

Universidad Libre Seccional Cali

Métricas de Software

**“Programa que retorna cuántos pares hay en un arreglo de tamaño n, por ejemplo, a y b, que sumados estos dos dan un número que es divisible por el número preguntado por pantalla”**

Juan Camilo Mamian Ruiz

Aldemar Bernal Hernández

Prof. Rafael Alberto Moreno

Facultad de Ingeniería

Programa Ingeniería de Sistemas

2021-1

**Contenido**

[**Programa en JAVA Suma de Pares** 3](#_Toc65891440)

[**Código Fuente** 4](#_Toc65891441)

[**Tabla de Recolección de Datos** 8](#_Toc65891442)

[**Gráfico Estadístico** 10](#_Toc65891443)

# **Programa en JAVA Suma de Pares**

El siguiente programa realizado en el lenguaje de programación Java tiene como objetivo contar cuántos pares de elementos hay en un arreglo de tamaño n, que sumados estos dos elementos da un resultado que sea divisible por un número pedido en pantalla. Por ejemplo:

Tenemos un arreglo de tamaño n = 6

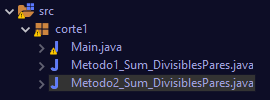
Y pedimos un número en pantalla que será el divisible, en este ejemplo utilizaremos el 5, divisible = 5.

Entonces, el programa tomará dos elementos del arreglo ; los sumará; y el resultado lo verifica con el número pedido por pantalla “divisible”; si el resultado del par de números es divisible con el entregado en pantalla, entonces lo contamos, por ejemplo.

Como observamos, los resultados de los pares dan un número que son divisibles por el preguntado en pantalla, el cual es 5. Por último, el programa nos retornará que la cantidad de estos pares que son 3.

# **Código Fuente**

El código está distribuido de la siguiente forma:

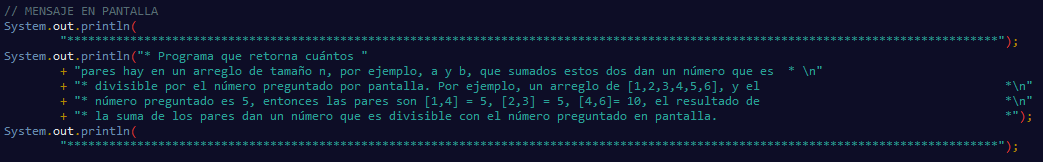


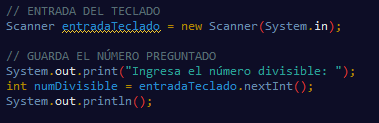
La clase Main es donde iniciamos el programa y las otras dos clases que sirven para hacer el trabajo de la suma de pares.

**Contenido del Método Main con Comentarios Explicativos**

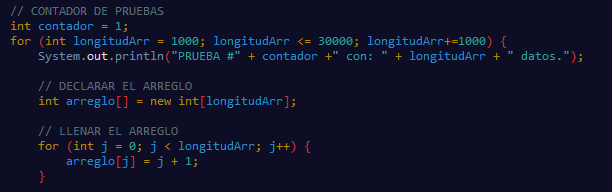
1. Variables para utilizar las métricas



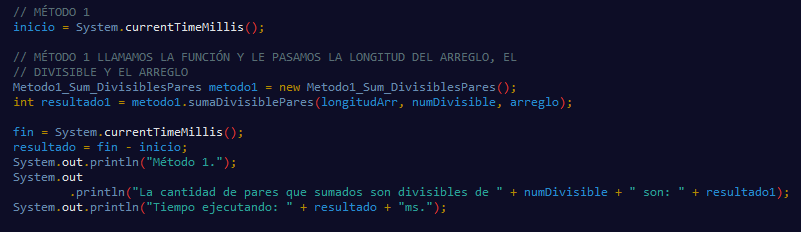
1. Mensaje en pantalla sobre que trata el programa.
2. Entrada de teclado y almacenamiento del dato pedido



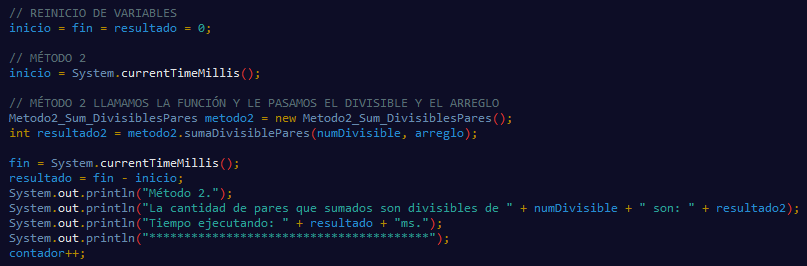
1. Contador para las pruebas que ayuda a verificar en que prueba nos encontramos “Es un plus para la salida de pantalla”. Seguido tenemos un ciclo donde incrementamos el tamaño del arreglo en cada vuelta, su incremento va de 1000 en 1000. Y, por último, declaramos el arreglo y le pasamos el tamaño que tendrá, luego lo llenamos con números.



