**Exercise 7: Financial Forecasting Hands-On**

**Scenario:**

You are developing a financial forecasting tool that predicts future values based on past data.

**Steps:**

1. **Understand Recursive Algorithms:**
   * Explain the concept of recursion and how it can simplify certain problems.
2. **Setup:**
   * Create a method to calculate the future value using a recursive approach.
3. **Implementation:**
   * Implement a recursive algorithm to predict future values based on past growth rates.
4. **Analysis:**
   * Discuss the time complexity of your recursive algorithm.
   * Explain how to optimize the recursive solution to avoid excessive computation.

**1.Understand Recursive Algorithms :**

**Concept of Recursion :**

a. Recursion is a programming technique where a method calls itself to solve a problem. It breaks down a problem into smaller, more manageable sub-problems.

b. Recursion can simplify certain problems, especially those that can be defined in terms of smaller instances of the same problem, such as calculating factorials, Fibonacci numbers, or in this case, predicting future values based on past growth rates.

**2. Set-Up / Procedure :**

**1. Create a New Java Project:**

Create Java Project and Name the project FinancialForecastingTool.

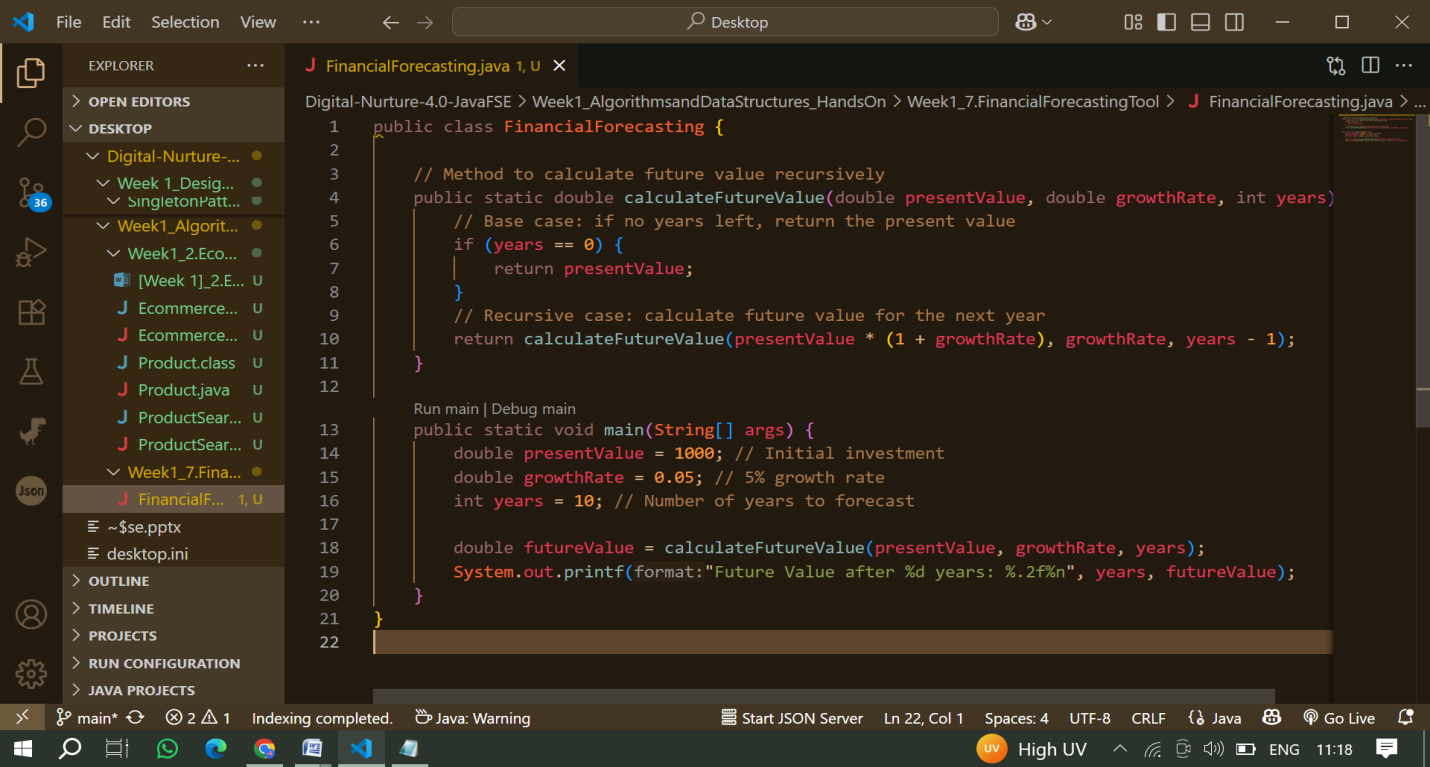
**2. Create a New Java Class:**

Name the file FinancialForecasting.java. Now, write the required code.

