

Exercise 2: Implementing the Factory Method Pattern

Document.java:

```
package week1.FactoryMethodPatternExample;

public interface Document {

    void open();

}
```

WordDocument.java:

```
package week1.FactoryMethodPatternExample;

public class WordDocument implements Document {

    @Override

    public void open() {

        System.out.println("Opening Word Document.");

    }

}
```

ExcelDocument.java:

```
package week1.FactoryMethodPatternExample;

public class ExcelDocument implements Document {

    @Override

    public void open() {

        System.out.println("Opening Excel Document.");

    }

}
```

PdfDocument.java:

```
package week1.FactoryMethodPatternExample;

public class PdfDocument implements Document {

    @Override

    public void open() {

        System.out.println("Opening PDF Document.");

    }

}
```

DocumentFactory.java:

```
package week1.FactoryMethodPatternExample;

public class WordDocumentFactory extends DocumentFactory {

    @Override

    public Document createDocument() {

        return new WordDocument();

    }

}
```

ExcelDocumentFactory.java:

```
package week1.FactoryMethodPatternExample;

public class ExcelDocumentFactory extends DocumentFactory {

    @Override

    public Document createDocument() {

        return new ExcelDocument();

    }

}
```

PdfDocumentFactory.java:

```
package week1.FactoryMethodPatternExample;

public class PdfDocumentFactory extends DocumentFactory {

    @Override

    public Document createDocument() {

        return new PdfDocument();

    }

}
```

WordDocumentFactory.java:

```
package week1.FactoryMethodPatternExample;

public class WordDocumentFactory extends DocumentFactory {

    @Override

    public Document createDocument() {

        return new WordDocument();

    }

}
```

DocumentTest.java

```
package week1.FactoryMethodPatternExample;

public class DocumentTest {

    public static void main(String[] args) {

        DocumentFactory wordFactory=new WordDocumentFactory();

        Document word=wordFactory.createDocument();

        word.open();


        DocumentFactory pdfFactory=new PdfDocumentFactory();

        Document pdf=pdfFactory.createDocument();

        pdf.open();


        DocumentFactory excelFactory=new ExcelDocumentFactory();

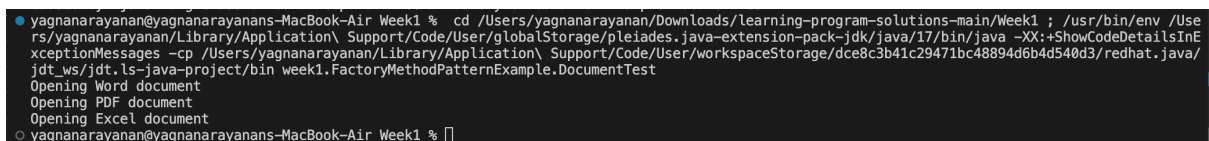
        Document excel=excelFactory.createDocument();

        excel.open();

    }

}
```

Output:

A terminal window screenshot showing the execution of the Java program. The prompt is 'yagnanarayanan@yagnanarayanans-MacBook-Air Week1 %'. The command entered is 'cd /Users/yagnanarayanan/Downloads/learning-program-solutions-main/Week1 ; /usr/bin/env /Use'. The output shows the program running successfully, printing 'Opening Word document', 'Opening PDF document', and 'Opening Excel document' on separate lines. The prompt returns as 'yagnanarayanan@yagnanarayanans-MacBook-Air Week1 %' with a cursor.

```
yagnanarayanan@yagnanarayanans-MacBook-Air Week1 % cd /Users/yagnanarayanan/Downloads/learning-program-solutions-main/Week1 ; /usr/bin/env /Use
rs/yagnanarayanan/Library/Application\ Support/Code/User/globalStorage/pleiades.java-extension-pack-jdk/java/17/bin/java -XX:+ShowCodeDetailsInE
xceptionMessages -cp /Users/yagnanarayanan/Library/Application\ Support/Code/User/workspaceStorage/dce8c3b41c29471bc48894d6b4d540d3/redhat.java/
jdt_ws/jdt.ls-java-project/bin week1.FactoryMethodPatternExample.DocumentTest
Opening Word document
Opening PDF document
Opening Excel document
yagnanarayanan@yagnanarayanans-MacBook-Air Week1 %
```