PRIME NUMBERS::

① for (int i=2; i< N ; i++){
    if (N%. i == 0){
        Not prime;
        break;
    }
}
prime;
}

② int c = 2;
if (n <= 1){
    return false;
}
while (c*c <= n){
    if (n%. C == 0)
        return false;
    }
    c++;
}
return true;
}

// c*c <= n means the loop runs only square root of n times. Coz the loop runs same twice so, they are neglected.

$\therefore c <= \sqrt{n} \Rightarrow \boxed{c*c <= n}$

Prime of range

①
```
for (int i=2; i<=n; i++){
    System.out.println(i++" "
                        isPrime(i));
}
```

② Only the prime numbers are printed between the range. (Optimized way)

Q N = 40 Print prime numbers between 0 & 40

Ans: 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37

Note : Ignore the things that gets repeated.

For example:

* If we know 2 is prime, the multiple of 2 is not prime.
* If 3 is prime, the multiple of 3 is not prime

~~1~~ (2) (3) ~~4~~ (5) ~~6~~ 7 ~~8~~ ~~9~~ ~~10~~
11 ~~12~~ 13 ~~14~~ ~~15~~ ~~16~~ 17 ~~18~~ 19 ~~20~~
~~21~~ ~~22~~ 23 ~~24~~ ~~25~~ ~~26~~ ~~27~~ ~~28~~ 29 ~~30~~
31 ~~32~~ ~~33~~ ~~34~~ ~~35~~ ~~36~~ 37 ~~38~~ ~~39~~ ~~40~~

0 → Prime - False
x → Not prime - True

* When you eliminate all multiples of 2, then only little bit multiples of 3 will get eliminated.

For example:

⇒ Via 2, when we eliminate 18, we don't need to eliminte 18 again when multiples of 3 getting eliminated.

* Checking for 6, in this case, 6 is itself not a prime, i.e, the factor of 6 has appeared before, no need to check for it.

And for 18, 18 is also not a prime since one of its factor has appeared before.

For that reason, running the loop only sqrt(n) times that will ignore the repeated case.

This is known as Sieve of Eratosthenes Algorithm. which is an ancient algorithm that is used to find all the primes less than given number N.

TIME COMPLEXITY:

How many numbers exist in a range of number that is divisible by 2.

lets say n=40 no's divisible by 2

$\frac{40}{2} = 20$, by 3 $\frac{40}{3} = 13$

by 5 = $\frac{40}{5} = 8$,

i.e., $\frac{N}{2} + \frac{N}{3} + \frac{N}{5} + \frac{N}{7} + \cdots + \frac{N}{P}$

where P = Highest prime that is less than N

$= N\left(\frac{1}{2} + \frac{1}{3} + \frac{1}{5} + \frac{1}{7} + \cdots \frac{1}{P}\right)$

Solving by HP

Harmonic Progression for prime is $\log(\log N)$

$\therefore N(\log(\log N))$

$\therefore$ Time Complexity $= O(N(\log(\log(N))))$