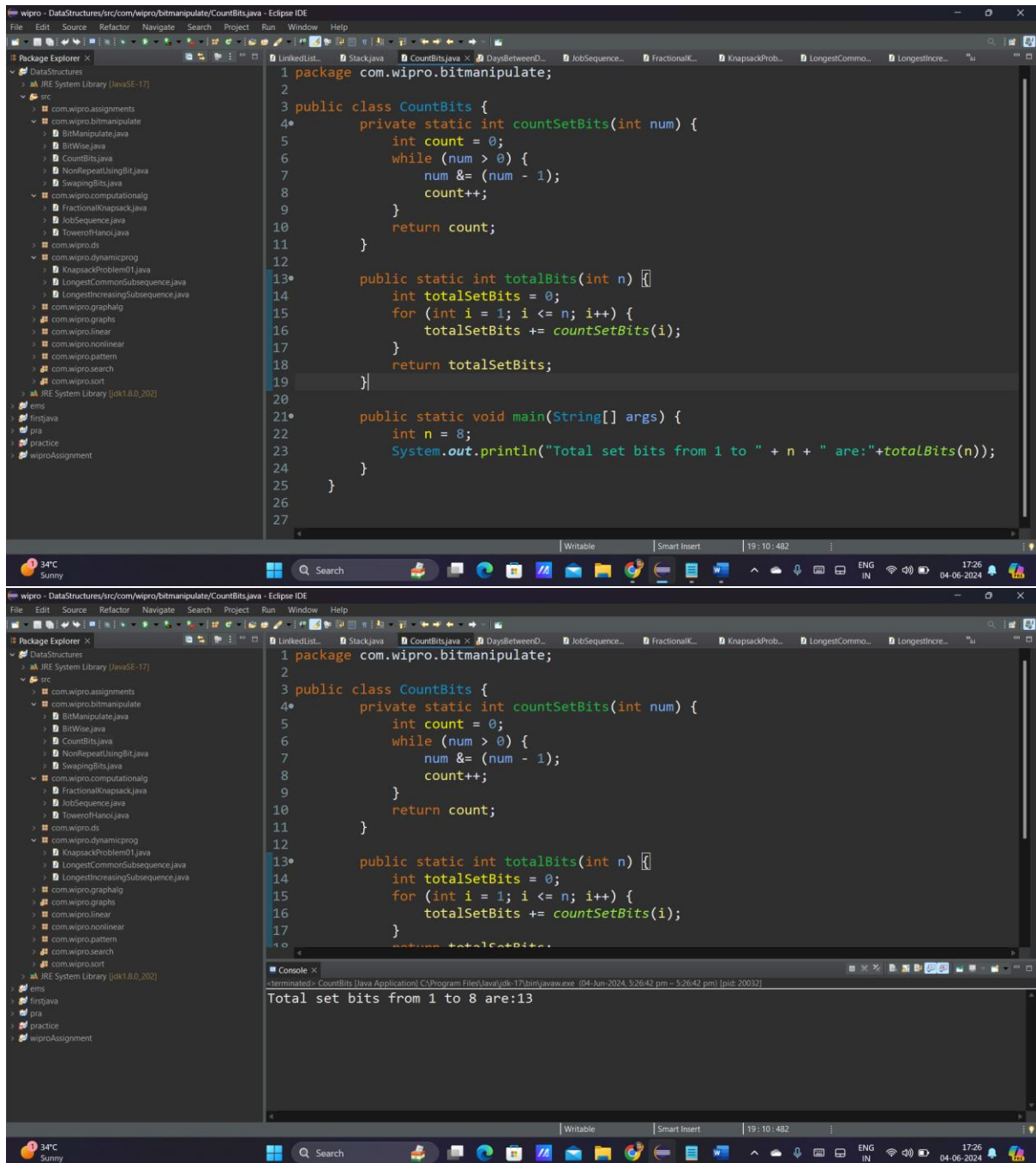


Assignment-Day12

Day 12:

Task 1: Bit Manipulation Basics

Create a function that counts the number of set bits (1s) in the binary representation of an integer. Extend this to count the total number of set bits in all integers from 1 to n.



The screenshot displays the Eclipse IDE with a Java project named 'wipro'. The Package Explorer on the left shows the project structure, including a package 'com.wipro.bitmanipulate'. The main editor shows the source code for 'CountBits.java'.

