```
public class Prime {
  public static boolean isPrime(int num) {
    if(num <= 1) return false;
    for(int i = 2; i <= num/2; i++){
      if(num % i == 0){
        return false;
      }
    }
    return true;
  }
}
```

```
public static int countBits(int n){
        Stream<Character> st = Integer.toBinaryString(n).chars().mapToObj(i ->
(char)i);
        return st.filter(i -> i.equals('1')).mapToInt(i ->
Character.getNumericValue(i)).sum();
    }
```

For this code review i decided to do two Rank Up level challenges, this means that i completed challenges at or above my level in code wars.

The first challenge was to detect if a number was prime; the function would take in an int and then return a boolean answer. To start, I first had to figure out what exactly was a prime number; in essence, a prime number is only divisible by 1 and itself, so for starters, I would check to see if a number was less than or equal to 1 if so I would say that wasn't a prime number and return false if it was greater than one then I would iterate trough every number in between 2 and half of the original number(because the highest factor of a number is half of itself), if the original can be divided by that iteration and have 0 as the remainder then it wasn't a prime number, and I would return false, if by end of the sequence I hadn’t return false then the number was prime, and I could return true.

The next problem has been one of the most fun problems I've solved in a while, the core of the problem was to figure out how many 1’s were in the binary representation of a number to do this, I first took the number and created a Stream of chars based on the binary representation of the number, then i filtered these chars to leave only the ones, finally, I mapped the chars to ints and added them up, and that was my answer at first i tried doing this with an array and for loop, but after getting it working i knew i could do it fewer lines of could so i switched to stream, which i find to be very similar to ArrayList but with more methods, in the end, i went from about 10 lines of code to only 2, but once i submitted and saw the first answer, i facepalmed on how simple the answer was, turners out javas Interger object has a method for precisely the problem that i had to solve, so i could have done it in one line of code `Interger.bitCount(n);` that method specifically returns the number of 1’s in the binary representation of an int, which honestly just leads me to ask how often does this particular problem need to be solved for oracle to add this method to base java, just makes me laugh.