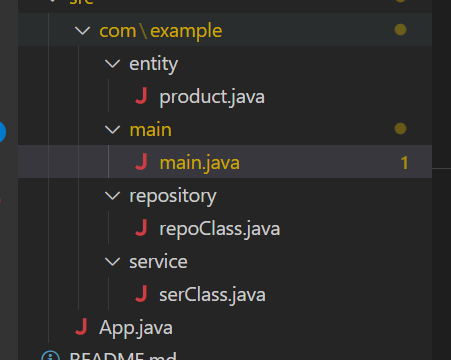
**Financial Forecasting**

Future Value = Present Value × (1 + Rate) ^ Years

Where:

* FV = future value
* PV = present value (initial amount)
* r = growth rate (e.g., 0.05 for 5%)
* n = number of years

**CODE:**



product.java

package com.example.entity;

public class product {

private String productName;

private double initialPrice;

private int growthRate;

private int noyears;

public product(String productName, double initialPrice, int growthRate, int noyears) {

this.productName = productName;

this.initialPrice = initialPrice;

this.growthRate = growthRate;

this.noyears = noyears;

}

public String getProductName() {

return productName;

}

public void setProductName(String productName) {

this.productName = productName;

}

public double getInitialPrice() {

return initialPrice;

}

public void setInitialPrice(double initialPrice) {

this.initialPrice = initialPrice;

}

public int getGrowthRate() {

return growthRate;

}

public void setGrowthRate(int growthRate) {

this.growthRate = growthRate;

}

public int getNoyears() {

return noyears;

}

public void setNoyears(int noyears) {

this.noyears = noyears;

}

}

main.java

package com.example.main;

import java.util.Scanner;

import com.example.service.serClass;

public class main {

private static serClass ser = new serClass();

private static Scanner sc = new Scanner(System.in);

public static void main(String args[]) {

System.out.println();

System.out.println("To calculate Future Value : ");

System.out.println();

System.out.print("Enter the name of the product = ");

String pro = sc.nextLine();

System.out.print("Enter the initial price = ");

Double price = sc.nextDouble();

System.out.print("Enter the growth rate (in %) = ");

int rate = sc.nextInt();

System.out.print("Enter the number of years = ");

int years = sc.nextInt();

System.out.println();

Double rateCon = (double) rate / 100;

Double ans = ser.finalValue(price, rateCon, years);

System.out.println();

System.out.printf("The Final Price of %s is %.2f\n", pro, ans);

System.out.println();

}

}

repoClass.java

package com.example.repository;

public class repoClass {

public Double finalValue(Double initialPrice, Double rate, int years) {

if (years == 0) {

return initialPrice;

}

return (1 + years) \* finalValue(initialPrice, rate, years - 1);

}

}

serClass.java

package com.example.service;

import com.example.repository.repoClass;

public class serClass {

private static repoClass repo = new repoClass();

public Double finalValue(Double initialPrice,Double rate,int years)

{

return repo.finalValue(initialPrice, rate, years);

}

}

## 

| **Approach** | **Code Simplicity** | **Performance** | **Stack Usage** |
| --- | --- | --- | --- |
| **Recursive** | 👍 Simple | ⚠️ Slower | ⚠️ Risk of stack overflow |
| **Iterative** | ✅ Fast & safe | ✅ Better | ✅ No stack growth |
| **Formula** | ✅ Best | ✅ Optimal | ✅ Constant memory |

## 

## **Output :**

