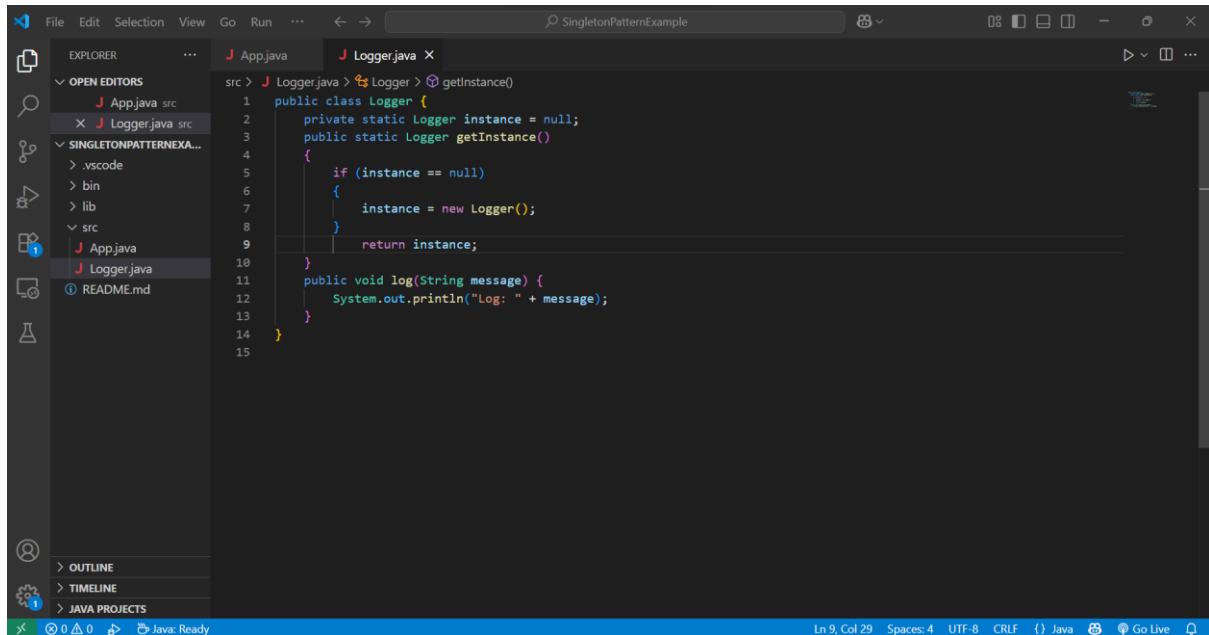


# DESIGN PRINCIPLES AND PATTERNS

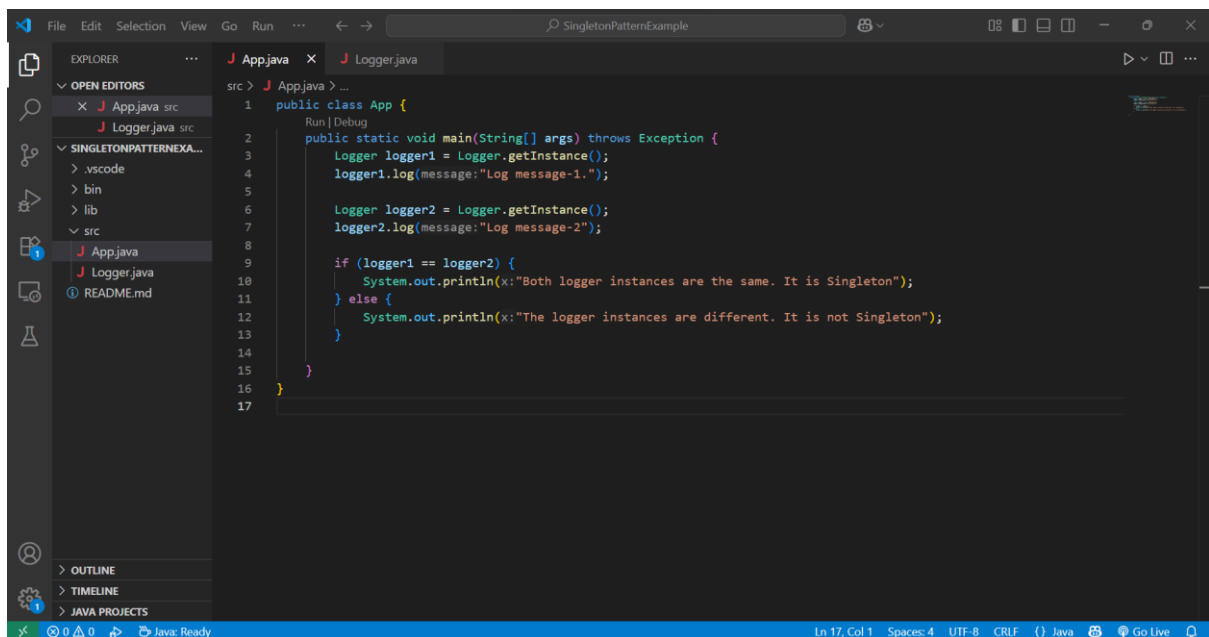
## EX 1 : IMPLEMENTING THE SINGLETON PATTERN

CODING:



The screenshot shows the Visual Studio Code editor with the file Explorer on the left. The Explorer shows a project named 'SingletonPatternExample' with a 'src' folder containing 'App.java' and 'Logger.java'. The 'Logger.java' file is open in the editor, showing the following code:

