

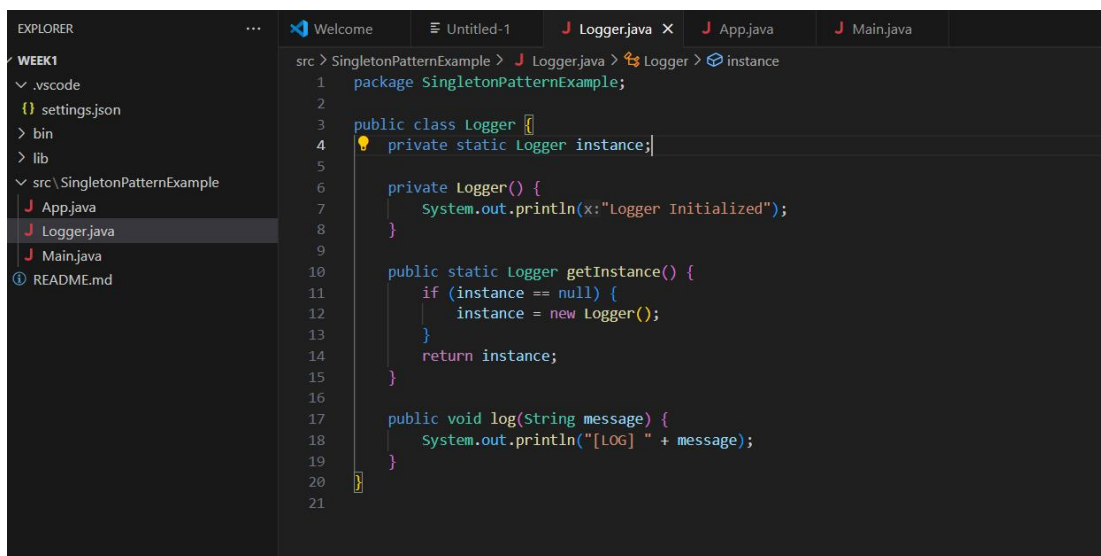
# Digital Nurture 4.0 – Deep Skilling

## Week 1 - Mandatory Hands - on

### Hands - on : 1

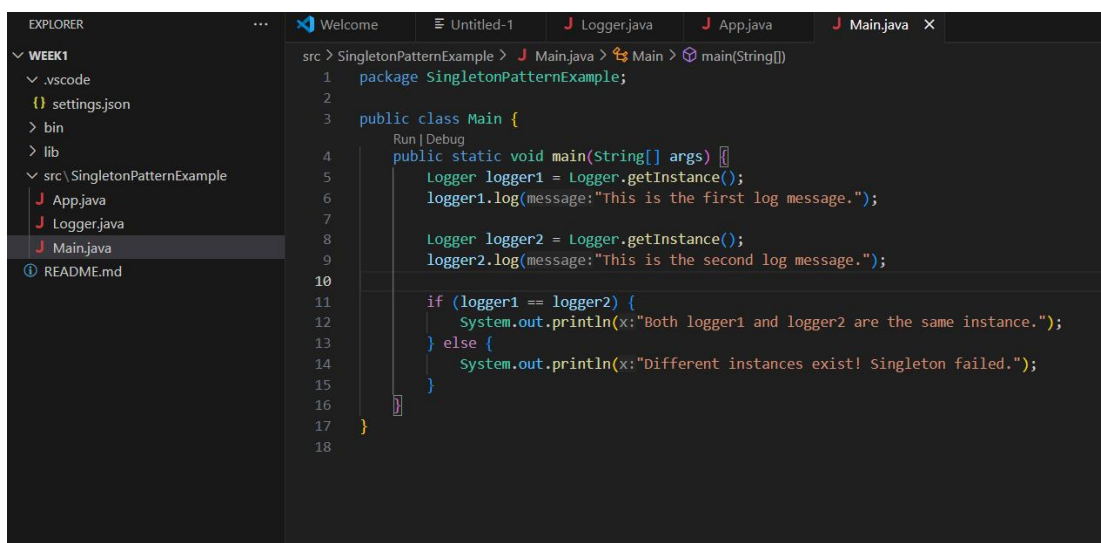
#### 1) Implementing the Singleton Pattern

