# **Assignment: Exercise 7 – Financial Forecasting**

## **3. Implementation: Recursive Forecast Function**

FinancialForecast.java

public class FinancialForecast {

public static double forecastValue(double currentValue, double growthRate, int years) {

if (years == 0) {

return currentValue;

}

return forecastValue(currentValue, growthRate, years - 1) \* (1 + growthRate);

}

public static void main(String[] args) {

double presentValue = 1000;

double growthRate = 0.1; // 10%

int years = 5;

double futureValue = forecastValue(presentValue, growthRate, years);

System.out.println("Forecasted value after " + years + " years: " + Math.round(futureValue \* 100.0) / 100.0);

}

}

## **4. Analysis: Time Complexity and Optimization**

### **Time Complexity:**

The time complexity of this recursive method is **O(n)** where n is the number of years. This is because the function is called once for each year until we reach year 0.

### **Optimization:**

Even though recursion looks neat, it’s not the best for large inputs. It may cause stack overflow or just become slower. We can optimize it by:

* Using **memoization** to store already calculated values so we don’t do the same work again.
* Or simply use a **loop (iteration)** which is more efficient in this case.

## **5. Output**

