Debugging Exercise 1: Array Manipulation

In the given Java program, there is an error in the for loop condition. It should be i < numbers.length instead of i <= numbers.length. This is because array indices start from 0, so the valid indices range from 0 to length - 1. Here's 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-1; i++) {

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

        }

    }

}

Debugging Exercise 2: Object-Oriented Programming

In the given Java program, there is an error in the Main class. The stop() method is being called on the car object, but the Car class does not have a stop() method defined. To fix this, you can either add a stop() method to the Car class or remove the car.stop() line from the Main class.

Here's an example of adding a stop() method to the Car class:

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 void stop() {

        System.out.println("Stoping 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

Divide by Zero Exception: In the divide method, you are attempting to divide by zero, which is not allowed in mathematics and will throw an ArithmeticException.

To fix these issues, you can handle the ArithmeticException by adding a try-catch block in the main method.

Here's the corrected code:

import java.util.\*;

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.");

        }

        try{

    int result = divide(10, 0);

    System.out.println("Result: " + result);

        }

    catch(ArithmeticException e){

        System.out.println("NOT DIVISIBLE BY ZERO");}

    }

    public static int divide(int a, int b) {

        return a / b;

    }

}

Exercise 4:

When we try to calculate the Fibonacci series with calling Fibonacci(n) It yields the wrong output as the answer will be greater than expected but if the calling statement is changed to Fibonacci(n-1), it gives out the required output.

This is caused due to it does not consider 0 as starting number.

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 = 8;

        int result = fibonacci(n-1);

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

    }

}

Exercise4:

The bug in the code lies in the condition for checking prime numbers. The inner loop should go up to the square root of i, not i - 1. This is because, to determine if i is prime, it's enough to check divisibility up to the square root of i.

Here's the corrected code:

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 < Math.sqrt(i); j++) {

                if (i % j == 0) {

                    isPrime = false;

                    break;

                }

            }

            if (isPrime) {

                primes.add(i);

            }

        }

        return primes;

    }

    public static void main(String[] args) {

        int n = 25;

        List<Integer> primeNumbers = findPrimes(n);

        System.out.println("Prime numbers up to " + n + ": " + primeNumbers);

    }

}