Exercise 7: Financial Forecasting

# Objective

To develop a financial forecasting tool using recursive algorithms that predicts future values based on past growth rates. This exercise demonstrates the implementation of recursion and optimization techniques in Java.

# Java Code

import java.util.Scanner;  
  
public class FinancialForecast {  
  
 // Recursive method to forecast future value  
 public static double forecastValue(double presentValue, double rate, int years) {  
 if (years == 0) return presentValue;  
 return forecastValue(presentValue, rate, years - 1) \* (1 + rate);  
 }  
  
 // Optimized method using memoization (optional)  
 public static double forecastValueMemo(double presentValue, double rate, int years, double[] memo) {  
 if (years == 0) return presentValue;  
 if (memo[years] != 0) return memo[years];  
 memo[years] = forecastValueMemo(presentValue, rate, years - 1, memo) \* (1 + rate);  
 return memo[years];  
 }  
  
 public static void main(String[] args) {  
 Scanner sc = new Scanner(System.in);  
  
 System.out.print("Enter present value: ");  
 double pv = sc.nextDouble();  
  
 System.out.print("Enter annual growth rate (e.g., 0.1 for 10%): ");  
 double rate = sc.nextDouble();  
  
 System.out.print("Enter number of years: ");  
 int years = sc.nextInt();  
  
 // Call simple recursive method  
 double future = forecastValue(pv, rate, years);  
 System.out.printf("Future value (recursive): %.2f\n", future);  
  
 // Call optimized recursive method  
 double[] memo = new double[years + 1];  
 double optimized = forecastValueMemo(pv, rate, years, memo);  
 System.out.printf("Future value (memoized): %.2f\n", optimized);  
 }  
}

# Sample Input & Output

Input:  
Enter present value: 10000  
Enter annual growth rate (e.g., 0.1 for 10%): 0.08  
Enter number of years: 5  
  
Output:  
Future value (recursive): 14693.28  
Future value (memoized): 14693.28

# Analysis

Time Complexity:  
- Recursive: O(n) — one recursive call for each year.  
- Memoized: O(n) — avoids recomputation using storage.  
  
Optimization:  
Memoization avoids redundant recursive calls by storing already computed values in an array. This significantly reduces computation for large values of 'n' and makes recursion more efficient.