### 

| ***Exercise 7: Financial Forecasting*** |
| --- |

### 

### **1. Understand Recursive Algorithms ->**

* **Recursion** is a method where the solution to a problem depends on solutions to smaller instances of the same problem.
* A recursive function **calls itself** with simpler inputs, moving toward a **base case** that ends the recursion.
* Recursion often mirrors the natural definition of the problem (e.g., future value depending on current value).
* It **eliminates complex loops**, making the code more readable and easier to implement for problems like tree traversal, dynamic programming, and mathematical sequences.

### **2. Setup ->**

**Method to Calculate Future Value Recursively**

private double predictFutureValue(double currentValue, int periods)

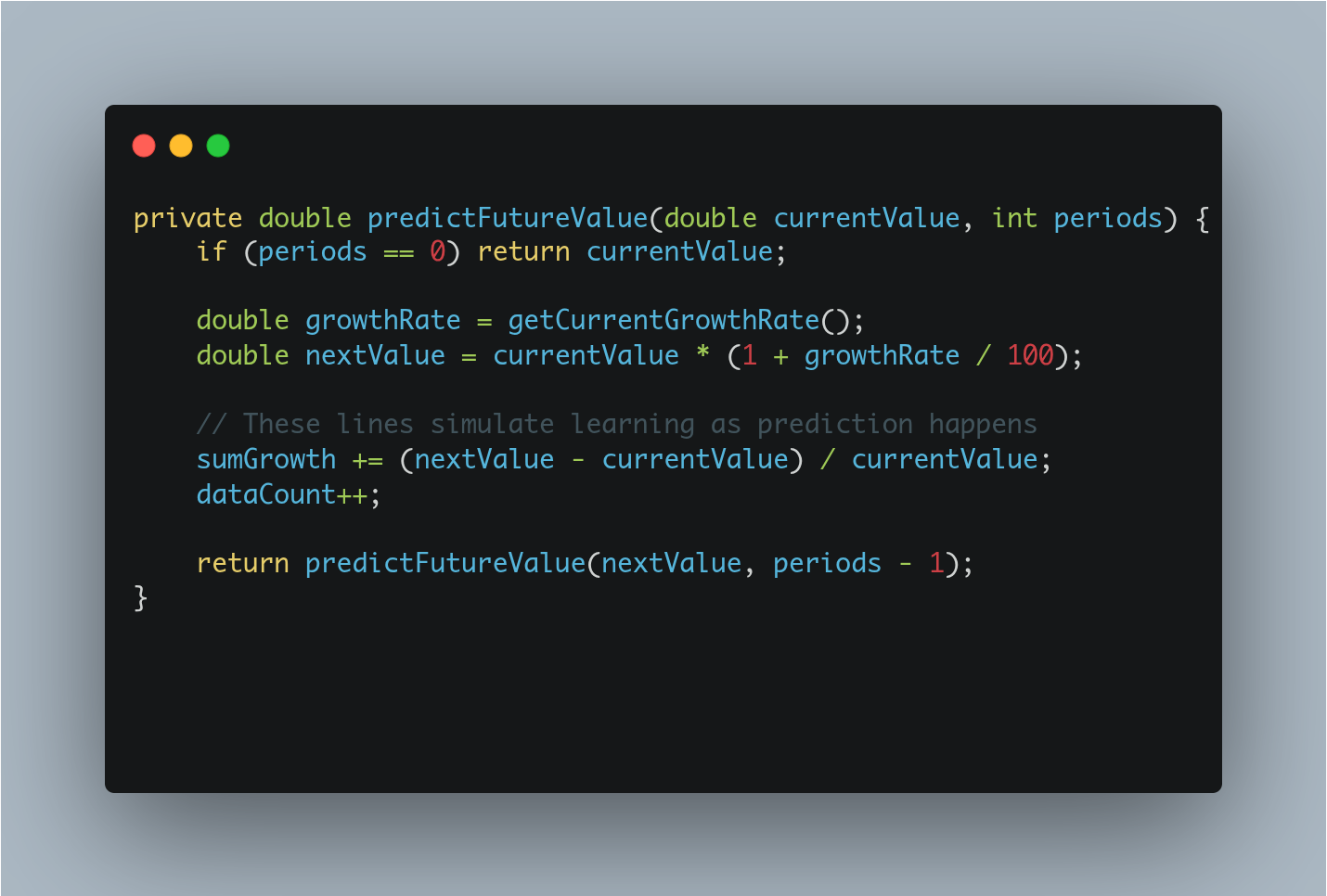
It uses:

* currentValue: the last known financial value.
* periods: how many future steps (time units) to predict.

### **3. Implementation**

FULL SOURCE CODE : https://github.com/AnujMishra-0/WeekWise-Learning-Solutions-Digital-Nurture-4.0-JavaFSE-2025/blob/main/Week%201%20Algorithms\_Data%20Structures/Exercise%207%3A%20Financial%20Forecasting/code/LiveFinancialForecaster.java

**Recursive Forecasting Logic:**



**Explanation:**

* **Base case**: when periods == 0, return the current value — no further prediction needed.
* **Recursive step**:  
  + Calculate the next value based on the current average growth rate.
  + Update internal data (growth, count), and recursively call the function with next value and periods - 1;

### **4. Analysis**

#### **Time Complexity**

Since This recursion runs once for each period, so:  
**Time Complexity**: O(periods)  
**Space Complexity** (due to call stack): O(period)

#### **Optimization to Avoid Excessive Computation**

Recursive methods can lead to **Stack overflow** for large periods..  


**Convert to Iterative Approach**:  
 Replace recursion with a loop to avoid deep call stacks.