**3. IMPLEMENTATION :**

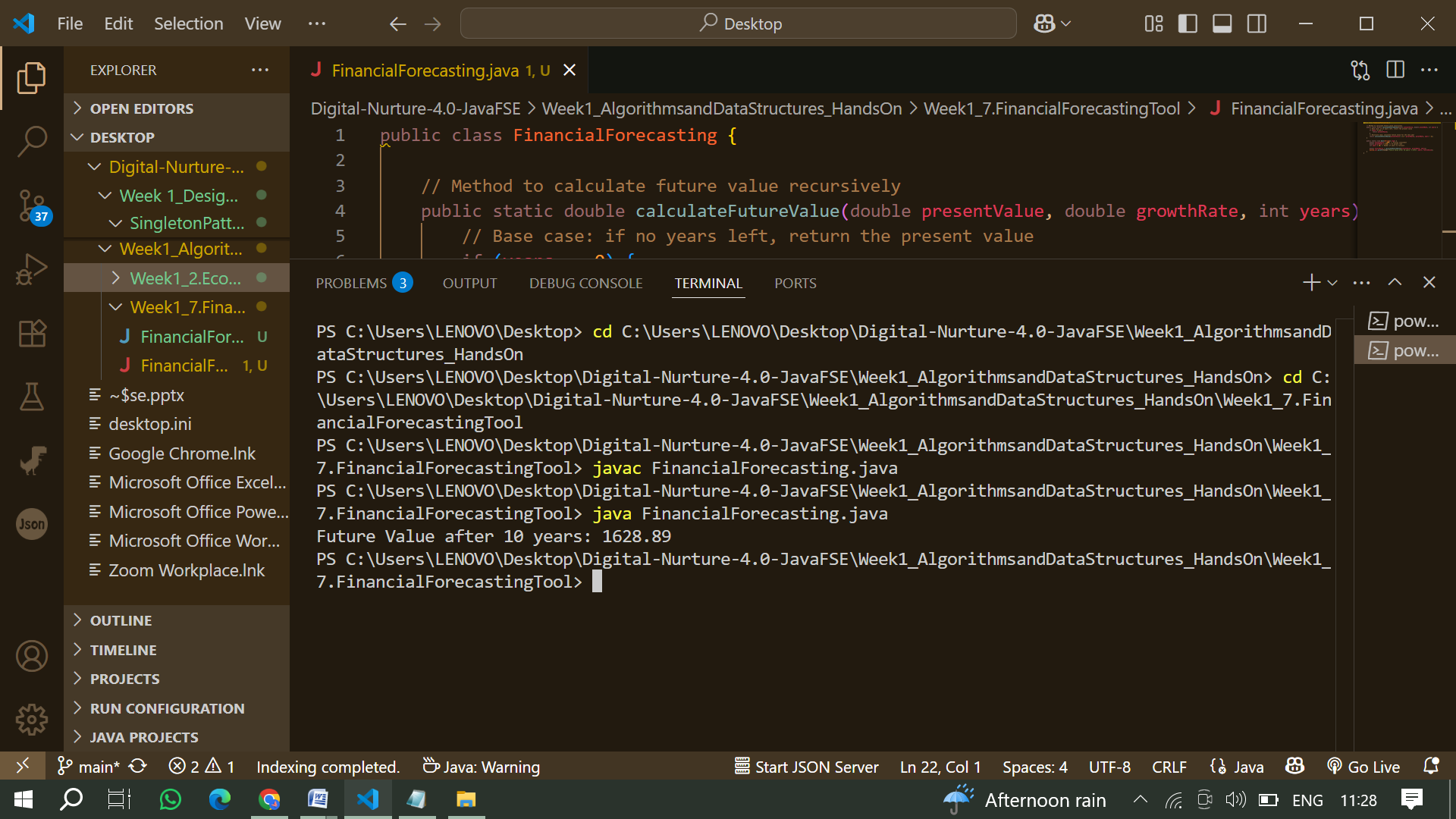
Implement the Recursive Method:

Open FinancialForecasting.java and add the following code:



**Run the Application:**

Compile the Java File and Run the test cases.



**4. Analysis :**

**a. Time Complexity:**

The time complexity of the recursive algorithm is, where is the number of years. This is because the function makes a single recursive call for each year until it reaches the base case.

**b. Optimizing the Recursive Solution:**

**Memoization:**

To avoid excessive computation, you can use memoization to store previously computed results. This way, if the same calculation is needed again, it can be retrieved from memory instead of recalculated.

**Iterative Approach:**

Alternatively, you can implement an iterative approach to calculate the future value, which can be more efficient in terms of space complexity.