Check for Prime in Java

Given a positive integer, check if the number is prime or not. A prime is a natural number greater than 1 that has no positive divisors other than 1 and itself. Examples of first few prime numbers are {2, 3, 5, ...}.

Examples :

```
Input: n = 11
Output: true


Input: n = 15
Output: false


Input: n = 1
Output: false
```

Naive Method:

Iterate from 2 to (n-1) and check if any number in this range divides n. If the number divides n, then it is not a prime number.

```
// A naive method based JAVA program
// to check if a number is prime
class GfG {
    public static void main(String args[]) {
        int n = 11;
        boolean prime = true;
        if (n <= 1)
            prime = false;
        else {
            // Check from 2 to n-1
            for (int i = 2; i < n; i++)
```

```
                if (n % i == 0)
                    prime = false;
        }
        if (prime)
            System.out.println(" true");
        else
            System.out.println(" false");
    }
}
```

Output
true

Time complexity: O(n)
Auxilary Space: O(1)

Efficient Method :

Iterate through all numbers from 2 to square root of n and for every number check if it divides n [because if a number is expressed as n = x*y and any of the x or y is greater than the root of n, the other must be less than the root value]. If we find any number that divides, we return false.

```
// A efficient JAVA program
// to check if a number is prime

import java.io.*;
import java.util.*;

class GfG {

    public static void main(String[] args) {

        int n = 65;
        boolean prime = true;
        if (n == 1)
```

```
            prime = false;

        for (int i = 2; i * i <= n; i++) {
            if (n % i == 0)
                prime = false;
        }

        System.out.println(prime);
    }
}
```

```
Output
false
```

Time complexity: O(sqrt(n))

Auxilary Space: O(1)

More Efficient Code(for large numbers) :

Steps:

- We will deal with a few numbers such as 1, 2, and 3, and the numbers which are divisible by 2 and 3 in separate cases.
- For the remaining numbers, we iterate from 5 to sqrt(n) and check for each iteration whether (that value) or (that value + 2) divides n or not and increment the value by 6 [because any prime can be expressed as 6n+1 or 6n-1].
- If we find any number that divides, we return false, else true.

```
// A more efficient JAVA program
// to check if a number is prime
```

```java
import java.io.*;
import java.util.*;

class GfG {

    public static void main(String[] args) {

        int n = 1031;
        boolean prime = true;
        if (n == 1)
            prime = false;

        if (n == 2 || n == 3)
            prime = true;

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

        for (int i = 5; i * i <= n; i = i + 6) {
            if (n % i == 0 || n % (i + 2) == 0)
                prime = false;
        }

        System.out.println(prime);
    }
}
```

Output

```
true
```

Time complexity: O(sqrt(n))

Auxilary Space: O(1)