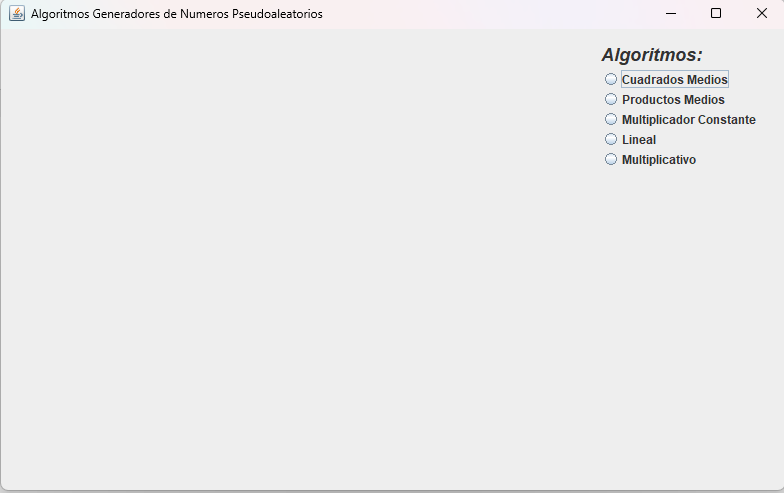
Hola, este documento está destinado a la explicación del proyecto de la generación de números pseudoaleatorios junto con las pruebas estadísticas. Espero quede comprendido, cualquier duda háganlo saber.

Ahora… para empezar explicaré desde la interfaz hasta las entrañas de todo el funcionamiento. Primeramente, tenemos nuestra interfaz gráfica para el usuario o GUI (Graphics User Interface):



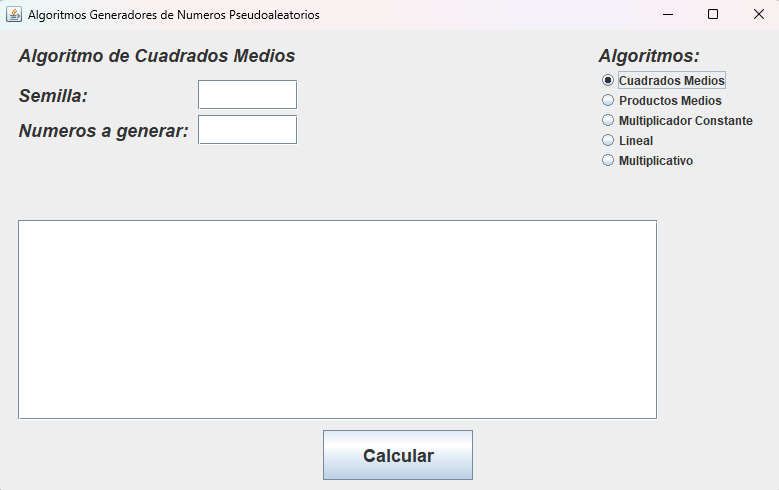
Al inicio nos aparece una interfaz vacia con los 5 primeros algoritmos (el aditivo no lo programe porque la neta está feo)

Como se puede ver, dependiendo cual se seleccione se desplegarán los componentes para ingresar ciertos valores.

Empecemos con cuadrados medios.

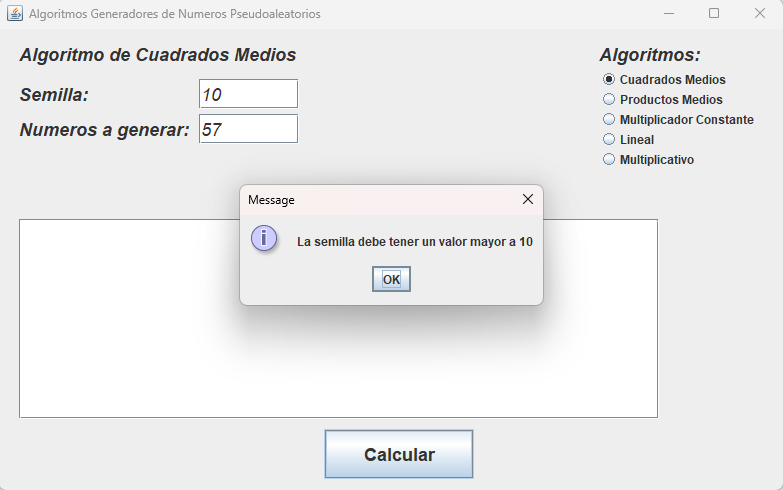
Cuando hablamos del método de cuadrados medios, tiene que ver con todos los métodos no congruenciales, ya que, estos se derivan del mismo procedimiento en realidad, sólo varían algunas cosas entre ellos.

