod2();  
 }  
   
 private static void testFromMethod1() {  
 Logger l = Logger.getInstance();  
 l.log("Message from method 1");  
 System.out.println("Method 1 ID: " + l.getInstanceId());  
 }  
   
 private static void testFromMethod2() {  
 Logger l = Logger.getInstance();  
 l.log("Message from method 2");  
 System.out.println("Method 2 ID: " + l.getInstanceId());  
 }  
}

Output:



# 2. FactoryMethodPatternExample.java

interface Document {  
 void open();  
 void save();  
 void close();  
}  
  
class WordDocument implements Document {  
 public void open() {  
 System.out.println("Opening Word document");  
 }  
   
 public void save() {  
 System.out.println("Saving Word document");  
 }  
   
 public void close() {  
 System.out.println("Closing Word document");  
 }  
}  
  
class PdfDocument implements Document {  
 public void open() {  
 System.out.println("Opening PDF document");  
 }  
   
 public void save() {  
 System.out.println("Saving PDF document");  
 }  
   
 public void close() {  
 System.out.println("Closing PDF document");  
 }  
}  
  
class ExcelDocument implements Document {  
 public void open() {  
 System.out.println("Opening Excel document");  
 }  
   
 public void save() {  
 System.out.println("Saving Excel document");  
 }  
   
 public void close() {  
 System.out.println("Closing Excel document");  
 }  
}  
  
abstract class DocumentFactory {  
 public abstract Document createDocument();  
}  
  
class WordFactory extends DocumentFactory {  
 public Document createDocument() {  
 return new WordDocument();  
 }  
}  
  
class PdfFactory extends DocumentFactory {  
 public Document createDocument() {  
 return new PdfDocument();  
 }  
}  
  
class ExcelFactory extends DocumentFactory {  
 public Document createDocument() {  
 return new ExcelDocument();  
 }  
}  
  
public class FactoryMethodPatternExample {  
 public static void main(String[] args) {  
 System.out.println("Factory Method Pattern Demo\n");  
   
 DocumentFactory wf = new WordFactory();  
 Document w = wf.createDocument();  
 w.open();  
 w.save();  
 w.close();  
   
 System.out.println();  
   
 DocumentFactory pf = new PdfFactory();  
 Document p = pf.createDocument();  
 p.open();  
 p.save();  
 p.close();  
   
 System.out.println();  
   
 DocumentFactory ef = new ExcelFactory();  
 Document e = ef.createDocument();  
 e.open();  
 e.save();  
 e.close();  
   
 System.out.println();  
   
 testDifferentWay();  
 }  
   
 static void testDifferentWay() {  
 String[] types = {"word", "pdf", "excel"};  
   
 for (String t : types) {  
 DocumentFactory f = getFactory(t);  
 Document d = f.createDocument();  
 d.open();  
 d.save();  
 d.close();  
 System.out.println();  
 }  
 }  
   
 static DocumentFactory getFactory(String type) {  
 if (type.equals("word")) {  
 return new WordFactory();  
 } else if (type.equals("pdf")) {  
 return new PdfFactory();  
 } else if (type.equals("excel")) {  
 return new ExcelFactory();  
 }  
 return null;  
 }  
}

Output:

A screenshot of a computer

AI-generated content may be incorrect.