**Laboratorio unidad 1**

Manuel Castaño – A00358994, Andrés Mayor – A00359333

Josué – A00359703 , Jhonatan Arboleda – A00358993

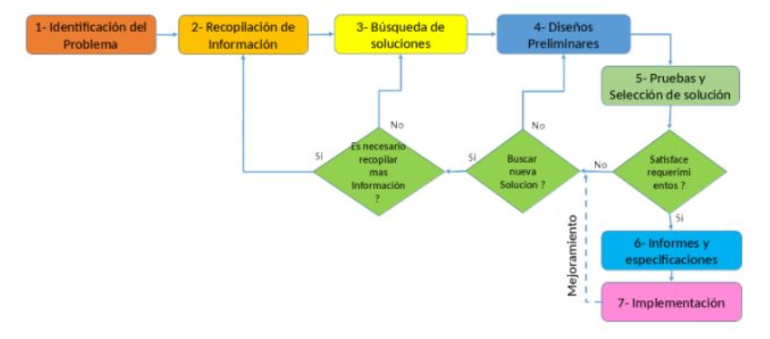
**Contexto del problema.**

En la actualidad, la rama de la informática llamada Criptografía desarrolla un papel muy importante para los softwares desarrollados a satisfacer la seguridad de los datos. Esta rama es apoyada en la grandiosa aliada de los ingenieros, la matemática, que se usa para definir algunos algoritmos de encriptación con los números primos.

Una empresa contrató a mi equipo para implementar más seguridad a sus sistemas y realizar sus propios programas de encriptación.

**Desarrollo de la solución.**

Para llevar a cabo la solución de la situación previa, hemos decidido seguir los pasos del siguiente gráfico, el cual es presentado en el libro “Introduction to Engineering” escrito por Paul Wright.



**Paso 1. Identificación del problema**

Definición del problema:

La empresa requiere un algoritmo que genere números primos de diferentes formas para procesos de encriptación.

Se reconocen las necesidades de la situación, definiéndose los términos y las condiciones bajo las cuales se debe resolver.

* El programa debe tener una interfaz de usuario para la comodidad de los clientes. Esta interfaz cumple la función de recibir un número entero que es solicitado para los métodos que generan los números primos.
* Se deben hacer tres métodos diferentes con los cuales generar números primos hasta un cierto número n (los tres métodos cumplen la misma función de diferentes maneras).
* Generación de una matriz de números enteros que vaya desde 1 hasta n. Esta matriz debe ser lo más cuadrada posible (es decir, si n = 100, la matriz debe ser 10 x 10, o si es n = 20, la matriz debe ser 5x4 o 4x5).
* En la matriz, los números generados que sean categorizados como números primos deben ir pintados de color verde, los demás de color rojo.
* Se debe poder ver en tiempo real lo que tarda el algoritmo en terminar su ejecución.

**Paso 2. Recopilación de información.**

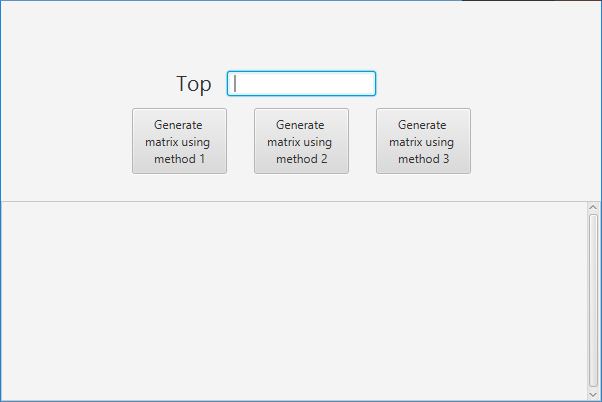
Para tener claridad sobre la situación planteada debemos saber concretamente algunos términos (más que todo matemáticos), ya que así podremos tener una buena base al diseñar una posible solución.

Conceptos:

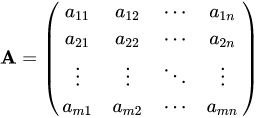
* Número primo: Un número primo es un numero natural mayor que 1 que solo tiene dos divisores, el mismo número y el 1. El número 1 tiene propiedades de ser un número primo y un número compuesto, por lo tanto, no se le considera con ninguna de esas características.

Según el teorema de Euclides, los números primos son infinitos, y algunos de ellos son 2, 3, 5, 7, 11, …

* Interfaz gráfica: La interfaz gráfica es un tipo de interfaz de usuario que utiliza imágenes, íconos y menús para facilitar la interacción del cliente o usuario con un programa, proporcionando también un ambiente más amigable.



* Matriz: Una matriz es un arreglo bidimensional de números para la matemática. Para la programación, una matriz puede contener cualquier tipo de dato. En nuestro caso utilizaremos una matriz compuesta de números enteros.



**Paso 3. Búsqueda de soluciones.**

En internet podemos encontrar diferentes métodos para generar matrices cuadradas o generar números primos, que es la necesidad principal. En este paso, podemos entrar a evaluar nuestras propias soluciones o también las ya halladas en internet.

* Métodos para generar números primos.

1. Opción 1: Este método consiste en recorrer todos los números naturales que hay desde el número 2 hasta el número que se está evaluando y entra como parámetro.

Para lograr esto, dentro de un ciclo (ya sea for o while) se crea un contador inicializado en el número 2 y que va hasta el número que se evalúa (sin incluirlo). Dentro de este ciclo se comprueba el módulo del número a evaluar con el contador, si este es igual a cero entonces se cambia una variable semáforo (que en este caso indica si es primo) a valor “false” y el bucle terminaría, indicándonos si el número evaluado es primo o no.

