

EXERCISE 2: IMPLEMENTING THE FACTORY METHOD PATTERN

```
// Document interface
interface Document {
    void open();
}

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

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

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

// Abstract Factory
abstract class DocumentFactory
{
    public abstract Document
createDocument();
}
```

```
// Concrete Factories
class WordFactory extends
DocumentFactory {
    public Document
createDocument() {
        return new
WordDocument();
    }
}

class PdfFactory extends
DocumentFactory {
    public Document
createDocument() {
        return new PdfDocument();
    }
}


class ExcelFactory extends
DocumentFactory {
    public Document
createDocument() {
        return new
ExcelDocument();
    }
}

// Main test class
public class Main {
    public static void
main(String[] args) {
        DocumentFactory
wordFactory = new
WordFactory();
        Document wordDoc =
wordFactory.createDocument();
        wordDoc.open();

        DocumentFactory
pdfFactory = new PdfFactory();
        Document pdfDoc =
pdfFactory.createDocument();
        pdfDoc.open();

        DocumentFactory
excelFactory = new
ExcelFactory();
        Document excelDoc =
excelFactory.createDocument();
        excelDoc.open();
    }
}
```

USING JDoodle COMPILER



```
Output  Generated files

Opening a Word document.
Opening a PDF document.
Opening an Excel document.
|

Compiled and executed in 1.972 sec(s)
```