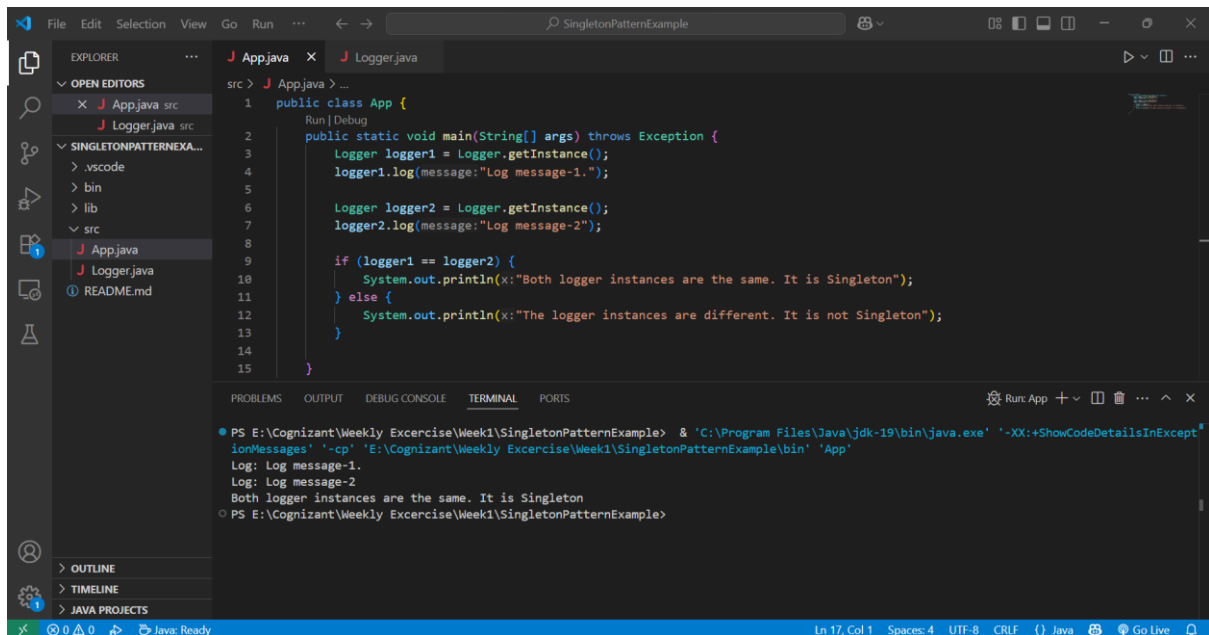
```
1 public class Logger {
2     private static Logger instance = null;
3     public static Logger getInstance()
4     {
5         if (instance == null)
6         {
7             instance = new Logger();
8         }
9         return instance;
10    }
11    public void log(String message) {
12        System.out.println("Log: " + message);
13    }
14 }
15
```



The screenshot shows the Visual Studio Code editor with the file Explorer on the left. The Explorer shows the same project 'SingletonPatternExample' with 'App.java' and 'Logger.java' in the 'src' folder. The 'App.java' file is open in the editor, showing the following code:

```
1 public class App {
2     public static void main(String[] args) throws Exception {
3         Logger logger1 = Logger.getInstance();
4         logger1.log(message:"Log message-1.");
5
6         Logger logger2 = Logger.getInstance();
7         logger2.log(message:"Log message-2");
8
9         if (logger1 == logger2) {
10            System.out.println(x:"Both logger instances are the same. It is Singleton");
11        } else {
12            System.out.println(x:"The logger instances are different. It is not Singleton");
13        }
14    }
15 }
16
17
```

OUTPUT :



```
src > J App.java x J Logger.java
EXPLORER
  OPEN EDITORS
    J App.java src
    J Logger.java src
  SINGLETONPATTERNEXA...
    .vscode
    bin
    lib
    src
      J App.java
      J Logger.java
      README.md
  OUTLINE
  TIMELINE
  JAVA PROJECTS

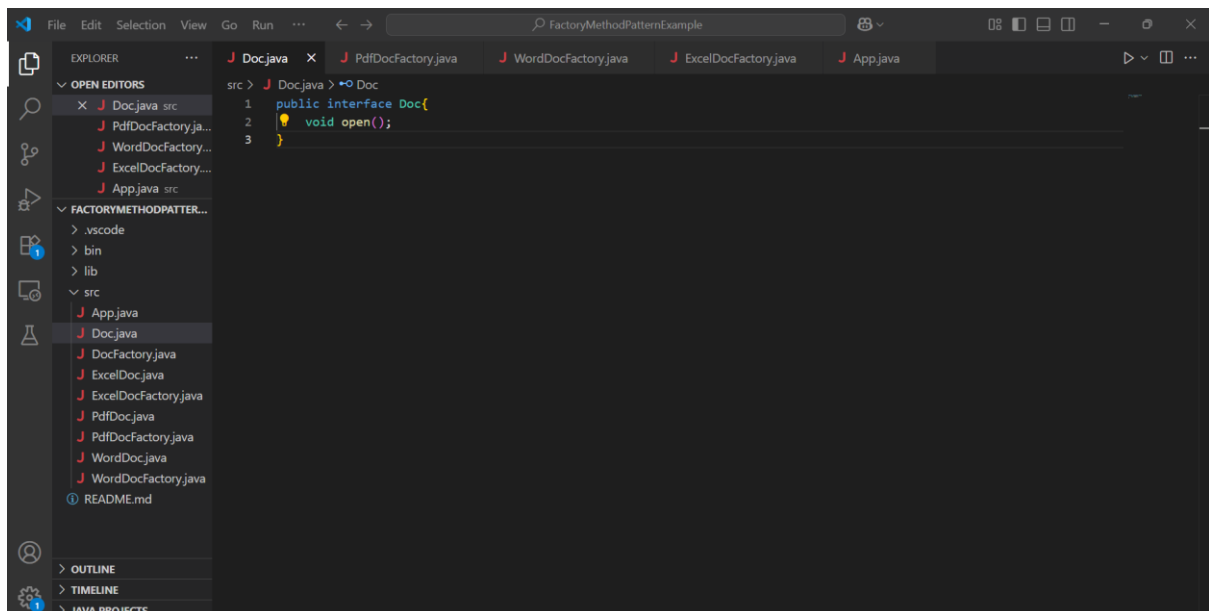
src > J App.java > ...
1 public class App {
2     public static void main(String[] args) throws Exception {
3         Logger logger1 = Logger.getInstance();
4         logger1.log(message:"Log message-1.");
5
6         Logger logger2 = Logger.getInstance();
7         logger2.log(message:"Log message-2.");
8
9         if (logger1 == logger2) {
10             System.out.println(x:"Both logger instances are the same. It is Singleton");
11         } else {
12             System.out.println(x:"The logger instances are different. It is not Singleton");
13         }
14     }
15 }

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Run: App + - + ... ^ x

PS E:\Cognizant\Weekly Exercise\Week1\SingletonPatternExample> & 'C:\Program Files\Java\jdk-19\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'E:\Cognizant\Weekly Exercise\Week1\SingletonPatternExample\bin' 'App'
Log: Log message-1.
Log: Log message-2
Both logger instances are the same.. It is Singleton
PS E:\Cognizant\Weekly Exercise\Week1\SingletonPatternExample>
```

