**Exercise 7: Financial Forecasting**

1. **Understand Recursive Algorithms**

**What is Recursion?** Recursion is a method in which a function calls itself to solve smaller instances of a problem. It is especially useful when the problem can be divided into similar sub-problems.

For example, to calculate the future value based on a compound growth formula, we can apply recursion for simplicity.

**2. Setup**

We will calculate future value using:

FV = PV \* (1 + r)^n Where:

* PV = Present Value
* r = Annual growth rate
* n = Number of years

We’ll implement recursive methods for calculating this.

1. **Implementation**

public class Financial\_Forecasting {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.println("----- Financial Forecast Tool -----");

System.out.print("Enter Present Value (\u20B9): ");

double presentValue = sc.nextDouble();

System.out.print("Enter Annual Growth Rate (%): ");

double growthRatePercent = sc.nextDouble();

double growthRate = growthRatePercent / 100.0;

System.out.print("Enter Number of Years: ");

int years = sc.nextInt();

double future1 = futureValueRecursive(presentValue, growthRate, years);

System.out.printf("Future Value (Recursive): \u20B9%.2f%n", future1);

double future2 = futureValueTail(presentValue, growthRate, years);

System.out.printf("Future Value (Tail-Recursive): \u20B9%.2f%n", future2);

double future3 = futureValueIterative(presentValue, growthRate, years);

System.out.printf("Future Value (Iterative): \u20B9%.2f%n", future3);

}

public static double futureValueRecursive(double pv, double rate, int years) {

if (years == 0) return pv;

return futureValueRecursive(pv \* (1 + rate), rate, years - 1);

}

public static double futureValueTail(double pv, double rate, int years) {

return helper(pv, rate, years);

}

private static double helper(double value, double rate, int years) {

if (years == 0) return value;

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

}

public static double futureValueIterative(double pv, double rate, int years) {

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

pv \*= (1 + rate);

}

return pv;

}

}

**4. Analysis**

**Time Complexity:**

* Recursive and Tail-Recursive: O(n) time, O(n) space (due to recursion stack)
* Iterative: O(n) time, O(1) space (more optimal in Java)

**Conclusion**

Recursive techniques make logic simpler, but iterative solutions are more efficient in Java due to lack of tail-call optimization. Always choose the method based on readability, efficiency, and platform behavior.