Así que no sólo empezaré con el algoritmo de cuadrados medios, explicaré la matemática de los algoritmos no congruenciales, pero comenzando con el de cuadrados medios (parece lo mismo, pero existe diferencia).

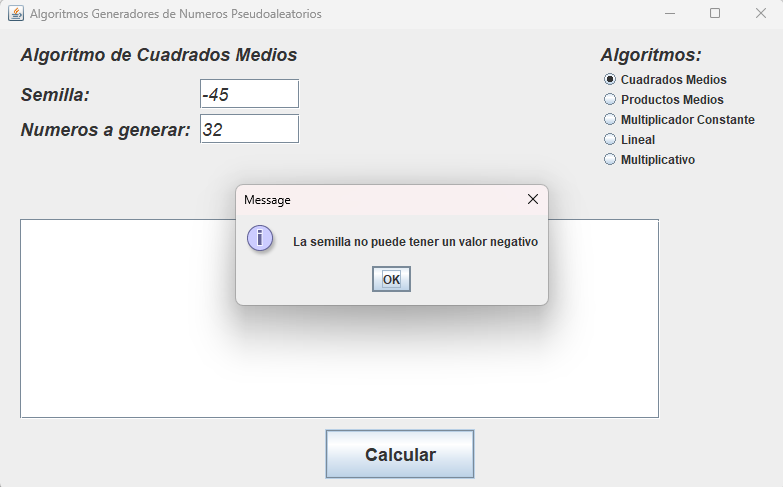


Cuando seleccionamos el algoritmo de Cuadrados medios nos despliega los componentes gráficos mostrados, pero lo importante por ahora es la semilla y los números a generar.

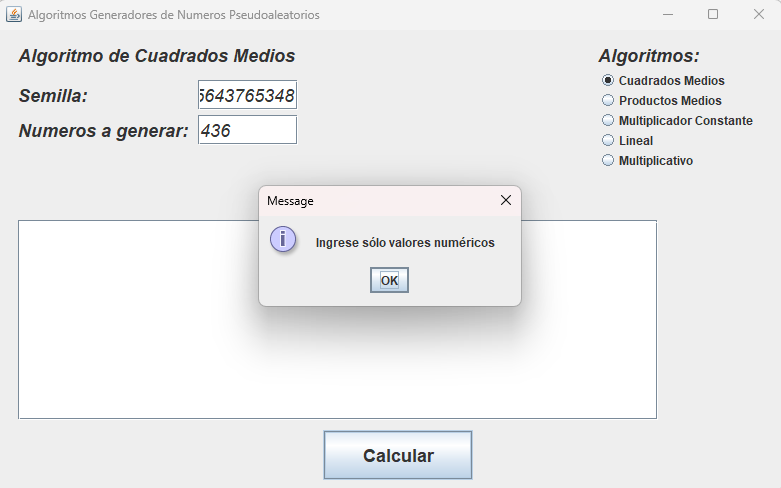
Ingresaremos la semilla que queramos, el programa no permite números igual o menores a 10, si hacemos esto, nos saldrá un mensaje

(haganle zoom)

De igual forma si ingresamos valores negativos, nos saldrá otro mensaje



Y si ingresamos valores por encima del máximo número entero permitido, este no será detectado como numero y saldrá un mensaje diferente (como si ingresaras cadena de texto y no dígitos numéricos)



A su vez, si ingresamos letras o cualquier otro carácter diferente de valores numéricos, nos saldrá el mismo mensaje

