**Debugging Exercise 1: Array Manipulation**

public class ArrayManipulation {

public static void main(String[] args) {

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

for (int i = 0; i < numbers.length; i++) { // Fix the loop condition

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

}

}

}

**Issue:**

The error in this code is the loop condition. It should be `i < numbers.length` instead of `i <= numbers.length` to prevent an "ArrayIndexOutOfBoundsException."

**Debugging Exercise 2: Object-Oriented Programming**

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() { // Add the stop method

System.out.println("Stopping the car.");

}

}

public class Main {

public static void main(String[] args) {

Car car = new Car("Toyota", "Camry");

car.start();

car.stop(); // Add a call to the stop method

}

}

**Issue:**

The error in this code is that the `stop()` method is called without being defined in the `Car` class. You need to add the `stop()` method to the `Car` class to fix this issue.

**Debugging Exercise 3: 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.");

}

try {

int result = divide(10, 0);

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

} catch (ArithmeticException e) {

System.out.println("Division by zero.");

}

}

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

if (b == 0) {

throw new ArithmeticException("Division by zero");

}

return a / b;

}

}

**Issue:**

In this code, I've added another try-catch block to handle the division by zero exception. Additionally, I've thrown an `ArithmeticException` when attempting to divide by zero in the `divide` method to handle this specific error case.

**Debugging Exercise 4: Fibonacci**

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);

}

}

**Debugging Exercise 5: PrimeNumbers**

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 \* j <= i; j++) { // Correct the condition for checking prime numbers

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);

}

}

**Issue:**

The error in this code is in the condition for checking prime numbers. The condition should be `j \* j <= i` instead of `j < i` for more efficient prime checking. This change prevents unnecessary iterations and speeds up the prime number calculation.