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

**Code:**

// 1. Product Interface

interface Document {

void open();

void save();

void close();

}

// 2. Concrete Products

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.");

}

}

// 3. Creator (Abstract Factory)

abstract class DocumentCreator {

// The factory method - returns a Document

public abstract Document createDocument();

// Other operations that use the created document

public void processDocument() {

Document doc = createDocument(); // Call the factory method

doc.open();

doc.save();

doc.close();

System.out.println("Document processed successfully.\n");

}

}

// 4. Concrete Creators

class WordDocumentCreator extends DocumentCreator {

@Override

public Document createDocument() {

return new WordDocument();

}

}

class PdfDocumentCreator extends DocumentCreator {

@Override

public Document createDocument() {

return new PdfDocument();

}

}

// 5. Client Code

public class FactoryMethodExample {

public static void main(String[] args) {

// Create a creator for Word documents and process one

DocumentCreator wordCreator = new WordDocumentCreator();

wordCreator.processDocument();

// Create a creator for PDF documents and process one

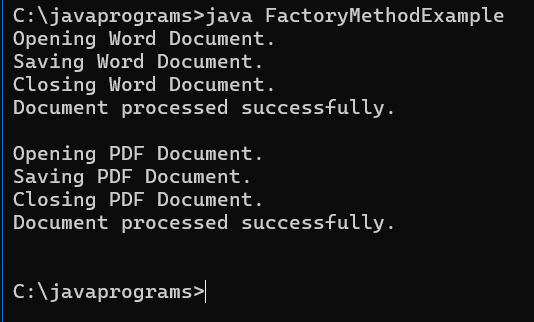
DocumentCreator pdfCreator = new PdfDocumentCreator();

pdfCreator.processDocument();

}

}

**Output:**

****