Exercise 1; Array manipulation

objective: to identify and fix err program that manupulates arrays.

public class ArrayManipulation { public static void main(String[] args) {

- **Explanation of code:**

int[]numbers={1,2,3,4,5};

for (int i=0;i<=numbers.length;i++){

System.out.println(numbers[i]); } } }

The loop condition i <= numbers.length causes the loop to run one extra iteration beyond the last index of the array. Since array indices in Java are zero-based, numbers.length is out of bounds, leading to an ArrayIndexOutOfBoundsException.

✈ **Here is the corrected code:-**

```
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]);
        }
    }
}
```

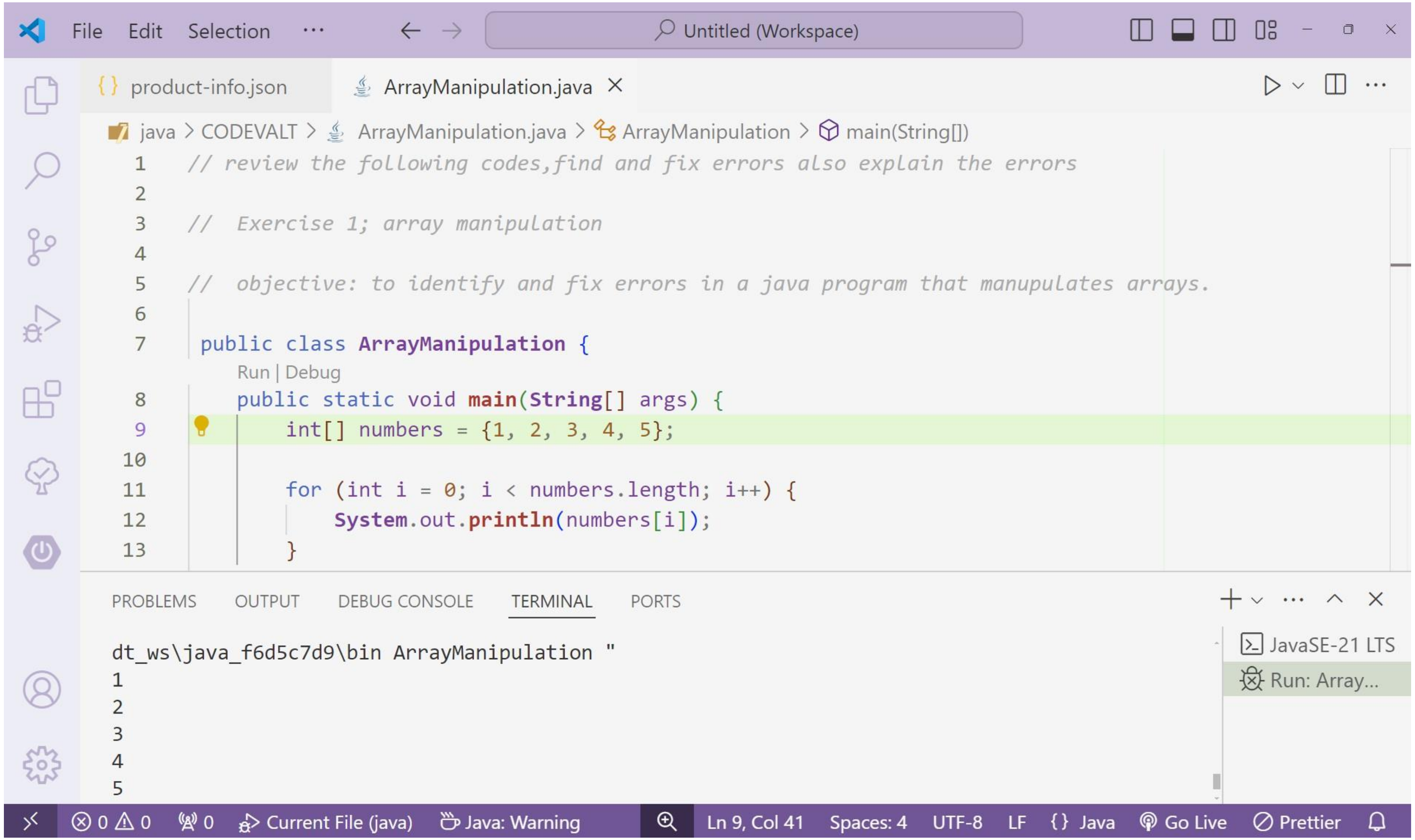
- In this code there is error:

