# DESIGN PATTERN AND PRINCIPLE - FACTORY METHOD PATTERN IMPLEMENTATION

## Project Structure

Create a Java project named FactoryMethodPatternExample with the following structure:  
  
FactoryMethodPatternExample/  
├── src/  
│ ├── documents/  
│ │ ├── Document.java  
│ │ ├── WordDocument.java  
│ │ ├── PdfDocument.java  
│ │ └── ExcelDocument.java  
│ ├── factories/  
│ │ ├── DocumentFactory.java  
│ │ ├── WordDocumentFactory.java  
│ │ ├── PdfDocumentFactory.java  
│ │ └── ExcelDocumentFactory.java  
│ └── Main.java

## 1. Document Interface (documents/Document.java)

package documents;  
  
public interface Document {  
 void open();  
 void save();  
 void close();  
}

## 2. Concrete Document Classes

### WordDocument.java

package documents;  
  
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 (.docx)");  
 }  
  
 @Override  
 public void close() {  
 System.out.println("Closing Word document");  
 }  
   
 @Override  
 public String toString() {  
 return "Microsoft Word Document";  
 }  
}

### PdfDocument.java

package documents;  
  
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 (.pdf)");  
 }  
  
 @Override  
 public void close() {  
 System.out.println("Closing PDF document");  
 }  
   
 @Override  
 public String toString() {  
 return "Adobe PDF Document";  
 }  
}

### ExcelDocument.java

package documents;  
  
public class ExcelDocument implements Document {  
 @Override  
 public void open() {  
 System.out.println("Opening Excel spreadsheet...");  
 }  
  
 @Override  
 public void save() {  
 System.out.println("Saving Excel workbook (.xlsx)");  
 }  
  
 @Override  
 public void close() {  
 System.out.println("Closing Excel spreadsheet");  
 }  
   
 @Override  
 public String toString() {  
 return "Microsoft Excel Workbook";  
 }  
}

## 3. Factory Classes

### DocumentFactory.java (Abstract Creator)

package factories;  
  
import documents.Document;  
  
public abstract class DocumentFactory {  
 public abstract Document createDocument();  
  
 public void processDocument() {  
 Document doc = createDocument();  
 System.out.println("\nProcessing " + doc.getClass().getSimpleName());  
 doc.open();  
 doc.save();  
 doc.close();  
 }  
}

### WordDocumentFactory.java

package factories;  
  
import documents.Document;  
import documents.WordDocument;  
  
public class WordDocumentFactory extends DocumentFactory {  
 @Override  
 public Document createDocument() {  
 return new WordDocument();  
 }  
}

### PdfDocumentFactory.java

package factories;  
  
import documents.Document;  
import documents.PdfDocument;  
  
public class PdfDocumentFactory extends DocumentFactory {  
 @Override  
 public Document createDocument() {  
 return new PdfDocument();  
 }  
}

### ExcelDocumentFactory.java

package factories;  
  
import documents.Document;  
import documents.ExcelDocument;  
  
public class ExcelDocumentFactory extends DocumentFactory {  
 @Override  
 public Document createDocument() {  
 return new ExcelDocument();  
 }  
}

## 4. Test Class (Main.java)

import documents.Document;  
import factories.\*;  
  
public class Main {  
 public static void main(String[] args) {  
 System.out.println("DOCUMENT MANAGEMENT SYSTEM\n");  
  
 DocumentFactory wordFactory = new WordDocumentFactory();  
 DocumentFactory pdfFactory = new PdfDocumentFactory();  
 DocumentFactory excelFactory = new ExcelDocumentFactory();  
  
 System.out.println("--- Creating Word Document ---");  
 Document wordDoc = wordFactory.createDocument();  
 System.out.println("Created: " + wordDoc);  
 wordFactory.processDocument();  
  
 System.out.println("\n--- Creating PDF Document ---");  
 Document pdfDoc = pdfFactory.createDocument();  
 System.out.println("Created: " + pdfDoc);  
 pdfFactory.processDocument();  
  
 System.out.println("\n--- Creating Excel Document ---");  
 Document excelDoc = excelFactory.createDocument();  
 System.out.println("Created: " + excelDoc);  
 excelFactory.processDocument();  
  
 System.out.println("\n=== Document Type Verification ===");  
 System.out.println("WordDoc is WordDocument: " + (wordDoc instanceof documents.WordDocument));  
 System.out.println("PdfDoc is PdfDocument: " + (pdfDoc instanceof documents.PdfDocument));  
 System.out.println("ExcelDoc is ExcelDocument: " + (excelDoc instanceof documents.ExcelDocument));  
 }  
}

## 5. Expected Output

DOCUMENT MANAGEMENT SYSTEM  
  
--- Creating Word Document ---  
Created: Microsoft Word Document  
  
Processing WordDocument  
Opening Word document...  
Saving Word document (.docx)  
Closing Word document  
  
--- Creating PDF Document ---  
Created: Adobe PDF Document  
  
Processing PdfDocument  
Opening PDF document...  
Saving PDF document (.pdf)  
Closing PDF document  
  
--- Creating Excel Document ---  
Created: Microsoft Excel Workbook  
  
Processing ExcelDocument  
Opening Excel spreadsheet...  
Saving Excel workbook (.xlsx)  
Closing Excel spreadsheet  
  
=== Document Type Verification ===  
WordDoc is WordDocument: true  
PdfDoc is PdfDocument: true  
ExcelDoc is ExcelDocument: true

## 6. Pattern Implementation Notes

Key Components:  
- Product (Document): Interface defining document operations  
- Concrete Products (Word/Pdf/ExcelDocument): Actual document implementations  
- Creator (DocumentFactory): Abstract factory with template method  
- Concrete Creators: Factories specializing in specific document types  
  
Advantages:  
1. Decoupling: Client code works with abstract interfaces, not concrete classes  
2. Extensibility: Add new document types without modifying existing code  
3. Single Responsibility: Each factory handles one product type  
4. Consistent Workflow: Template method ensures uniform processing

## 7. Potential Extensions

1. Document Properties:  
 public interface Document {  
 void setAuthor(String author);  
 void setTitle(String title);  
 }  
  
2. Factory Registry:  
 public class DocumentFactoryRegistry {  
 private static Map<String, DocumentFactory> factories = new HashMap<>();  
  
 static {  
 register("word", new WordDocumentFactory());  
 register("pdf", new PdfDocumentFactory());  
 }  
  
 public static Document createDocument(String type) {  
 DocumentFactory factory = factories.get(type.toLowerCase());  
 if (factory != null) return factory.createDocument();  
 throw new IllegalArgumentException("Unknown document type");  
 }  
 }  
  
3. Enhanced Processing:  
 public abstract class DocumentFactory {  
 public void fullProcess() {  
 Document doc = createDocument();  
 doc.open();  
 // Add editing, formatting, etc.  
 doc.save();  
 doc.close();  
 }  
 }