Recurso: <http://lineadecodigo.com/java/numeros-primos-en-java/>

1. Opción 3: Para este caso se recorre desde el número 2 hasta el número límite.

En un bucle se crea un contador con valor 2. La condición del ciclo es: contador\*contador < = límite, lo que quiere decir que si el resultado al multiplicarse el contador por el mismo es mayor al número que se evalúa, entonces el ciclo no entra y se define al número evaluado como primo.

Recurso: <http://ayudasprogramacionweb.blogspot.com/2012/11/los-numeros-primos-en-java.html> (Se hicieron algunos cambios importantes a el método que aparece en la página).

1. Opción 2: En este método se recorre desde el número 1 hasta el número que se evalúa.

En un ciclo se crea una variable inicializada en 1 que va hasta el número que se está evaluando, dentro del ciclo se evalúa que, si el módulo del contador con el número evaluando es igual a 0. Cuando es igual a 0, una variable x inicializada en cero se le suma 1.

Fuera del ciclo hay dos condicionales que analizan el valor de la variable x, si el valor es menor o igual a 2 entonces se define que el número que se evalúa es primo. Si es mayor que dos, entonces se define como no primo.

Recurso: <https://www.lawebdelprogramador.com/codigo/Java/3567-Determinar-si-un-numero-es-primo.html>

* Método para generar una matriz:
* Para generar una matriz lo más cuadrada posible dependiendo del número límite, primero debemos sacar la raíz cuadrada del número límite para así evaluarla y saber si este resultado en un número entero o decimal. Si es un número entero, entonces la matriz se crea con el resultado de la raíz cuadrada como valor de sus filas y columnas. Si el resultado es un número decimal, entonces la matriz se crea con el resultado de la raíz cuadrada sumándole el número 1 como valor de las filas y columnas.

**Paso 4. Diseños preliminares.**

En este paso vamos a analizar las opciones para ver si se pueden corregir o se deben descartar.

1. Opción 1: Para esta solución el único cambio que haremos para que sea un poco más eficiente será convertir este método iterativo en un método recursivo.
2. Opción 2: Este método, será implementado con su misma estructura.
3. Opción 3: En este algoritmo encontramos un error que nos indica que el número 1 también es un número primo, esto ocurre por una indicación que se debe corregir en la condición que define si es un número primo. De resto, identifica correctamente que número son primos y cuales no. A este algoritmo también se le hará el cambio de método iterativo a método recursivo.
4. El método que genera la matriz es un algoritmo completamente correcto que se implementará sin ningún cambio.

**Paso 5. Evaluación y selección.**

En este paso evaluaremos algunos criterios de las opciones para la solución que han sido propuestas. El peso de cada criterio será definido con un número el cual irá al lado de cada opción.

Criterio 1: Análisis de complejidad temporal.

1: O(nLog(n)), 1.5: O(Log(n)), 2: O(n)

Criterio 2: Para todas las entradas de números el método proporciona la respuesta correcta.

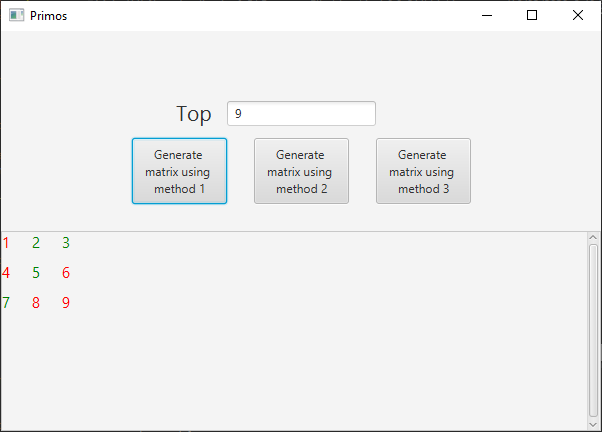
0: No, 1: Si

**Esta evaluación se toma con los algoritmos ya recursivos, pero sin las correcciones hechas.**

|  |  |  |  |
| --- | --- | --- | --- |
| **Evaluación de los algoritmos que generan números primos** | | | |
|  | Criterio 1 | Criterio 3 | Total |
| Opción 1 | 2 | 1 | 3 |
| Opción 2 | 1.5 | 0 | 1.5 |
| Opción 3 | 2 | 1 | 3 |

**Conclusión.**

La situación planteada por la empresa nos pide que el programa tenga 3 métodos diferentes con los cuales generar los números primos, por lo tanto, las tres opciones son elegidas soluciones eficientes siempre y cuando se hayan hecho las correcciones que fueran necesarias. En el caso de la matriz, el método seleccionado es totalmente eficiente y cumple con todos los requisitos que plantea el problema, incluyendo a los números coloreados dependiendo de la propiedad que tenga. La interfaz gráfica es diseñada con todos los requisitos solicitados y adelante se muestra una captura de pantalla de esta.



**Paso 6. Preparación de reportes y especificaciones.**

* [Requerimientos funcionales](FuntionalRequirements.pdf)
* [Análisis de complejidad temporal](Temporal.pdf)
* [Análisis de complejidad espacial](Análisis%20de%20complejidad%20espacial.pdf)
* [Diagrama de clases](DiagramLab1Estructuras.pdf)

**Paso 7. Implementación del diseño.**

En el siguiente repositorio se encuentra la implementación de la solución propuesta anteriormente:

<https://github.com/manuelcastano/Primos>