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

- **here i corrected**

Changing the **loop** condition to i < **numbers.length** ensures that the **loop** only **iterates** over **valid** indices of the array, preventing the **ArrayIndexOutOfBoundsException** error.

- A rray manipulation: output



```
// review the following codes,find and fix errors also explain the errors

//  Exercise 1; array manipulation

//  objective: to identify and fix errors in a java program that manupulates 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]);
      }
```

```
dt_ws\java_f6d5c7d9\bin ArrayManipulation "
1
2
3
4
5
```

Exercise 2:- Oop objective:-To identify and fix the error in a java program that demonstrate basic oops principles.

- **Here is the corrected code:-**

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

    // Constructor without 'class' keyword and proper naming
    public Car(String make, String model) {
        this.make = make;

        this.model = model;
    }
```

Explanations of the corrections*:*

1. *Constructor Syntax*: The Car constructor is defined correctly without the class keyword. The proper syntax is public Car(String make, String model).

By making these corrections, the code will compile and run as intended.

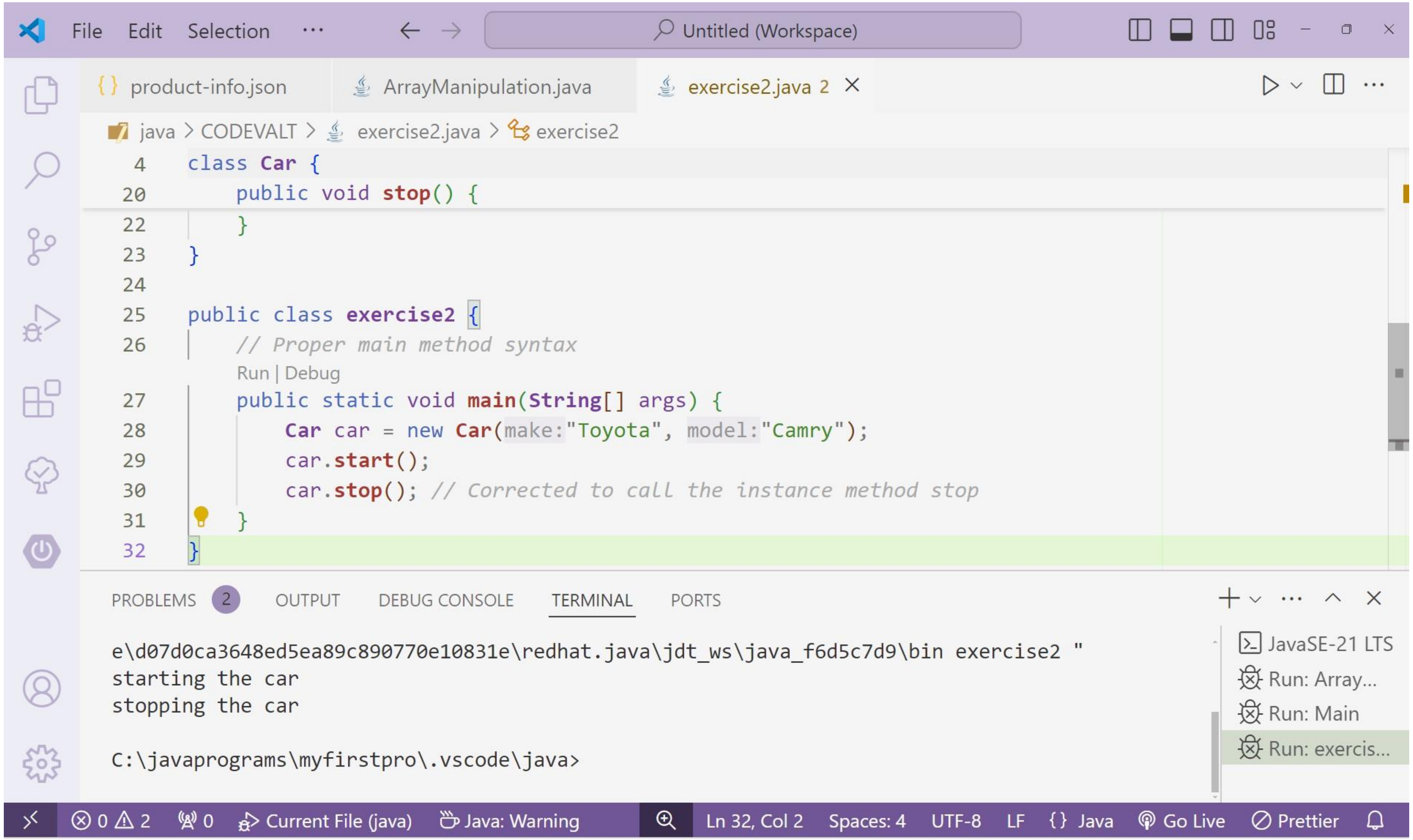
```java
    // Instance method to start the car
    public void start() {
        System.out.println("starting the car");
    }