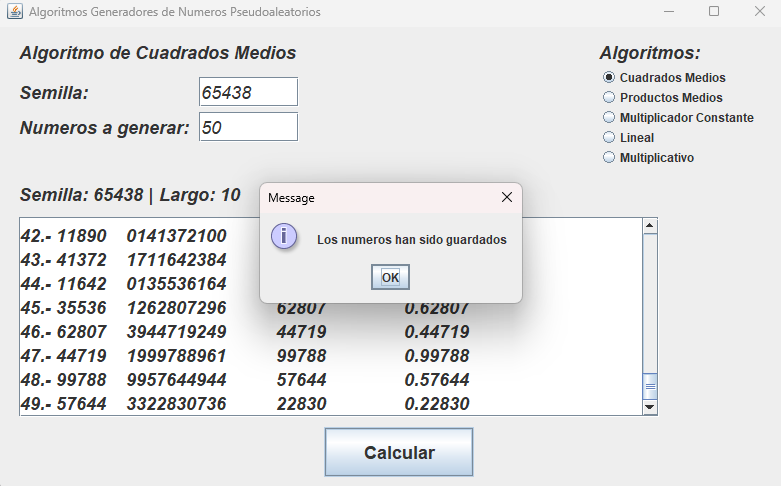


Esto es lo que se llama “programa a prueba de balas”, mas adelante explicaré como es que esto funciona.

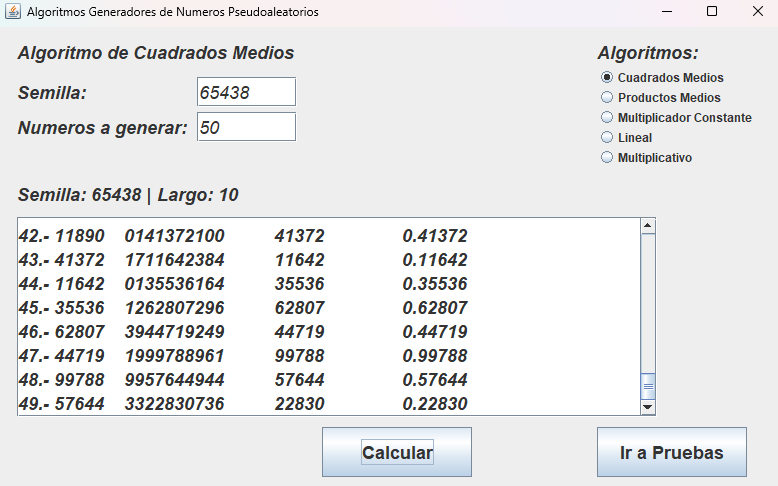
Ahora realicemos una prueba real.

* Primero ingresamos un valor valido para la semilla.
* Posteriormente ingresamos cuantos números pseudoaleatorios queremos generar.
* Por último, presionamos el botón calcular.

Si los números pasaron la primera prueba, la cual, en el caso de los no congruenciales es que ninguno de los números generados se repita, entonces se mostrara de la misma manera que a continuación:



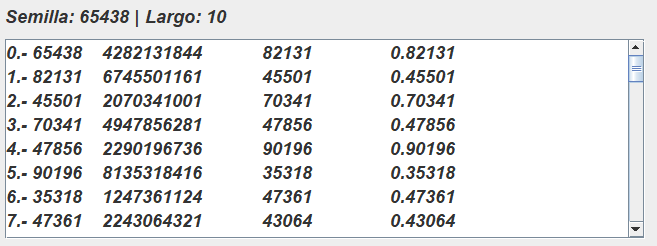
Posteriormente, es necesario presionar el botón que dice “OK” para continuar.



Si todo sale bien, nos aparecerá el botón para realizar las pruebas estadísticas 😊

Pero aún no vamos a eso.

Ahora, como se puede ver, lo que se muestra en el área del campo de texto es:



1. Con un Label mostramos la semilla y el largo del producto obtenido (o sea, la semilla elevada al cuadrado en este caso)
2. Posteriormente, dentro del campo de texto, se despliega el primer valor introducido (la semilla).
3. La semilla elevada al cuadrado.
4. Los 5 dígitos de la parte de en medio (el numero de dígitos a tomar es el mismo que el numero de dígitos de la semilla).
5. Y, por último, nuestro número pseudoaleatorio.

Ahora sí… ¿Cómo se hace esto?

