WEEK-1

1.E-commerce Platform Search Function:

Understand Asymptotic Notation

Big O Notation:

Big O notation is used to describe the time or space complexity of an algorithm in terms of input size n. It helps estimate the worst-case performance of algorithms as inputs grow, giving developers a tool to compare and choose efficient solutions.

Best, Average, and Worst-Case Scenarios

For search operations:

Scenario	**Linear Search**	**Binary Search**
Best Case	O(1) – First item matched	O(1) – Middle element match
Average Case	O(n/2) → O(n)	O(log n)
Worst Case	O(n) – Not found at all	O(log n) – Keep halving

- Linear search checks each item in sequence – simple but slow for large datasets.
- Binary search cuts the search space in half each step – much faster, but requires sorted data**.**

For an e-commerce platform**,** which deals with thousands of products, binary search would be much more efficient than linear search.

2.Financial Forecasting:

Understand Recursive Algorithms:

Recursion is a programming technique where a function calls itself to solve smaller instances of a problem until it reaches a base case.

Example analogy:
Calculating compound interest year after year. Each year builds on the result of the previous year.

Why Use Recursion?

- Simplifies problems that have repeating patterns, such as growth over time.

- Natural fit for problems with self-similar structure**,** like financial projections or tree traversals.

Time Complexity

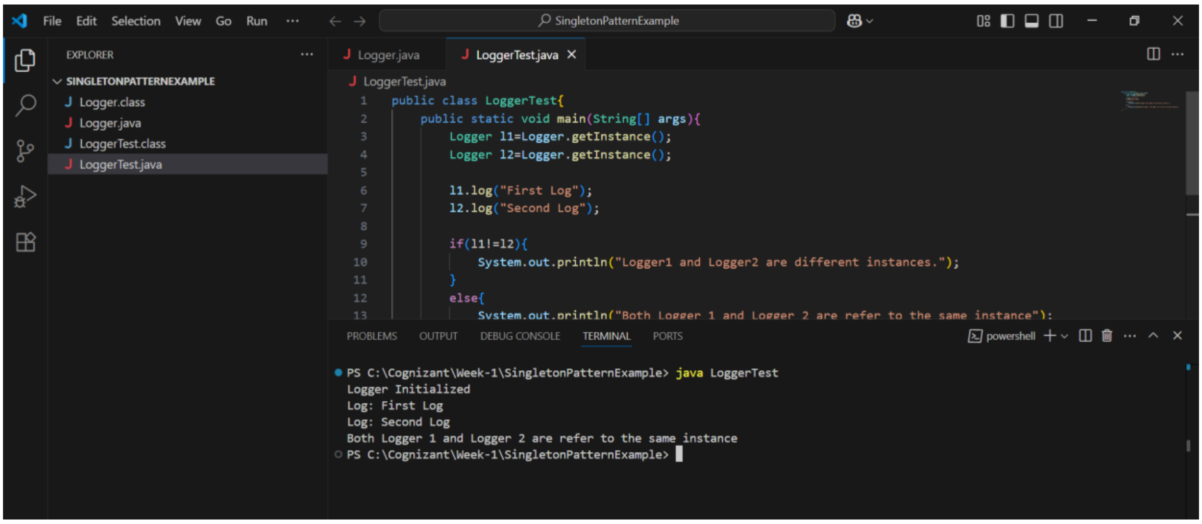
The recursive algorithm has O(n) time and space complexity, as it makes one recursive call per year. Each call builds on the previous year's result.

Optimization

To avoid excessive computation and stack overflow:

- Use memoization to store already computed results.
- Or use an iterative approach to eliminate recursive calls.

SingletonPattern Output:



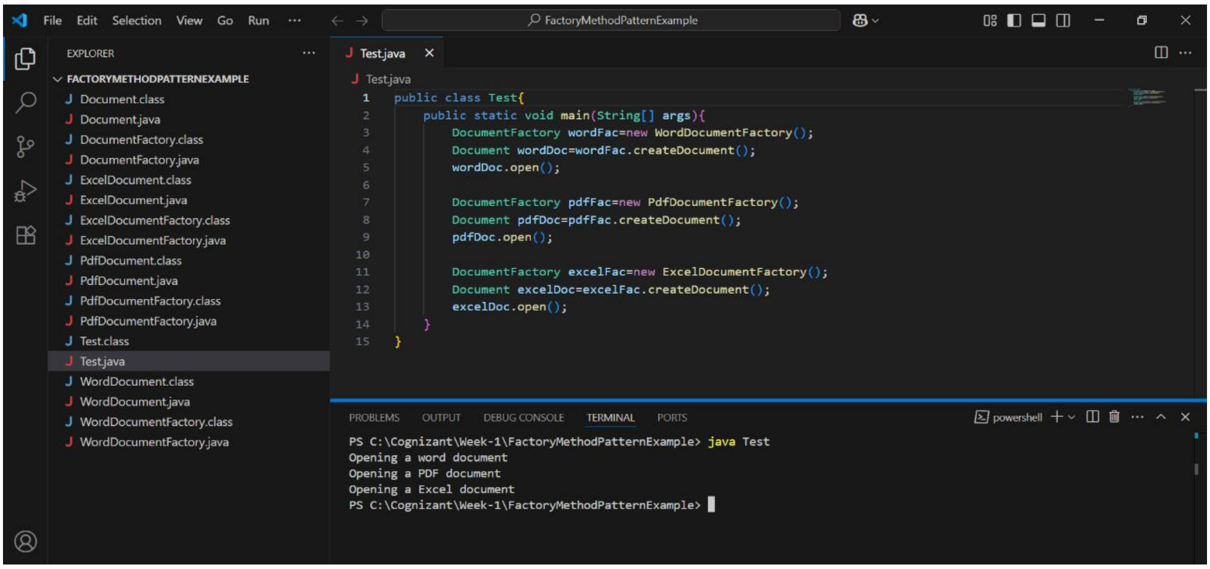
```
public class LoggerTest{
    public static void main(String[] args){
        Logger l1=Logger.getInstance();
        Logger l2=Logger.getInstance();

        l1.log("First Log");
        l2.log("Second Log");

        if(l1!=l2){
            System.out.println("Logger1 and Logger2 are different instances.");
        }
        else{
            System.out.println("Both Logger 1 and Logger 2 are refer to the same instance");
```

```
PS C:\Cognizant\Week-1\SingletonPatternExample> java LoggerTest
Logger Initialized
Log: First Log
Log: Second Log
Both Logger 1 and Logger 2 are refer to the same instance
PS C:\Cognizant\Week-1\SingletonPatternExample> 
```

FactotyMethodPattern Output:



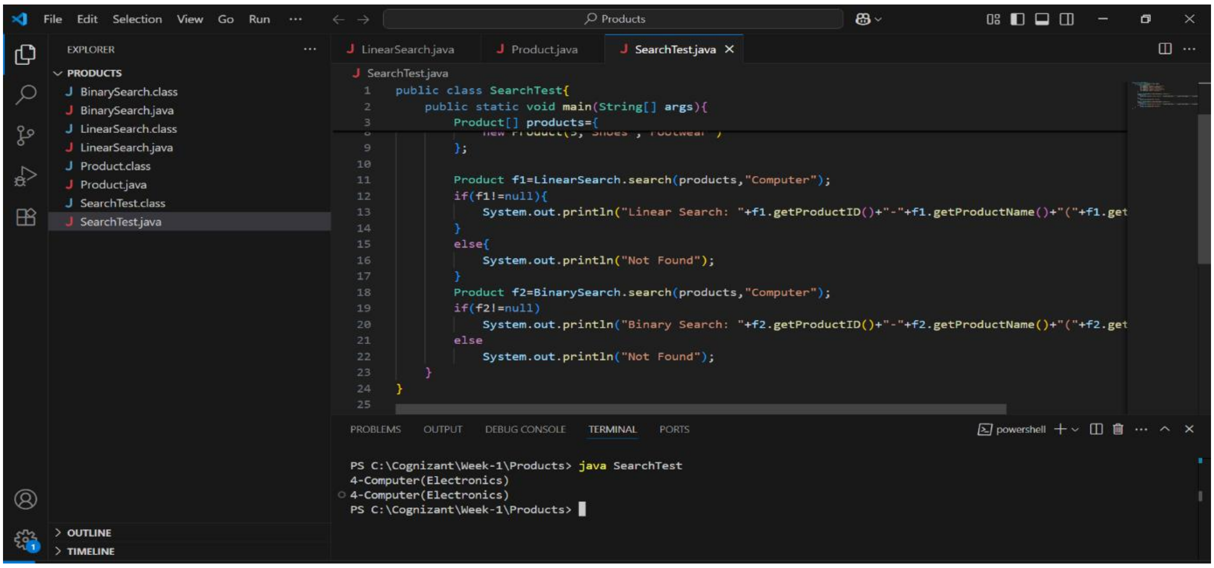
```
public class Test{
    public static void main(String[] args){
        DocumentFactory wordFac=new WordDocumentFactory();
        Document wordDoc=wordFac.createDocument();
        wordDoc.open();

        DocumentFactory pdfFac=new PdfDocumentFactory();
        Document pdfDoc=pdfFac.createDocument();
        pdfDoc.open();

        DocumentFactory excelFac=new ExcelDocumentFactory();
        Document excelDoc=excelFac.createDocument();
        excelDoc.open();
    }
}
```

```
PS C:\Cognizant\Week-1\FactoryMethodPatternExample> java Test
Opening a word document
Opening a PDF document
Opening a Excel document
PS C:\Cognizant\Week-1\FactoryMethodPatternExample> 
```

