Universidad Autónoma del Estado de México

Unidad Académica Profesional Tianguistenco

Ingeniería en software

Unidad de aprendizaje:

Programación paralela

Suma de subconjuntos

Profesor:

Gustavo Gómez Vergara

Alumno:

Leandro Gómez Flores

Fecha de entrega: viernes, 16 de diciembre de 2020

Índice

Problemática

Dado un conjunto cuya longitud sea n, sacar todos los subconjuntos posibles cuya suma sea 0, omitiendo así el subconjunto vacío {} y los subconjuntos cuya longitud sea 1: {-1}.

Análisis

Requerimientos funcionales:

- Dar un conjunto aleatorio cuya longitud sea aleatorio dado solo su rango máximo ej. 3 a 15.
- Dar un conjunto aleatorio cuya longitud sea dada por el usuario.
- Adaptable a Programación paralela.
- Se crean ciclos no infinitos.
- Se crean todas las combinaciones posibles de subconjuntos omitiendo así el subconjunto vacío y los subconjuntos cuya longitud sea 1.
- Verificar el tiempo de ejecución cuando se inicie la creación de los subconjuntos y cuando finalice.
- Imprimir en pantalla todos los subconjuntos.
- Imprimir en pantalla todos los subconjuntos cuya suma sea 0.
- Imprimir el numero total de subconjuntos y el numero total de subconjuntos que suman 0.

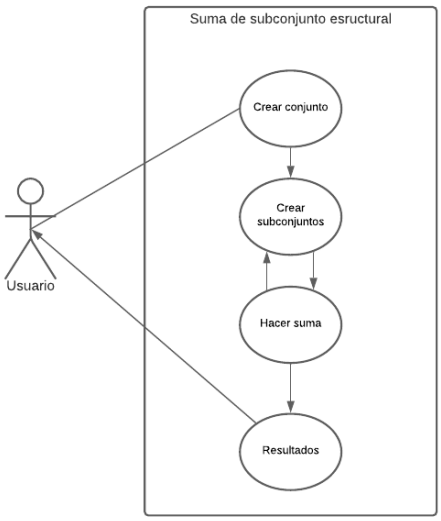
Requerimientos no funcionales:

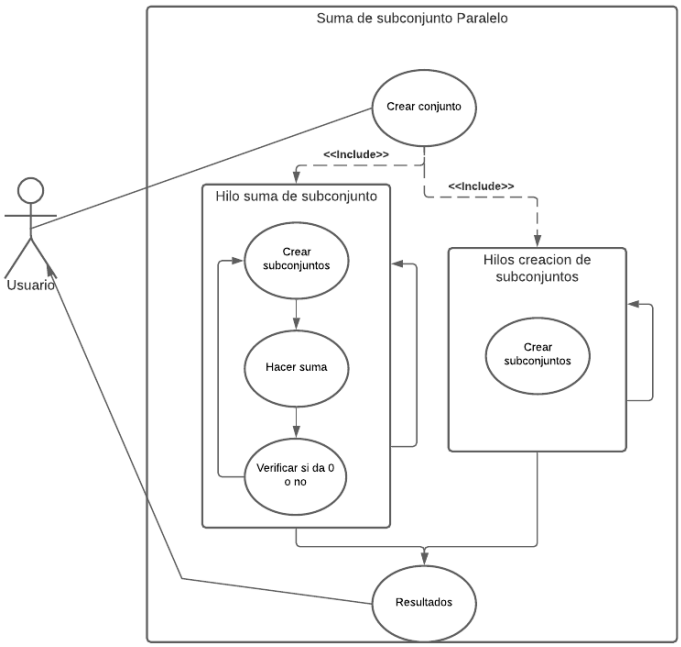
- Imprimir los subconjuntos de longitud mayor a la longitud menor.
- Tener un tiempo de respuesta aceptable.
- Se mostrará en una interfaz.
- Nombre de variables aceptables.
- Color de interfaz.
- Seleccionar entre el programa echo en paralélelo o Estructural.

