**Java Programming Internship**

First task: Debugging errors

1.

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

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]);

}

}

}

The problem was in loop condition that is in 7th line of the given code.’ArrayIndexOutOfBoundsException’ when you run it.

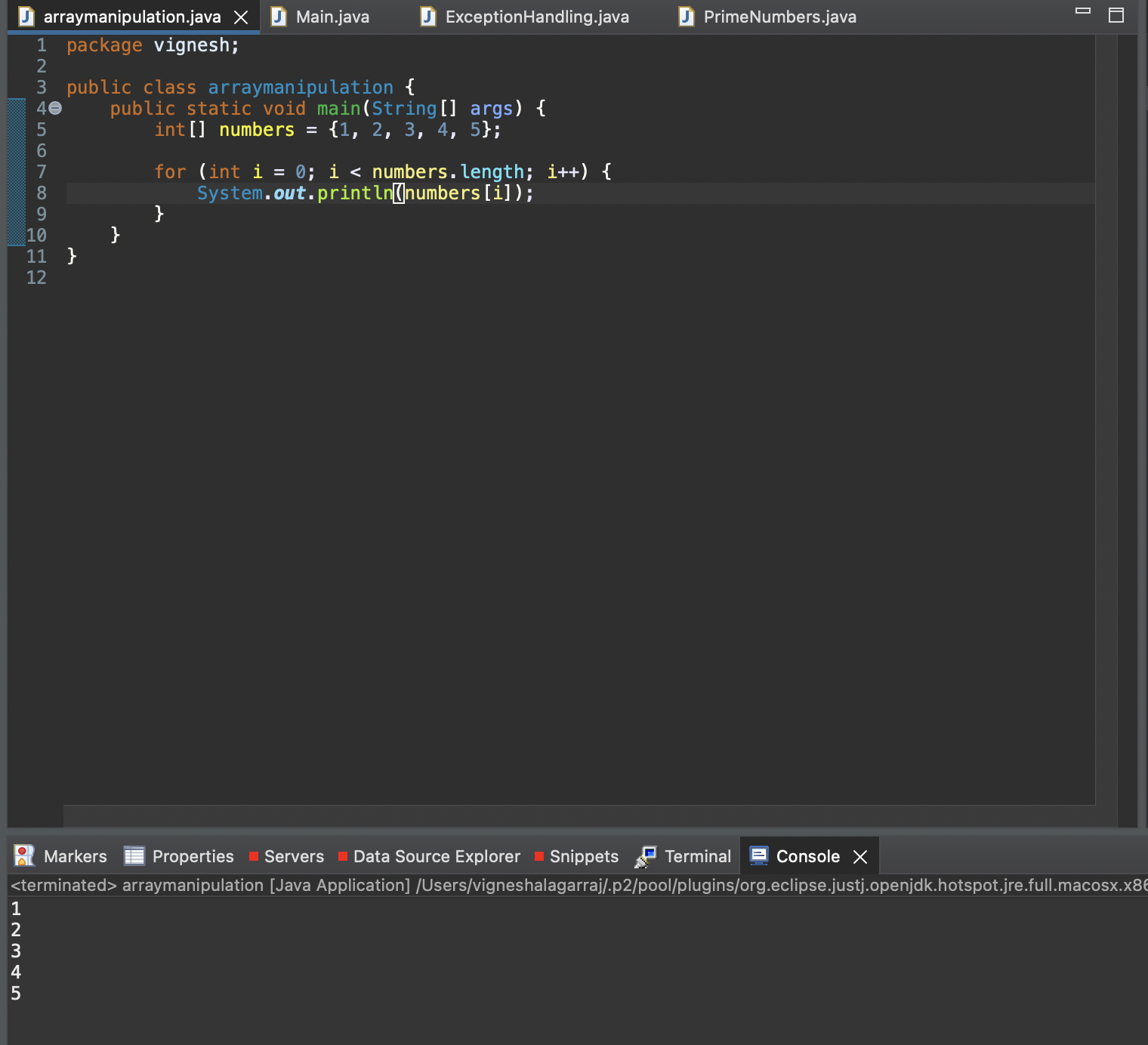
To fix this issue, you should change the loop condition to ‘i < numbers.length’.

