**Exercise 7: Financial Forecasting**

**1)Explain the concept of recursion and how it can simplify certain problems?**

Recursion is a technique wherea function call itself.

**Recursive Case**: the function calls itself with a smaller input.

**Base Case**: the condition where the recursion stops.

**Implementation:**

package com.EcommercePlatform;

import java.util.\*;

//Class Definition

public class FinancialForecast {

// here using Recursive Forecasting Method

public static double predictFutureValue(double value, double rate, int years) {

if (years == 0)

return value;

return *predictFutureValue*(value \* (1 + rate), rate, years - 1);

}

//Optimized Method Using Math.pow

public static double predictEfficient(double value, double rate, int years) {

return value \* Math.*pow*(1 + rate, years);

}

//Main Method to Accept Input and Show Output

public static void main(String[] args) {

Scanner first = new Scanner(System.*in*);

// Input

System.*out*.print("Enter current value: ");

double currentValue = first.nextDouble();

System.*out*.print("Enter annual growth rate (e.g., 0.05 for 5%): ");

double growingRate = first.nextDouble();

System.*out*.print("Enter number of years: ");

int years = first.nextInt();

// Call recursive method

double futureRecursive = *predictFutureValue*(currentValue, growingRate, years);

// Call optimized method

double futureEfficient = *predictEfficient*(currentValue, growingRate, years)

// Output

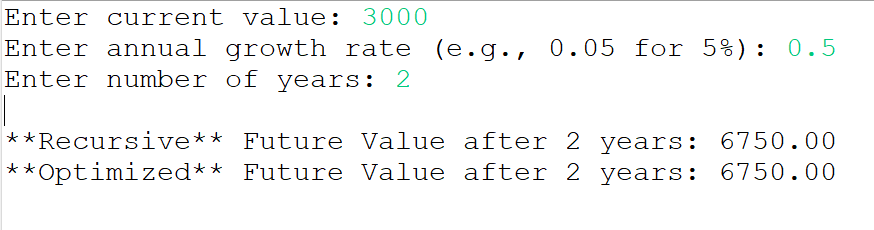
System.*out*.printf("\n\*\*Recursive\*\* Future Value after %d years: %.2f\n", years, futureRecursive);

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

}

}

**Output :**



**Analysis:**

Time Complexity of the Recursive Algorithm

The recursive financial forecasting algorithm calls itself once for each year to project future values. This results in a time complexity of O(n), where n is the number of years. Each recursive call is kept in the call stack until it reaches the base case, leading to a space complexity of O(n). For a large number of years, this can cause stack overflow or slow performance.

Optimizing the Recursive Solution

To cut down on unnecessary computation and stack usage, you can improve the recursive solution in two ways:

Convert to Iterative: Swap recursion with a loop that multiplies the current value by the growth rate each year. This reduces the space complexity to O(1) and eliminates the risk of stack overflow.

Use a Mathematical Formula: The most efficient method uses the compound interest formula:

Future Value = Current Value × (1 + Rate) ^ n

This method has a constant time and space complexity of O(1) and works well for production systems.