**Exercise 7: Financial Forecasting**

**Scenario:**  
You are developing a financial forecasting tool that estimates future values by analyzing trends in historical data. The goal is to implement a recursive approach to project future financial figures based on prior growth patterns.

**Steps to Complete the Exercise**

**1. Understand Recursion Conceptually**

* Explain how recursion works by solving a problem through smaller, repeated subproblems.
* Highlight its usefulness in scenarios where problems exhibit self-similarity or follow a pattern based on previous outcomes (like compound growth).

**2. Design the Recursive Framework**

* Define a recursive function that forecasts financial values.
* Identify inputs such as:
  + currentValue – the known financial value.
  + growthRate – expected percentage increase per period.
  + periodsRemaining – how many future periods to forecast.

**3. Implement the Recursive Forecast Function**

* Write a recursive function where each call computes the next period’s value as:

ini

CopyEdit

nextValue = currentValue \* (1 + growthRate)

* Recursively call the function until the desired number of periods has been forecasted.

**4. Evaluate Performance**

* Analyze the **time and space complexity** of the recursive solution.
* Discuss potential inefficiencies, especially with overlapping subproblems.
* Propose **memoization** to cache already-computed values, improving performance significantly.

Source code :

public class FinancialForecasting {  
 public static void main(String[] args) {  
 double initialAmount = 10000;  
 double growthRate = 0.08;  
 int years = 5;  
  
 double futureValue = ForecastCalculator.*calculateFutureValue*(initialAmount, growthRate, years);  
 System.*out*.println("Estimated Future Value (Recursive): " + futureValue);  
 }  
}  
class ForecastCalculator {  
 public static double calculateFutureValue(double currentValue, double rate, int yearsLeft) {  
 if (yearsLeft == 0) return currentValue;  
 return *calculateFutureValue*(currentValue, rate, yearsLeft - 1) \* (1 + rate);  
 }  
}

Output :