Here is the correct code:

git remote add origin https://github.com/ZilLion777/VIgneshjavacodes.git

git branch -M main

git push -u origin main



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

class Car {

private String make;

private String model;

public Car(String make, String model) {

this.make = make;

this.model = model;

}

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();

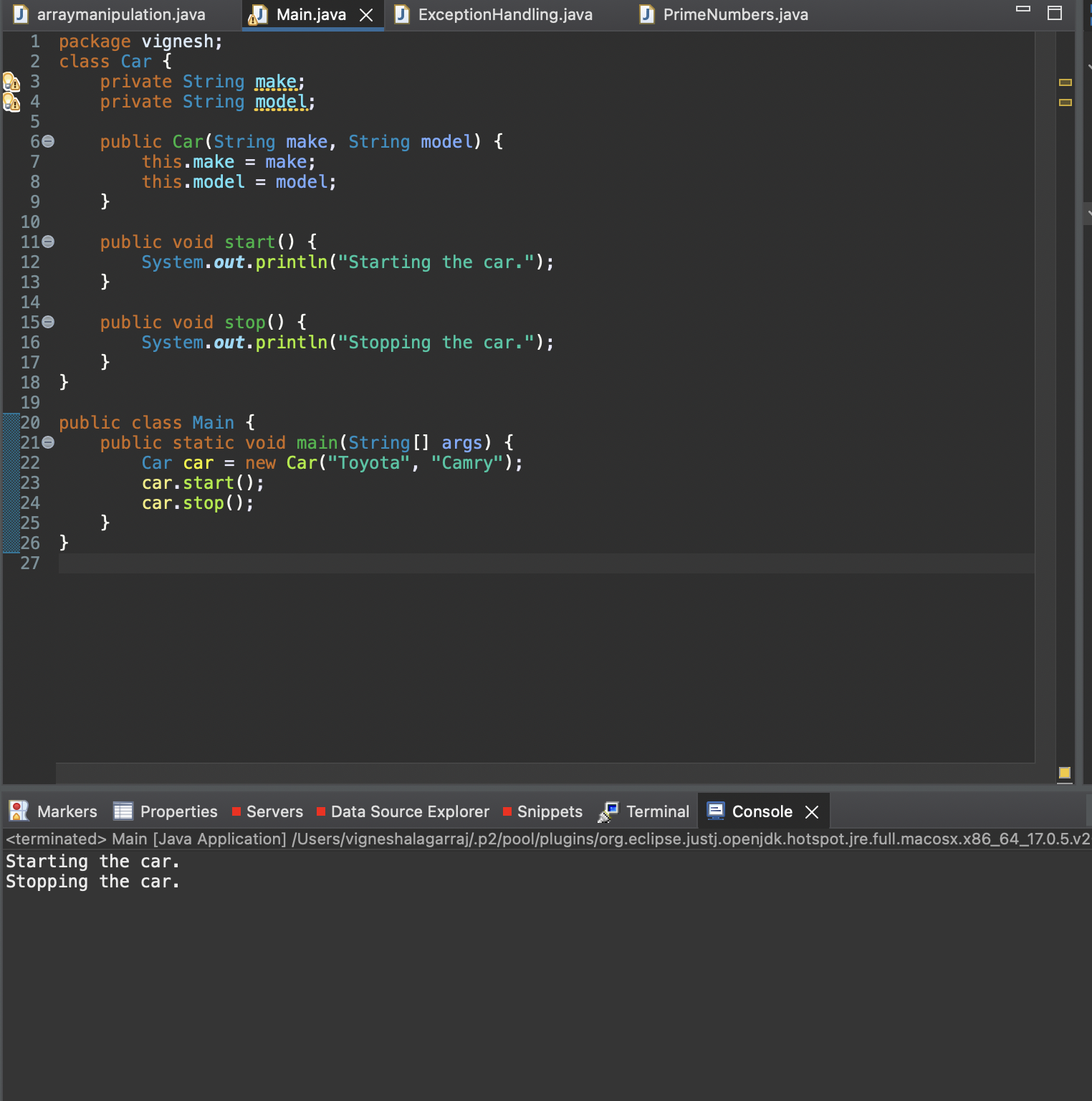
car.stop();

}

}

Error:

There is a error in the given code that the car class doesn’t have a stop method. Here is the correct code:



3. To identify and fix errors in a Java program that demonstrates exception handling.

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;