    // Instance method to stop the car
    public void stop() {
        System.out.println("stopping the car");
    }
}

public class exercise2 {
    // Proper main method syntax
    public static void main(String[] args) {
        Car car = new Car("Toyota", "Camry");
        car.start();
        car.stop(); // Corrected to call the instance method stop
    }
}
```

2. *Inner Class Removal*: The Car class should be defined independently without embedding another class inside it.
3. *Main Method Syntax*: The main method should be declared as public static void main(String[] args) to be the entry point of the program.
4. *Method Call Correction*: The stop method should be called on the car object instance, not on the class itself. Therefore, car.stop(); is the correct way to call the method.

Output:-



```java
class Car {
    public void stop() {
    }
}

public class exercise2 {
    // Proper main method syntax
    Run | Debug
    public static void main(String[] args) {
        Car car = new Car(make:"Toyota", model:"Camry");
        car.start();
        car.stop(); // Corrected to call the instance method stop
    }
}
```

```
e\d07d0ca3648ed5ea89c890770e10831e\redhat.java\jdt_ws\java_f6d5c7d9\bin exercise2 "
starting the car
stopping the car

C:\javaprograms\myfirstpro\.vscode\java>
```

Excercise 3:-ExceptionHandling

- **corrected code**:-

```
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) { //
Corrected 'Catch' to 'catch' and added missing parenthesis
            System.out.println("array index out of
            bounds");
        }

