Day 1 - Code Java Smart

Check for Prime Number

We know that any integer number can be written in the form of 6k+i, where ‘k’ is a non-negative integer (like 0, 1, 2, 3, ...) and ‘i’ is a number between 0 and 5 (so i can be 0, 1, 2, 3, 4, or 5). If we look closely, we’ll notice that when i is 0, 2, 3, or 4, the numbers 6k, 6k+2, 6k+3, and 6k+4 are all divisible by either 2 or 3. But prime numbers greater than 3 can't be divisible by 2 or 3. Therefore, the only forms left that a prime number can have are 6k+1 or 6k+5 (since these forms are not divisible by 2 or 3).

Instead of checking every number up to the √n to see if it divides n, we only check numbers of the form 6k+1 and 6k+5. This optimized approach for calculating √n reduces the number of operations and achieves a threefold performance improvement compared to a standard method.

Program

```
static boolean isPrime (int n) {


    // Check if n is 1 or 0
    if (n <= 1)
        return false;


    // Check if n is 2 or 3
    if (n == 2 || n == 3)
        return true;
```

```
    // Check whether n is divisible by 2 or 3
    if (n % 2 == 0 || n % 3 == 0)
        return false;

    // Check from 5 to square root of n
    // Iterate i by (i+6)
    for (int i = 5; i <= Math.sqrt(n); i = i + 6)
        if (n % i == 0 || n % (i + 2) == 0)
            return false;

    return true;
}
```