**Exercise 2 – Implementation of Factory Method Design Pattern**

**What is Factory Method Pattern?**

* The Factory Method Pattern is a **Creational Pattern.**
* The Factory Method Pattern defines an interface for creating an object, but it lets the subclasses decide which class to instantiate. It lets a class defer instantiation to its subclasses.

**Key Principles:**

1. Factory method – method that creates and returns an object.
2. Subclass responsibility – subclass override the factory method to specify the type.
3. Loose Coupling – client code depends only on abstract types but not on specific implementations.

**Objective:**

Defines an interface for creating objects, but allow subclasses to decide which class to instantiate. It supports **open/closed principle** by delegating object creation to subclasses rather than instantiating directly in client code.

**Steps for Implementing Factory Method Design Pattern:**

**Step 1:** Create a new Java Project

* Name of the Project: **FactoryMethodPattern**
* The project folder is as follows:

designpatterns

├──FactoryMethodPattern/

├── Document.java

├── WordDocument.java

├── PdfDocument.java

├── DocumentCreator.java

├── WordDocumentCreator.java

├── PdfDocumentCreator.java

├── ExcelDocument.java

├── ExcelDocumentCreator.java

└── Main.java

**Step 2:** Defining Document interface

A common interface all type of documents. Each document should have open () method.

|  |
| --- |
| interface Document {  void open (); // Every document must be able to open  } |

**Step 3:** Create concrete Document Classes

The specific document classes:

* WordDocument
* PdfDocument
* ExcelDocument

**WordDocument.java**

|  |
| --- |
| class WordDocument implements Document {  public void open () {  System.out.println("Opening a Word document...");  }  } |

**PdfDocument.java**

|  |
| --- |
| class PdfDocument implements Document {  public void open () {  System.out.println("Opening a PDF document...");  }  } |

**ExcelDocument.java**

|  |
| --- |
| class ExcelDocument implements Document {  public void open () {  System.out.println("Opening an Excel document...");  }  } |

**Step 4:** Implementation of Factory Method

* Define an abstract class **DocCreator** with the factory method **createDoc()**.
* Each concrete subclass will follow this method implementation.

**DocCreator.java**

|  |
| --- |
| //a manager class to handle document creation  abstract class DocCreator{  // This is the "factory method" that will create the document  public abstract Document createDoc();  // This method uses the created document  public void openDocument() {  Document doc = createDoc(); // ask subclass to create document  doc.open();  }  } |

**WordDocumentCreator.java**

|  |
| --- |
| //creates Word documents  class WordDocumentCreator extends DocCreator{  public Document createDoc() {  return new WordDocument();  }  } |

**PdfDocumentCreator.java**

|  |
| --- |
| // creates PDF documents  class PdfDocumentCreator extends DocCreator {  public Document createDoc() {  return new PdfDocument();  }  } |

**ExcelDocumentCreator.java**

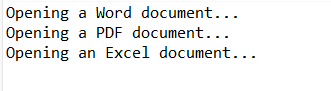
|  |
| --- |
| // creates excel documents  class ExcelDocumentCreator extends DocCreator {  public Document createDoc() {  return new ExcelDocument();  }  } |

**Step 5:** Testing the Factory method

**Main.java**

|  |
| --- |
| public class Main {  public static void main (String[] args) {  DocCreator creator;  creator = new WordDocumentCreator();  creator.openDocument();  creator = new PdfDocumentCreator();  creator.openDocument();  creator = new ExcelDocumentCreator();  creator.openDocument();  }  } |

**Expected Output:**

****