E-commerce Platform Search Function:

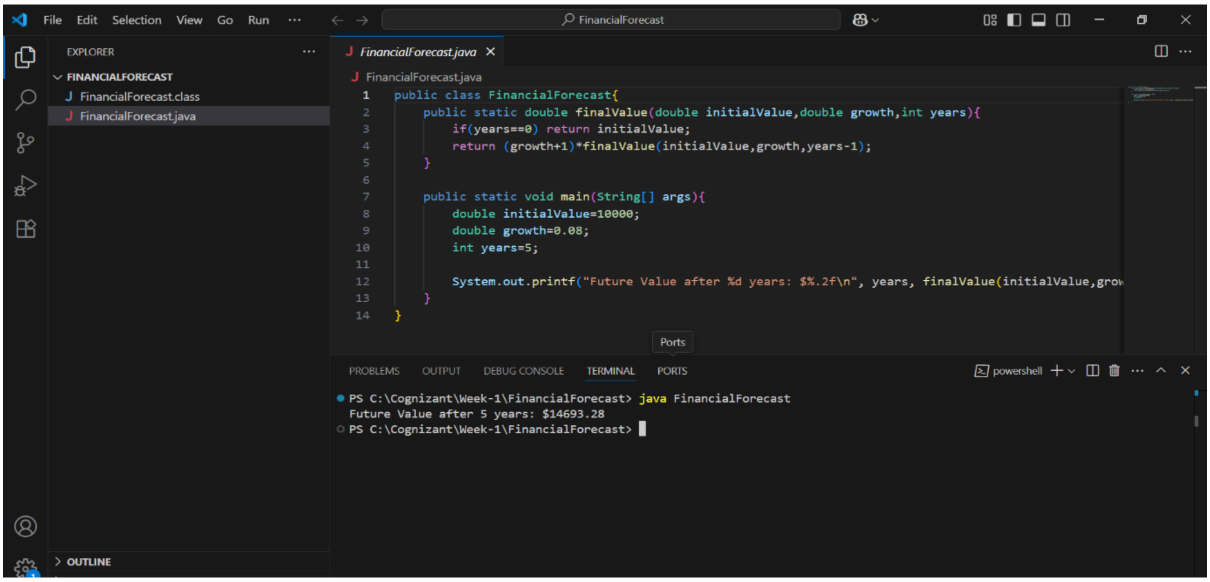


```java
public class SearchTest{
    public static void main(String[] args){
        Product[] products={
            new Product(5,"Shoes","Footwear")
        };

        Product f1=LinearSearch.search(products,"Computer");
        if(f1!=null){
            System.out.println("Linear Search: "+f1.getProductID()+"-"+f1.getProductName()+"("+f1.get
        }
        else{
            System.out.println("Not Found");
        }
        Product f2=BinarySearch.search(products,"Computer");
        if(f2!=null)
            System.out.println("Binary Search: "+f2.getProductID()+"-"+f2.getProductName()+"("+f2.get
        else
            System.out.println("Not Found");
    }
}
```

```
PS C:\Cognizant\Week-1\Products> java SearchTest
4-Computer(Electronics)
4-Computer(Electronics)
PS C:\Cognizant\Week-1\Products>
```

Financial Forecasting:



```java
public class FinancialForecast{
    public static double finalValue(double initialValue,double growth,int years){
        if(years==0) return initialValue;
        return (growth+1)*finalValue(initialValue,growth,years-1);
    }

    public static void main(String[] args){
        double initialValue=10000;
        double growth=0.08;
        int years=5;

        System.out.printf("Future Value after %d years: $%.2f\n", years, finalValue(initialValue,grow
    }
}
```

```
PS C:\Cognizant\Week-1\FinancialForecast> java FinancialForecast
Future Value after 5 years: $14693.28
PS C:\Cognizant\Week-1\FinancialForecast>
```