Practical 6

1. Write a recursive method `countUp` that displays the count-up from 1 to n, where n is a positive integer.
Hint: A recursive call will occur before you display anything.

```
//Jun Yan
public void countUp(int n)
{
    if (n == 1)
        ;
    else
    {
     countUp(n - 1);
    }
    System.out.println(n);
}

public static void main(String[] args)
{
    countUp(3);
}
```

2. Write a recursive method that takes two integers and finds the `greatest common divisor` (GCD). Complete the Extra Challenge question in Practical 3 Q1 if you have not done so.

```
//Jia Jian
public class GCD{
      public int gcd(int num1, int num2){
            if(num2 == 0){
                  return num1;
            }
            return gcd(num2, num1 % num2);
      }

      public static void main(String[ ] args){
            GCD g = new GCD();
            Scanner scan = new Scanner(System.in);
            System.out.print(“Enter 1st Digit: “);
            Int num1 = scan.nextInt();
            System.out.print(“Enter 2nd Digit: “);
            Int num2 = scan.nextInt();
            System.out.println(“\nGCD for “+ num1 +” &”+
                                    num2+”is”+g.gcd(num1,num2));
      }
}
```

3. Write code to compare the performance of 3 different implementations of the `Fibonacci` numbers: using recursion, iteration and array.

```
//Lee Ling - recursion
public int fibonacciRecursive(int n){
        if(n == 0)
            return 0;
        else if(n == 1)
            return 1;
        else
            return fibonacciRecursive(n - 1) + fibonacciRecursive(n - 2);
}
```

//Lee Yong - iteration

```
public static int fibonacciIteration(int number){
      int num1, num2 = 0, current = 1;
      for(int i = 1; i  <= number ; i++){
            num1 = num2;
            num2 = current;
            current = num1 + num2;
      }
      return current;
}
```

//Han Yao - array

```
public void fibonacciArray(int n){
        ArrayList<Integer> arr = new ArrayList<Integer>();//ArrayList
        int i = 0, j = 1;
        int s = 0;
        while(s < n){
            int temp = i + j;
            arr.add(temp);
            i = j;
            j = temp;
            s++;
        }
        for(Integer v : arr)
            System.out.print(v+" ");
}
```

4. Write a recursive method `displayBackward` that writes a given array backward. Consider the last element of the array first.

You may add your code to the Chapter 6 NetBeans project's `RecursiveDisplayArray` class.

```
//Kah Yee
public static void main(String[] args) {
    Integer[] intArray = {10, 20, 30, 40, 50, 60};

    System.out.println("\n\nInvoking displayBackward()...");
    displayBackward(intArray, 0, intArray.length - 1);
}


public static void displayBackward(Object[] array, int first, int last){
      if (first <= last) {
            System.out.print(array[last] + " ");
            displayBackward(array, first, last - 1);

      }
}
```

5. Write a recursive method `countNodes` that counts and returns the number of nodes in a chain of linked nodes.

You may add your code to the Chapter 6 NetBeans project's `SimpleList` class.

```
//Kuan Xian
private int countNodes(Node<T> data){
      if(data == null)
            return 0;
      return 1 + countNodes(data.next);
}
```