**Exercise 7: Financial Forecasting**

**Recursion** is a programming concept where a function calls **itself** in order to solve a problem.

Instead of solving a problem all at once, recursion breaks it down into **smaller sub-problems** of the same type — solving each one until it reaches a **base case** (the simplest version of the problem, which can be solved directly).

**FinancialForecast.java**

interface ForecastStrategy {

double forecast(double currentValue, double growthRate, int years);

}

class RecursiveForecastStrategy implements ForecastStrategy {

@Override

public double forecast(double currentValue, double growthRate, int years) {

if (years == 0) {

return currentValue;

}

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

}

}

class ProductValueForecaster {

private final ForecastStrategy strategy;

public ProductValueForecaster(ForecastStrategy strategy) {

this.strategy = strategy;

}

public double calculateFutureValue(double currentValue, double growthRate, int years) {

return strategy.forecast(currentValue, growthRate, years);

}

}

public class FinancialForecast {

public static void main(String[] args) {

double currentValue = 10000;

double growthRate = 0.10;

int years = 5;

ForecastStrategy strategy = new RecursiveForecastStrategy();

ProductValueForecaster forecaster = new ProductValueForecaster(strategy);

double futureValue = forecaster.calculateFutureValue(currentValue, growthRate, years);

System.out.printf("Forecasted value after %d years: $%.2f\n", years, futureValue);

}

}

**OUTPUT SCREENSHOT:**

