Explicación de Listas

Ejercicio: Criba de Eratóstenes

Criba de Eratóstenes

El algoritmo de la Criba de Eratóstenes permite la obtención de todos los números primos menores que un número dado.

***Número primo**: número natural mayor que 1 que tiene únicamente dos divisores: él mismo y el 1*

Criba de Eratóstenes

¿Cómo funciona el algoritmo?

1. Hacer una lista con todos los números naturales desde 2 hasta un número n dado
2. Marcar 2 como primer primo y tachar todos sus múltiplos
3. Marcar como primo el siguiente número no tachado y luego tachar todos sus múltiplos
4. Repetir el paso anterior hasta que el primo marcado sea mayor que la raíz cuadrada de N

Criba de Eratóstenes

Grafiquemos los 4 pasos..

1. Hacer una lista de todos los naturales desde 2 hasta un número ***n*** dado:

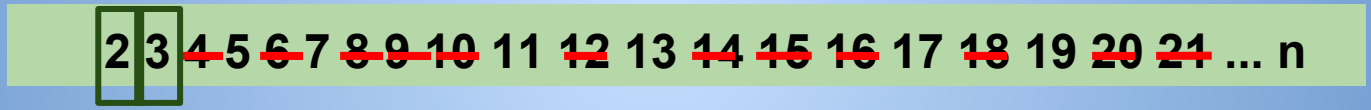
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 … n

2. Marcar el ***2*** como primer primo y tachar de ahí en adelante todos sus múltiplos:

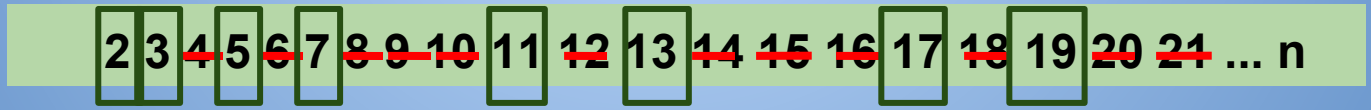
2 3 ~~4~~ 5 ~~6~~ 7 ~~8~~ 9 ~~10~~ 11 ~~12~~ 13 ~~14~~ 15 ~~16~~ 17 ~~18~~ 19 ~~20~~ 21 … n

Criba de Eratóstenes

3. Marcar como primo el siguiente número no tachado y luego tachar todos sus múltiplos. En este caso el 3 es el siguiente primo. Marcamos el ***3*** y tachamos todos sus múltiplos:



4. Repetir el paso anterior hasta que el primo marcado sea mayor que la raíz cuadrada de N



➡ *Los **números marcados son todos los primos** entre 1 y n.*

Criba de Eratóstenes

Ejercicio 4 - Práctica 3

Escriba una clase llamada ***tp03.ejercicio4.CribaDeEratostenes*** con un método llamado ***obtenerPrimos()*** que tome como parámetro un objeto de tipo ***ListaDeEnteros*** que contenga los primeros 1000 números naturales y retorne la lista de los primos correspondientes siguiendo el procedimiento antes descrito.

Criba de Eratóstenes

```
package tp03.ejercicio4;


public class CribaDeEratostenes {

    public static void main(String[] args) {

        ListaDeEnteros enteros = new ListaDeEnterosEnlazada();
        for (int i = 1; i <= 1000; i++) {
            enteros.agregarFinal(i);
        }
        System.out.print(obtenerPrimos(enteros));
    }

    public static ListaDeEnteros obtenerPrimos(ListaDeEnteros l) {
    ...


    }
}
```

Criba de Eratóstenes

```
public static ListaDeEnteros obtenerPrimos(ListaDeEnteros l) {
    ListaDeEnteros res = new ListaDeEnterosEnlazada();
    ListaDeEnteros noPrimos = new ListaDeEnterosEnlazada();
    int e;
    l.comenzar();
    e=l.proximo(); //empiezo a procesar desde el 2
    while (!l.fin()) {
        e=l.proximo();
        if (!noPrimos.incluye(e)) {
            res.agregarFinal(e);
            if(e<=Math.sqrt(l.tamanio())) //ver (*)
                for (int i = e * 2; i < l.tamanio() + 1; i += e)
                    noPrimos.agregarFinal(i);
        }
    }
    return res;
}
```

(*) el agregado de esta sentencia if es porque resulta suficiente con procesar hasta la raíz de n