Design principles & Patterns

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

**Document.java**

package com.factory.example;

public interface Document {

void open();

}

**ExcelDocument.java**

package com.factory.example;

public class ExcelDocument implements Document{

@Override

public void open()

{

System.*out*.println("Open Excel");

}

}

**PdfDocument,java**

package com.factory.example;

public class PdfDocument implements Document{

@Override

public void open()

{

System.*out*.println("Open Pdf");

}

}

**WordDocument.java**

package com.factory.example;

public class WordDocument implements Document{

@Override

public void open()

{

System.*out*.println("Open Word");

}

}

**DocumentFactory.java**

package com.factory.example;

public abstract class DocumentFactory {

public abstract Document CreateDocument();

}

**ExcelDocumentFactoty.java**

package com.factory.example;

public class ExcelDocumentFactory extends DocumentFactory{

@Override

public Document CreateDocument()

{

return new ExcelDocument();

}

}

**PdfDocumentFacotory.java**

package com.factory.example;

public class PdfDocumentFactory extends DocumentFactory{

@Override

public Document CreateDocument()

{

return new PdfDocument();

}

}

**WordDocumentFactory.java**

package com.factory.example;

public class WordDocumentFactory extends DocumentFactory{

@Override

public Document CreateDocument()

{

return new WordDocument();

}

}

**Main.java**

package com.factory.example;

class Main {

public static void main(String args[])

{

DocumentFactory word=new WordDocumentFactory();

Document wd=word.CreateDocument();

wd.open();

DocumentFactory pdf=new PdfDocumentFactory();

Document pd=pdf.CreateDocument();

pd.open();

DocumentFactory excel=new ExcelDocumentFactory();

Document exc=excel.CreateDocument();

exc.open();

}

}