        try { // Added try-catch block for divide by zero
exception int result = divide(10, 0);
            System.out.println("Result: " +
            result);
        } catch (ArithmeticException e) {
            System.out.println("Cannot divide by zero");
        }
    }
```

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

- **Corrections** and Explanations:

1. *Syntax Error in Catch Block:*
   - Original: Catch(ArrayIndexOutOfBoundsException e{
   - Corrected: catch (ArrayIndexOutOfBoundsException e) {
   - Explanation: The catch keyword should be in lowercase, and there was a missing closing parenthesis before the curly brace.

2. *Handling Divide by Zero Exception:*
   - Original code called the divide method without handling the possibility of division by zero.
   - Added a try-catch block around the divide(10, 0) call to catch ArithmeticException, which is thrown when an integer is divided by zero.

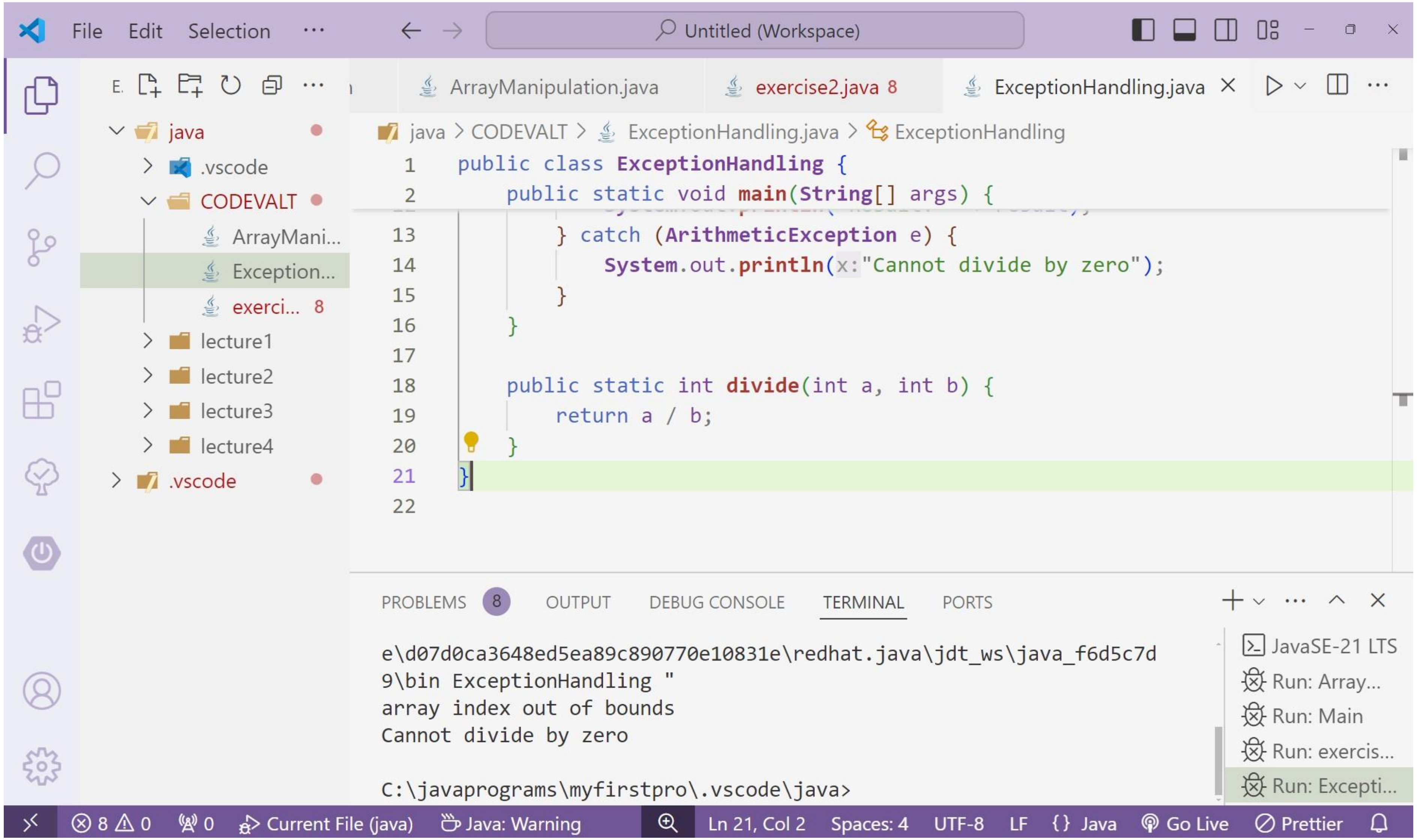
-

Improved Code Execution Flow:
   - The corrected code first attempts to print an element at index 10 of the numbers array. Since this index is out of bounds, it catches the ArrayIndexOutOfBoundsException and prints an appropriate message.

- It then attempts to divide 10 by 0, which causes an ArithmeticException. This exception is caught, and a message indicating that division by zero is not allowed is printed.

By including the try-catch blocks, the program can handle these specific exceptions gracefully, providing useful error messages without crashing.

Output:-



```
public class ExceptionHandling {
    public static void main(String[] args) {
        } catch (ArithmeticException e) {
            System.out.println(x:"Cannot divide by zero");
        }
    }

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

```
e\d07d0ca3648ed5ea89c890770e10831e\redhat.java\jdt_ws\java_f6d5c7d
9\bin ExceptionHandling "
array index out of bounds
Cannot divide by zero

C:\javaprograms\myfirstpro\.vscode\java>
```

Excercise 4:- fibonacci sequence.