##### Logger.java



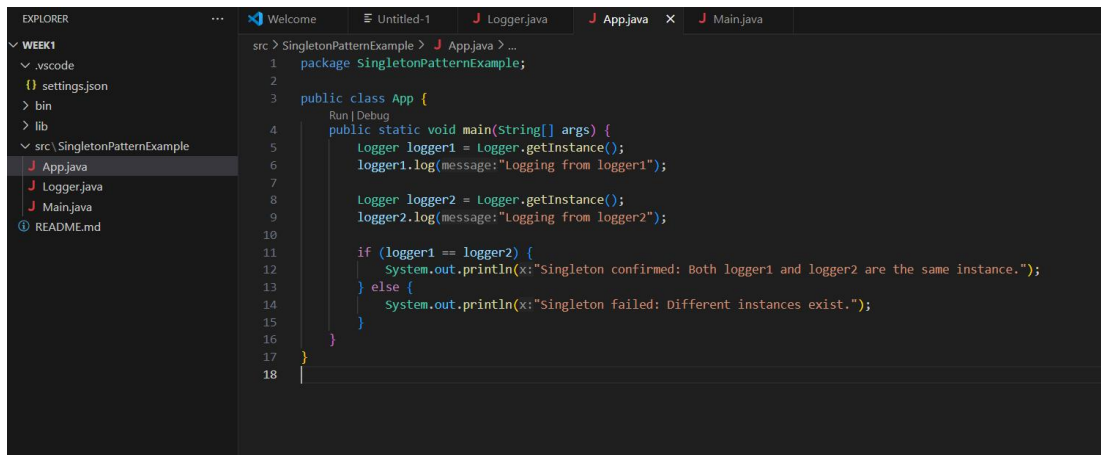
```
src > SingletonPatternExample > J Logger.java > instance
1 package SingletonPatternExample;
2
3 public class Logger {
4     private static Logger instance;
5
6     private Logger() {
7         System.out.println(x:"Logger Initialized");
8     }
9
10    public static Logger getInstance() {
11        if (instance == null) {
12            instance = new Logger();
13        }
14        return instance;
15    }
16
17    public void log(String message) {
18        System.out.println("[LOG] " + message);
19    }
20 }
21
```

##### Main.java



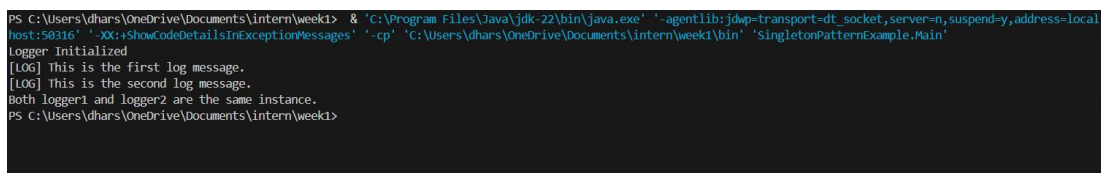
```
src > SingletonPatternExample > J Main.java > Main > main(String[])
1 package SingletonPatternExample;
2
3 public class Main {
4     public static void main(String[] args) {
5         Logger logger1 = Logger.getInstance();
6         logger1.log(message:"This is the first log message.");
7
8         Logger logger2 = Logger.getInstance();
9         logger2.log(message:"This is the second log message.");
10
11         if (logger1 == logger2) {
12             System.out.println(x:"Both logger1 and logger2 are the same instance.");
13         } else {
14             System.out.println(x:"Different instances exist! Singleton failed.");
15         }
16     }
17 }
18
```

# App.java



```
src > SingletonPatternExample > App.java > ...
1  package SingletonPatternExample;
2
3  public class App {
4      public static void main(String[] args) {
5          Logger logger1 = Logger.getInstance();
6          logger1.log(message:"Logging from logger1");
7
8          Logger logger2 = Logger.getInstance();
9          logger2.log(message:"Logging from logger2");
10
11         if (logger1 == logger2) {
12             System.out.println(x:"Singleton confirmed: Both logger1 and logger2 are the same instance.");
13         } else {
14             System.out.println(x:"Singleton failed: Different instances exist.");
15         }
16     }
17 }
18
```

