ASSIGNMENT 1 D. SIVAKARTHIKEYAN

TASK 1:

DEBUGGING EXERCISE 1: ARRAY MANIPULATION

class ArrayManipulation {

public static void main(String args[]) {

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

// Corrected loop: iterate up to numbers.length - 1 to avoid out-of-bounds access

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

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

}

}

}

**Explanation of the Fix:**

The original code contained an error in the loop condition:

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

This condition attempts to access numbers[numbers.length], which is outside the valid range of the array. Arrays in Java are zero-indexed, meaning their indices start from 0 and go up to numbers.length - 1.

**Corrected Output:**

When you run the corrected code, it will print each element of the numbers array on a separate line:

1

2

3

4

5

DEBUGGING EXERCISE 2: OBJECT ORINTED PROGRAMMING

**Error:**

* The stop() method is missing from the Car class. It's referenced in the main method but not defined.

**Fix:**

We'll add the stop() method to the Car class:

**Code with Getter Methods:**

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("Stopping the car");

}

// Getter methods for make and model

public String getMake() {

return make;

}

public String getModel() {

return model;

}

}

DEBUGGING EXERCISE 3: EXCEPTION HANDLING

**Error 1:**

* The try block attempts to access numbers[10], which is outside the valid range of the array (numbers.length is 5). This will cause an ArrayIndexOutOfBoundsException.

**Error 2:**

* The divide method tries to divide by zero, which will result in an ArithmeticException at runtime.

**Fixes:**

1. **Correct Array Access:**

Inside the try block, change numbers[10] to access an element within the valid range (e.g., numbers[4]).

1. **Handle Division by Zero:**

There are two approaches to handle the division by zero:

* + **Throw an exception:** Modify the divide method to throw an ArithmeticException when b is zero:

Java

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

if (b == 0) {

throw new ArithmeticException("Division by zero");

}

return a / b;

}

**Corrected Program**

public class ExceptionHandling {

public static void main(String args[]) {

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

try {

System.out.println(numbers[4]); // Access valid element

} catch (ArrayIndexOutOfBoundsException e) {

System.out.println("Array index out of bounds");

}

int result;

try {

result = divide(10, 0);

} catch (ArithmeticException e) {

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

result = 0; // Or handle the error differently

}

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

}

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

return a / b;

}

}

EXERCISE 4:

In the main method, you print the result using System.out.println("The Fibonacci number at position"+n+"is:"result);. This line attempts to concatenate a string with an integer (n) and another string directly. However, in Java, string concatenation requires using the + operator between strings or converting non-strings to strings.

**Corrected Code**

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);

}

}

EXERCISE: 5

The inner loop (for(int j=2;j<i;j++)) iterates from 2 up to i-1 to check if i is divisible by any number between 2 and itself (excluding i). However, inside the loop, isPrime is always set to true before the divisibility check. This means even if i is not prime, isPrime will remain true, leading to incorrect results.

**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; // Moved outside the inner loop

for (int j = 2; j < i; j++) {

if (i % j == 0) {

isPrime = false;

break; // Optimization: exit loop if a divisor is found

}

}

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);

}

}