Sieve of Eratosthenes

Given a number n, print all primes smaller than or equal to n. It is also given that n is a small number.
For example, if n is 10, the output should be "2, 3, 5, 7". If n is 20, the output should be "2, 3, 5, 7, 11, 13, 17, 19".

The sieve of Eratosthenes is one of the most efficient ways to find all primes smaller than n when n is smaller than 10 million or so (Ref Wiki).

We strongly recommend that you click here and practice it, before moving on to the solution.

Following is the algorithm to find all the prime numbers less than or equal to a given integer *n* by Eratosthenes' method:

1. Create a list of consecutive integers from 2 to *n*: (2, 3, 4, ..., *n*).
2. Initially, let *p* equal 2, the first prime number.
3. Starting from *p*, count up in increments of *p* and mark each of these numbers greater than *p* itself in the list. These numbers will be 2*p*, 3*p*, 4*p*, etc.; note that some of them may have already been marked.
4. Find the first number greater than *p* in the list that is not marked. If there was no such number, stop. Otherwise, let *p* now equal this number (which is the next prime), and repeat from step 3.

When the algorithm terminates, all the numbers in the list that are not marked are prime.

Explanation with Example:
Let us take an example when n = 50. So we need to print all print numbers smaller than or equal to 50.

We create a list of all numbers from 2 to 50.

	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
41	42	43	44	45	46	47	48	49	50

According to the algorithm we will mark all the numbers which are divisible by 2.

	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
41	42	43	44	45	46	47	48	49	50

Now we move to our next unmarked number 3 and mark all the numbers which are multiples of 3.

	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
41	42	43	44	45	46	47	48	49	50

We move to our next unmarked number 5 and mark all multiples of 5.

	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
41	42	43	44	45	46	47	48	49	50

We continue this process and our final table will look like below:

	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
41	42	43	44	45	46	47	48	49	50

So the prime numbers are the unmarked ones: 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47.

Thanks to **Krishan Kumar** for providing above explanation.

Implementation:
Following is C++ implementation of the above algorithm. In the following implementation, a boolean array arr[] of size n is used to mark multiples of prime numbers.

C/C++ Java

```java
// Java program to print all primes smaller than or equal to
// n using Sieve of Eratosthenes

class SieveOfEratosthenes
{
    void sieveOfEratosthenes(int n)
    {
        // Create a boolean array "prime[0..n]" and initialize
        // all entries it as true. A value in prime[i] will
        // finally be false if i is Not a prime, else true.
        boolean prime[] = new boolean[n+1];
        for(int i=0;i<n;i++)
            prime[i] = true;

        for(int p = 2; p*p <=n; p++)
        {
            // If prime[p] is not changed, then it is a prime
            if(prime[p] == true)
            {
                // Update all multiples of p
                for(int i = p*2; i <= n; i += p)
                    prime[i] = false;
            }
        }

        // Print all prime numbers
        for(int i = 2; i <= n; i++)
        {
            if(prime[i] == true)
                System.out.print(i + " ");
        }
    }

    // Driver Program to test above function
    public static void main(String args[])
    {
        int n = 30;
        System.out.print("Following are the prime numbers ");
        System.out.println("smaller than or equal to " + n);
        SieveOfEratosthenes g = new SieveOfEratosthenes();
        g.sieveOfEratosthenes(n);
    }
}

// This code has been contributed by Amit Khandelwal.
```

Run on IDE

Output:

```
Following are the prime numbers below 30
2 3 5 7 11 13 17 19 23 29
```