**Exercise 7: Financial Forecasting**

1. Explain the concept of recursion and how it can simplify certain problems.

Recursion is a programming concept where a function calls itself to solve a problem. It works by breaking down a complex problem into smaller, self-similar subproblems, solving those, and then combining the results. This approach is particularly useful when dealing with problems that can be naturally divided into smaller instances of the same problem.

Example: calculating the factorial of a number, n!.

factorial(n) = n × factorial(n - 1)

factorial(1) = 1 ← This is the base case

So, factorial(5) becomes:  
 5 × 4 × 3 × 2 × 1

1. Implementation:

**package** FinancialForecast;

**public** **class** FinancialForecast

{

**public** **static** **double** futureValue(**double** principal, **double** rate, **int** years) {

**if** (years == 0)

**return** principal;

**return** *futureValue*(principal, rate, years - 1) \* (1 + rate); }

**public** **static** **void** main(String[] args) {

**double** initialAmount = 1000;

**double** growthRate = 0.05;

**int** years = 5;

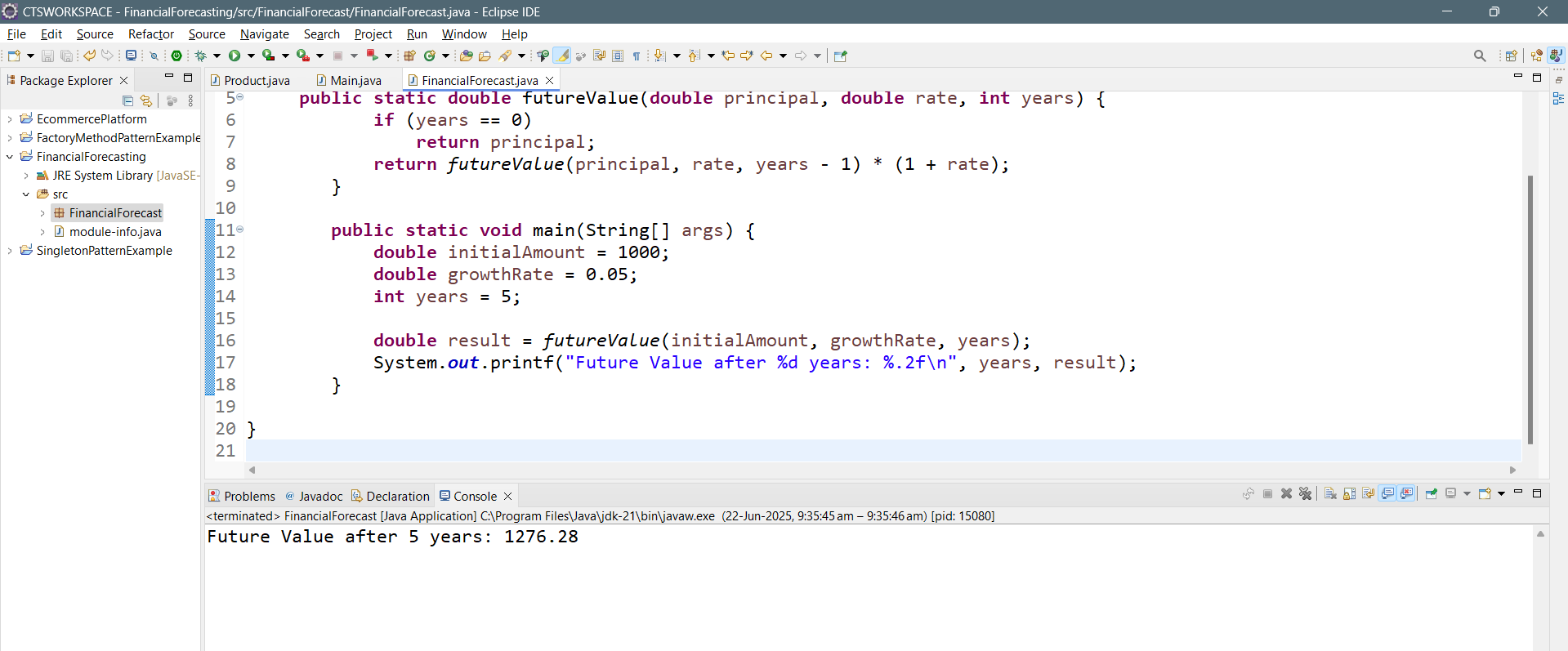
**double** result = *futureValue*(initialAmount, growthRate, years);

System.***out***.printf("Future Value after %d years: %.2f\n", years, result);

}

}

Output:



1. Analysis:

**Time Complexity:**

* **T(n) = T(n-1) + O(1)** ⇒ **O(n)**
* Each recursive call reduces years by 1 until it hits 0.

**Space Complexity:**

* Due to recursion stack: **O(n)**

**Optimization Tips**

1. **Memoization (Top-Down DP)**:
   * Store intermediate results to avoid recalculating them.
2. **Convert to Iterative (Bottom-Up DP)**:
   * Use a loop to compute the result without recursion stack overhead.