Learning Experience:

This was a very nice refresher on recursion. We learned about recursion briefly during CMSC 203 but only really glanced over it. The sumOfArray method was pretty straight forward. I was able to first calculate the last value at a given index and then work up the array, adding to each total. I created a fibonacci recursion method for practice. I then worked on the dynamic processing for fibonacci. I first created an int array that had 2 plus the given number to account for the index 0 . Then I assigned index 0 to 0 and index 1 to 1. Then I used a for loop to start at index 2. This loop runs until the given number has been reached. It assigns each value of the array it's given fibonacci number. Dynamic processing seems very efficient and concise to me. For the sumOfArray method, I can see recursion making sense since it's adding on to each value in the array. For the fibonacci recursion, although it does still work correctly, there is a some more thinking involved. I have to write down the recursion tree to fully understand what I am dealing with. This takes extra time and processing compared to the dynamic processing. I can definitely see when each one might be a better choice.

Question:
Is there an easier way to think about recursions (like in my fibonacci method that used recursion)? I find myself always using tree diagrams to understand my work. Is this bad or will things eventually click?

Test Run:

```
     * @param arr array of Integers
     * @param num index of array to sum all previous index values (including num
     * @return sum of array values
     */
    public static int sumOfArray(Integer[] arr, int num) {
        if (num < 0)
            return 0;

        return arr[num] + sumOfArray(arr, num-1);
    }

    public static int fibonacci(int n) {
        if (n <= 1)
            return n;

        return fibonacci(n-1) + fibonacci(n-2);
    }

    public static int fibonacciDynamic(int n){
        int value[] = new int [n+2];
        value[0] = 0;
        value[1] = 1;

        for (int i = 2; i <= n; i++)
            value[i] = value[i-1] + value[i-2];

        return value[n];
    }
}
```

```
Console
<terminated> ArraySumDriver [Java Application] /Library/Java/JavaVirtualMachines/jdk-18.0.1.1.jdk/Contents/Home/bin/j
36
44
fibonaccc(3) = 2
fibonacci(6) = 8
fibonacci(9) = 34
fibonacciDynamic(3) = 2
fibonacciDynamic(6) = 8
fibonacciDynamic(9) = 34
```