Tipo de datos:

- if
- if else
- for i, i, i++;
- long
- BigInteger
- int []
- do while
- int
- String
- ArrayList<>

Caso de uso:





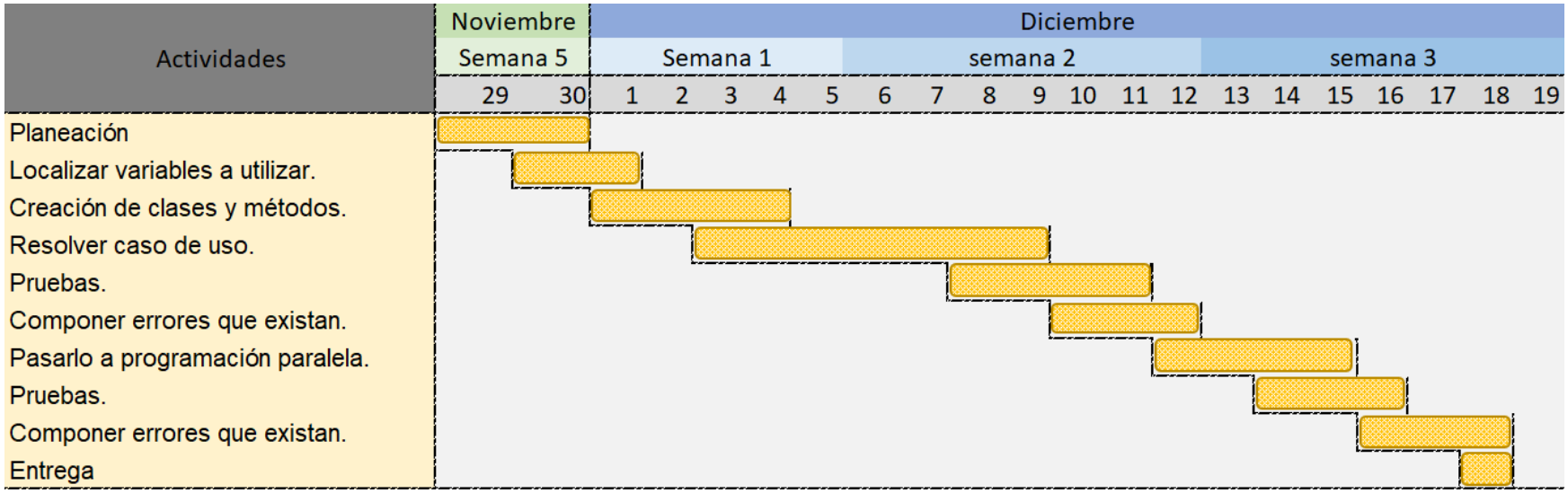
Metodología:

- Incremental.

Tipo de caso a los que me podría enfrentar:

- Tardar mucho en crear subconjuntos.
- Se podrían crear ciclos y no de respuesta.
- Tiempo no óptimo.
- Métodos no optimizados.

Cronograma de actividades:



Diseño

Tipo de programación:

- Programación estructurada.
- Programación paralela.

Lenguaje de programación:

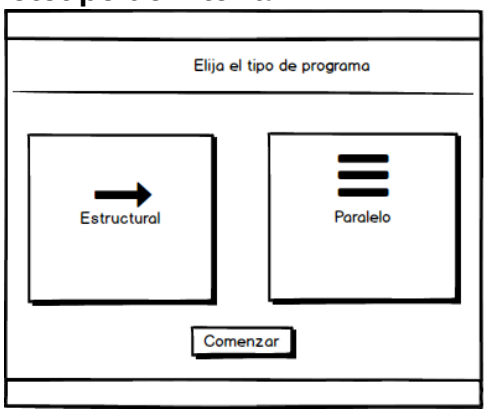
- Java

Estructuras de control:

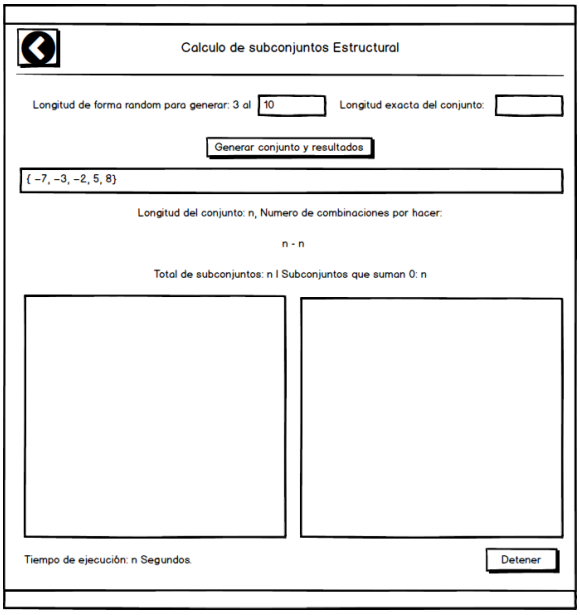
```
if(condición)
{
        Instrucción;
}

if(condición)
{
        Instrucción;
}else
{
        Instrucción;
}
```

```
while(condición)
{
        Instrucción;
}

for(int i = N; i>=n; i++)
{
        Instrucción;
}
```