## Output

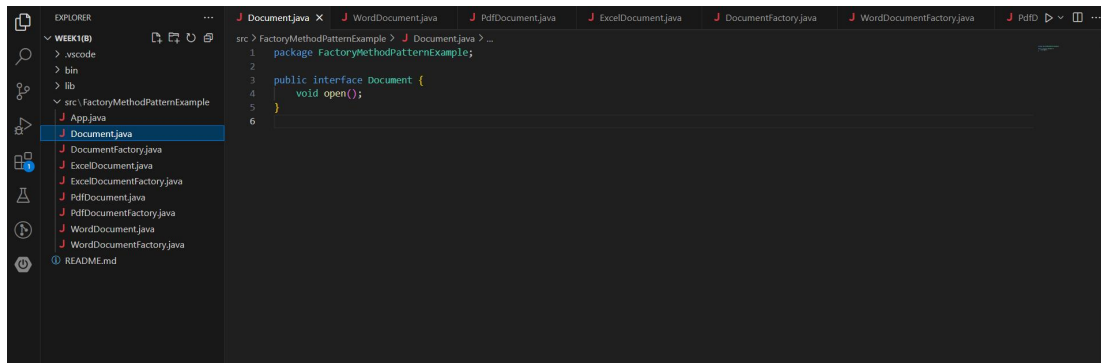


```
PS C:\Users\dhars\OneDrive\Documents\intern\week1> & 'C:\Program Files\Java\jdk-22\bin\java.exe' '-agentlib:jdwp=transport=dt_socket,server=n,suspend=y,address=local
host:58316' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\dhars\OneDrive\Documents\intern\week1\bin' 'SingletonPatternExample.Main'
Logger Initialized
[LOG] This is the first log message.
[LOG] This is the second log message.
Both logger1 and logger2 are the same instance.
PS C:\Users\dhars\OneDrive\Documents\intern\week1>
```

## Hands - on : 2

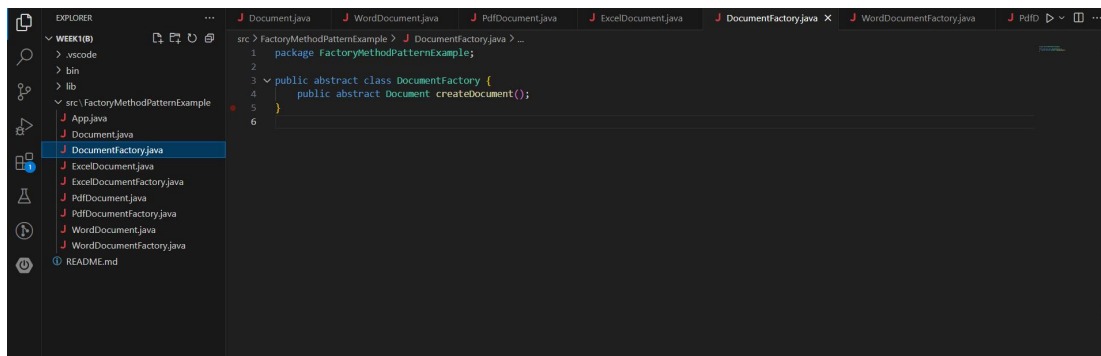
### 2) Implementing the Factory Method Pattern

#### Document.java



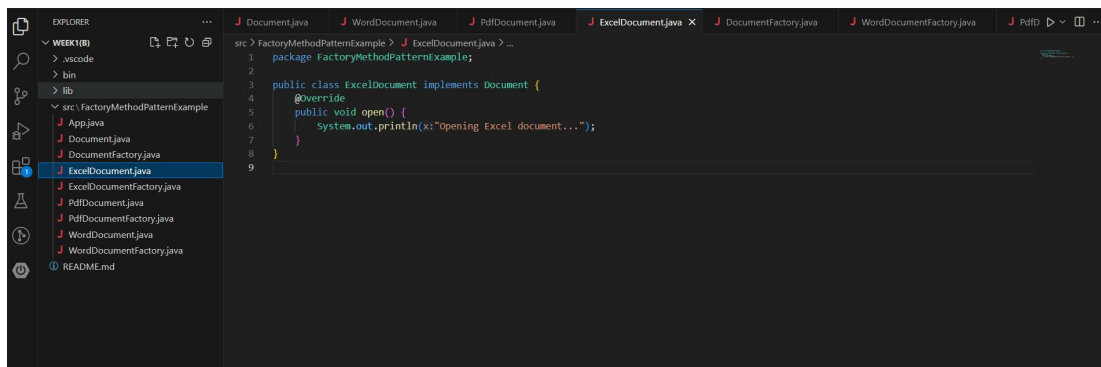
```
src > FactoryMethodPatternExample > J Document.java > ...
1 package FactoryMethodPatternExample;
2
3 public interface Document {
4     void open();
5 }
6
```

