**CTS DIGITAL NURTURE - 4.0 JAVA FSE**

**WEEK 1 – ALGORITHMS\_DATA STRUCTURES**

MANDATORY HANDS-ON 2:

QUESTION:

**Financial Forecasting**

**Scenario:**

You are developing a financial forecasting tool that predicts future values based on past data.

**Steps:**

1. **Understand Recursive Algorithms:**
   * Explain the concept of recursion and how it can simplify certain problems.
2. **Setup:**
   * Create a method to calculate the future value using a recursive approach.
3. **Implementation:**
   * Implement a recursive algorithm to predict future values based on past growth rates.
4. **Analysis:**
   * Discuss the time complexity of your recursive algorithm.
   * Explain how to optimize the recursive solution to avoid excessive computation.

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

Recursion is a programming technique where a function calls itself to solve smaller instances of the same problem. It continues to do this until it reaches a base case, which stops the recursion. Recursion helps to solve complex problems by breaking them down into simpler sub-problems, especially when the task has a repetitive or a nested structure.

Example : Factorial, Fibonacci Series, Tower of Hanoi

PROGRAM:

**FinanceForecaster.java**

package financeForecasting;

public class FinanceForecaster {

public double calculateFutureValue(double presentValue, double growthRate, int periods) {

if (periods == 0) {

return presentValue;

} else {

double previousPeriodValue = calculateFutureValue(presentValue, growthRate, periods - 1);

return previousPeriodValue \* (1 + growthRate);

}

}

public static void main(String[] args) {

FinanceForecaster forecaster = new FinanceForecaster();

double initialInvestment = 1000.00;

double annualGrowthRate = 0.07;

int forecastPeriods = 5;

double futureValue = forecaster.calculateFutureValue(initialInvestment, annualGrowthRate, forecastPeriods);

System.***out***.printf("Initial Investment: $%.2f%n", initialInvestment);

System.***out***.printf("Annual Growth Rate: %.2f%%%n", annualGrowthRate \* 100);

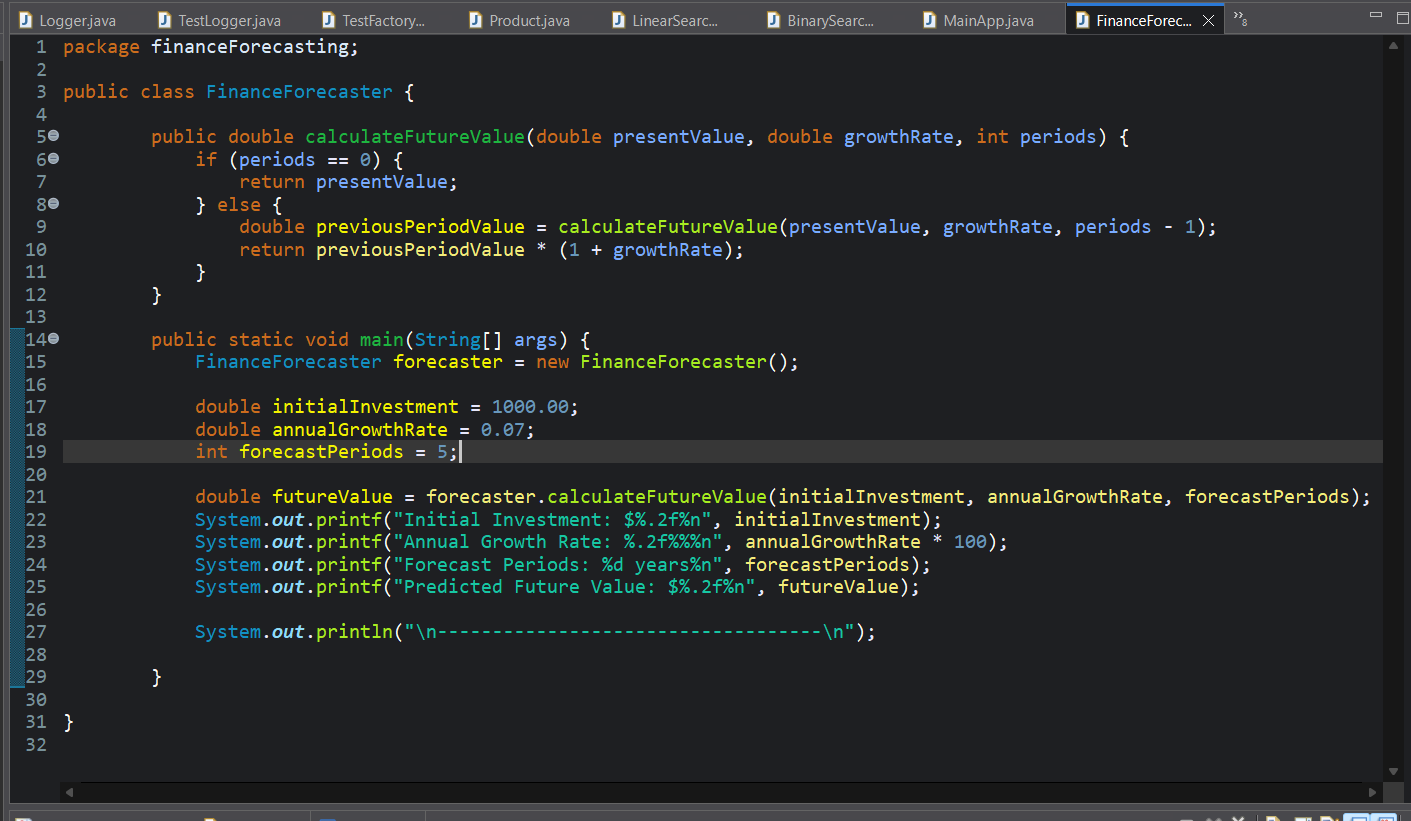
System.***out***.printf("Forecast Periods: %d years%n", forecastPeriods);

System.***out***.printf("Predicted Future Value: $%.2f%n", futureValue);

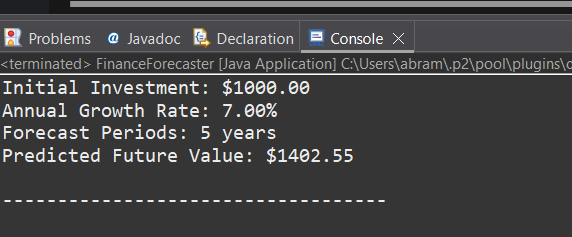
System.***out***.println("\n-----------------------------------\n");

}

}



OUTPUT:



**4. Discuss the time complexity of your recursive algorithm.**

The recursive method calculates the future value by calling itself for each period. Each recursive call processes one period, so the total number of calls is equal to the number of periods.

So, Time Complexity: O(n)  
Where n = number of periods (years/months).

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

**1.Iterative Method:**

Recursion adds overhead due to repeated function calls and stack usage. An iterative version avoids this and is more efficient.

For example,

double result = presentValue;

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

result \*= (1 + growthRate);

}

This does the same calculation with less memory usage and no risk of stack overflow.

**2.Direct Mathematical Formula:**

Future Value=Present Value×(1+r) n,

Implementing this formula avoids recursion entirely and provides the most optimal way as it reduces the problem to a constant number of operations.

**Time Complexity:** **O(1).**

For Example,

public double optimizedFutureValue(double presentValue, double rate, int periods) {

return presentValue \* Math.pow(1 + rate, periods);

}