Debugging Exercise 1: Array Manipulation

Correct code

```
public class ArrayManipulation {
    public static void main(String[] args) {
        int[] numbers = {1, 2, 3, 4, 5};
        // The loop condition should be "i < numbers.length" instead of "i <= numbers.length"
        for (int i = 0; i < numbers.length; i++) {
            System.out.println(numbers[i]);
        }
    }
}
```

Explanation:

1. Array Indexing: In Java, array indices start from 0. The array `numbers` has indices from 0 to 4 for a total of 5 elements (`numbers[0]`, `numbers[1]`, `numbers[2]`, `numbers[3]`, and `numbers[4]`).

2. Off-by-One Error: The original code uses `i <= numbers.length` as the loop condition. This is incorrect because the last valid index in the array is `numbers.length - 1`. Therefore, the loop should run while `i` is less than `numbers.length`.

3.Loop Condition Fix: The corrected code uses `i < numbers.length` as the loop condition, ensuring that the loop iterates from `i = 0` to `i = numbers.length - 1`, inclusive.

Debugging Exercise 2: Object-Oriented Programming

Correct code

```
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.");
    }
    // Adding a stop method to the Car class
    public void stop() {
        System.out.println("Stopping the car.");
    }
}
public class Main {
    public static void main(String[] args) {
        // Creating an instance of the Car class
        Car car = new Car("Toyota", "Camry");
        car.start();
        car.stop();
    }
}
```

Explanation:

1. **Added stop Method:** I added a **stop** method to the **Car** class to match the attempt to call **car.stop()** in the **Main** class. This prevents a compilation error.
2. **Instance Creation:** In the **Main** class, an instance of the **Car** class is created using the constructor with the parameters "Toyota" and "Camry."
3. **Method Calls:** The **start** method is called to indicate that the car is starting, and then the newly added **stop** method is called to indicate that the car is stopping.

Debugging Exercise 3: Exception Handling

Correct code

```
public class Fibonacci {
    public static int fibonacci(int n) {
        if (n < 0) {
            System.out.println("Invalid input. Fibonacci sequence is not defined for negative numbers.");
            return -1;
        } else 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);
    }
}
```

Explanation:

1. **Handling Negative Input:** Added a check for **n < 0** at the beginning of the **fibonacci** method. If **n** is negative, it prints an error message and returns a specific value (in this case, -1) to indicate an error.
2. **Base Case:** The existing base case **if (n <= 1)** remains, ensuring that the correct values are returned for positions 0 and 1.

Exercise4:

```java
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++) {

            // Skip even numbers greater than 2

            if (i > 2 && i % 2 == 0) {

                continue;

            }

            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 = 20;

        List<Integer> primeNumbers = findPrimes(n);

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

    }

}
```