#### DocumentFactory.java



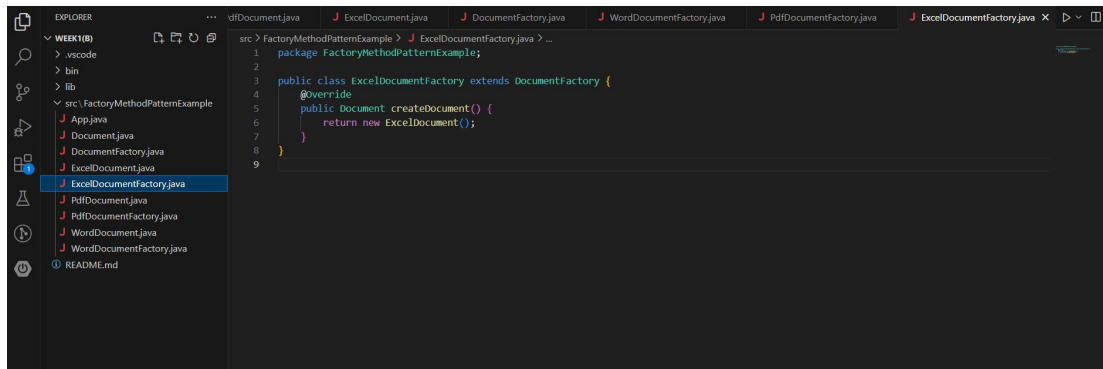
```
src > FactoryMethodPatternExample > J DocumentFactory.java > ...
1 package FactoryMethodPatternExample;
2
3 public abstract class DocumentFactory {
4     public abstract Document createDocument();
5 }
6
```

#### ExcelDocument.java



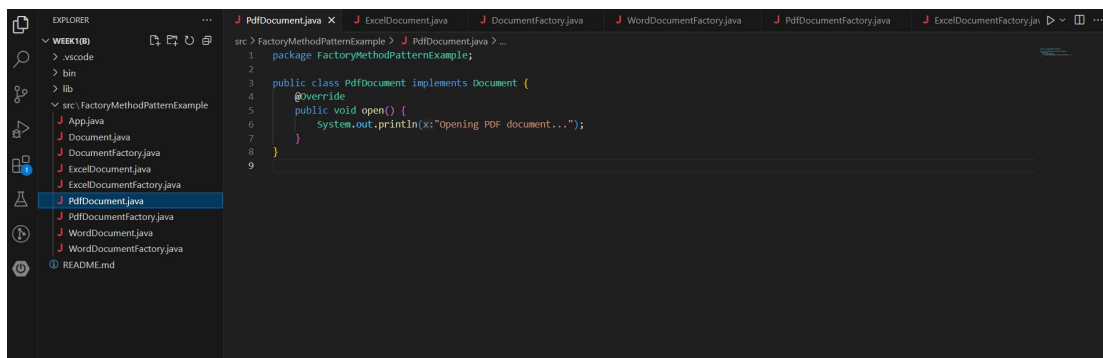
```
src > FactoryMethodPatternExample > J ExcelDocument.java > ...
1 package FactoryMethodPatternExample;
2
3 public class ExcelDocument implements Document {
4     @Override
5     public void open() {
6         System.out.println("Opening Excel document...");
7     }
8 }
9
```

## ExcelDocumentFactory.java



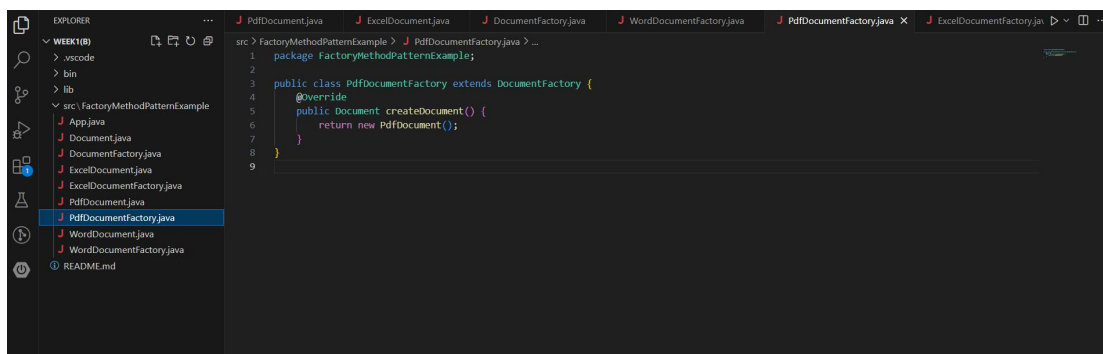
```
1 package FactoryMethodPatternExample;
2
3 public class ExcelDocumentFactory extends DocumentFactory {
4     @Override
5     public Document createDocument() {
6         return new ExcelDocument();
7     }
8 }
9
```