Bueno, como mencioné anteriormente, los algoritmos no congruenciales se basan en la misma dinámica, es decir, en el caso del algoritmo de cuadrados medios es una semilla que se eleva al cuadrado, en este caso visto en la imagen, la semilla es **65438.**

Esta semilla se recibe y se eleva al cuadrado: **65438²**

**65438² = 65438\*65438**

**X² = X\*X**

Hasta aquí todo claro, sin embargo, utilizamos la misma clase tanto para el algoritmo de cuadrados medios, productos medios y multiplicador constante.

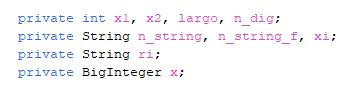
Esto es porque si somos observadores, en todos los casos son productos de 2 numeros, es decir, un número por otro número, es decir, usamos la lógica de **A\*B** en el caso del de cuadrados medios, A y B tendrán el mismo valor, ya que un numero elevado al cuadrado es ese mismo numero multiplicado por sí mismo **X² = X\*X,** pero en un contexto mas general esto es igual a tener **A\*B**, pero **A = X** y **B = X**, entonces nos basamos en eso, y juntamos todo en una misma clase.

Para el de productos medios, son 2 semillas, es decir **A\*B**, ambas pueden variar entre sí o tener el mismo valor y serán totalmente validas si cumplen con lo antes mencionado (que tengan un valor mayor a 10)

Para el de multiplicador constanto es una semilla por un valor constante, es decir, ese nunca va a cambiar en el desarrollo del algoritmo, y esto tambien es igual a **A\*B**

Comprendido todo esto, podemos continuar con la explicación de la clase.

Primero, mostraré los atributos o parámetros de la clase No\_Congruenciales, la cual es la clase destinada para el cálculo de los algoritmos no congruenciales.

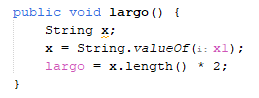


Como podemos observar tenemos x1 y x2, siendo estos los valores para ambas semillas, pero para no confundirnos tanto, digamos que x1 es como si fuera A y x2 como si fuera B. Siguiendo, tenemos el largo, el largo va a ser el valor del producto de x1\*x2 (A\*B), este lo obtenemos calculando el número de dígitos máximo posible obtenido del producto de ambos números con la misma cantidad de caracteres, es decir, por ejemplo, para nuestro algoritmo de cuadrados medios, teniendo nuestra semilla anteriormente propuesta **65438,** el producto de esta sería:

x1 x2

**65438² = 65438\*65438 = 4282131844**

Ahora, este sería el método para obtener el largo:



Como podemos ver, declaramos una variable de tipo String llamada x, luego a x le damos el valor de x1, pero este valor tiene al ser numerico, es necesario ser convertido a tipo String para poder obtener el largo de esta cadena, entonces simplemente utilizamos el metodo String.valueOf para convertir nuestro valor entero a String y la longitud de este valor la multiplicamos por el doble, ya que al estar tratando con dos valores de la misma longitud, el numero con la longitud mas grande que puede llegarse a obtener es del doble de la longitud de la semilla o semillas, por esto es que necesitamos que las semillas sean de la misma longitud.

Vayamos un poco más despacio, básicamente tenemos a

**65438**, ahora imaginemos que cada valor de este numero está dentro de un arreglo (esto para futuro entendimiento), entonces tendríamos

6

8

3

4

5

0 1 2 3 4

Entonces tendríamos que el numero en cuestión tiene una longitud de 5, ya que, ocupa 5 espacios para ser representado o ejemplificado.

¿Pero ahora, por qué multiplicamos por 2?

Buenos, eso es porque el numero mas grande de esta longitud inicial, sería 99999, si multiplicamos 99999\*99999, nos da un resultado de

9,999,800,001, como podemos observar, la longitud máxima que se puede alcanzar es de 10 dígitos, esto quiere decir que la longitud máxima del producto de 2 numeros con la misma longitud, es el doble de esa longitud inicial, en este caso del ejemlo de 65438 es 5\*2 = 10.