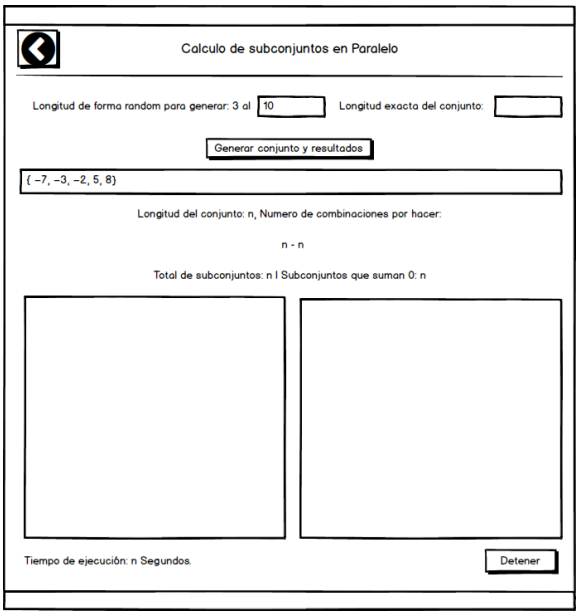
Prototipo de interfaz:



Prototipo 1. Selección de programa

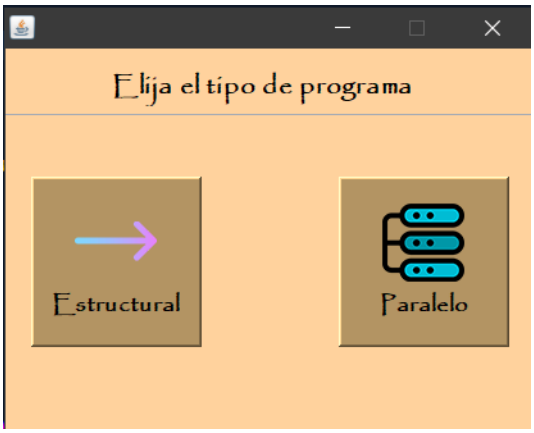


Prototipo 2. Programa Programado de manera estructural

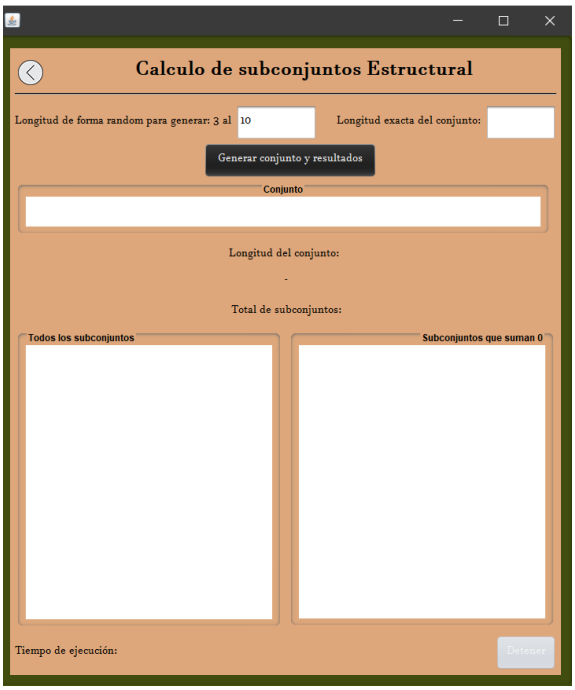


Prototipo 3. Programa realizado paralelo

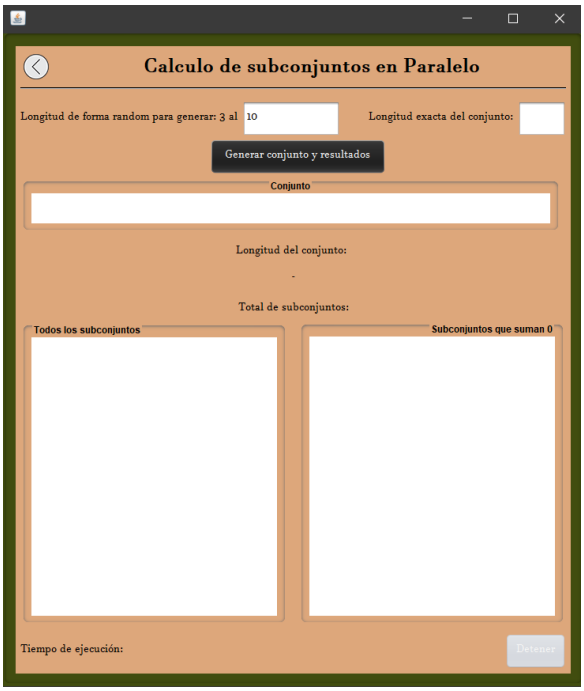
Interfaz:



Interfaz 1. Selección de programa



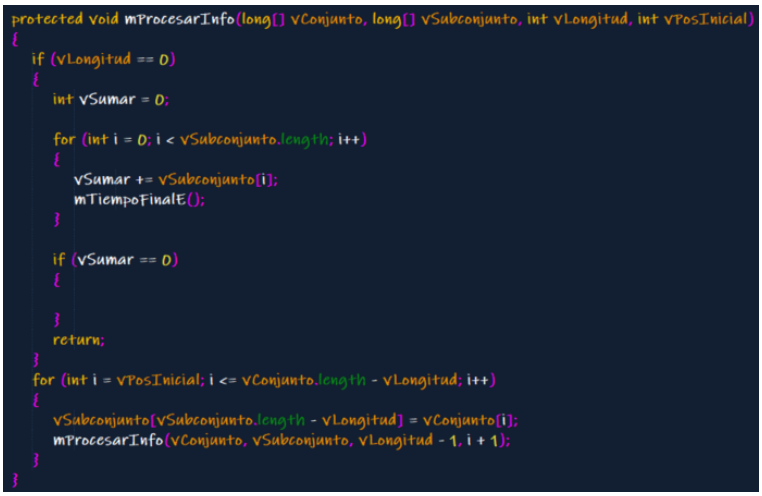
Interfaz 2. Programa Programado de manera estructural



Interfaz 3. Programa realizado paralelo

Implementación

Para llevar a cabo el proyecto primero se debe que planear como se empelara que lenguaje se usara, se me menciono en un apartado se usó la programación estructural entonces teniendo en cuenta esa parte se comienza a planear como se solucionara el problema, se realizaron investigaciones sobre como son las combinaciones, sacar su numero de combinaciones totales que pueda haber. Una vez resulto esa parte se empieza a poner lógica para poder transformar esas palabras a código por lo que se decide hacerlo en un método recursivo para que pueda crear los subconjuntos sin que se repitan, la creación del conjunto se dijo en un requerimiento funcional que debe ser la construcción de este de forma automática.



