**Debugging Exercise 1: Array Manipulation**

**Objective:** 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]);

        }

    }

}

**Solution:**

**Error:** The code contains a logical looping error causing an ‘ArrayIndexOutOfBounds’ exception since the loop runs from i=0 to i<=length of the array.

**Explanation:** The indexing of an array is generally from ‘0’ to ‘n-1’ where ‘n’ refers to the length of the array. If we consider the above example array, it stores 5 integers. So the indexing goes from 0 to 4. While the loop runs from 0 to 5(as 5 is the length of the array), which doesn’t exist. Hence it gives an ‘ArrayIndexOutOfBoundsException’ as we’re trying to access an element that is out of the index range of the array i.e. 0 to 4.

**Fixed Code:** The fix is simple. We’ve to just replace i<=numbers.length to i<numbers.length OR i<=numbers.length-1.

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.  
  
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();

    }

}

**Solution:**

**Error:** The code contains an error as the method we’re trying to access in the main function is not declared.

**Explanation:** The execution of a Java Program starts from the ‘main’ function. In the ‘main’ function, we’re calling two methods- start and stop. Since the ‘stop’ method doesn’t exist, we get an ‘undeclared method’ error as we’re trying to call a method that doesn’t exist in the program.

**Fixed Code:** To fix this, we can either remove the function calling statement in the ‘main’ method OR we can simply declare a ‘stop’ method.

**Solution 1:**

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.");

    }

}// closing of class ‘Car’

public class Main {

    public static void main(String[] args) {

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

        car.start();

//cat.stop() is removed    }

}

**Solution 2:**

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(){ //stop method added

System.out.println("Car Stopped");}

} //close of class ‘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.

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;

    }

}

**Solution:**

**Error:** The code contains an ‘ArithmeticException’ as we’re trying to divide a number by zero.

**Explanation:** In the above code when we pass the denominator as 0 to the ‘divide’ function, an ‘ArithmeticException’ occurs, leading to a runtime error as we cannot divide a number by zero.

**Fixed Code:** To fix the code, we can simply add a ‘try-catch’ block for the ‘Arithmetic Exception’ and add a condition to check if the denominator passed in the ‘divide’ function is equal to zero.

public class Car {

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 is not allowed.");

}}

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

if (b == 0) {

throw new ArithmeticException("Division by zero is not allowed.");

}

return a / b;}

}

**Exercise 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);

    }

}

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.

**Solution:**

The code seems to be technically correct as it prints the number of the Fibonacci Series at the specified position.

**Exercise 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);

    }

}

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.

**Solution:**

The above code has no error and it works fine.