## PdfDocument.java



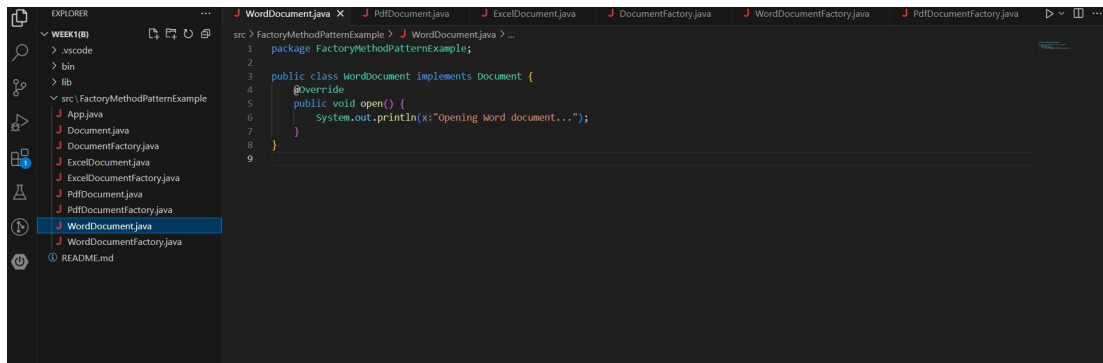
```
1 package FactoryMethodPatternExample;
2
3 public class PdfDocument implements Document {
4     @Override
5     public void open() {
6         System.out.println("Opening PDF document...");
7     }
8 }
9
```

## pdfDocumentFactory.java



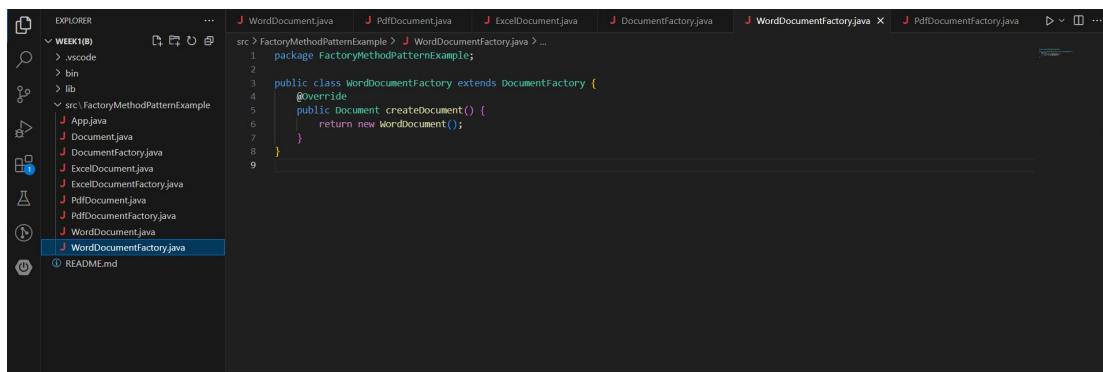
```
1 package FactoryMethodPatternExample;
2
3 public class PdfDocumentFactory extends DocumentFactory {
4     @Override
5     public Document createDocument() {
6         return new PdfDocument();
7     }
8 }
9
```

## WordDocument.java

A screenshot of the Visual Studio Code editor. The Explorer sidebar on the left shows a project structure with a folder 'src' containing 'FactoryMethodPatternExample'. Inside this folder, 'WordDocument.java' is selected and highlighted. The main editor window displays the code for 'WordDocument.java'. The code is as follows:

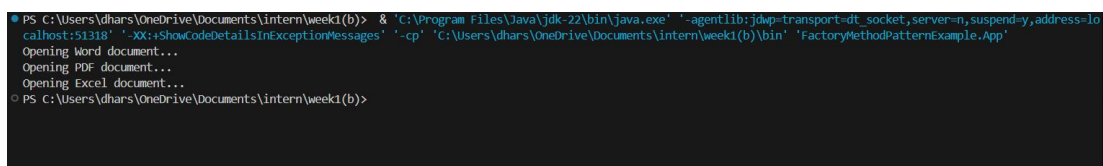
```
src > FactoryMethodPatternExample > WordDocument.java > ...
1 package FactoryMethodPatternExample;
2
3 public class WordDocument implements Document {
4     @Override
5     public void open() {
6         System.out.println("Opening Word document...");
7     }
8 }
9
```

