**Week 1**

**Hands-on :-**

**Exercise 7: Financial Forecasting**

1. **Explain the concept of recursion and how it can simplify certain problems?**

* **Recursion is a programming concept where a method calls itself to solve smaller instances of the same problem.**

**-**Code is often **shorter and cleaner**

**-**Makes **complex problems easier to understand**

1. **Setup :**

**FV(n) = FV(n-1) \* (1 + rate)**

This is the formula for predicting the future value for n years.

FV(n) – Future Value for n years.

1. **Implementation:**

**Code:**

**Forecasting.java**

public class Forecasting {

public static double forecastRecursive(double initial, double rate, int years) {

if (years == 0) {

return initial;

}

return forecastRecursive(initial, rate, years - 1) \* (1 + rate);

}

}

**Optimized.java:**

public class Optimized {

public static double forecastIterative(double initial, double rate, int years) {

double value = initial;

for (int i = 0; i < years; i++) {

value \*= (1 + rate);

}

return value;

}

}

**Test.java:**

import java.util.Scanner;

public class test {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.print("Enter initial value: ");

double initial = sc.nextDouble();

System.out.print("Enter growth rate : ");

double rate = sc.nextDouble();

System.out.print("Enter number of years: ");

int years = sc.nextInt();

double recursiveResult = Forecasting.forecastRecursive(initial, rate, years);

System.out.println("FOR RECURSIVE ");

System.out.printf("Future value : ₹%.2f%n", recursiveResult);

double iterativeResult = Optimized.forecastIterative(initial, rate, years);

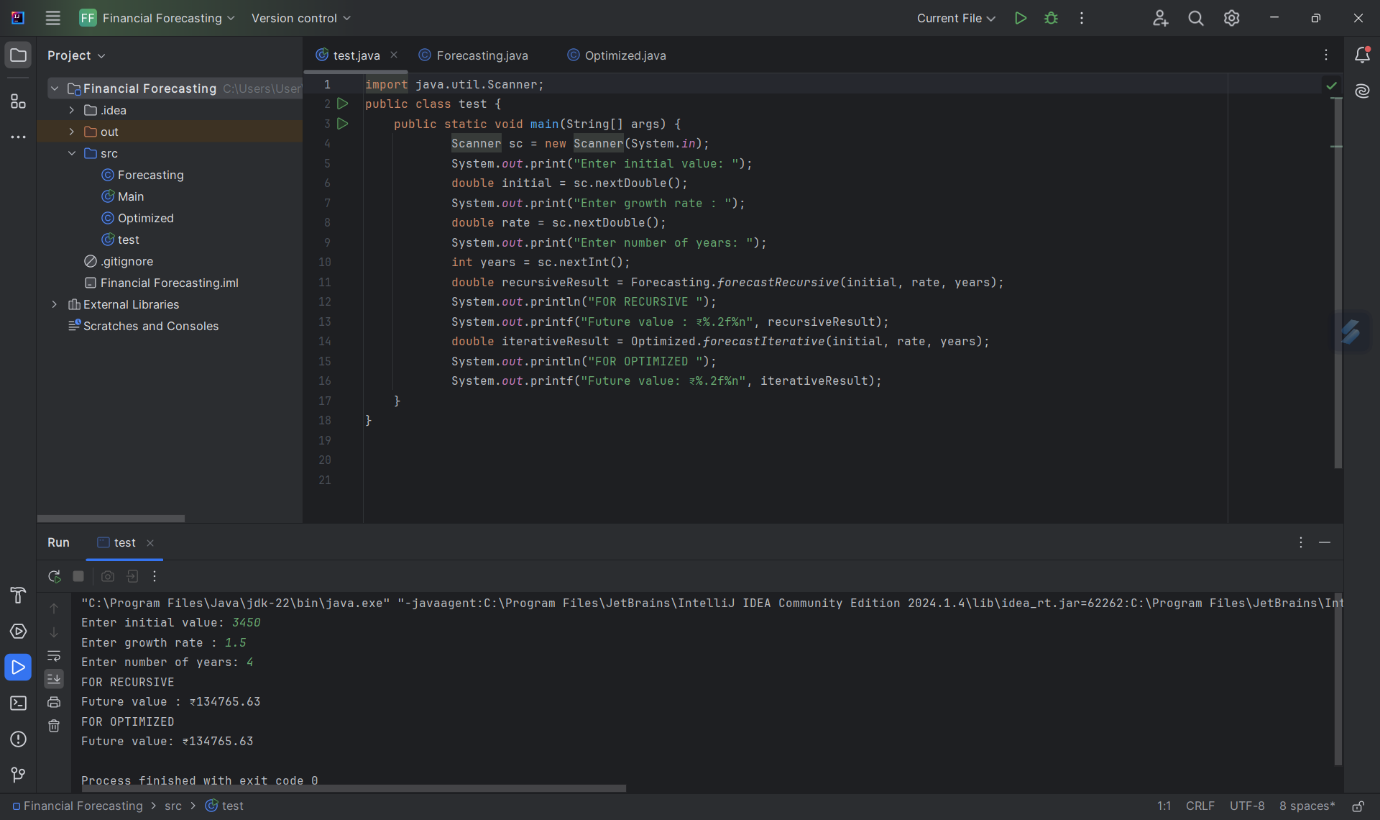
System.out.println("FOR OPTIMIZED ");

System.out.printf("Future value: ₹%.2f%n", iterativeResult);

}

}

**OUTPUT : -**



1. **Analysis:**

**Time complexity of My recursive algorithm : - O(n)**

So for n years, the function is called **n+1 times (including base case)**

**There is one multiplication per call**

**Hence, Time Complexity = O(n)**

**Explain how to optimize the recursive solution to avoid excessive computation.**

* **Use Iteration Instead of Recursion**

**Replace the recursive calls with a simple loop.**

1. **Uses constant space**
2. **Eliminates the need for function call overhead**

**Benefits:**

Time Complexity: **O(n)**

Space Complexity: **O(1)**

Efficient for **large n**(No. of Years)