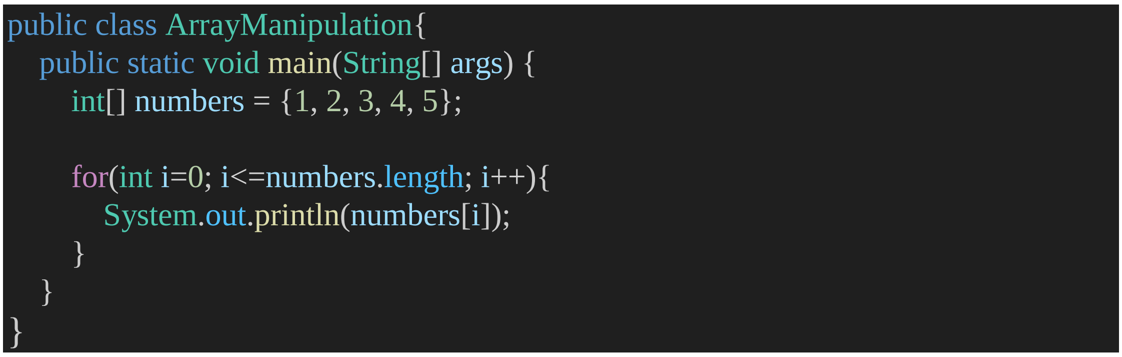
Assignment No: 1

Task 1:

Exercise 1: Array Manipulation

➢ Given 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]);
      }
   }
}
```

➢ Error:

Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: Index 5 out of bounds for length 5 at ArrayManipulation.main (ArrayManipulation.java:6)

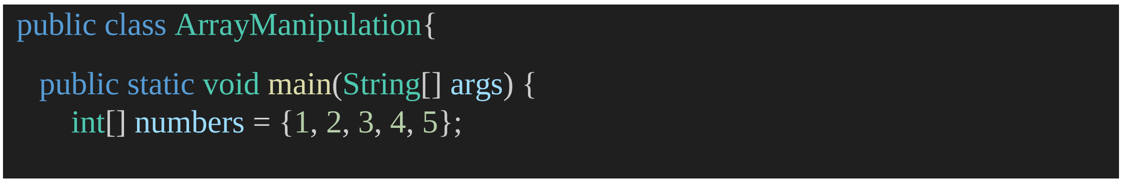
➢ Explanation:

The error in the provided code is due to an ArrayIndexOutOfBoundsException, which occurs when trying to access an index outside the bounds of the array. In this case, the loop condition i <= numbers.length allows i to reach the value of numbers.length, which is one index beyond the last element of the array. Since array indices start from 0, the valid indices for numbers array are from 0 to numbers.length - 1.

```
for(int i=0; i<=numbers.length; i++){
   System.out.println(numbers[i]);
}
```

The loop runs while i is less than or equal to numbers.length. When i becomes equal to numbers.length, it tries to access numbers[numbers.length], which is one element beyond the last element of the array. This results in an ArrayIndexOutOfBoundsException.

➢ 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]);
        }
    }
}
```

Exercise 2: Object-Oriented Programming

- **<u>Given Code:</u>**

```
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();
        }
    }
}
```

- **<u>Error:</u>**

  The method stop() is undefined for the type Car

- **<u>Explanation:</u>**

  This error occurs because the Car class does not have a stop() method defined, but you're trying to call it on a car object.

- **<u>Corrected code:</u>**

```
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();
    }
}
```

Exercise 3: Exception Handling

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

- **Error:**

  Array index out of bound.
  Exception in thread "main" java.lang.ArithmeticException: / by zero
  	at ExceptionHandling.divide(ExceptionHandling.java:14)
  	at ExceptionHandling.main(ExceptionHandling.java:10)

- **Explanation:**

  The error in the code is a runtime exception: ArithmeticException. This occurs because of the attempt to divide by zero in the divide method.

