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.

```
public class CountSetBits {


  public static int countSetBits(int n) {
    int count = 0;
    while (n > 0) {
      if ((n & 1) == 1) {
        count++;
      }
      n >>= 1;
    }
    return count;
  }
  public static int countTotalSetBits(int n) {
    int totalSetBits = 0;


    for (int i = 1; i <= n; i++) {
      totalSetBits += countSetBits(i);
    }
    return totalSetBits;
  }


  public static void main(String[] args) {
    int number = 5;
    int setBits = countSetBits(number);
```

```
        System.out.println("The number of set bits in " + number + " is: " + setBits);

        int n = 10;
        int totalSetBits = countTotalSetBits(n);
        System.out.println("The total number of set bits in all integers from 1 to " + n + " is: " +
totalSetBits);
    }
}
```

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.

```
public class FindNonRepeating {

    public static int[] findNonRepeating(int[] nums) {
        int xor = nums[0];
        for (int num : nums) {
            xor ^= num;
        }
        int rightmostSetBit = xor & ~(xor - 1);
        int group1 = 0;
        int group2 = 0;
        for (int num : nums) {
            if ((num & rightmostSetBit) != 0) {
                group1 ^= num;
            } else {
                group2 ^= num;
            }
        }
        return new int[]{group1, group2};
    }

    public static void main(String[] args) {
        int[] nums = {1, 2, 1, 3, 2, 4};
        int[] nonRepeating = findNonRepeating(nums);
        System.out.println("The two non-repeating elements are: " + nonRepeating[0] + " and " + nonRepeating[1]);
    }
}
```