ASSIGNMENT 1

TASK 1:

Debugging Exercise 1: Array Manipulation:

Error 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]);
		}
	}
}
```

Reason for the Error:

Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException

Index 5 out of bounds for length 5.

We are trying to print 5^{th} index element which is not present, therefore we get this Exception.

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 will be printed with line breaks

Debugging Exercise 2: Object-Oriented Programming

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

Reason for the Error:

Error: cannot find symbol car.stop();

symbol: method stop()

location: variable car of type Car

Here we did not define the stop() method in the class. So, the compiler throws exception, cannot find symbol. To solve this problem, we have to define the method or function 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();
	}
}
```

Output:

Starting the car.

Stopping the car.

Debugging Exercise 3: Exception Handling

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

Reason for Error:

Exception in thread "main" java.lang.ArithmeticException: / by zero

When we are trying to invoke the divide() method, compiler throws Exception. This Exception is an Unchecked Exception. We can solve this error by surrounding the statements with try-catch block and display the Exception messages and non-technical guiding messages.

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("Cannot divide by zero. ");
                }
        }
        public static int divide(int a , int b){
           return a/b;
        }
    }
}
```

Output:

Array index out of bounds.

Cannot divide by zero.

Exercise 4:

Wrong 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 Sequence at position "+n+" is: "+result);
         }
}
```

Error in the Code:

1. Highly inefficient: It recalculates the same Fibonacci numbers many times
(e.g., fib(3) is calculated multiple times when computing fib(5)).
This leads to exponential time complexity (O(2n)).

2. Can lead to StackOverflowError for larger n due to deep recursion. It is inefficient method.

Correct Code: Iterative Approach (Most Efficient for General Use)

```
public class FibonacciIterative {
    public static int fibonacci(int n) {
            if (n < 0)
                throw new IllegalArgumentException("Input cannot be negative.")
            if (n <= 1)
                return n;
            int a = 0; // Represents F(i-2)
            int b = 1; // Represents F(i-1)
            int result = 0; // Represents F(i)

            for (int i = 2; i <= n; i++) {
                result = a + b; // Current Fibonacci is sum of previous two
                a = b;          // Move F(i-1) to become F(i-2) for next iteration
                b = result;     // Move F(i) to become F(i-1) for next iteration
            }
            return result;
    }
    public static void main(String[] args) {
            int n = 10;
            System.out.println("Fibonacci(" + n + ") = " + fibonacci(n));
            System.out.print("Fibonacci Sequence up to " + n + ": ");
            for (int i = 0; i <= n; i++) {
                System.out.print(fibonacci(i) + " ");
            }
                System.out.println();
                int largeN = 40; // Test with a larger number
                System.out.println("Fibonacci(" + largeN + ") = " + fibonacci(largeN));
    }
}
```

Exercise 5:

Error Code:

```
import java.util.*;
public class primeNumbers{
	public static List<Integer>findPrimes(int n){
		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 upto "+n+":"+primeNumbers);
		}
	}
}
```

Reason for Error:

The List<Integer> primes variable is not declared or initialized.

The inner loop condition for checking primality is inefficient and incorrect.

Correct Code:

```
import java.util.*;
public class primeNumbers {
   public static List<Integer> findPrimes(int n) {
              List<Integer> primes = new ArrayList<>();
               if (n < 2) {
                  return primes;
              }
              for (int i = 2; i <= n; i++) {
                        boolean isPrime = true;
                        // Optimized: Only check divisors up to the square root of i
                    for (int j = 2;  j * j <= i;  j++) {
                         if (i % j == 0) {
                              isPrime = false;
                              break; // No need to check further divisors for 'i'
                         }
                    }
                if (isPrime) {
                     primes.add(i); // Add the prime number to the list
                }
             }
      return primes;
  }

 public static void main(String[] args) {
      // Corrected primitive type
      int n = 20;
      List<Integer> primeNumbers = findPrimes(n);
      System.out.println("Prime Numbers upto " + n + ": " + primeNumbers);
```

```
        int n2 = 100;
        List<Integer> primeNumbers2 = findPrimes(n2);
        System.out.println("Prime Numbers upto " + n2 + ": " + primeNumbers2);

        int n3 = 1; // Test edge case
        List<Integer> primeNumbers3 = findPrimes(n3);
        System.out.println("Prime Numbers upto " + n3 + ": " + primeNumbers3);
    }
}
```