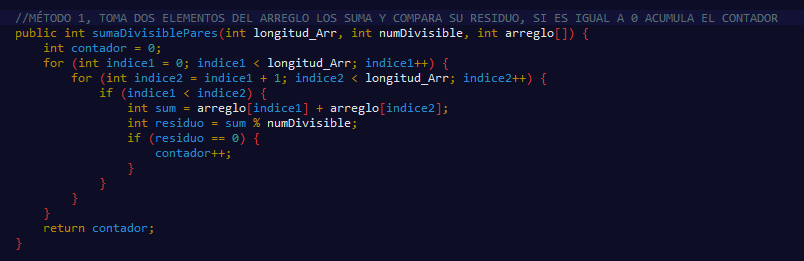
1. Aquí utilizamos las variables para medir el tiempo de ejecución del método 1, además, utilizamos el método llamando Metodo1\_Sum\_DivisiblesPares creamos un objeto de esta clase y traemos la función sumaDivisiblesPares(); le pasamos por parámetro la longitud del arreglo, el número pedido por pantalla y el arreglo.



1. Reiniciamos las variables para medir el tiempo de ejecución de nuevo; utilizamos el método 2 como en el método 1. Pero en este caso solo mandamos el número pedido por pantalla y el arreglo. Por último, incrementamos la variable “contador” que nos ayuda a mirar en que prueba nos encontramos.

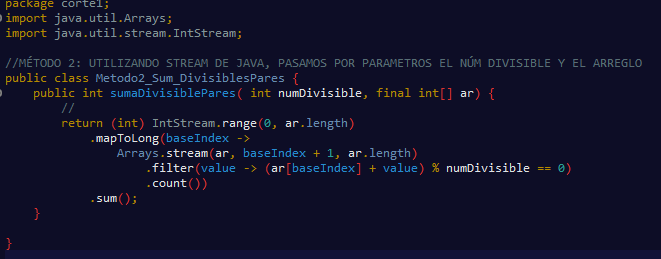


1. El método 1 lo que hace es recorrer dos índices en el arreglo, los suma y con el resultado le hace modulo con el número pedido por pantalla, si el residuo da 0, entonces se incrementa contador.



1. El segundo método se apoya en dos librerías, Array y Stream. Hace lo mismo que el método 1 por lo que vemos que saca residuo dentro del filter, la única diferencia es que se apoya de estas librerías. Este código lo observamos en un foro de retos de programación donde las personas compartían otras alternativas.