- **corrected code:-**

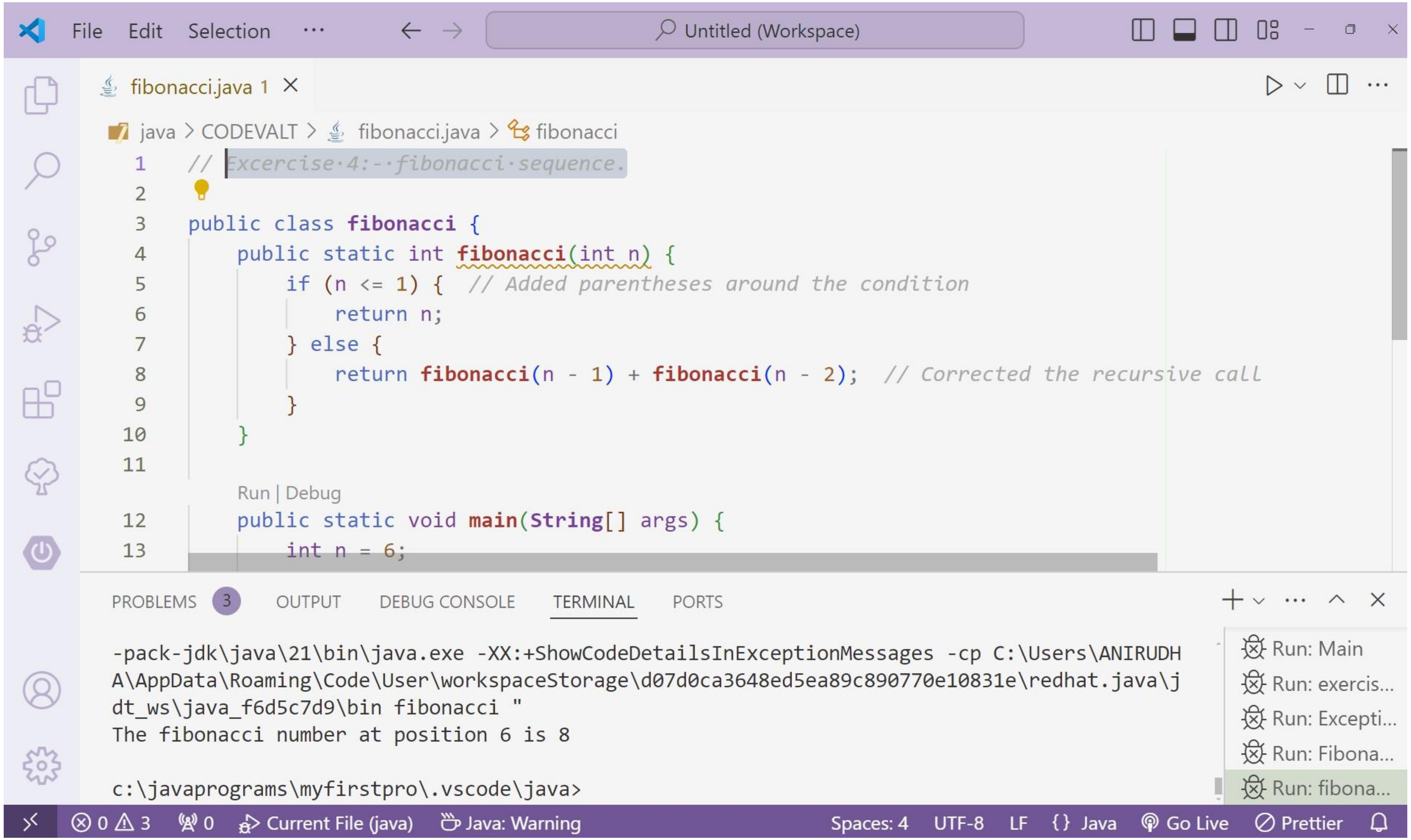
```
public class fibonacci {
    public static int fibonacci(int n) {
        if (n <= 1) {  // Added parentheses around the condition
            return n;
        } else {
            return fibonacci(n - 1) + fibonacci(n - 2);  // Corrected the recursive call
        }
    }

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

- **Corrections and Explanations:**

1. Syntax Error in the If Statement: - Original: if n<=1 - Corrected: if (n <= 1) - Explanation: Conditions in Java must be enclosed in parentheses.
2. Corrected Recursive Call: - Original: return fibonacci(n-1)+(n-2); - Corrected: return fibonacci(n - 1) + fibonacci(n - 2); - Explanation: The original code mistakenly tried to add (n - 2) directly. It should recursively call fibonacci(n - 2) instead.
3. System.out.println Statement Placement: - Original: System.out.println("The fibonacci number at position"+n+"is"+result); was outside the main method. - Corrected: Moved the System.out.println statement inside the main method. - Explanation: Statements outside of any method are not allowed in Java.

- **Output:-**



