**Algorithms Data Structures**

**Exercise 7: Financial Forecasting**

**SCENARIO:**

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

**SOURCE CODE :**

**FinancialForecast.java**

package com.forecasting.tool;

public class FinancialForecast {

// Recursive method to calculate future value

public static double calculateFutureValueRecursive(double presentValue, double growthRate, int years) {

if (years == 0) {

return presentValue; // base case

}

return (1 + growthRate) \* calculateFutureValueRecursive(presentValue, growthRate, years - 1); // recursive case

}

// Optimized: Iterative version

public static double calculateFutureValueIterative(double presentValue, double growthRate, int years) {

double futureValue = presentValue;

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

futureValue \*= (1 + growthRate);

}

return futureValue;

}

public static void main(String[] args) {

double presentValue = 10000; // Starting amount

double annualGrowthRate = 0.08; // 8% growth rate

int years = 5;

// Recursive calculation

long startTimeRec = System.nanoTime();

double futureValueRec = calculateFutureValueRecursive(presentValue, annualGrowthRate, years);

long endTimeRec = System.nanoTime();

// Iterative calculation

long startTimeIter = System.nanoTime();

double futureValueIter = calculateFutureValueIterative(presentValue, annualGrowthRate, years);

long endTimeIter = System.nanoTime();

System.out.printf("Recursive Future Value after %d years: %.2f\n", years, futureValueRec);

System.out.printf("Iterative Future Value after %d years: %.2f\n", years, futureValueIter);

System.out.println("\n=== Time Complexity Analysis ===");

System.out.println("Recursive Method: O(n), Space: O(n) due to call stack");

System.out.println("Iterative Method: O(n), Space: O(1)");

System.out.println("\nExecution Time:");

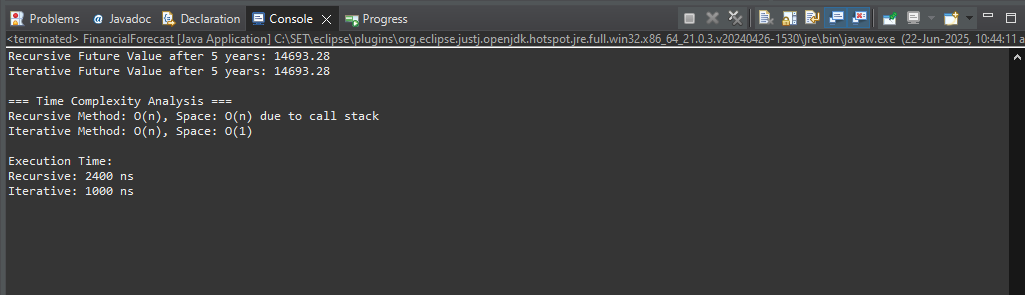
System.out.println("Recursive: " + (endTimeRec - startTimeRec) + " ns");

System.out.println("Iterative: " + (endTimeIter - startTimeIter) + " ns");

}

}

**OUTPUT :**

****