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.

```
package BitManipulation;


public class BitCount {
public static void main(String args[]) {
    int n=13;
    int count=0;

    while(n>0) {
        count += n&1;
        n>>=1;
    }
    System.out.println(count);
}
}
```

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.

```
package BitManipulation;
import java.util.Arrays;
```

```java
public class UniqueElementsIdentification {

    public static int[] findUniqueElements(int[] nums) {
        // Step 1: XOR all elements to get the XOR of the two unique numbers
        int xorResult = 0;
        for (int num : nums) {
            xorResult ^= num;
        }

        // Step 2: Find a set bit in xorResult (any set bit will work)
        int setBit = xorResult & -xorResult; // This isolates the rightmost set bit

        // Step 3: Divide the numbers into two groups and XOR them separately
        int unique1 = 0, unique2 = 0;
        for (int num : nums) {
            if ((num & setBit) == 0) {
                unique1 ^= num;
            } else {
                unique2 ^= num;
            }
        }

        // Return the two unique numbers
        return new int[]{unique1, unique2};
    }

    public static void main(String[] args) {
        int[] nums = {1, 2, 1, 3, 2, 5};
        int[] uniqueElements = findUniqueElements(nums);
        System.out.println("The two unique elements are: " + 
Arrays.toString(uniqueElements));
    }
}
```