**Exercise 7: Financial Forecasting**

**Understanding the Problem:**  
Financial forecasting often involves projecting future values based on growth rates. Recursion helps solve such problems when current results depend on previous results.

**Setup and Implementation:**  
The program reads:

* Initial investment amount
* Growth rate (as a decimal)
* Number of years

It recursively calculates the future value using compound interest logic.

**Java Code:**

import java.util.\*;

public class FinancialForecast {

static double futureValue(double value, double rate, int years) {

if (years == 0) return value;

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

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.print("Enter initial value: ");

double value = sc.nextDouble();

System.out.print("Enter growth rate (e.g., 0.1 for 10%): ");

double rate = sc.nextDouble();

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

int years = sc.nextInt();

double result = futureValue(value, rate, years);

System.out.printf("Predicted future value: %.2f\n", result);

}

}

**Time Complexity Analysis:**

* Recursive Calls: O(n), where n = number of years
* Space Complexity: O(n) due to call stack

**Optimization Discussion:**  
This recursive approach is readable but can lead to stack overflow for very large n. It can be optimized by:

* Using iteration instead of recursion (avoids recursion overhead)
* Using memoization (not needed here since no overlapping subproblems)

For simple linear projections, iterative logic is generally preferred in professional finance software.