## EX 2 : IMPLEMENTING FACTORY METHOD PATTERN

CODING:



```
src > J Doc.java x J PdfDocFactory.java J WordDocFactory.java J ExcelDocFactory.java J App.java
EXPLORER
  OPEN EDITORS
    J Doc.java src
    J PdfDocFactory.java src
    J WordDocFactory.java src
    J ExcelDocFactory.java src
    J App.java src
  FACTORYMETHODPATTER...
    .vscode
    bin
    lib
    src
      J App.java
      J Doc.java
      J DocFactory.java
      J ExcelDocFactory.java
      J ExcelDocFactory.java
      J PdfDocFactory.java
      J PdfDocFactory.java
      J WordDocFactory.java
      J WordDocFactory.java
      README.md
  OUTLINE
  TIMELINE
  JAVA PROJECTS

src > J Doc.java > Doc
1 public interface Doc{
2     void open();
3 }
```

This screenshot shows the Visual Studio Code editor with the 'FactoryMethodPatternExample' project open. The Explorer sidebar on the left displays the project structure, including the 'src' directory with files like 'App.java', 'Doc.java', 'DocFactory.java', 'ExcelDoc.java', 'ExcelDocFactory.java', 'PdfDoc.java', 'PdfDocFactory.java', 'WordDoc.java', 'WordDocFactory.java', and 'README.md'. The 'WordDoc.java' file is selected and open in the editor. The code in the editor is as follows:

```
src > J WordDoc.java > ...
1 public class WordDoc implements Doc{
2     public void open() {
3         System.out.println(x:"Opening a Word document.");
4     }
5 }
6
```

The status bar at the bottom indicates the cursor is at line 6, column 1, with 4 spaces, UTF-8 encoding, and CRLF line endings. The Java language is selected.

This screenshot shows the Visual Studio Code editor with the 'FactoryMethodPatternExample' project open. The Explorer sidebar on the left displays the project structure, including the 'src' directory with files like 'App.java', 'Doc.java', 'DocFactory.java', 'ExcelDoc.java', 'ExcelDocFactory.java', 'PdfDoc.java', 'PdfDocFactory.java', 'WordDoc.java', 'WordDocFactory.java', and 'README.md'. The 'PdfDoc.java' file is selected and open in the editor. The code in the editor is as follows:

```
src > J PdfDoc.java > PdfDoc
1 public class PdfDoc implements Doc{
2     public void open() {
3         System.out.println(x:"Opening an PDF document.");
4     }
5 }
```

The status bar at the bottom indicates the cursor is at line 1, column 35, with 4 spaces, UTF-8 encoding, and CRLF line endings. The Java language is selected.

This screenshot shows the Visual Studio Code editor with the 'ExcelDoc.java' file open. The Explorer sidebar on the left shows the project structure, including the 'src' directory with files like 'App.java', 'Doc.java', 'DocFactory.java', 'ExcelDoc.java', 'ExcelDocFactory.java', 'PdfDoc.java', 'PdfDocFactory.java', 'WordDoc.java', and 'WordDocFactory.java'. The main editor area displays the following Java code:

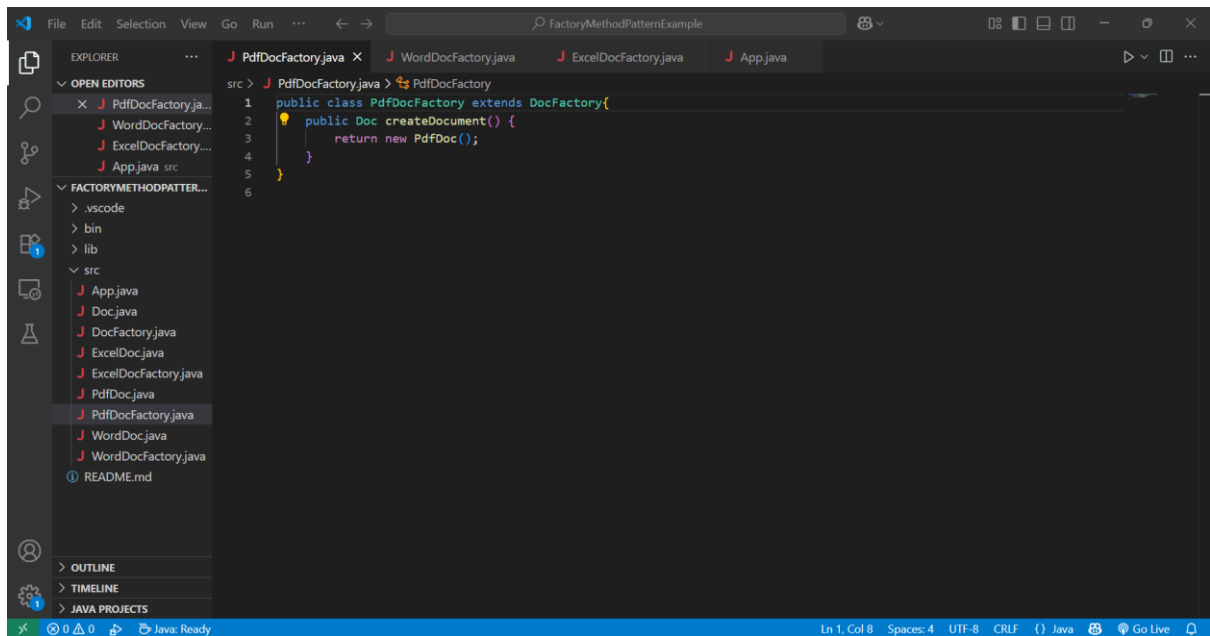
```
src > J ExcelDoc.java > ExcelDoc
1 public class ExcelDoc implements Doc{
2     public void open() {
3         System.out.println(x:"Opening an Excel document.");
4     }
5 }
6
```

The status bar at the bottom indicates 'Ln 1, Col 37', 'Spaces: 4', 'UTF-8', 'CRLF', and 'Java'.

