Debugging Exercise 1: Array Manipulation
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; i++) {
            System.out.println(numbers[i]);
        }
    }
}
```

Output:
1
2
3
4
5

Corrected Errors:
ArrayIndexOutOfBounds Exception:
It occurs when a program attempts to access an invalid index in an array i.e an index that is less than 0 or equal to or greater than the length of the array.
In the above code if we give i <= numbers.length as a condition in the for loop, then we are trying to access the index greater than the length of the array.
So the correct condition is i <numbers.length.

Debugging Exercise 2: Object-Oriented Programming

Corrected code:

```
class Car {
    public String make;
    public 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();
    }
}
```

Output:

Starting the car.

Corrected Errors:

- Private access specifier:

The access level of the private modifier is only within the class. It cannot be accessed from outside of the class. Here make and model attributes are given as private but accessing them outside the class. So we need to define them as public.

- Stop is an undefined method . we cannot access undefined methods.

Debugging Exercise 3: Exception Handling

Corrected code:

```
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("Divided by zero is not possible");
        }
    }

    public static int divide(int a, int b) {
        return a / b;
    }
}
```

Output:
Array index out of bounds.
Divided by zero is not possible

Corrected errors:

Arithmetic Exception: Any number divided by zero gives the answer infinity. No data structure in the programming can store an infinite amount of data. So any number divided by zero throws arithmetic exception.
So we used a try catch block to handle the exception.

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;
        if(n<0)
        {
           System.out.println("fibonacci number is not defined");
        }
        else {
        int result = fibonacci(n);
        System.out.println("The Fibonacci number at position " + n + " is: " + result);
        }
```

```
    }
}
```
Output:
The Fibonacci number at position 6 is: 8

Exercise4:

```
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);
    }
}
```

Output:

Prime numbers up to 20: [2, 3, 5, 7, 11, 13, 17, 19]