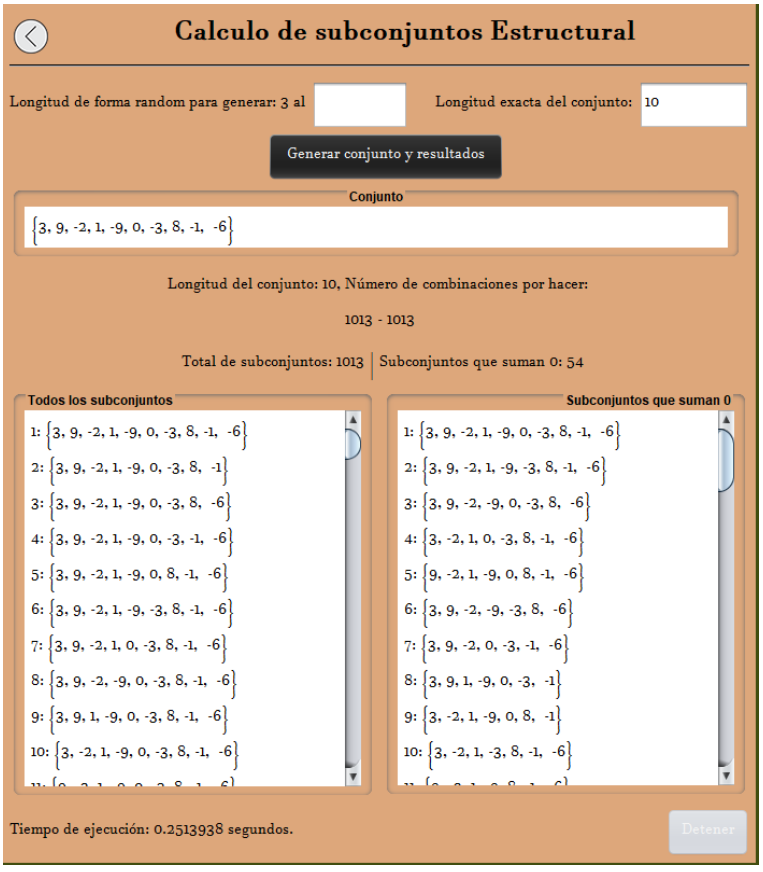
```
protected void mProcesarInfo(long[] vConjunto, long[] vSubconjunto, int vLongitud, int vPosInicial)
{
    if (vLongitud == 0)
    {
        int vSumar = 0;

        for (int i = 0; i < vSubconjunto.length; i++)
        {
            vSumar += vSubconjunto[i];
            mTiempoFinalE();
        }

        if (vSumar == 0)
        {

        }
        return;
    }
    for (int i = vPosInicial; i <= vConjunto.length - vLongitud; i++)
    {
        vSubconjunto[vSubconjunto.length - vLongitud] = vConjunto[i];
        mProcesarInfo(vConjunto, vSubconjunto, vLongitud - 1, i + 1);
    }
}
```

Método para crear los subconjuntos de forma recursiva.

La primera versión se pensó la impresión de los conjuntos era en consola una de las ventajas de usar impresiones en consola es que no te consume mucha memoria con esta ventaja se puede obtener un tiempo de ejecución mucho mas favorable pero no era muy visible para mostrar todos los subconjuntos y todos los subconjuntos que sumaran 0 por lo que se decidió que la impresión de estos subconjuntos fuera en una interfaz gráfica.

```
run:
Puchaste el Estructual
Es
{3, 9, -2, 1, -9, 0, -3, 8, -1,  -6}
{3, 9, -2, 1, -9, 0, -3, 8,  -1}
{3, 9, -2, 1, -9, 0, -3, 8,  -6}
{3, 9, -2, 1, -9, 0, -3, -1,  -6}
{3, 9, -2, 1, -9, 0, 8, -1,  -6}
{3, 9, -2, 1, -9, -3, 8, -1,  -6}
{3, 9, -2, 1, 0, -3, 8, -1,  -6}
{3, 9, -2, -9, 0, -3, 8, -1,  -6}
{3, 9, 1, -9, 0, -3, 8, -1,  -6}
{3, -2, 1, -9, 0, -3, 8, -1,  -6}
{9, -2, 1, -9, 0, -3, 8, -1,  -6}
{3, 9, -2, 1, -9, 0, -3,  8}
{3, 9, -2, 1, -9, 0, -3,  -1}
{3, 9, -2, 1, -9, 0, -3,  -6}
{3, 9, -2, 1, -9, 0, 8,  -1}
```

Impresión de subconjuntos mediante la consola

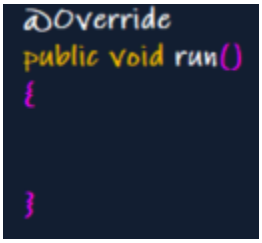


Impresión de subconjuntos y subconjuntos que su suma fuera 0.

Cuando el programa estructural empezó a tener resultados finales se empezaron a tratar los errores obtenidos para tener una mejor calidad. Posteriormente se empezó a trabajar sobre el mismo programa, pero esta vez dirigido a la programación paralela la cual conllevaba subdividir procesos para que el tiempo de ejecución sea el mejor obtenido, La subdivisión de los procesos que hacia el sistema estructural se crearon con hilos, estos pueden tener diferentes formas de ejecutoras en este caso se implemento que fuera un hilo para procesos, esto se hace poniendo una extensión a la clase que se quiere subdividir como se puede mostrar en la siguiente imagen.