This screenshot shows the Visual Studio Code editor with the 'WordDocFactory.java' file open. The Explorer sidebar on the left shows the same project structure as the previous screenshot. The main editor area displays the following Java code:

```
src > J WordDocFactory.java > WordDocFactory
1 public class WordDocFactory extends DocFactory{
2     public Doc createDocument() {
3         return new WordDoc();
4     }
5 }
6
```

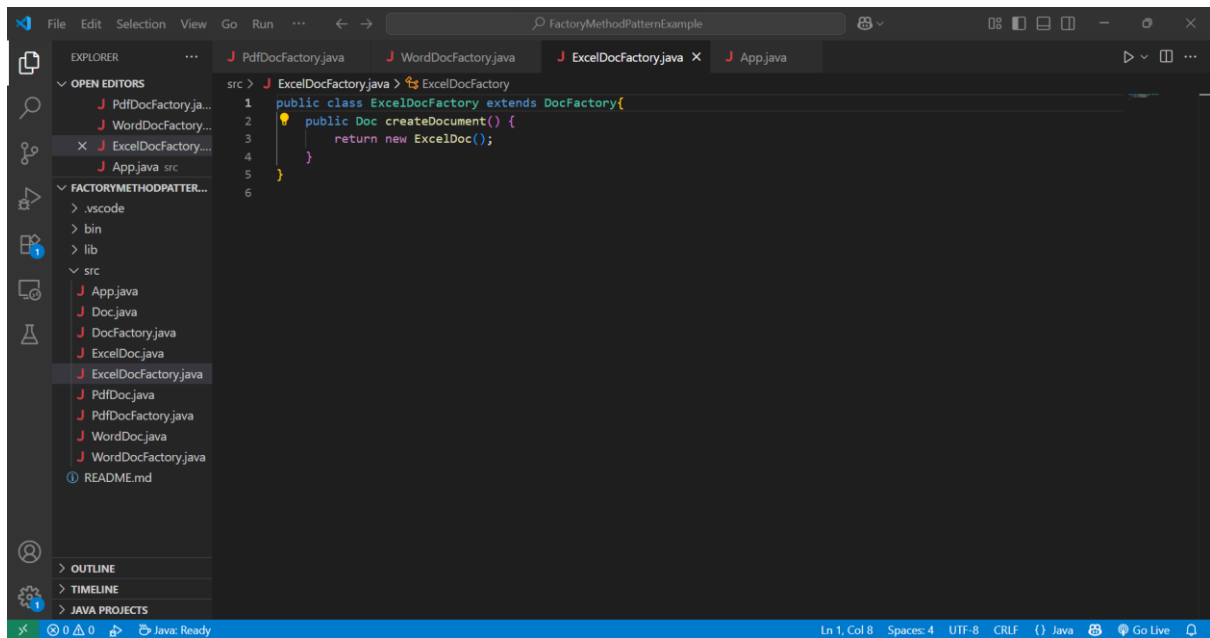
The status bar at the bottom indicates 'Ln 1, Col 8', 'Spaces: 4', 'UTF-8', 'CRLF', and 'Java'.



This screenshot shows the Visual Studio Code editor with the file `FactoryMethodPatternExample` open. The `EXPLORER` sidebar on the left displays the project structure, including the `src` directory with files like `App.java`, `Doc.java`, `DocFactory.java`, `ExcelDoc.java`, `ExcelDocFactory.java`, `PdfDoc.java`, `PdfDocFactory.java`, `WordDoc.java`, `WordDocFactory.java`, and `README.md`. The `OPEN EDITORS` list shows `PdfDocFactory.java` as the active file. The main editor area displays the code for `PdfDocFactory`, which extends `DocFactory` and implements the `createDocument()` method to return a new `PdfDoc` object.

```
src > PdfDocFactory.java > PdfDocFactory
1 public class PdfDocFactory extends DocFactory{
2     public Doc createDocument() {
3         return new PdfDoc();
4     }
5 }
6
```

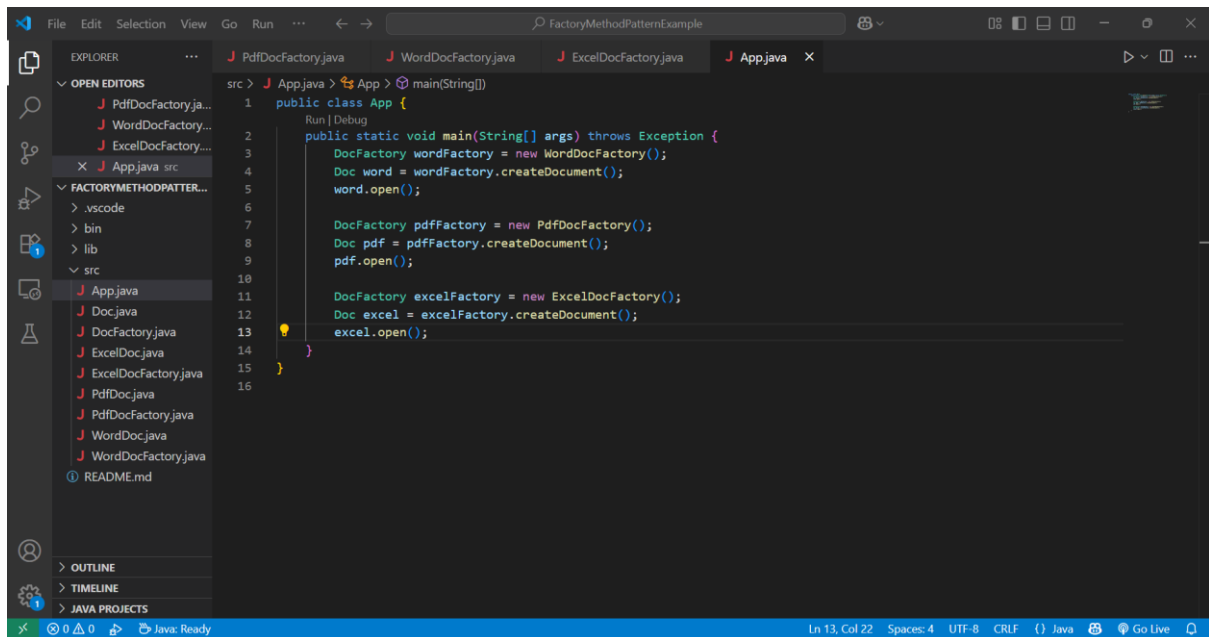
The status bar at the bottom indicates the current position is `Ln 1, Col 8`, with `Spaces: 4`, `UTF-8` encoding, `CRLF` line endings, and the `Java` language mode.



