WEEK-1

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++) {
         System.out.println(numbers[i]);
      }
   }
}
```

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
```

Explanation of code:

The loop iterates from i = 0 to i <= numbers.length. The problem here is that when i equals numbers.length, it's trying to access an element at an index that is beyond the bounds of the array. Array indices in Java are zero-based, meaning they start from 0, so the last valid index for an array of length n is n-1.

So, when i becomes equal to numbers.length, numbers[i] tries to access an element outside the bounds of the array, leading to the "ArrayIndexOutOfBoundsException".

To fix this issue, you should change the loop condition to i < numbers.length, ensuring that i never exceeds the valid range of array indices.

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 class Main {
   public static void main(String[] args) {
      Car car = new Car("Toyota", "Camry");
      car.start();
      car.stop();
   }
}
```

Corrected 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.");
```

```
    }
    public void stop() {
        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();  // Now this line is valid since Car class has a stop() method
    }
}
```

Explanation of error:

The error in the provided code is due to an attempt to call a method stop() on the car object which is not defined in the Car class.

Here's a breakdown of the error:

1.In the Car class definition, there's only a start() method provided. There is no stop() method defined.

2.In the Main class, after creating an instance of the Car class named car using Car car = new Car("Toyota", "Camry");, the start() method is invoked successfully on the car object using car.start();.

3.However, the next line car.stop(); tries to invoke a stop() method on the car object, but such a method does not exist in the Car class. Therefore, this line causes a compilation error.

To resolve this error, you either need to remove the invocation of stop() method from the Main class if it's unnecessary, or define a stop() method in the Car class if it's intended to have such functionality.

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.");
        }
        int result = divide(10, 0);
        System.out.println("Result: " + result);
    }
    public static int divide(int a, int b) {
        return a / b;
    }
}
```

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("Division by zero error.");
        }
    }
    public static int divide(int a, int b) {
        if (b == 0) {
            throw new ArithmeticException("Division by zero.");
        }
        return a / b;
    }
}
```

Explanation of error:

The error in the provided code occurs due to an attempt to perform integer division by zero. Here's a step-by-step explanation of the error:

1.In the main() method, after handling the ArrayIndexOutOfBoundsException for numbers[10] using a try-catch block, the program proceeds to execute the next statement, which is int result = divide(10, 0);.

2.The divide() method is invoked with arguments 10 and 0. Inside the divide() method, there's a division operation return a / b;, where a is 10 and b is 0.

3.Division by zero is mathematically undefined and not allowed in Java. Therefore, attempting to divide by zero results in an ArithmeticException.

4.Since there's no try-catch block surrounding the call to divide(10, 0) in the main() method, the ArithmeticException is thrown, and since it's not caught, it propagates up the call stack and ultimately terminates the program prematurely.

To fix this error, you can handle the possibility of division by zero inside the divide() method itself. This can be done by adding a check to ensure that the divisor (b) is not zero before performing the division. If it is zero, you can either throw an exception or handle it in another appropriate way depending on your application's requirements.

Exercise 4:

NO ERROR

Excersie-5:

```
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);
    }
}
```

Corrected code:

```
public class PrimeNumbers {
    public static List<Integer> findPrimes(int n) {
        List<Integer> primes = new ArrayList<>();
        if (n >= 2) {
            primes.add(2);
            for (int i = 3; i <= n; i += 2) {
                boolean isPrime = true;
                int sqrt = (int) Math.sqrt(i);
                for (int j = 3; j <= sqrt; j += 2) {
                    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);
    }
}
```