Lab 1 – Sieve of Eratosthenes

The first few numbers primes are:

2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, ...

One technique for finding all the prime numbers in the range from 2 through to any given number *N* is known as the Sieve of Eratosthenes [see wikipedia page: Sieve of Eratosthenes]. The technique starts by listing all the numbers from 2 through to the given number N :

2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, ..., N.

The first number 2 is a prime number, but the higher multiples of 2 (4, 6, 8, 10, 12, 14, ...) are not, and so they are crossed out in the list :

2, 3, [4], 5, [6], 7, [8], 9, [10], 11, [12], 13, [14], 15, [16], 17, [18], 19, [20], 21, [22], 23, [24], 25, ..., N.

where crossed out numbers are encased in square brackets. The next number after 2 that was not crossed out is 3, the next prime, but then all higher multiples of 3 (6, 9, 12, 15, 18, 21, ...) are not prime, and so they are crossed out in the list (if they are not already crossed out) :

2, 3, [4], 5, [6], 7, [8], [9], [10], 11, [12], 13, [14], [15], [16], 17, [18], 19, [20], [21], [22], 23, [24], 25, ..., N.

The next number after 3 that was not crossed out is 5, the next prime, but then all higher multiples of 5 (10, 15, 20, 25, 30, 35, ...) are not prime, and so they are crossed out in the list (if they are not already crossed out) :

2, 3, [4], 5, [6], 7, [8], [9], [10], 11, [12], 13, [14], [15], [16], 17, [18], 19, [20], [21], [22], 23, [24], [25], ..., N.

This procedure is repeated until a number is reach in the list that has not been crossed out and whose value is greater than the square root of N, then all the numbers that are not crossed out in the list will be the primes numbers from 2 through to the given number N.

Write a Java class `Sieve` to compute the prime numbers from 2 up to a given number *N* which is entered by the user. As part of your solution you must use an array to represent the sequence of numbers (which may contain both crossed out and non-crossed out numbers). You must produce input and output in the precise form shown below (in the Example Interaction) and you must write and use (at least) the following public functions as part of your solution:

`public boolean[] createSequence(int)` - which takes a positive integer *N* (whose value is at least 2) and which returns an array initialised to represent the sequence of numbers

from 2 through to *N*. The method should set the first two positions in the array to `true` i.e. those values are crossed out.

`public void crossOutHigherMultiples(boolean[], int)` - which takes a sequence of numbers *σ* (which may contain both crossed out and non-crossed out numbers) and a number *n*, and which crosses out all higher multiples of the given number *n* from the given sequence of numbers *σ*.

`public String sequenceToString(boolean[])` - which takes a sequence of numbers *σ* (which may contain both crossed out and non-crossed out numbers) and which returns a `String` representation of the given sequence of numbers *σ* (where crossed out numbers are denoted by encasement in square bracket characters) separated by commas characters, e.g., given 2nd last sequence above, of crossed out and non-crossed out numbers, this method should return a reference to the `String` object "2, 3, [4], 5, [6], 7, [8], [9], [10], 11, [12], 13, [14], [15], [16], 17, [18], 19, [20], [21], [22], 23, [24], 25, ... ".

`public String nonCrossedOutSubseqToString(boolean[])` - which takes a sequence of numbers *σ* (which may contain both crossed out and non-crossed out numbers) and which returns a `String` representation of the sub-sequence of only non-crossed out numbers separated by comma characters, e.g., given the 2nd last sequence above, of crossed out and non-crossed out numbers, this method should return a reference to the `String` object "2, 3, 5, 7, 11, 13, 17, 19, 23, 25, ... ".

`main()` - which instantiates a `Sieve` object that allows a user to enter a positive integer *N* (whose value is at least 2), and which then prints to the standard output the prime numbers from 2 through to the entered number *N* separated by commas.

Example Interaction:

```
Enter int >= 2 : 10
2, 3, 4, 5, 6, 7, 8, 9, 10
2, 3, [4], 5, [6], 7, [8], 9, [10]
2, 3, [4], 5, [6], 7, [8], [9], [10]
2, 3, 5, 7

Enter int >= 2 : 19
2,3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19
2, 3, [4], 5, [6], 7, [8], 9, [10], 11, [12], 13, [14], 15, [16], 17, [18], 19
2, 3, [4], 5, [6], 7, [8], [9], [10], 11, [12], 13, [14], [15], [16], 17, [18], 19
2, 3, 5, 7, 11, 13, 17, 19
```

Note:

- You must upload your solution for automatic grading via Submitty using the link: https://submit.scss.tcd.ie/courses/2324ht/csu11012

- You may submit only one file Sieve.java which must contain `main()`. A skeleton of the class is provided for you on Blackboard.

- You should make use of the `Scanner` class for input from the user