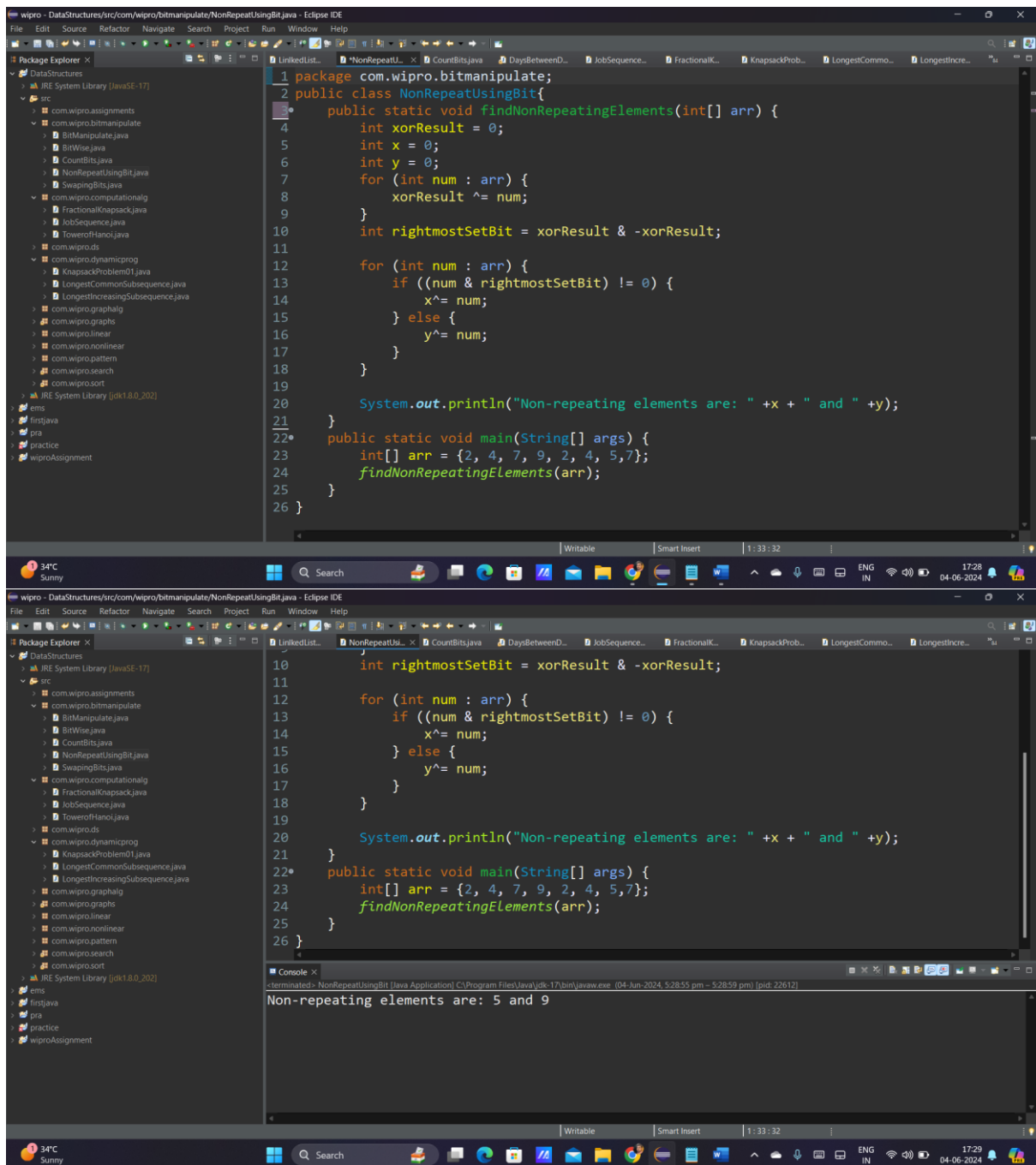
```
1 package com.wipro.bitmanipulate;
2
3 public class CountBits {
4     private static int countSetBits(int num) {
5         int count = 0;
6         while (num > 0) {
7             num &= (num - 1);
8             count++;
9         }
10        return count;
11    }
12
13    public static int totalBits(int n) {
14        int totalSetBits = 0;
15        for (int i = 1; i <= n; i++) {
16            totalSetBits += countSetBits(i);
17        }
18        return totalSetBits;
19    }
20
21    public static void main(String[] args) {
22        int n = 8;
23        System.out.println("Total set bits from 1 to " + n + " are:" + totalBits(n));
24    }
25 }
26
27
```

The console output at the bottom shows the result of the program execution:

```
Total set bits from 1 to 8 are:13
```

Task 2: Unique Elements Identification

Given an array of integers where every element appears twice except for two, write a function that efficiently finds these two non-repeating elements using bitwise XOR operations.



The screenshot displays the Eclipse IDE with a Java project named 'DataStructures'. The package explorer on the left shows the project structure, including a 'src' folder with various Java files. The main editor shows the file 'NonRepeatUsingBit.java'. The code implements a function 'findNonRepeatingElements' that takes an integer array and returns the two unique elements. The function uses bitwise XOR to find the rightmost set bit in the XOR of all elements, then uses this bit to partition the array into two groups, each containing one unique element. The main method tests the function with the array {2, 4, 7, 9, 2, 4, 5, 7}.

```
1 package com.wipro.bitmanipulate;
2 public class NonRepeatUsingBit{
3     public static void findNonRepeatingElements(int[] arr) {
4         int xorResult = 0;
5         int x = 0;
6         int y = 0;
7         for (int num : arr) {
8             xorResult ^= num;
9         }
10        int rightmostSetBit = xorResult & -xorResult;
11
12        for (int num : arr) {
13            if ((num & rightmostSetBit) != 0) {
14                x ^= num;
15            } else {
16                y ^= num;
17            }
18        }
19
20        System.out.println("Non-repeating elements are: " + x + " and " + y);
21    }
22    public static void main(String[] args) {
23        int[] arr = {2, 4, 7, 9, 2, 4, 5, 7};
24        findNonRepeatingElements(arr);
25    }
26 }
```

The console output shows the result of the function: "Non-repeating elements are: 5 and 9".