CS 212 - Lab 0

Sieve of Eratosthenes

One of the oldest and fastest methods for finding prime numbers is the Sieve of Eratosthenes. The following illustration uses this method to determine all prime numbers between 2 and 25 inclusive:

Initialize a list with the integers between 2 and 25 inclusive:

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

Start with p = 2 and cross out all numbers greater than p that are multiples of p , that is, cross out 4, 6, 8, 10 and so on.

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

Being multiples of 2, these numbers are not prime.

Now, find the next unmarked number p (p = 3), and again cross out all multiples of p that are greater than p (6, 9, 12, 15, . . .)

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

Once again, find the next unmarked number p (p = 5), and cross out all multiples (10, 15, 20, 25).

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

Continue the process. Stop when p exceeds the square root of 25. The numbers that remain unmarked (2, 3, 5, 7, 11, 13, 17, 19, and 23) are the prime numbers less than 25.

Step 1: Create a Java Project with your name. For example, if your name is Pedro Gomez, name the project csci212.GomezP.

Step 2: Add a new Java class called **SieveOfEratosthenes** to the project, and change the name of the package to **lab0**.

Add the necessary Java source code to the class, **SieveOfEratosthenes**, to create a program that displays prime numbers using the following algorithm:

Permit the user to enter a value for n, and then use the following algorithm to display the prime numbers in the range 2 to n.

Start with a table of numbers (e.g., 2, 3, 4, 5, . . ., n) and progressively cross off numbers in the table until the only numbers left are primes.

Specifically, we begin with the first number, p, in the table, and

1. Declare p to be prime, then display it
2. Cross off all the multiples of that number in the table, starting from p^2
3. Find the next number in the table after p that is not yet crossed off, and set p to that number
4. Repeat steps 1 to 3 until you check n.

The starting point of p^2 is a pleasing but minor optimization, which can be made because lower multiples will have already been crossed off when we found the primes prior to p. For a fixed size table of size n, once we have reached the p^2 entry in the table, we need perform no more crossings off— we can simply read the remaining table entries and know them all to be prime.

Therefore, we can adjust the above algorithm as follows:

Specifically, we begin with 2, the first number in the table, and

1. **If this number, p, has not been crossed off, it is prime, display it.**
2. **Cross off all the multiples of p in the table, starting from p^2 . As stated in the last sentence of the previous paragraph, this is only necessary if $p \leq \sqrt{n}$.**
3. **Repeat steps 1 to 2 until p > n, where p is the next number in the table.**

Example output:

```
Enter a number: 100

Prime Numbers in the range 2 to 100:
2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97

End of Program
```

Document the program as follows, and use this as a template for all labs and projects:

```
package lab0;
/**
* <b>Title:</b> Lab 0:<br>
* <b>Filename:</b> SieveOfEratosthenes.java<br>
* <b>Date Written:</b> January 31, 2018<br>
* <b>Due Date:</b> February 5, 2018<br>
* <p>
* <b>Description:</b><br>
* Displays prime numbers using The Sieve of Eratosthenes.
* </p>
* <p>
* The user is permitted to enter a value for n, and then all prime numbers
* in the range 2 to n are displayed.
* </p>
* <p><b>Algorithm:</b></p>
* <p>
* We start with a table of numbers (e.g., 2, 3, 4, 5, . . ., n) and progressively
* cross off numbers in the table until the only numbers left are primes. </p>
* <p>Specifically, we begin with the first number, p, in the table, and<br>
* 1. Declare p to be prime, then display it<br>
* 2. Cross off all the multiples of that number in the table, starting from p^2<br>
* 3. Find the next number in the table after p that is not yet crossed off and
* set p to that number; and then repeat steps 1 to 3.
*</p>
*@author Your name and your partner's name
*/
public class SieveOfEratosthenes {
		public static void main(String[] args) {
				// declare and initialize variables
				// permit the user to input a value for n
				// create a loop for p = [2, n]
		}
}
```

Submit the Lab:

Even though you will be working with a partner, each student must submit the lab for a grade.

Export the Eclipse project and name it csci212.YourLastName.lab0, then attach and send it via email to your lab instructor.

Right click on the name of the project in the Package Explorer and select **Export...** from the menu. In the Export dialog box, expand the **General** option and select **Archive File**. Then click the Next button.

Make sure .classpath and .project are checked. Without them, I will not be able to open and reproduce your project.

Click the **Browse...** button and choose a location and name for the archive file, csci212.YourName.lab0.zip.

Click the **Finish** button to save the zip file.