## WordDocumentFactory.java

A screenshot of the Visual Studio Code editor. The Explorer sidebar on the left shows the same project structure as the previous image, with 'WordDocumentFactory.java' now selected and highlighted. The main editor window displays the code for 'WordDocumentFactory.java'. The code is as follows:

```
src > FactoryMethodPatternExample > WordDocumentFactory.java > ...
1 package FactoryMethodPatternExample;
2
3 public class WordDocumentFactory extends DocumentFactory {
4     @Override
5     public Document createDocument() {
6         return new WordDocument();
7     }
8 }
9
```

## Output

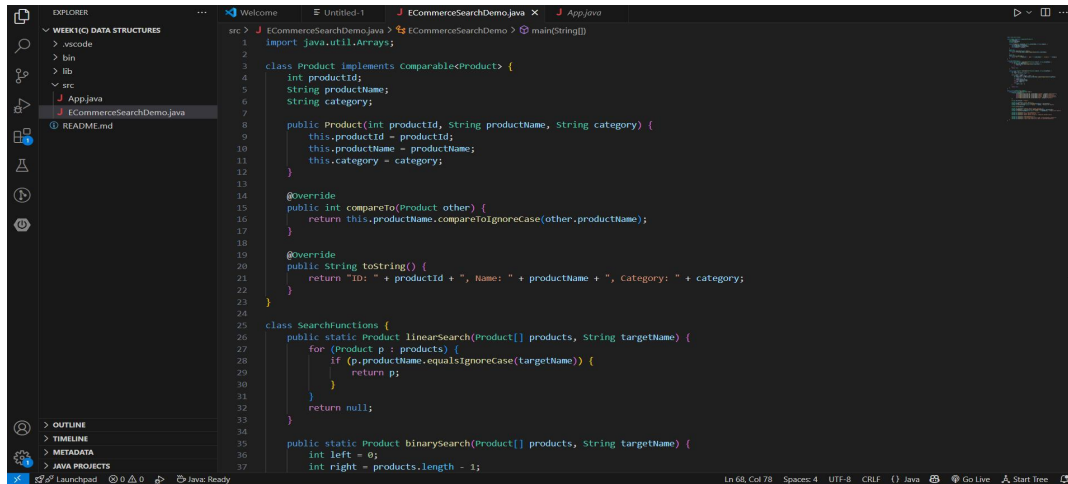
A screenshot of a terminal window. The prompt is 'PS C:\Users\dhars\OneDrive\Documents\intern\week1(b)>'. The user has entered a command to run the application. The output of the program is displayed on the next line:

```
PS C:\Users\dhars\OneDrive\Documents\intern\week1(b)> & 'C:\Program Files\Java\jdk-22\bin\java.exe' '-agentlib:jdwp=transport=dt_socket,server=n,suspend=y,address=10.0.0.1:51318' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\dhars\OneDrive\Documents\intern\week1(b)\bin' 'FactoryMethodPatternExample.App'
Opening Word document...
Opening PDF document...
Opening Excel document...
PS C:\Users\dhars\OneDrive\Documents\intern\week1(b)>
```

