Introduction
This exercise is meant to introduce you to simple procedural programming in Java, as well as the NetBeans IDE. You will be implementing a simple **prime number sieve** to find all prime numbers between 2 and 500. Your application class should be called `Sieve.java`.

Background
The **sieve of Eratosthenes** is a very efficient algorithm for quickly finding all numbers from 2 up to some upper limit. The basic idea is to start with a list of all numbers between 2 and 500, and then successively divide each by all primes less than it (2, 3, 5, etc.). If it is divisible by any of those numbers, it is crossed off of the list. If we have tested a number with all primes less than it, and it still hasn’t been crossed off yet, than that number is also a prime, and is then used to test the numbers larger than it. A more complete explanation (along with a neat animation) may be found at [http://en.wikipedia.org/wiki/Sieve_of_Eratosthenes](http://en.wikipedia.org/wiki/Sieve_of_Eratosthenes).

Algorithm
The algorithm you are to implement can be defined in pseudocode as follows:

- Create a **Boolean array** with indices up to an including the upper limit of 500. All elements should initially be **true** (which means they are initially all considered to be prime).

- For all integers `p` from 2 to the square root of the limit (23 in this case):
  - If the `p^{th}` element of the array is true (which means that `p` is not divisible by any number less than `p`, and is therefore prime):
    - For all integers `q` from `p + 1` to the limit:
      - If `q` is divisible by `p`, then it is not prime, and the array element corresponding to `q` is set to **false**.

- Print the index of all elements of the array that are still **true**.
**Program Structure**
You are to implement this algorithm as a simple java program, using static methods in an application class called `Sieve`.

In order to get practice with the method syntax of Java, you are to implement it at least two other functions besides `main`. Specifically, it should:

- The `main` method should create the Boolean array and set all elements to `true`. It should also run the main loop for all integers `p` from 2 to the square root of the limit (23 in this case), calling the below function to eliminate all numbers divisible by `p`.

- A function that takes the integer `p` and the array as parameters, and then loops for all integers `q` from `p + 1` to the limit setting the corresponding array element to `false` if `q` is divisible by `p`.

- A function that takes the Boolean array as its parameter and prints the index of all elements that still have a value of `true`.

**Useful Hints and Information**

- Remember that all of your methods in this program must be declared as `static` (`main` is static, and static methods can only call other static methods).
- Don’t forget that you must `construct` the array using `new` before you use it.
- A simple way to determine whether one number is divisible by another is to use the `%` operator (that is, `q` is divisible by `p` if `q % p == 0`).
- As in C, arrays in Java are passed by `reference`. That is, if an array (such as your Boolean array) is passed to a method as a parameter, then changing it in the method (by setting elements to false) changes the array in the caller as well.

**What to Turn In**
Email me the `source code` for your `Sieve.java` class at `jrsullins@ysu.edu`.

You can find the source code for your `Sieve.java` class on your computer under the `Documents/NetBeansProjects/Sieve/src` directory.

The subject line of your email should be `CSIS 3701 Exercise 1 from yourname`.

As with any program, your source code should be `well documented`!