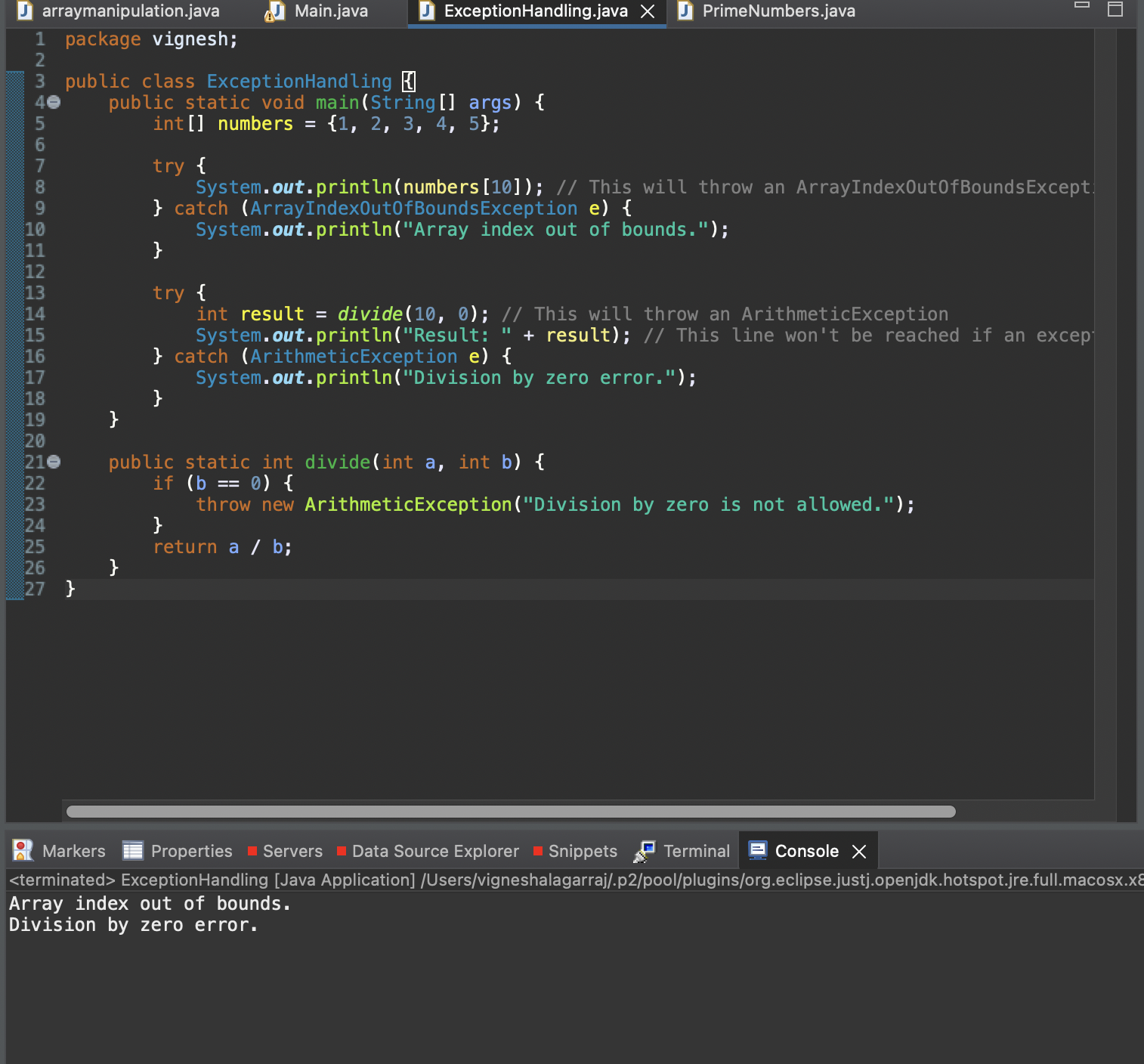
    }

}

Error:

There are two error in this given code

They are Division by zero error and Accessing an array index out of bounds.Here is the correct code:

4. 

4.

