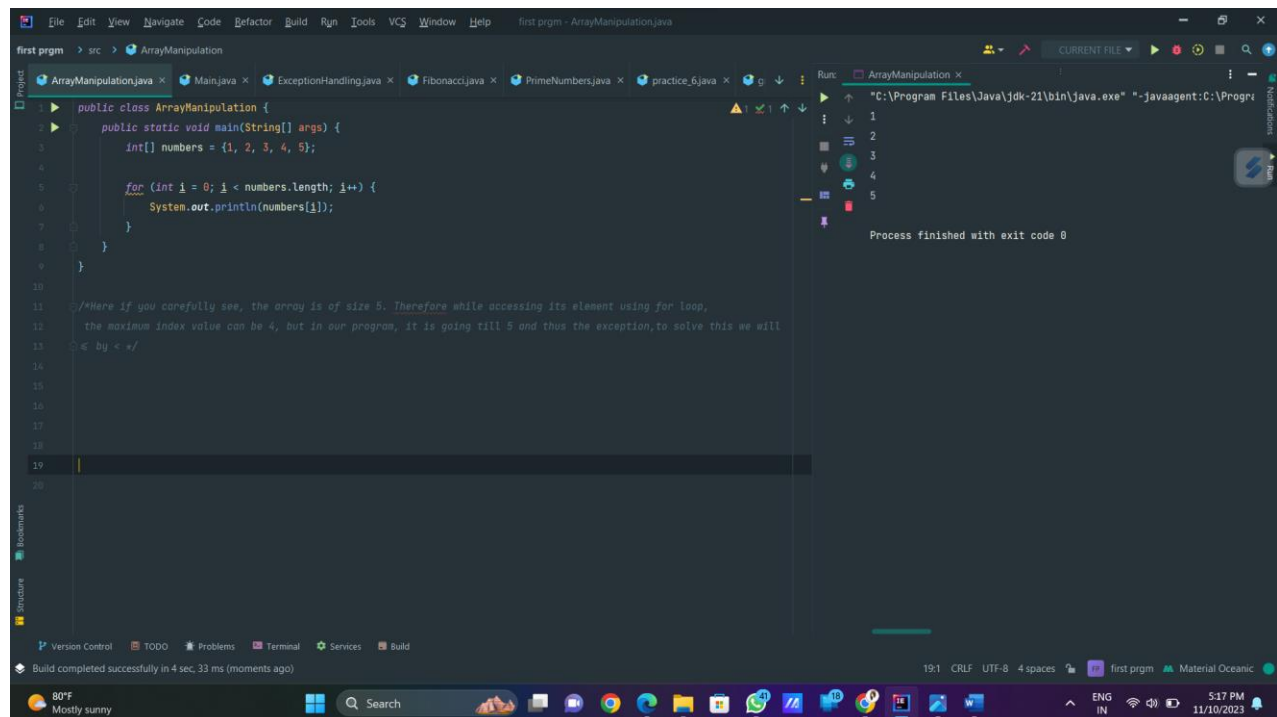Debugging Exercise 1: Array Manipulation

Objective: To identify and fix errors in a Java program that manipulates arrays.

```java
public class ArrayManipulation {
    public static void main(String[] args) {
        int[] numbers = {1, 2, 3, 4, 5};

        for (int i = 0; i <= numbers.length; i++) {
            System.out.println(numbers[i]);
        }
    }
}
```


Debugging Exercise 2: Object-Oriented Programming

Objective: To identify and fix errors in a Java program that demonstrates basic object-oriented programming principles.

```java
class Car {
    private String make;
    private String model;

    public Car(String make, String model) {
        this.make = make;
        this.model = model;
    }
```

```java
    public void start() {
        System.out.println("Starting the car.");
    }
}

public class Main {
    public static void main(String[] args) {
        Car car = new Car("Toyota", "Camry");
        car.start();
        car.stop();
    }
}
```

