public class FinancialForecast {

// Recursive function for constant growth rate

public static double forecastRecursive(double initialAmount, double growthRate, int years) {

// Base case

if (years == 0) {

return initialAmount;

}

// Recursive case

return forecastRecursive(initialAmount, growthRate, years - 1) \* (1 + growthRate);

}

// Iterative version (optimized)

public static double forecastIterative(double initialAmount, double growthRate, int years) {

double result = initialAmount;

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

result \*= (1 + growthRate);

}

return result;

}

// Recursive version for variable growth rates with memoization

public static double forecastWithVariableGrowth(double initialAmount, double[] growthRates, int year, Double[] memo) {

if (year == 0) return initialAmount;

if (memo[year] != null) return memo[year];

memo[year] = forecastWithVariableGrowth(initialAmount, growthRates, year - 1, memo) \* (1 + growthRates[year - 1]);

return memo[year];

}

public static void main(String[] args) {

double initialAmount = 10000;

double constantGrowthRate = 0.08; // 8%

int years = 5;

// 1. Recursive Forecast

double valueRecursive = forecastRecursive(initialAmount, constantGrowthRate, years);

System.out.printf("Recursive Forecast (%.2f%% growth for %d years): %.2f\n",

constantGrowthRate \* 100, years, valueRecursive);

// 2. Iterative Forecast

double valueIterative = forecastIterative(initialAmount, constantGrowthRate, years);

System.out.printf("Iterative Forecast (%.2f%% growth for %d years): %.2f\n",

constantGrowthRate \* 100, years, valueIterative);

// 3. Variable Growth Rates with Memoization

double[] variableGrowthRates = {0.05, 0.06, 0.07, 0.08, 0.09}; // Growth rate changes each year

Double[] memo = new Double[years + 1];

double valueVariable = forecastWithVariableGrowth(initialAmount, variableGrowthRates, years, memo);

System.out.printf("Variable Growth Forecast (5 years): %.2f\n", valueVariable);

}

}

**Output Example:**