## Hands - on : 3

### 3) E-commerce Platform Search Function

#### ECommerceSearchDemo.java



```
src > J ECommerceSearchDemo.java > ECommerceSearchDemo ? main(String[])
import java.util.Arrays;

2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37

class Product implements Comparable<Product> {
    int productid;
    String productName;
    String category;

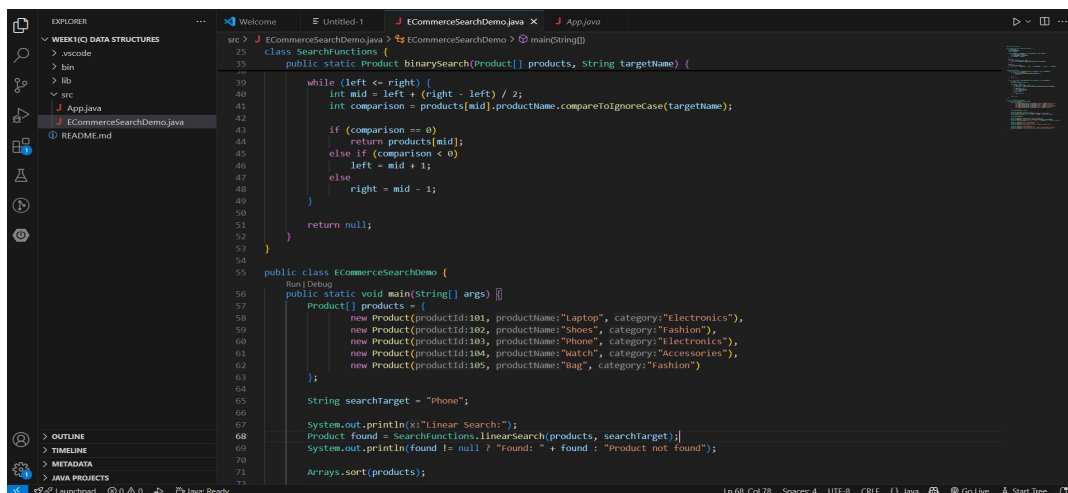
    public Product(int productid, String productName, String category) {
        this.productid = productid;
        this.productName = productName;
        this.category = category;
    }

    @Override
    public int compareTo(Product other) {
        return this.productName.compareToIgnoreCase(other.productName);
    }

    @Override
    public String toString() {
        return "ID: " + productid + ", Name: " + productName + ", category: " + category;
    }
}

class SearchFunctions {
    public static Product linearSearch(Product[] products, String targetName) {
        for (Product p : products) {
            if (p.productName.equalsIgnoreCase(targetName)) {
                return p;
            }
        }
        return null;
    }

    public static Product binarySearch(Product[] products, String targetName) {
        int left = 0;
        int right = products.length - 1;
```



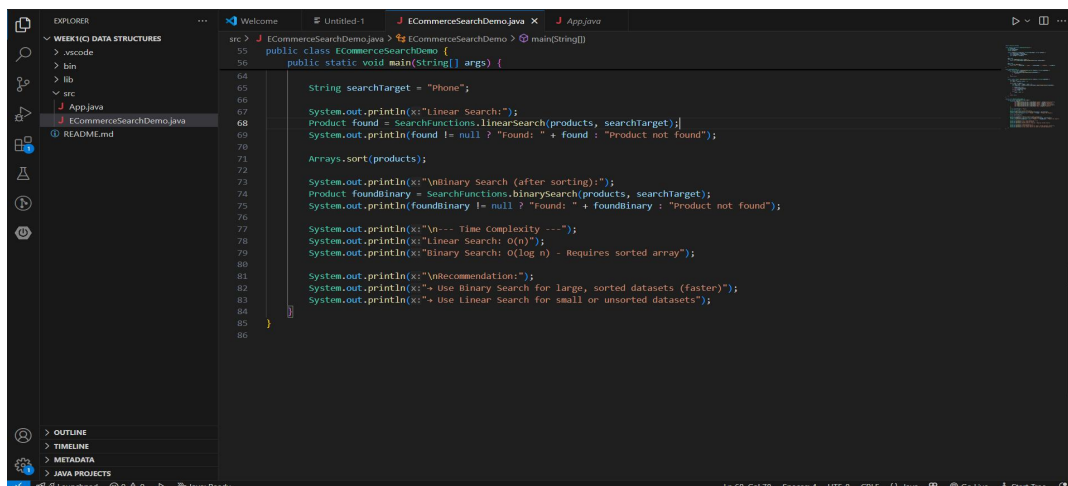
```
src > J ECommerceSearchDemo.java > ECommerceSearchDemo ? main(String[])
class SearchFunctions {
    public static Product binarySearch(Product[] products, String targetName) {
        while (left <= right) {
            int mid = left + (right - left) / 2;
            int comparison = products[mid].productName.compareToIgnoreCase(targetName);
            if (comparison == 0) {
                return products[mid];
            } else if (comparison < 0) {
                left = mid + 1;
            } else {
                right = mid - 1;
            }
        }
        return null;
    }
}

public class ECommerceSearchDemo {
    public static void main(String[] args) {
        Product[] products = {
            new Product(productid:101, productName:"Laptop", category:"Electronics"),
            new Product(productid:102, productName:"Shoes", category:"Fashion"),
            new Product(productid:103, productName:"Phone", category:"Electronics"),
            new Product(productid:104, productName:"Watch", category:"Accessories"),
            new Product(productid:105, productName:"Bag", category:"Fashion")
        };

        String searchTarget = "Phone";

        System.out.println("Linear Search:");
        Product found = SearchFunctions.linearSearch(products, searchTarget);
        System.out.println(found != null ? "Found: " + found : "Product not found");

        Arrays.sort(products);
    }
}
```



```
src > J ECommerceSearchDemo.java > ECommerceSearchDemo ? main(String[])
public class ECommerceSearchDemo {
    public static void main(String[] args) {
        String searchTarget = "Phone";

        System.out.println("Linear Search:");
        Product found = searchFunctions.linearSearch(products, searchTarget);
        System.out.println(found != null ? "Found: " + found : "Product not found");

        Arrays.sort(products);

        System.out.println("Binary Search (after sorting):");
        Product foundBinary = SearchFunctions.binarySearch(products, searchTarget);
        System.out.println(foundBinary != null ? "Found: " + foundBinary : "Product not found");

        System.out.println("Time Complexity ---");
        System.out.println("Linear Search: O(n)");
        System.out.println("Binary Search: O(log n) - Requires sorted array");

        System.out.println("Recommendation:");
        System.out.println("Use Binary Search for large, sorted datasets (faster)");
        System.out.println("Use Linear Search for small or unsorted datasets");
    }
}
```

