**Exercise 7: Financial Forecasting**

**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.

**Solution:**

**Recursion:**

Let us imagine that we have a complex problem. To solve this problem we require to complete multiple smaller calculations and aggregate them to receive the final answer.

This is where recursion comes into play, Recursion is when a method calls itself to solve a problem again and again.

Each recursive call simplifies the problem into a smaller problem, until it reaches a base case.

A simple example of recursion is the factorial problem. The code for the factorial problem is as follows,

int factorial(int n) {

if (n == 0) return 1;

return n \* factorial(n - 1);

}

As you can see ,we take a problem and split into multiple subproblems,we perform recursion call to solve the subproblems which gives us the final output.

**Code:**

For our financial forecasting recursion program let us consider the calculation of compound interest.

We have a fixed principal amount, a fixed growth rate and a fixed time period. Let us write the program to calculate the amount year by year.

package Coding.Recursion;

public class RecursionExample {

  public static double Calculate(double amount,double rate,int years){

    if(years==0){

      return amount;

    }

    double newAmount=amount\*(1+rate/100);

    return Calculate(newAmount,rate,years-1);

  }

  public static void main(String[] args) {

    double amount=100500.0;

    double rate=5.9;

    int years=12;

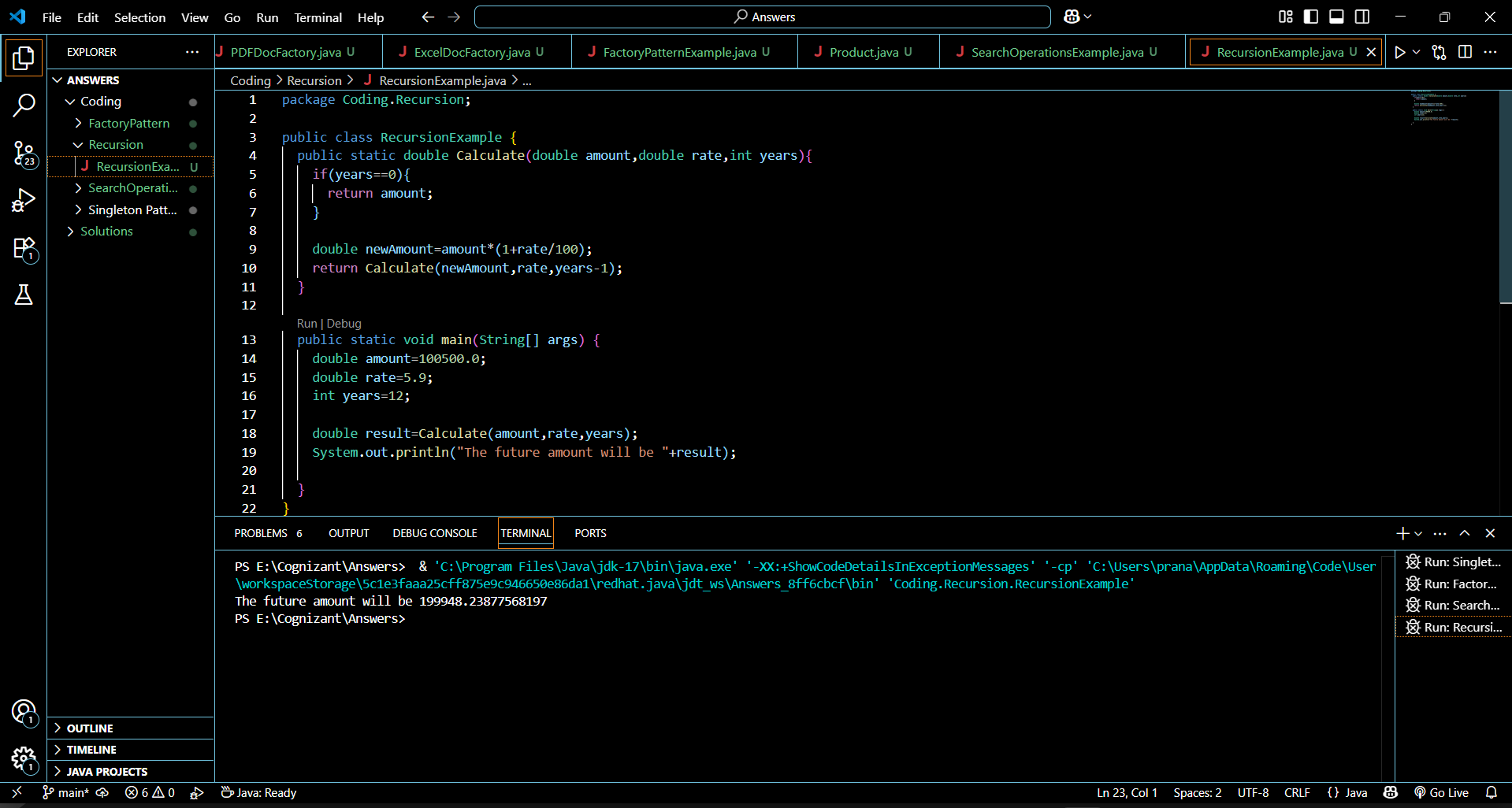
    double result=Calculate(amount,rate,years);

    System.out.println("The future amount will be "+result);

  }

}

**Output:**



**Time Complexity:**

The time complexity for the written program is O(n) , where n is the number of years.

This is because the program runs for exactly the number of years, however this can become problematic if the number of years is very large.

Since each recursive call uses memory, it can take a long time to compute and a large memory space.

**Optimization:**

To better optimize the recursion program, we stop using recursion itself. We use loops in the place of recursion, too reduce the time and memory to complete computation.

The code for this would look something like this,

Public static calculate(double amount,double rate,int years){

for(int i=0;i<years;i++){

amount\*=(1+rate/100);

}

return amount;

}

Using loops is much more efficient and safer.