Forma de como crear una clase y que pertenezca a un hilo.

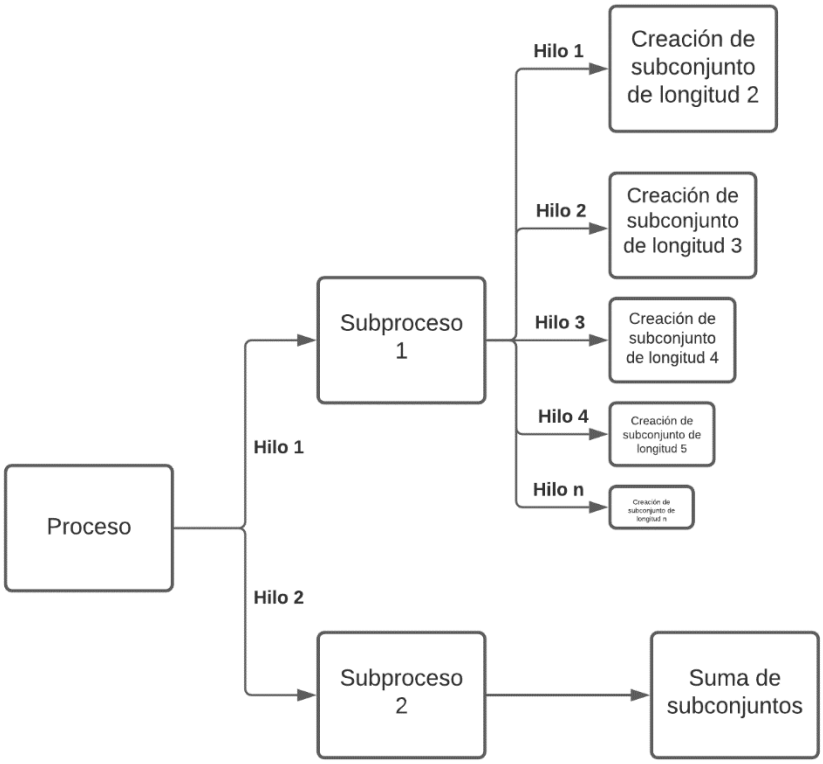
Lo cual al tener este tipo de extensión se necesita un método run para poder hacer los subprocesos.



método para correr los métodos de la clase.

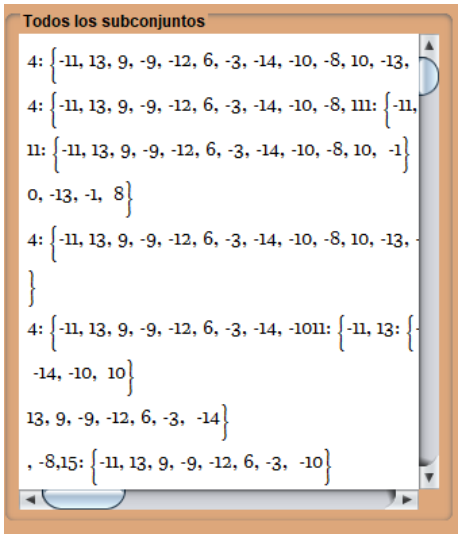
Cuando ya se tiene todo estructurado se empieza a ver como se subdividirá el proceso, se decidió que la creación de los hilos iba a ser referente a la longitud del conjunto, ej. si mi conjunto tiene una longitud de 3 entonces se crearan 3 hilos, el proceso principal se subdivide en 3 partes no iguales ya que cada parte tiene una combinación diferente, me refiero a que cada subconjunto tiene una longitud diferente desde longitud 2 hasta n. También se separaron en la función de suma de subconjuntos para que este sea un mas ligero y solo haga una acción.

Por lo cual se crea la siguiente estructura:



Estructura que se crea para que peda ser paralelo.

La información obtenida se va imprimiendo en pantalla, en ocasiones la información de los subconjuntos no es muy buena ya que esta entra en conflicto con los demás subprocesos y a la hora de mostrarlos estos se pueden encimar.



