

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.

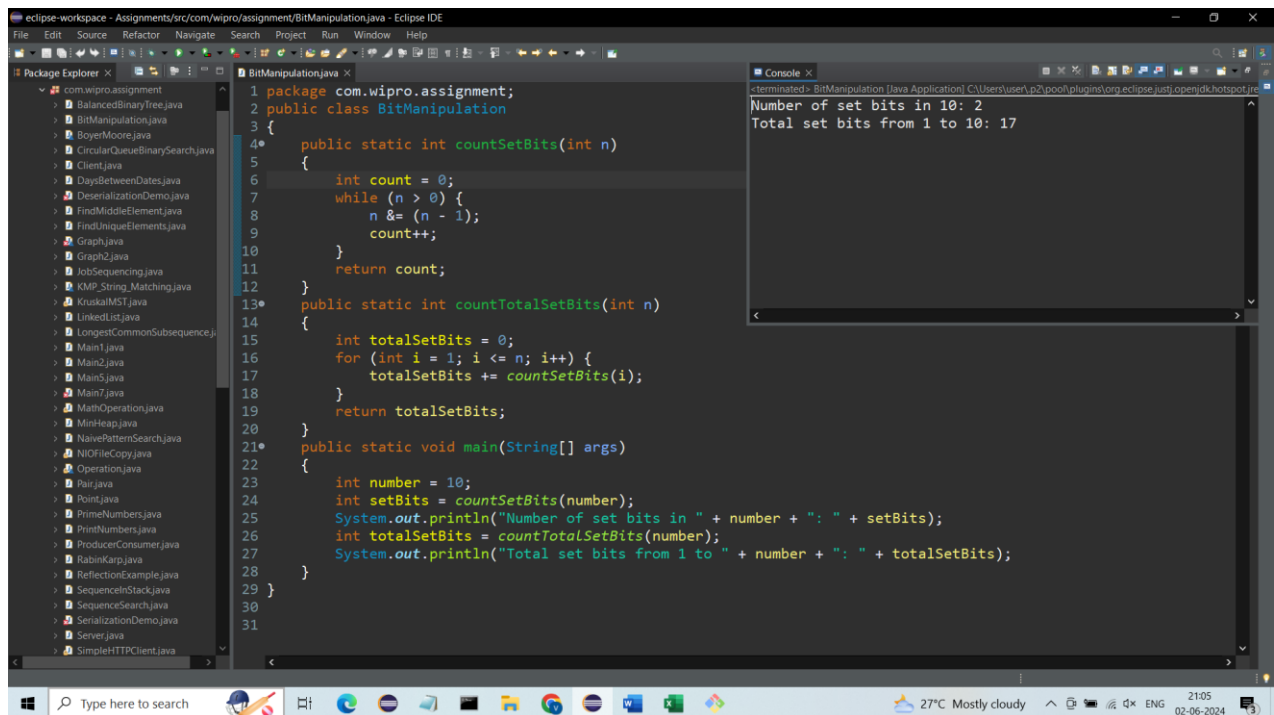
Function for counting number of set bits: -

```
public static int countSetBits(int n)
{
    int count = 0;
    while (n > 0) {
        n &= (n - 1);
        count++;
    }
    return count;
}
```

Function for counting total number of set bits in all integers from 1 to n: -

```
public static int countTotalSetBits(int n)
{
    int totalSetBits = 0;
    for (int i = 1; i <= n; i++) {
        totalSetBits += countSetBits(i);
    }
    return totalSetBits;
}
```

Output: -



The screenshot shows the Eclipse IDE interface. The Package Explorer on the left lists various Java files under the package 'com.wipro.assignment'. The main editor displays the file 'BitManipulation.java' with the following code:

```
1 package com.wipro.assignment;
2 public class BitManipulation
3 {
4     public static int countSetBits(int n)
5     {
6         int count = 0;
7         while (n > 0) {
8             n &= (n - 1);
9             count++;
10        }
11        return count;
12    }
13    public static int countTotalSetBits(int n)
14    {
15        int totalSetBits = 0;
16        for (int i = 1; i <= n; i++) {
17            totalSetBits += countSetBits(i);
18        }
19        return totalSetBits;
20    }
21    public static void main(String[] args)
22    {
23        int number = 10;
24        int setBits = countSetBits(number);
25        System.out.println("Number of set bits in " + number + ": " + setBits);
26        int totalSetBits = countTotalSetBits(number);
27        System.out.println("Total set bits from 1 to " + number + ": " + totalSetBits);
28    }
29 }
30
31
```

The Console window on the right shows the output of the program:

```
Number of set bits in 10: 2
Total set bits from 1 to 10: 17
```

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.

Function for utilizing XOR operations to find the two unique elements in an array where all other elements appear twice.

```

public static int[] findUniqueElements(int[] arr)
{
    int xor = arr[0];
    for (int num : arr)
    {
        xor ^= num;
    }
    int rightmostSetBit = xor & ~(xor - 1);
    int unique1 = 0, unique2 = 0;
    for (int num : arr) {
        if ((num & rightmostSetBit) != 0) {
            unique1 ^= num;
        } else {
            unique2 ^= num;
        }
    }
    return new int[]{unique1, unique2};
}

```

Output: -

The screenshot shows the Eclipse IDE interface. The Package Explorer on the left lists the project structure. The main editor displays the source code of the `FindUniqueElements` class. The Console on the right shows the output of the program, which is "Unique elements: 7 9". The bottom status bar shows the system temperature as 27°C and the date as 02-06-2024.