https://www.hackerrank.com/challenges/divisible-sum-pairs/forum/comments/154029



# **Tabla de Recolección de Datos**

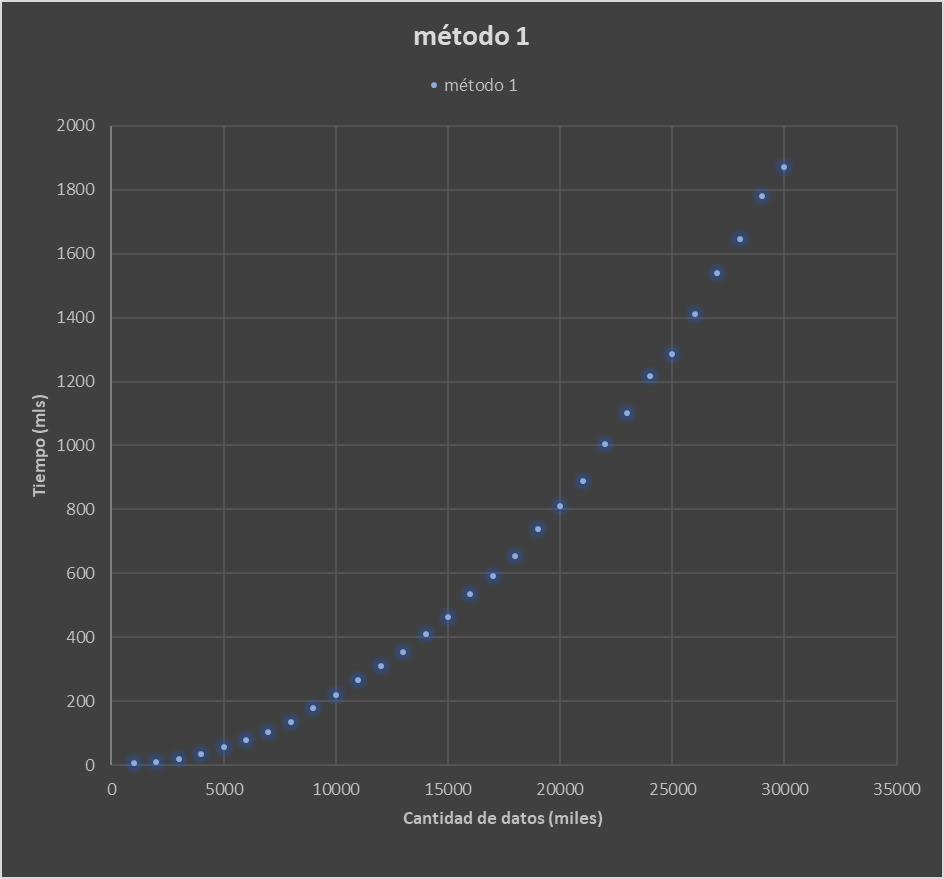
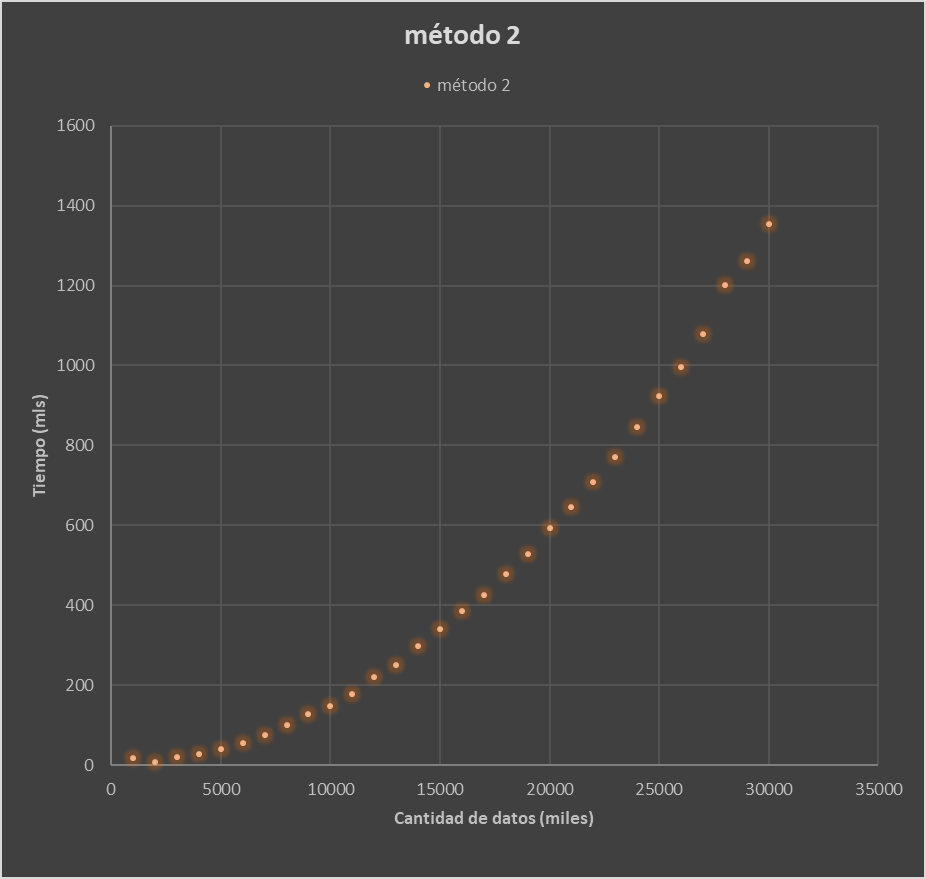
**Tabla Método 1**



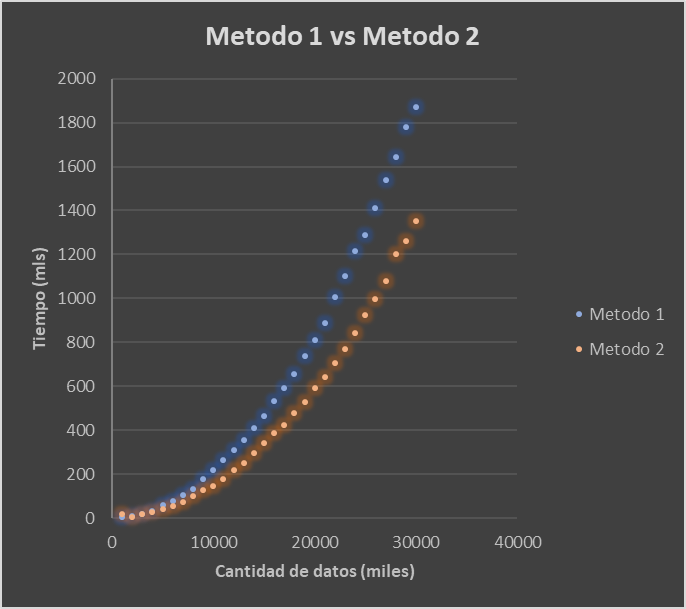
**Tabla Método 2**



# **Gráfico Estadístico**



**Comparación de los dos Métodos**

****

Lo que podemos concluir, es que el método 2 con pocos datos es lento y el método 1 destaca en velocidad. Pero cuando los datos crecen, el método 1 demora más tiempo de ejecución que el método 2.