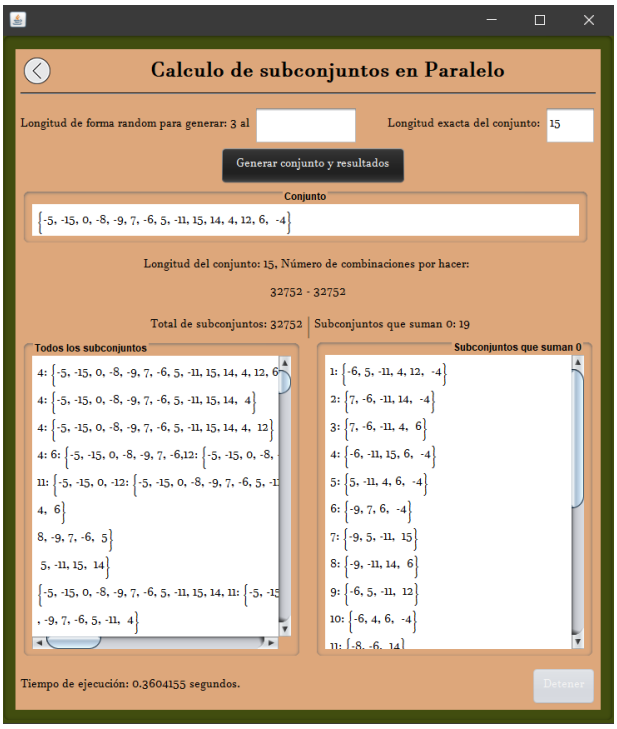
Información mal mostrada por el proceso paralelo.

Pruebas

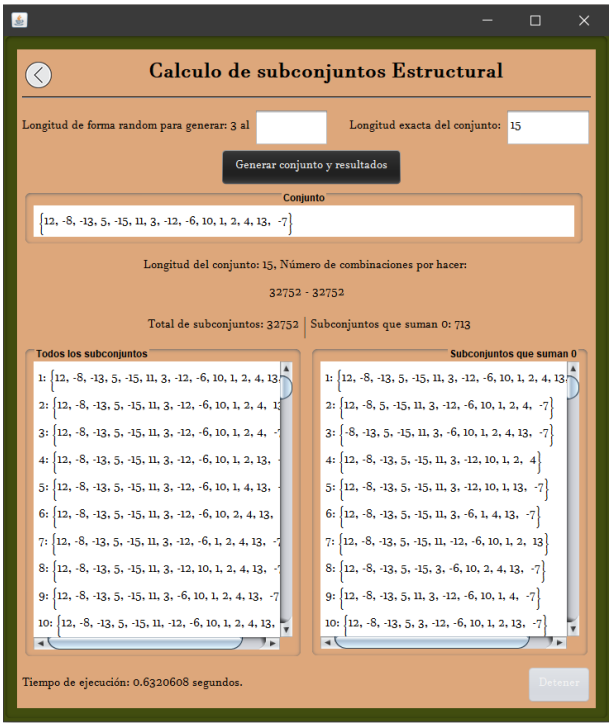
Tabla de pruebas con diferentes longitudes de conjuntos:

	Programa estructural		
Intentos realizados	Longitud del conjunto	Subconjuntos generados	Tiempo de ejecución
1	10	1013	0.1077434
2	10	1013	0.0491865
3	10	1013	0.0332959
1	15	32752	0.9788268
2	15	32752	0.7559559
3	15	32752	0.6320608
1	20	1048555	37.0297775
2	20	1048555	34.5920467
3	20	1048555	30.5967772
1	30	1073741793	12 min y 0.5821183 seg.
	Programa paralelo		
Intentos realizados	Tamaño del conjunto	Subconjuntos generados	Tiempo de ejecución
1	10	1013	0.226232
2	10	1013	0.0199908
3	10	1013	0.0178606
1	15	32752	0.4975943
2	15	32752	0.3604155
3	15	32752	0.4071154
1	20	1048555	22.8454552
2	20	1048555	15.0032942
3	20	1048555	16.6593408
1	30	1073741793	java.lang.OutOfMemoryError: Java heap space.

Imágenes de las pruebas:



Ejecución del programa paralelo con un conjunto de longitud 15



Ejecución del programa Estructural con un conjunto de longitud 15