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);

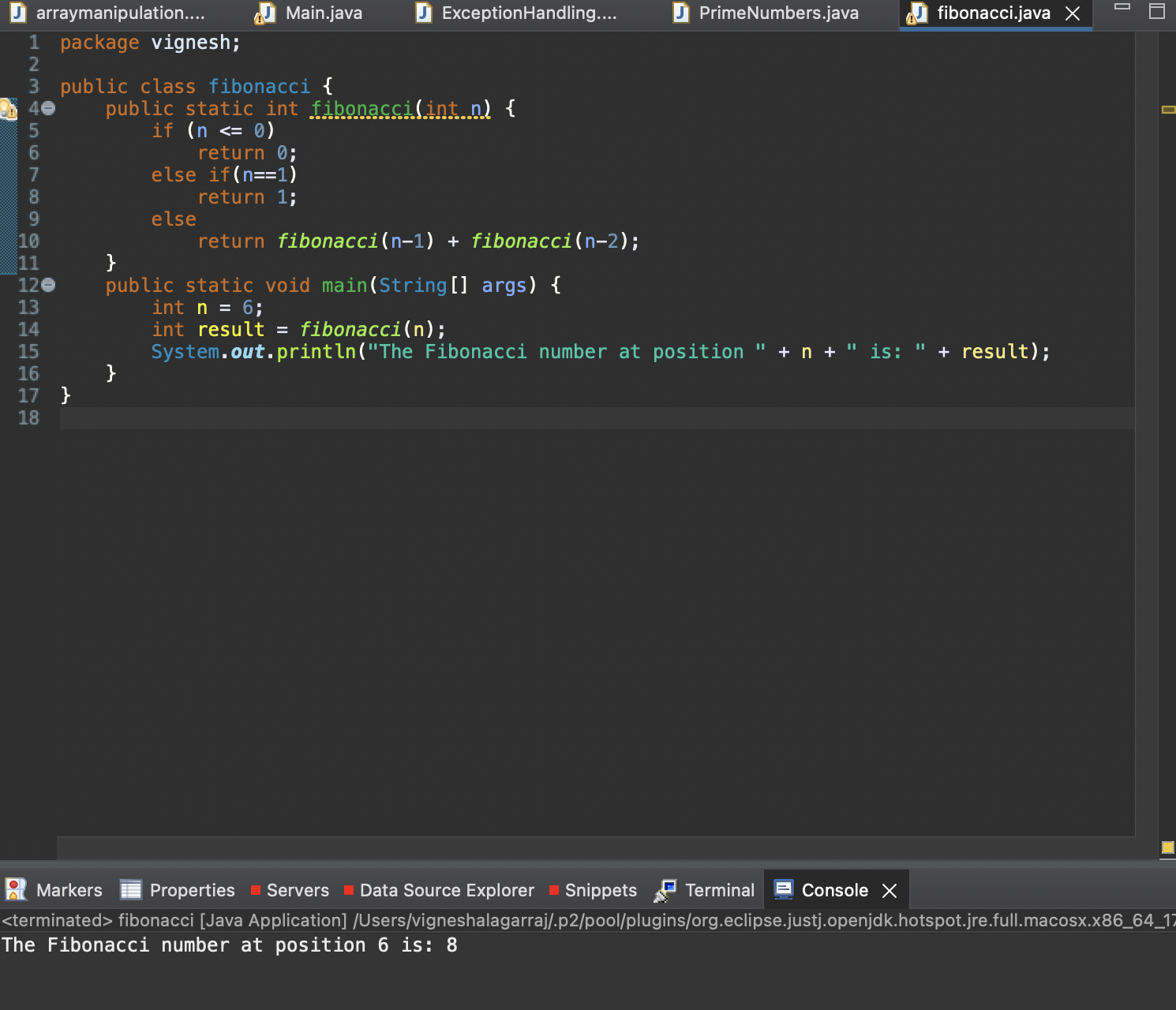
        System.out.println("The Fibonacci number at position " + n + " is: " + result);

    }

}

Error:

The error in the provided code is related to how it handles the starting values of the Fibonacci sequence. In the current implementation, when ‘n’ is less than or equal to 1, it returns ‘n’. However, this is incorrect because the Fibonacci sequence conventionally begins with the values 0 and 1, not 1 and 1.Here is the correct code



5.

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);

    }

}

Error:

The problem in the provided code lies in how it determines prime numbers. The code mistakenly assumes all numbers greater than 1 are prime by setting the isPrime variable to true initially and only changing it to false when a divisor is discovered. To address this issue, it's necessary to enhance the inner loop to more accurately identify divisors.