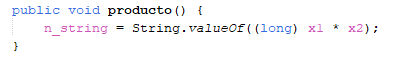
Algun caso en el que sea posible que una multiplicación con 2 numeros de la misma longitud no de el doble de esa? Sí, muy fácil, multipliquemos 10,000\*10,000, nos da como resultado 100,000,000

Un numero que como se puede ver, es de 9 dígitos, en este caso, si esto llegara a ocurrir en algún punto de nuestro programa, simplemente agregamos un 0 a la izquierda para que llegue al largo correspondiente, es decir, quedaría como 0100000000.

Terminada esta explicación, procedemos al siguiente punto…

n\_dig es un atributo el cual sólo se encarga de obtener la longitud de la primera semilla, es decir x1 y lo guarda para posteriormente en los casos de productos medios y multiplicador constante, se comparen ambos valores de semilla con semilla o semilla con constante y se vea que se tiene ambos valores de la misma longitud.

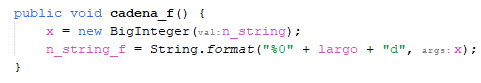
Posteriormente, tenemos n\_string que básicamente es el producto del número guardado en un atributo de tipo cadena para mas facilidad de manipular, ya que lo que haremos es sacar los n dígitos de en medio del producto de ese numero



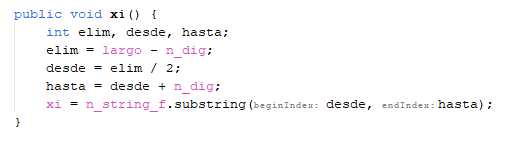
Aquí tenemos que el valor del producto de ambos números se convierte a String y se queda guardado para posteriormente ser mostrado, el porqué de la razón de castearlo a long, es porque existe la posibilidad de que el producto de ambos números sea mayor al maximo numero que permite el formato int, es decir, 2,147,483,647

como por ejemplo, el producto de 46341 al cuadrado, nos daría como resultado 2,147,488,281, el cual es mayor a 2,147,483,647 y esto podría ocasionar una excepción en la ejecución de nuestro programa (es decir, que truene, pues) y aplicando el casteo de (long) esto se evita.

Posteriormente tenemos n\_string\_f es utilizado para agregar ceros a la izquierda en caso de ser necesario, para esto lo que hacemos es que, nuestro parámetro x lo hacemos igual a el numero obtenido en n\_string, para evitar el problema de que el numero sea mayor al valor maximo de int, utilizamos la librería BigInteger, el cual trata al valor ingresado en formato String como si fuera un int, sin importar el tamaño o valor que este numero tenga, posteriormente este pasa a n\_string\_f donde se le da el valor de la variable x ahora en formato String para poder agregar los 0’s a la izquierda. Eso se hace en estas líneas de código:



Una vez obtenido este valor, pasamos a obtener los cinco números de en medio



Cómo?

Pues es muy simple, creamos 3 variables locales, elim, desde y hasta

Elim la hacemos igual a la resta del largo menos el numero de dígitos, en el ejemplo dado anteriormente sería:

elim = 10 – 5 = 5

desde va a ser igual a la división entera de elim/2, es decir, 5/2 = 2

hasta va a ser igual a desde mas el numero de dígitos, en este caso, 5, es decir, 2+5 = 7.

Por que?

Bueno, poniendo un ejemplo similar al anterios, tratamos a el valor de n\_string\_f, es decir, el valor final del producto de lo números junto con los 0’s agregados (de haber sido necesario).

0 1 2 3 4 5 6 7 8 9

2

8

4

4

8

1

3

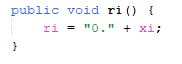
1

4

2

Entonces, en este caso, tomaremos desde el numero en la posición dos, hasta el que esté antes del numero en la posición 7, ¿Por qué antes de la posición 7? Porque el método substring lo que hace es que toma el digito anterior al ingresado en la posición en este caso, sería hasta el que está en la posición 6, entonces el numero sería **82131** el cual cumple con el largo de 5 digitos (al igual que su semilla) y este será el siguiente valor del que se hará el producto de si mismo hasta que se haya hecho el numero de veces ingresado por el usuario 😊

Por ultimo, para obtener nuestro ri simplemente al valor de xi le agregamos el “0.”+82131 y tendríamos nuestro ri.



Y eso sería todo para los no congruenciales (ya me cansé ☹, me voy a mimir)