

Week 1

Algorithms and Data Structures

Exercise 7: Financial Forecasting

Recursion is a method of solving a problem where the solution depends on solving smaller instances of the same problem.

A recursive function:

- Calls itself.
- Has a base case (to stop recursion).
- Has a recursive case (to reduce the problem size).

Why use recursion?

- Simplifies complex problems (e.g., Fibonacci, Tree Traversal).
- Natural fit for problems with repeated sub-structure.

Future Value Formula (for compound growth)

$$FV = PV \times (1+r)^n$$

Where:

- FV = Future Value
- PV = Present Value (initial amount)
- r = annual growth rate (in decimal)
- n = number of years

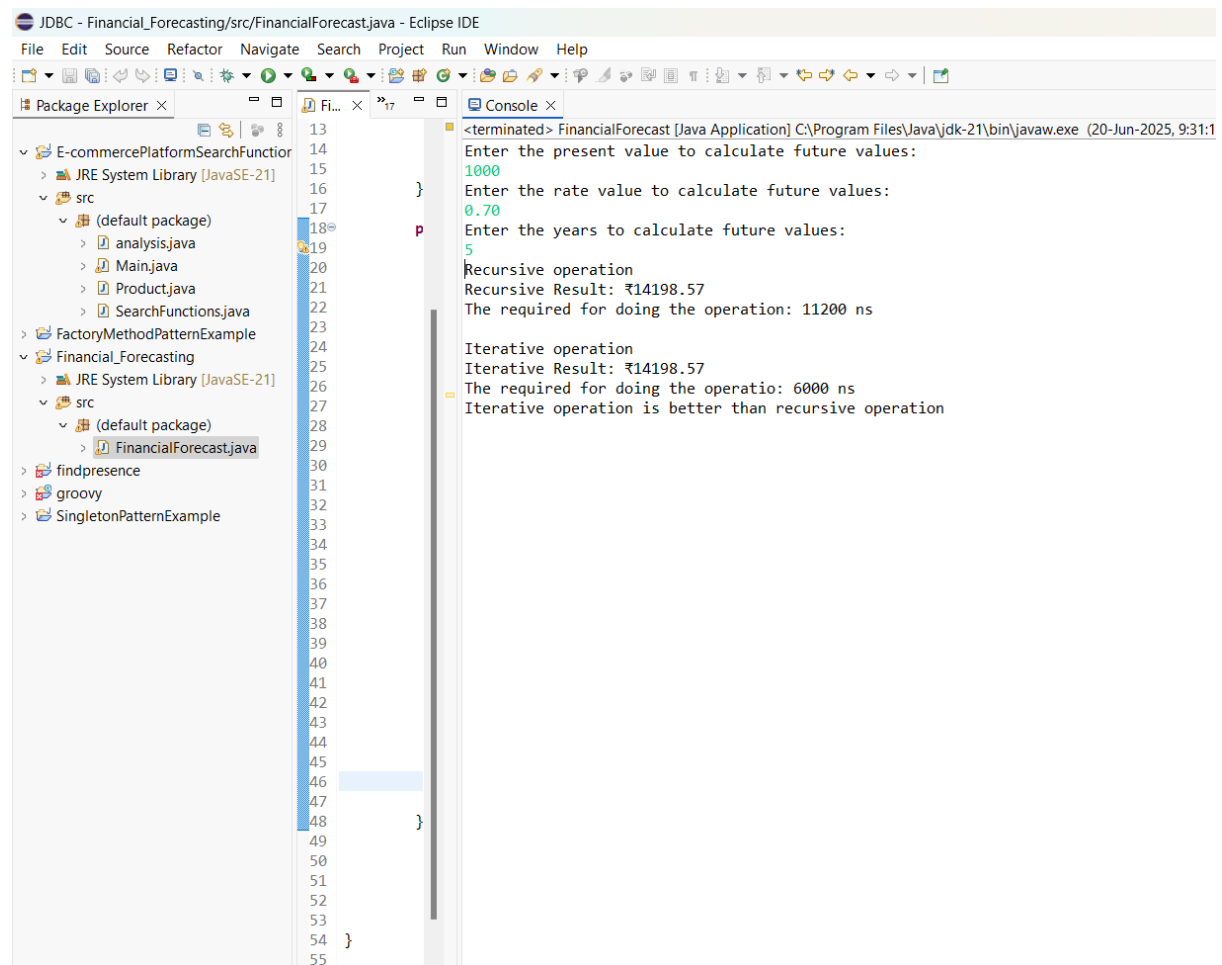
How to Optimize?

This recursion is already tail-recursive (no repeated subproblems), so it is efficient enough for small n.

Optimization Techniques:

- **Memoization:** Useful if results are reused (not needed here).
- **Iteration:** For large n, recursion could be replaced with iteration to avoid stack overflow.