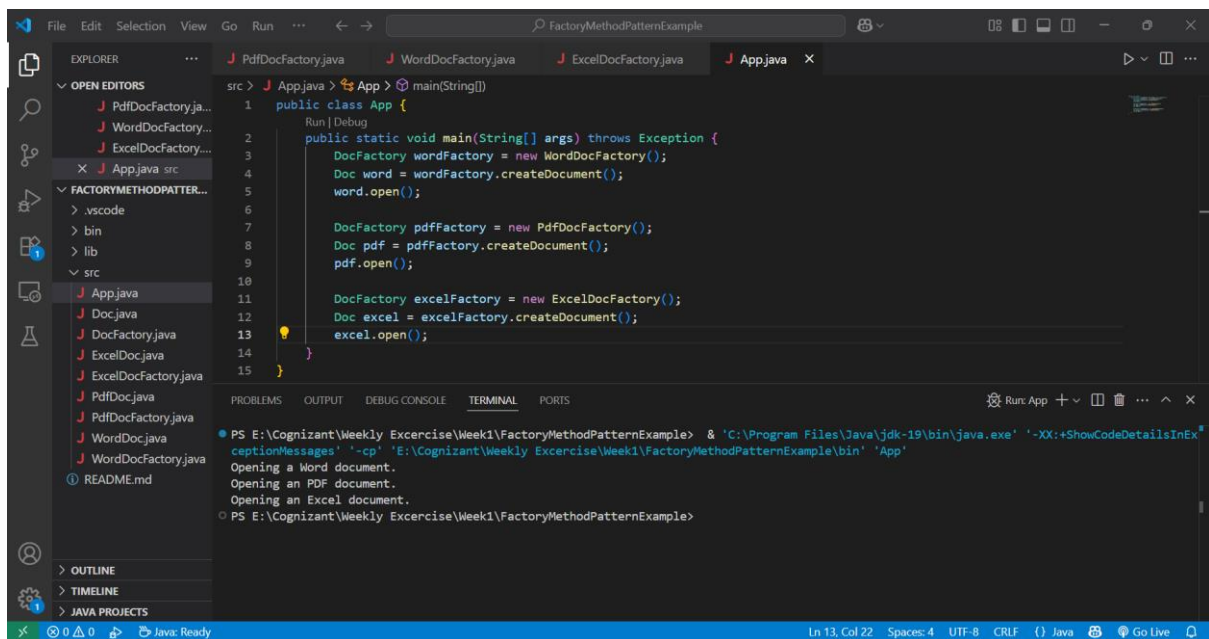
This screenshot shows the Visual Studio Code editor with the file `FactoryMethodPatternExample` open. The `EXPLORER` sidebar on the left displays the project structure, including the `src` directory with files like `App.java`, `Doc.java`, `DocFactory.java`, `ExcelDoc.java`, `ExcelDocFactory.java`, `PdfDoc.java`, `PdfDocFactory.java`, `WordDoc.java`, `WordDocFactory.java`, and `README.md`. The `OPEN EDITORS` list shows `ExcelDocFactory.java` as the active file. The main editor area displays the code for `ExcelDocFactory`, which extends `DocFactory` and implements the `createDocument()` method to return a new `ExcelDoc` object.

```
src > ExcelDocFactory.java > ExcelDocFactory
1 public class ExcelDocFactory extends DocFactory{
2     public Doc createDocument() {
3         return new ExcelDoc();
4     }
5 }
6
```

The status bar at the bottom indicates the current position is `Ln 1, Col 8`, with `Spaces: 4`, `UTF-8` encoding, `CRLF` line endings, and the `Java` language mode.



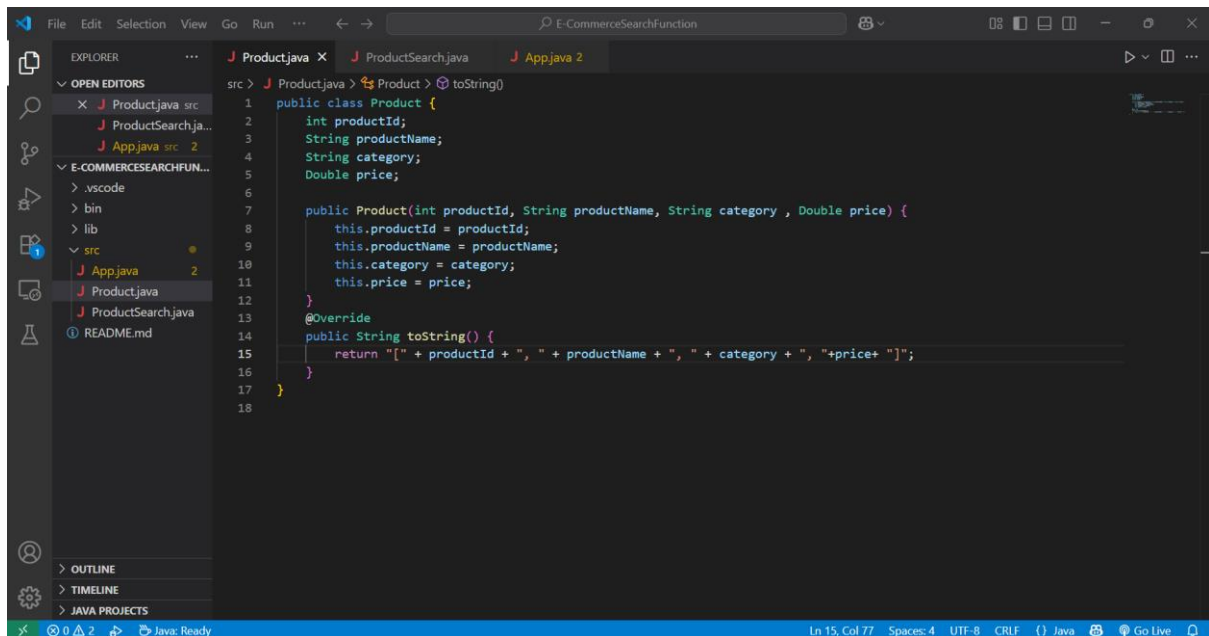
OUTPUT:



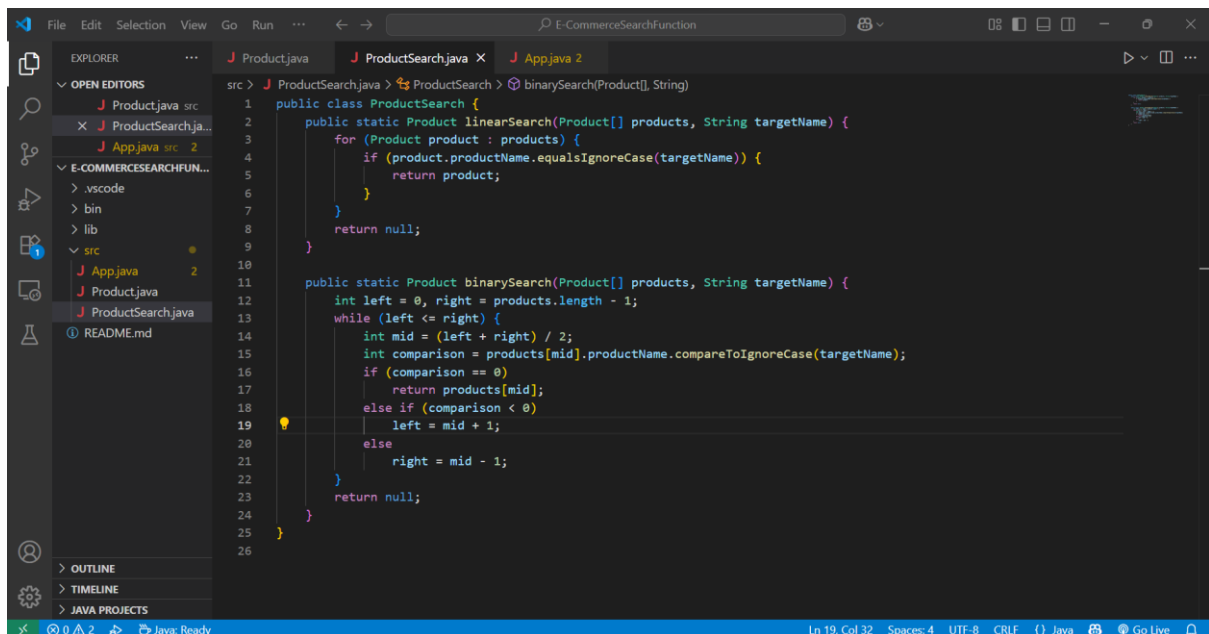
# DATA STRUCTURES AND ALGORITHMS

## EX 2 : E-COMMERCE PLATFORM SEARCH FUNCTION

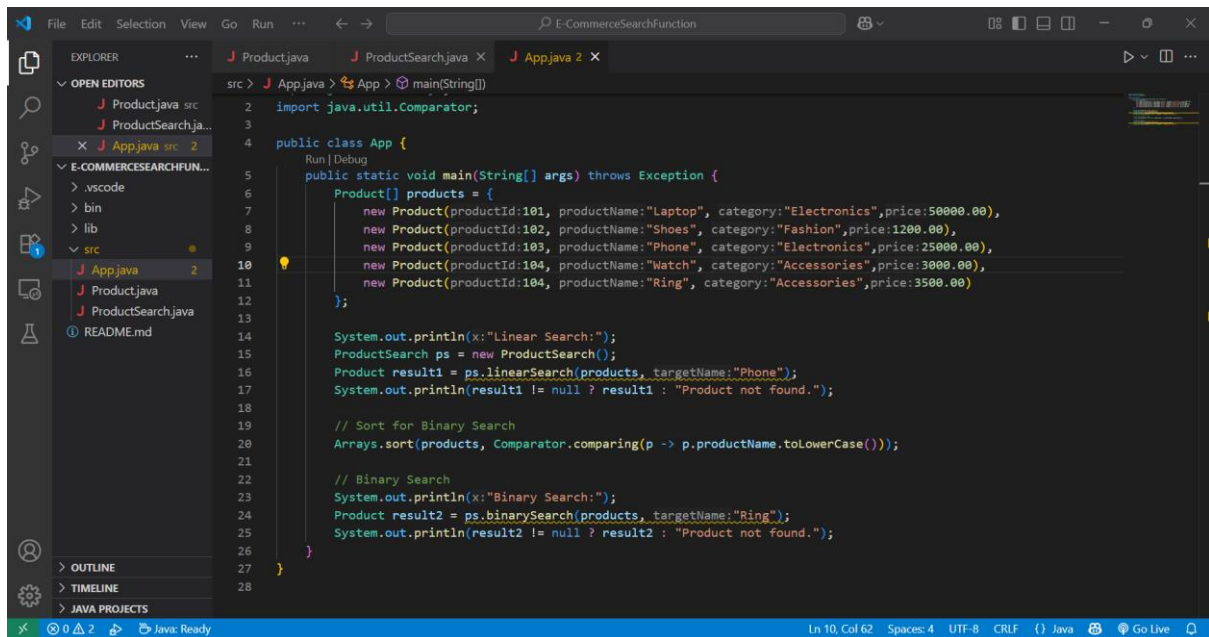
CODING:



```
src > J Product.java > Product > toString()
1 public class Product {
2     int productId;
3     String productName;
4     String category;
5     Double price;
6
7     public Product(int productId, String productName, String category, Double price) {
8         this.productId = productId;
9         this.productName = productName;
10        this.category = category;
11        this.price = price;
12    }
13    @Override
14    public String toString() {
15        return "[" + productId + ", " + productName + ", " + category + ", "+price+ "]";
16    }
17 }
18
```

