**Exercise 2: Financial Forecasting**

**Objective:**

To create a simple financial forecasting tool that can predict how much your investment or savings might grow in future based on past growth trends. It uses recursive approach to predict future values based on historical growth rates.

**Step 1:** Understanding Recursive Algorithms

**What is Recursion?**

Recursion is when a function call itself to solve smaller parts of a problem. It’s like solving a puzzle by breaking it into smaller, more manageable pieces.

**Use of recursion:**

* Makes code shorter and often more intuitive.
* Useful for problems that follow a repetitive pattern.

**Step 2:** Our Forecasting Formula

As we want to create a future value with the help of recursion.

Recursion usage in simple financial model.

Future value in simple financial model can be as follows:

Future Value = present\_value × (1 + growth\_rate) ^ n

Writing a recursive method that uses;

present\_value : amount you start with

growth\_rate : expected annual growth rate – e.g. 0.07 for 7%

duration (in years) : number of years into the future

So, if you invest ₹1000 at 10% annual growth for 5 years, it compounds like magic. We’ll write a Java method that uses recursion to model this year-by-year growth.

**financial\_forecast.java**:

|  |
| --- |
| // recursive method to calculate future value  public static double forecast (double present\_value, double growth\_rate, int duration) {  // base case  if (duration == 0) {  return present\_value;  }  // recursive function  return forecast (present\_value \* (1 + growth\_rate), growth\_rate, duration - 1);  } |

**Step 3**: Implementation

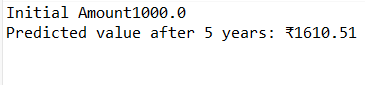
A simple Java program with a recursive method called forecast().  
Here’s how it works:

* It checks if duration = 0 → just returns the amount you started with.
* Or else → it grows the amount by one year, and calls itself for the remaining years

**ForecastRecursive.java:**

|  |
| --- |
| public class ForecastRecursive {  public static void main (String[] args) {  double initial\_amount = 1000; // Starting with ₹1000  double growth\_rate = 0.1; // 10% annual growth  int duration = 5; // Forecast for 5 years  double future\_value = forecast (initial\_amount, growth\_rate, duration);  System.out.println("Predicted value after %d duration: ₹%.2f%n", duration, future\_value);  }  } |

**Expected output:**

****

**Step 4:** Analysis

How fast the performance is?

Time complexity : O(n) --- linear

Space complexity : O(n) --- due to stack

Optimization ideas:

Recursive approaches are elegant but, they can be slow or become complex for large inputs which causes **stack overflow**.

Ways to improve:

* Usage of iteration for large inputs.
* Memorization can be used if values are reused.