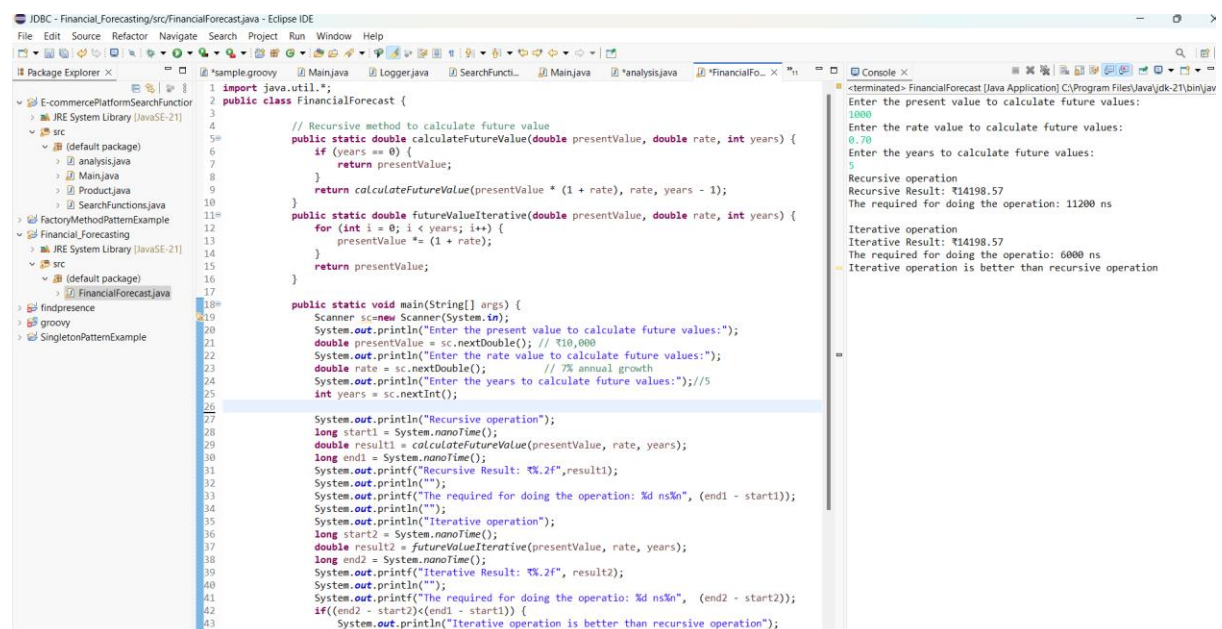
Output snapshot:



The screenshot shows the Eclipse IDE interface. The Package Explorer on the left displays the project structure, including the 'Financial_Forecasting' package and its sub-packages. The main editor window shows the 'FinancialForecast.java' file. The code is a Java class with two static methods: 'calculateFutureValue' and 'main'. The 'calculateFutureValue' method is a recursive method that calculates the future value of an investment. The 'main' method is a standard Java main method that takes command-line arguments and prints the results. The Console window on the right shows the output of the program, which includes the recursive calculation results and the time taken for the operation.

```
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  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52  
53  
54  
55
```

```
<terminated> FinancialForecast [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (20-Jun-2025, 9:31:1  
Enter the present value to calculate future values:  
1000  
Enter the rate value to calculate future values:  
0.70  
Enter the years to calculate future values:  
5  
Recursive operation  
Recursive Result: ₹14198.57  
The required for doing the operation: 11200 ns  
  
Iterative operation  
Iterative Result: ₹14198.57  
The required for doing the operatio: 6000 ns  
Iterative operation is better than recursive operation
```



The screenshot shows the Eclipse IDE interface. The Package Explorer on the left displays the project structure, including the 'Financial_Forecasting' package and its sub-packages. The main editor window shows the 'FinancialForecast.java' file. The code is a Java class with two static methods: 'calculateFutureValue' and 'main'. The 'calculateFutureValue' method is a recursive method that calculates the future value of an investment. The 'main' method is a standard Java main method that takes command-line arguments and prints the results. The Console window on the right shows the output of the program, which includes the recursive calculation results and the time taken for the operation.

```
1 import java.util.*;  
2 public class FinancialForecast {  
3  
4     // Recursive method to calculate future value  
5     public static double calculateFutureValue(double presentValue, double rate, int years) {  
6         if (years == 0) {  
7             return presentValue;  
8         }  
9         return calculateFutureValue(presentValue * (1 + rate), rate, years - 1);  
10    }  
11  
12    public static double futureValueIterative(double presentValue, double rate, int years) {  
13        for (int i = 0; i < years; i++) {  
14            presentValue *= (1 + rate);  
15        }  
16        return presentValue;  
17    }  
18  
19    public static void main(String[] args) {  
20        Scanner sc = new Scanner(System.in);  
21        System.out.println("Enter the present value to calculate future values:");  
22        double presentValue = sc.nextDouble(); // ₹10,000  
23        System.out.println("Enter the rate value to calculate future values:");  
24        double rate = sc.nextDouble(); // 7% annual growth  
25        System.out.println("Enter the years to calculate future values:"); // 5  
26        int years = sc.nextInt();  
27  
28        System.out.println("Recursive operation");  
29        long start1 = System.nanoTime();  
30        double result1 = calculateFutureValue(presentValue, rate, years);  
31        long end1 = System.nanoTime();  
32        System.out.printf("Recursive Result: ₹%.2f", result1);  
33        System.out.println("");  
34        System.out.printf("The required for doing the operation: %d ns", (end1 - start1));  
35        System.out.println("");  
36        System.out.println("Iterative operation");  
37        long start2 = System.nanoTime();  
38        double result2 = futureValueIterative(presentValue, rate, years);  
39        long end2 = System.nanoTime();  
40        System.out.printf("Iterative Result: ₹%.2f", result2);  
41        System.out.println("");  
42        System.out.printf("The required for doing the operation: %d ns", (end2 - start2));  
43        if ((end2 - start2) < (end1 - start1)) {  
44            System.out.println("Iterative operation is better than recursive operation");  
45        }  
46    }  
47 }  
48  
49  
50  
51  
52  
53  
54  
55
```

```
<terminated> FinancialForecast [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (20-Jun-2025, 9:31:1  
Enter the present value to calculate future values:  
1000  
Enter the rate value to calculate future values:  
0.70  
Enter the years to calculate future values:  
5  
Recursive operation  
Recursive Result: ₹14198.57  
The required for doing the operation: 11200 ns  
  
Iterative operation  
Iterative Result: ₹14198.57  
The required for doing the operatio: 6000 ns  
Iterative operation is better than recursive operation
```