Debugging Exercise 3: Exception Handling
Objective: To identify and fix errors in a Java program that demonstrates exception handling.

```java
public class ExceptionHandling {
    public static void main(String[] args) {
        int[] numbers = {1, 2, 3, 4, 5};

        try {
            System.out.println(numbers[10]);
        } catch (ArrayIndexOutOfBoundsException e) {
            System.out.println("Array index out of bounds.");
        }

        int result = divide(10, 0);
        System.out.println("Result: " + result);
    }

    public static int divide(int a, int b) {
        return a / b;
    }
}
```
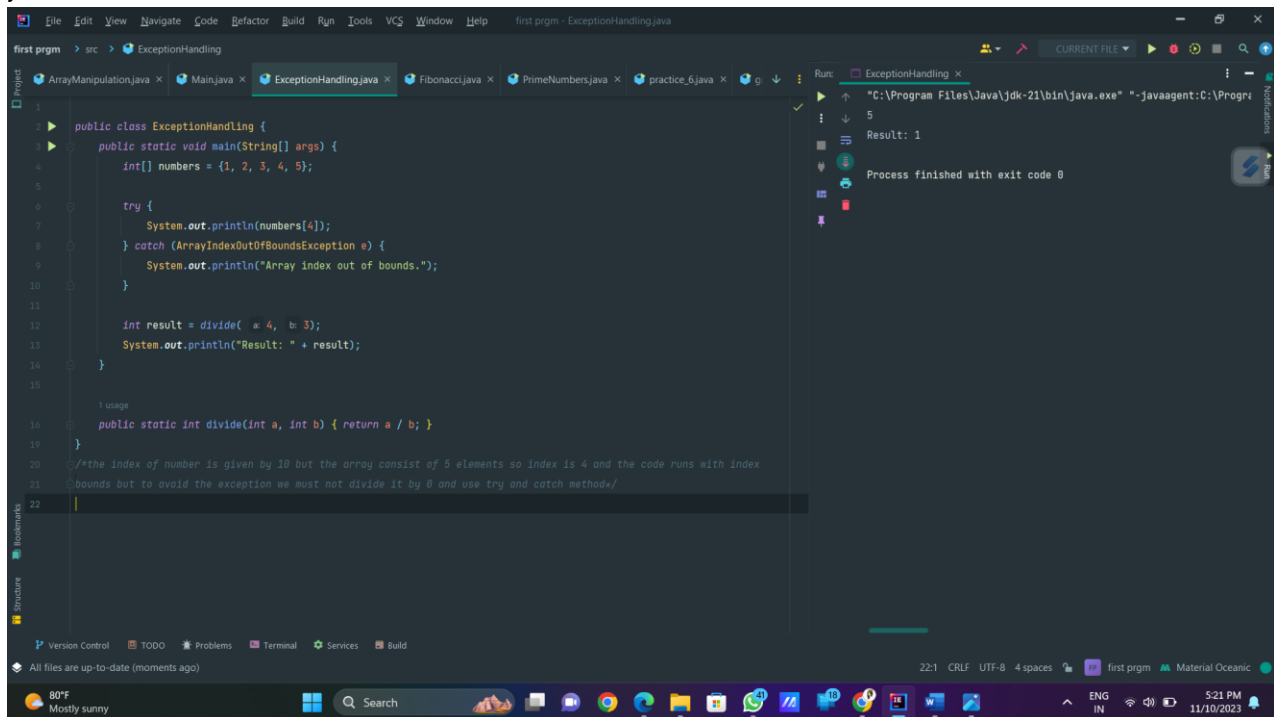

Exercise 4:
```java
public class Fibonacci {
    public static int fibonacci(int n) {
        if (n <= 1)
            return n;
```