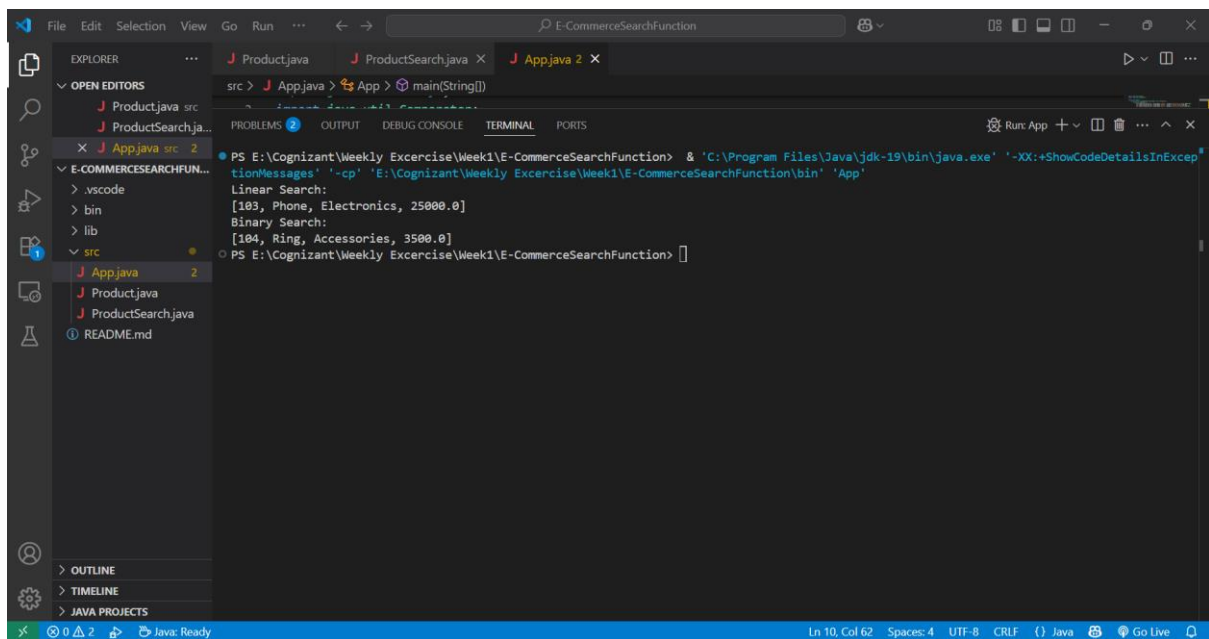


```
src > J ProductSearch.java > ProductSearch > binarySearch(Product[], String)
1 public class ProductSearch {
2     public static Product linearSearch(Product[] products, String targetName) {
3         for (Product product : products) {
4             if (product.productName.equalsIgnoreCase(targetName)) {
5                 return product;
6             }
7         }
8         return null;
9     }
10
11     public static Product binarySearch(Product[] products, String targetName) {
12         int left = 0, right = products.length - 1;
13         while (left <= right) {
14             int mid = (left + right) / 2;
15             int comparison = products[mid].productName.compareToIgnoreCase(targetName);
16             if (comparison == 0) {
17                 return products[mid];
18             } else if (comparison < 0) {
19                 left = mid + 1;
20             } else {
21                 right = mid - 1;
22             }
23         }
24         return null;
25     }
26 }
```



```
1 import java.util.Comparator;
2
3
4 public class App {
5     public static void main(String[] args) throws Exception {
6         Product[] products = {
7             new Product(productId:101, productName:"Laptop", category:"Electronics",price:50000.00),
8             new Product(productId:102, productName:"Shoes", category:"Fashion",price:1200.00),
9             new Product(productId:103, productName:"Phone", category:"Electronics",price:25000.00),
10            new Product(productId:104, productName:"Watch", category:"Accessories",price:3000.00),
11            new Product(productId:104, productName:"Ring", category:"Accessories",price:3500.00)
12        };
13
14        System.out.println(x:"Linear Search:");
15        ProductSearch ps = new ProductSearch();
16        Product result1 = ps.linearSearch(products, targetName:"Phone");
17        System.out.println(result1 != null ? result1 : "Product not found.");
18
19        // Sort for Binary Search
20        Arrays.sort(products, Comparator.comparing(p -> p.productName.toLowerCase()));
21
22        // Binary Search
23        System.out.println(x:"Binary Search:");
24        Product result2 = ps.binarySearch(products, targetName:"Ring");
25        System.out.println(result2 != null ? result2 : "Product not found.");
26    }
27 }
28
```

OUTPUT:

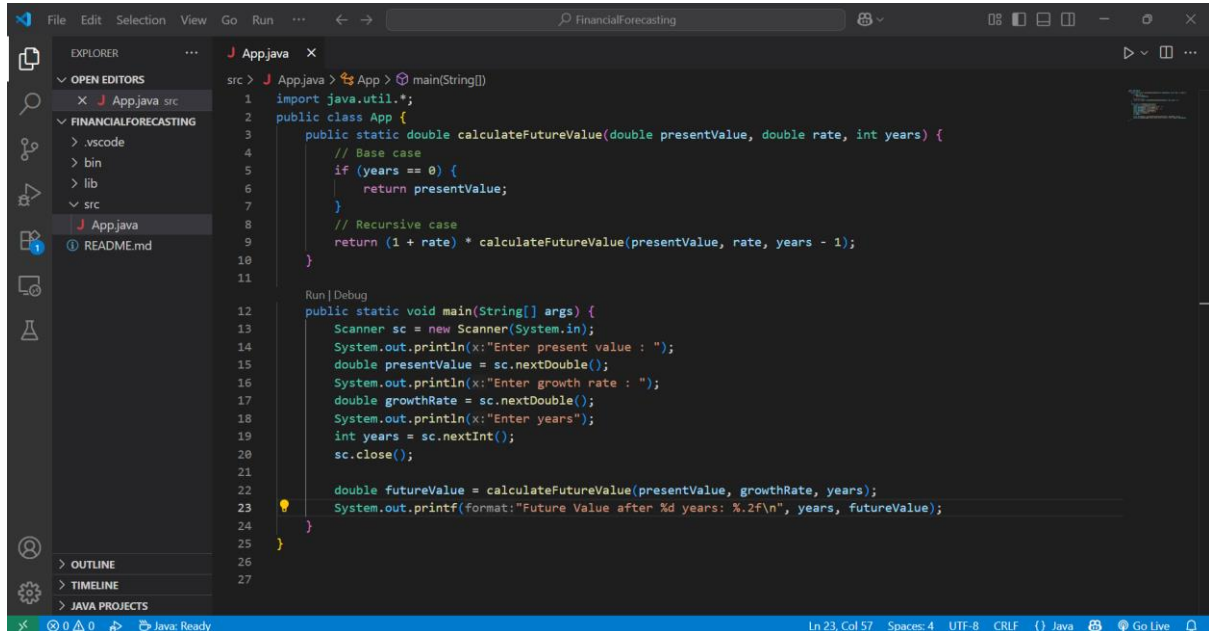


```
PS E:\Cognizant\Weekly Exercise\Week1\E-CommerceSearchFunction> & 'C:\Program Files\Java\jdk-19\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'E:\Cognizant\Weekly Exercise\Week1\E-CommerceSearchFunction\bin' 'App'
Linear Search:
[103, Phone, Electronics, 25000.0]
Binary Search:
[104, Ring, Accessories, 3500.0]
PS E:\Cognizant\Weekly Exercise\Week1\E-CommerceSearchFunction>
```



## EX 7 : FINANCIAL FORECAST

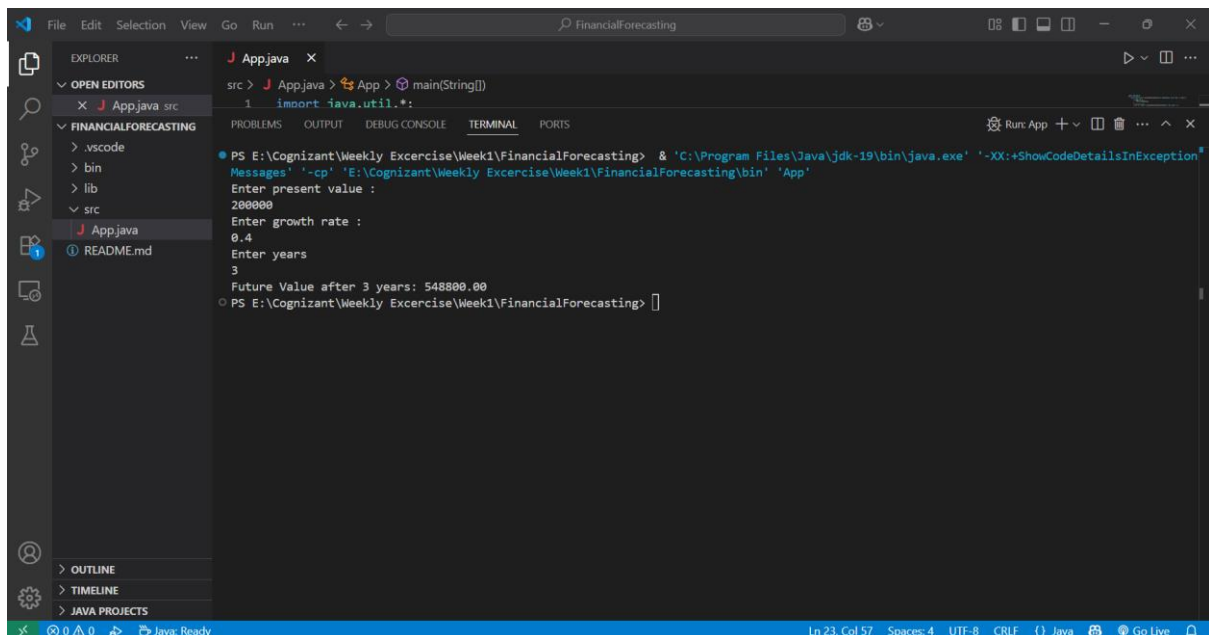
### CODING:



The screenshot shows the Visual Studio Code editor with a Java file named `App.java` open. The code implements a recursive method `calculateFutureValue` to calculate the future value of an investment based on a present value, a growth rate, and a number of years. The `main` method uses a `Scanner` to take user input for these three values and then calls the recursive method to calculate the result, which is printed to the console.

```
src > J App.java > App > main(String[])  
1  import java.util.*;  
2  public class App {  
3      public static double calculateFutureValue(double presentValue, double rate, int years) {  
4          // Base case  
5          if (years == 0) {  
6              return presentValue;  
7          }  
8          // Recursive case  
9          return (1 + rate) * calculateFutureValue(presentValue, rate, years - 1);  
10     }  
11  
12     Run | Debug  
13     public static void main(String[] args) {  
14         Scanner sc = new Scanner(System.in);  
15         System.out.println(x:"Enter present value : ");  
16         double presentValue = sc.nextDouble();  
17         System.out.println(x:"Enter growth rate : ");  
18         double growthRate = sc.nextDouble();  
19         System.out.println(x:"Enter years");  
20         int years = sc.nextInt();  
21         sc.close();  
22  
23         double futureValue = calculateFutureValue(presentValue, growthRate, years);  
24         System.out.printf(format:"Future Value after %d years: %.2f\n", years, futureValue);  
25     }  
26  
27 }
```

### OUTPUT:



The screenshot shows the Visual Studio Code editor with the terminal window open. The terminal displays the output of the Java application, showing the user input for present value, growth rate, and years, followed by the calculated future value.

```
PS E:\Cognizant\Weekly Exercise\Week1\FinancialForecasting> & 'C:\Program Files\Java\jdk-19\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'E:\Cognizant\Weekly Exercise\Week1\FinancialForecasting\bin' 'App'  
Enter present value :  
200000  
Enter growth rate :  
0.4  
Enter years  
3  
Future Value after 3 years: 548800.00  
PS E:\Cognizant\Weekly Exercise\Week1\FinancialForecasting>
```