```java
// Excercise 4:- fibonacci sequence.

public class fibonacci {
    public static int fibonacci(int n) {
        if (n <= 1) {  // Added parentheses around the condition
            return n;
        } else {
            return fibonacci(n - 1) + fibonacci(n - 2);  // Corrected the recursive call
        }
    }

    public static void main(String[] args) {
        int n = 6;
```

```
-pack-jdk\java\21\bin\java.exe -XX:+ShowCodeDetailsInExceptionMessages -cp C:\Users\ANIRUDH
A\AppData\Roaming\Code\User\workspaceStorage\d07d0ca3648ed5ea89c890770e10831e\redhat.java\j
dt_ws\java_f6d5c7d9\bin fibonacci "
The fibonacci number at position 6 is 8

c:\javaprograms\myfirstpro\.vscode\java>
```

Excercise 5 :- To find Prime Number

- **corrected code**:-

```
import java.util.ArrayList   // Import necessary packages;
import java.util.List;
```

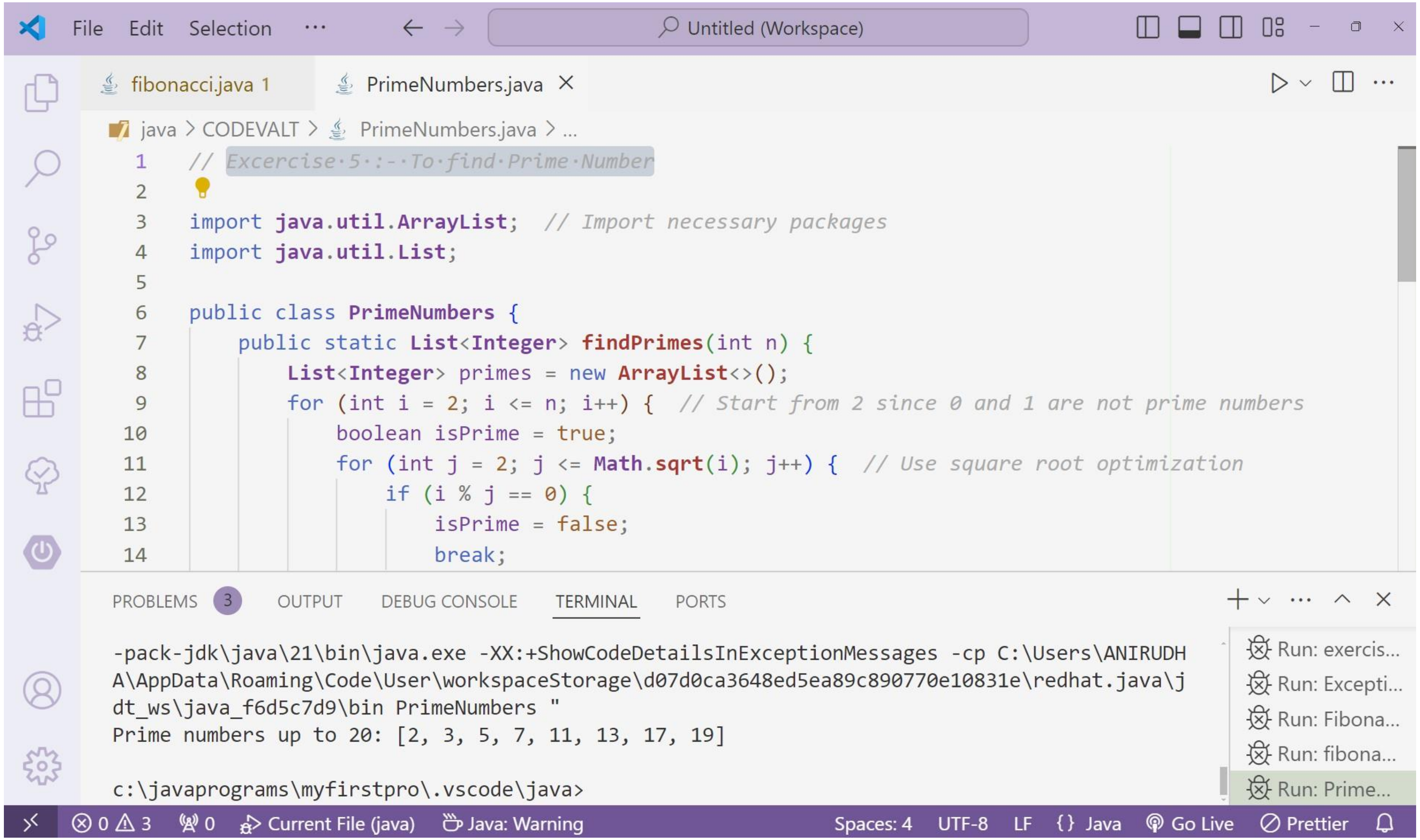
```java
public class PrimeNumbers {
    public static List<Integer> findPrimes(int n) {
        List<Integer> primes = new ArrayList<>();
        for (int i = 2; i <= n; i++) {  // Start from 2 since 0 and 1 are not prime numbers
            boolean isPrime = true;
            for (int j = 2; j <= Math.sqrt(i); j++) {  // Use square root optimization
                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);
    }
}
```

Corrections and Explanations:

1. Import Statements: - Added import java.util.ArrayList; and import java.util.List; to import the necessary classes for list handling.