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

    public static void main(String[] args) {
        int n = 6;
        int result = fibonacci(n);
        System.out.println("The Fibonacci number at position " + n + " is: " + result);
    }
}
```

The code aims to calculate the Fibonacci sequence. However, there is a bug in the code. When the student runs this code, it will raise an error or produce incorrect output. The student's task is to identify and correct the bug.

Hint: Pay close attention to the base case and recursive calls.
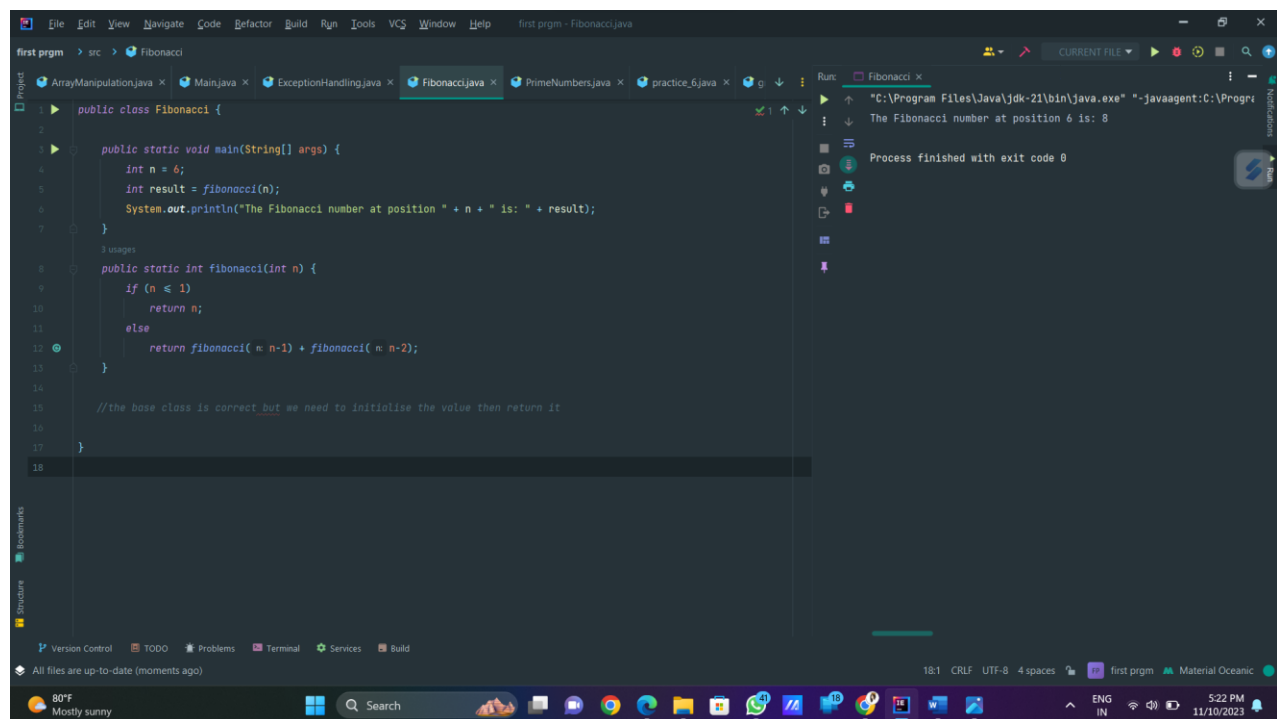
Exercise4:
```java
import java.util.*;

public class PrimeNumbers {
    public static List<Integer> findPrimes(int n) {
        List<Integer> primes = new ArrayList<>();
        for (int i = 2; i <= n; i++) {
            boolean isPrime = true;
            for (int j = 2; j < i; j++) {
                if (i % j == 0) {
                    isPrime = false;
                    break;
                }
            }
            if (isPrime) {
                primes.add(i);
            }
        }
        return primes;
    }

    public static void main(String[] args) {
        int n = 20;
        List<Integer> primeNumbers = findPrimes(n);
        System.out.println("Prime numbers up to " + n + ": " + primeNumbers);
    }
}
```

The code aims to find prime numbers up to a given limit. However, there is a bug in the code. When the student runs this code, it will raise an error or produce incorrect output. The student's task is to identify and correct the bug.

Hint: Check the condition for checking prime numbers.

```java
public class PrimeNumbers {
    1 usage
    public static List<Integer> findPrimes(int n) {
        List<Integer> primes = new ArrayList<>();
        for (int i = 2; i <= n-1; i++) {
            boolean isPrime = true;
            for (int j = 2; j < (i); j++) {
                if (i % j == 0) {
                    isPrime = false;
                    break;
                }
            }
            if (isPrime) {
                primes.add(i);
            }
        }
        return primes;
    }

    public static void main(String[] args) {
        int n = 20;
        List<Integer> primeNumbers = findPrimes(n);
        System.out.println("Prime numbers up to " + n + ": " + primeNumbers);
    }
}
    //there is no search error but in for condition we can add n-1
```

Run: PrimeNumbers
```
"C:\Program Files\Java\jdk-21\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDE
Prime numbers up to 20: [2, 3, 5, 7, 11, 13, 17, 19]

Process finished with exit code 0
```