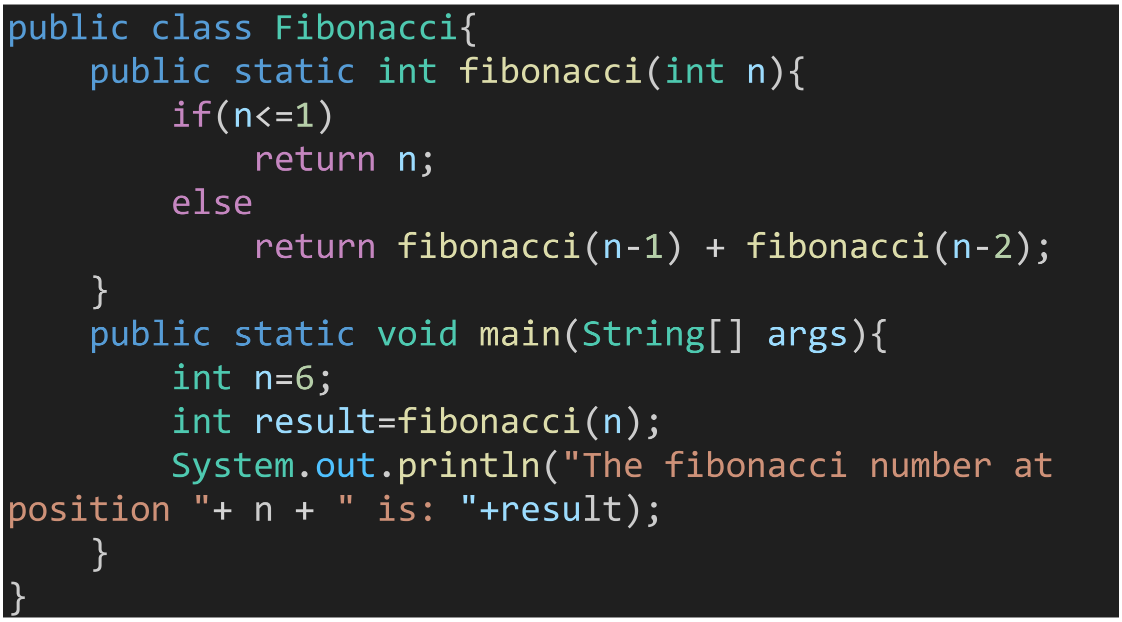
- **Corrected code:**

```
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
bound.");
        }try {
            int result = divide(10,0);
            System.out.println("Result: "+ result);
        }
        catch(ArithmeticException e){
            System.out.println("Cannot divide by
zero.");
        }
    } public static int divide(int a, int b){
        return a/b;
    }
}
```

Exercise 4: Fibonacci sequence

- **Given Code:**

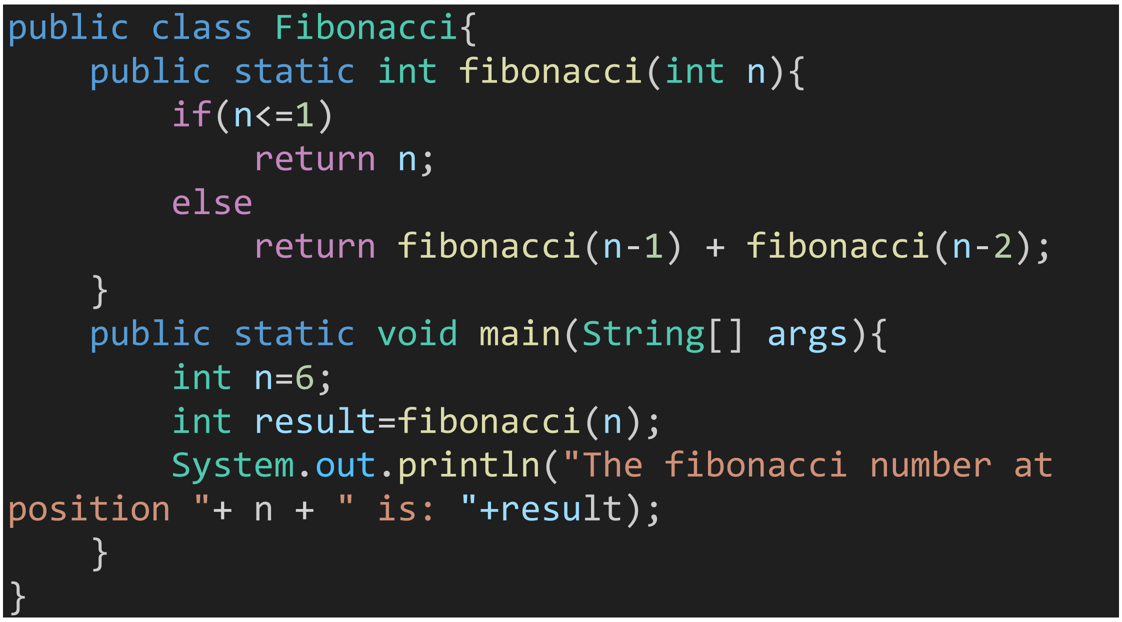


```
public class Fibonacci{
    public static int fibonacci(int n){
        if(n<=1)
            return n;
        else
            return fibonacci(n-1) + fibonacci(n-2);
    }
    public static void main(String[] args){
        int n=6;
        int result=fibonacci(n);
        System.out.println("The fibonacci number at
position "+ n + " is: "+result);
    }
}
```

- **Explanation:**

  The error in the code is related to inefficiency and potential stack overflow for larger values of n. The recursive approach for calculating Fibonacci numbers has exponential time complexity, which can lead to performance issues and stack overflow errors for larger values of n.