2. Corrected Type Declarations: - Changed List<integer> to List<Integer> since Integer is the correct wrapper class for primitive int in Java. Java is case-sensitive, and integer is not a valid type.
3. Fixed Method Call Syntax: - Corrected newArrayList<>(); to new ArrayList<>(); to properly instantiate the ArrayList.
4. Prime Checking Loop: - Changed the outer loop to start from 2 instead of 0 since 0 and 1 are not prime numbers.   - Optimized the inner loop condition to j <= Math.sqrt(i). This reduces the number of iterations by only checking up to the square root of i. 5. Corrected Printing Statement Placement: - Moved System.out.println("Prime numbers up to " + n + ": " + primeNumbers); inside the main method to ensure it is part of the program flow.

- **Output:-**



```java
// Excercise 5 :- To find Prime Number

import java.util.ArrayList;  // Import necessary packages
import java.util.List;

public class PrimeNumbers {
    public static List<Integer> findPrimes(int n) {
        List<Integer> primes = new ArrayList<>();
        for (int i = 2; i <= n; i++) {  // Start from 2 since 0 and 1 are not prime numbers
            boolean isPrime = true;
            for (int j = 2; j <= Math.sqrt(i); j++) {  // Use square root optimization
                if (i % j == 0) {
                    isPrime = false;
                    break;
```

```
-pack-jdk\java\21\bin\java.exe -XX:+ShowCodeDetailsInExceptionMessages -cp C:\Users\ANIRUDH
A\AppData\Roaming\Code\User\workspaceStorage\d07d0ca3648ed5ea89c890770e10831e\redhat.java\j
dt_ws\java_f6d5c7d9\bin PrimeNumbers "
Prime numbers up to 20: [2, 3, 5, 7, 11, 13, 17, 19]

c:\javaprograms\myfirstpro\.vscode\java>
```