## Output

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Debug: ECommerceSearchDemo

PS C:\Users\dhars\OneDrive\Documents\intern\week1(c) Data structures> & 'C:\Program Files\Java\jdk-22\bin\java.exe' '-agentlib:jdwp=transport=dt_socket,server=n,suspend=y,address=localhost:54825' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\dhars\OneDrive\Documents\intern\week1(c) Data structures\bin' 'ECommerceSearchDemo'
Linear Search:
Found: ID: 103, Name: Phone, Category: Electronics

Binary Search (after sorting):
Found: ID: 103, Name: Phone, Category: Electronics

--- Time Complexity ---
Linear Search: O(n)
Binary Search: O(log n) - Requires sorted array

Recommendation:
? Use Binary Search for large, sorted datasets (faster)
? Use Linear Search for small or unsorted datasets
PS C:\Users\dhars\OneDrive\Documents\intern\week1(c) Data structures>
```

## Hands - on : 4

### 4) Financial Forecasting

#### App.java

```
EXPLORER
WEEK1(D) DATA STRU...
> .vscode
> bin
> lib
> src
  App.java
  README.md
OUTLINE
TIMELINE
METADATA
JAVA PROJECTS

src > J App.java > App > forecastMemo(double, double, int, double[])
1 public class App {
2
3     public static double forecastRecursive(double baseValue, double growthRate, int years) {
4         if (years == 0) {
5             return baseValue;
6         }
7         return forecastRecursive(baseValue, growthRate, years - 1) * (1 + growthRate);
8     }
9
10    public static double forecastMemo(double baseValue, double growthRate, int years, double[] memo) {
11        if (years == 0) {
12            return baseValue;
13        }
14        if (memo[years] != 0) {
15            return memo[years];
16        }
17        memo[years] = forecastMemo(baseValue, growthRate, years - 1, memo) * (1 + growthRate);
18        return memo[years];
19    }
20
21    public static void main(String[] args) {
22        double baseValue = 10000;
23        double growthRate = 0.08;
24        int years = 10;
25
26        double recursiveResult = forecastRecursive(baseValue, growthRate, years);
27        System.out.printf("Future value (Recursive): %.2f\n", recursiveResult);
28
29        double[] memo = new double[years + 1];
30        double memoResult = forecastMemo(baseValue, growthRate, years, memo);
31        System.out.printf("Future value (Optimized with Memoization): %.2f\n", memoResult);
32    }
33 }
```

## Output

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Debug: App

PS C:\Users\dhars\OneDrive\Documents\intern\week1(d) Data Structures> & 'C:\Program Files\Java\jdk-22\bin\java.exe' '-agentlib:jdwp=transport=dt_socket,server=n,suspend=y,address=localhost:55109' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\dhars\OneDrive\Documents\intern\week1(d) Data Structures\bin' 'App'
Future value (Recursive): ?21589.25
Future value (Optimized with Memoization): ?21589.25
PS C:\Users\dhars\OneDrive\Documents\intern\week1(d) Data Structures>
```