- **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: Prime Number

- **Given 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;
            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);
    }
}
```

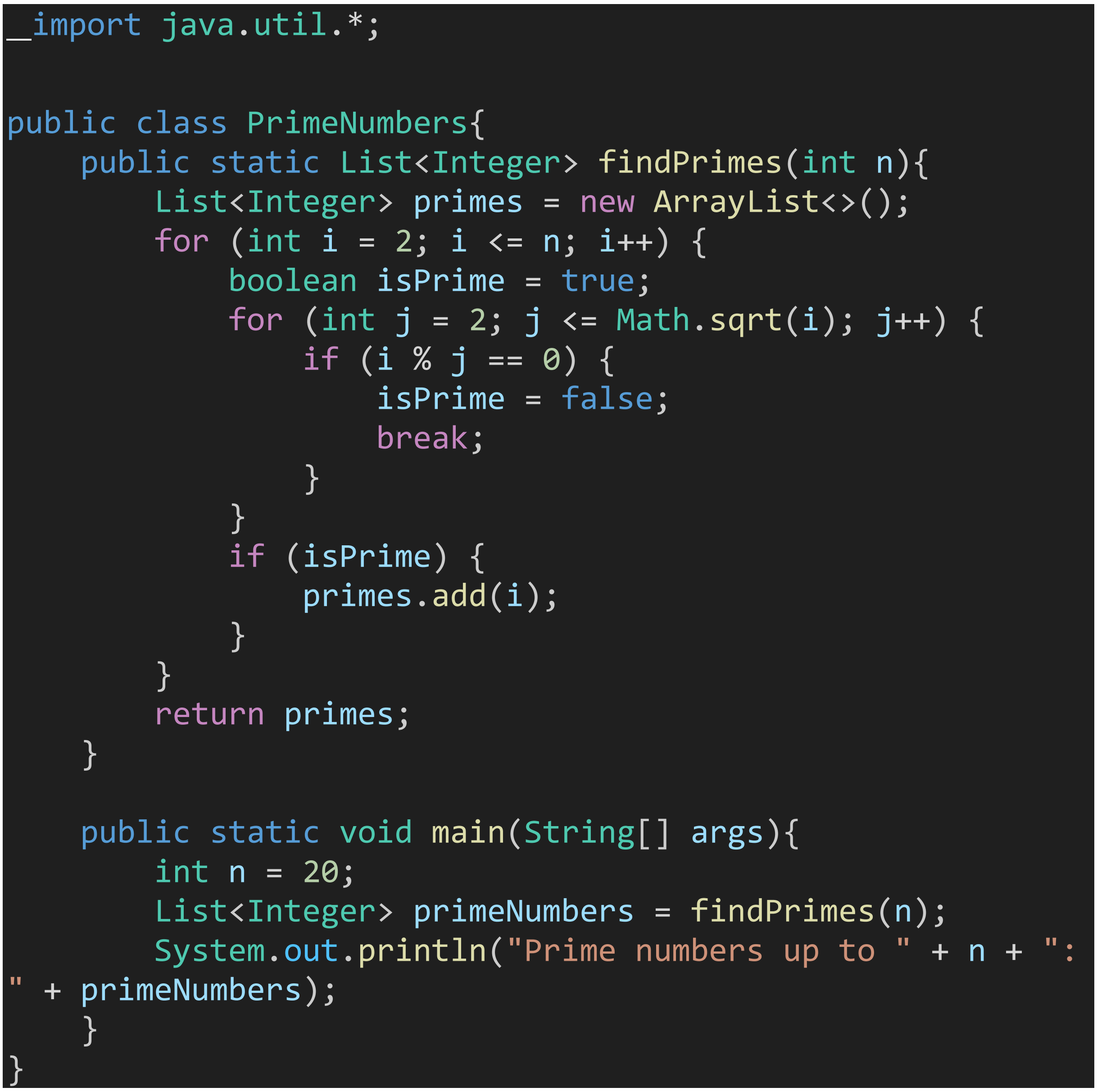
- **Error with Explanation:**

The error in the code is that it incorrectly identifies some numbers as prime numbers.

```
for(int j=2; j<i; j++){
   if(i%j==0){
      isPrime =false;
      break;
   }
}
```

In this loop, j iterates from 2 to i - 1. If i is divisible by any number in this range, it sets isPrime to false. However, the condition i % j == 0 only checks divisibility with numbers less than i. It should continue until j <= Math.sqrt(i) because if i is divisible by any number greater than its square root, it will also be divisible by a number smaller than its square root. This optimization reduces the time complexity of the algorithm.

➢ **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;
            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);
    }
}
```