**Exercise 7: Financial Forecasting**

**Scenario:**

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

**Recursion**

The process in which a function calls itself directly or indirectly is called recursion

For financial forecasting:

* Base case: When no periods remain (periods == 0), return the current value
* Recursive case: Apply growth rate and repeat for one fewer period
* Simplifies compound growth modeling:  
  future\_value = current\_value × (1 + growth\_rate) per period

**Implementation:**

public class FinancialForecast {

public static double calculateFutureValue(

double currentValue,

double growthRate,

int periods

) {

if (periods == 0) {

return currentValue;

}

double nextValue = currentValue \* (1 + growthRate);

return calculateFutureValue(nextValue, growthRate, periods - 1);

}

public static void main(String[] args) {

double investment = 1000.0;

double annualGrowth = 0.05;

int years = 3;

double result = calculateFutureValue(investment, annualGrowth, years);

System.out.printf("Projected value after %d years: $%.2f", years, result);

}

}

## **Time Complexity Analysis**

Time : O(n)

Space: O(n)

Optimization : (Iterative approach):

public class FinancialForecastIterative {

public static double calculateFutureValue(double currentValue, double growthRate, int periods) {

double result = currentValue;

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

result \*= (1 + growthRate);

}

return result;

}

public static void main(String[] args) {

double initialInvestment = 1000.0;

double annualGrowthRate = 0.05;

int years = 3;

double result = calculateFutureValue(initialInvestment, annualGrowthRate, years);

System.out.printf("Iterative: Projected value after %d years: $%.2f", years, result);

}

}

Complexity

* Time: O(n)
* Space: O(1)

Output:  
