I feel like even though it's implied in the top answers, it could be useful to the layman to state this more plainly. (I'm just rewording what can be found at Wikipedia: [Primality test](https://en.wikipedia.org/wiki/Primality_test" \t "_blank))  
  
Every single integer *n* greater than 4 can be written in the form  
*n* = 6*k* + *i*  
where *k* is any integer and *i* is either -1, 0, 1, 2, 3, or 4. This in itself isn't that remarkable, right? It's obvious because every integer is either a multiple of 6 or some other number between a multiple of 6 and the next multiple of 6.  
  
But then what's interesting is that considering different values for *i*, you can see that only ±1 could include the prime numbers. Let's try:  
  
6*k* + 0 -> not prime, because it's a number divisible by 6  
  
6*k* + 2 -> not prime, because 6*k* is divisible by 6 (an even number), and if you add 2 to it, you just get another even number  
  
6*k* + 3 -> not prime, because 6*k* is divisible by 3, and if you add 3 to any number that is already divisible by 3, then you'll get another number that's also divisible by 3  
  
6*k* + 4 -> not prime, because 6*k* is an even number, and if you add 4 to any even number, you'll get another even number.  
  
That leaves only 6*k* ± 1 which could include the prime numbers. It's important to note that it also includes numbers that are not prime, like 25 (which is 6\*4 + 1) and 35

(which is 6\*6 - 1), for example, but all prime numbers are also in that collection.  
  
That's it, pretty much.

// A optimized school method based Java

// program to check if a number is prime

import java.io.\*;

class GFG {

    static boolean isPrime(int n)

    {

        // Corner cases

        if (n <= 1) return false;

        if (n <= 3) return true;

        // This is checked so that we can skip

        // middle five numbers in below loop

        if (n % 2 == 0 || n % 3 == 0) return false;

**The loop condition could be (i <= Math.sqrt(n)), but if you square both sides you get the equivalent expression (i\*i <= n), which is much faster.**

**for (int i = 5; i \* i <= n; i = i + 6)**

**if (n % i == 0 || n % (i + 2) == 0)**

**return false;**

**return true;**

    }

    // Driver Program

    public static void main(String args[])

    {

        if(isPrime(11))

            System.out.println(" true");

        else

            System.out.println(" false");

        if(isPrime(15))

            System.out.println(" true");

        else

            System.out.println(" false");

    }

}

/\*This code is contributed by Nikita Tiwari.\*/