

Guatemala 03 de febrero de 2019

Gráfica de rendimiento de algoritmos

**Algoritmos y estructura de datos**

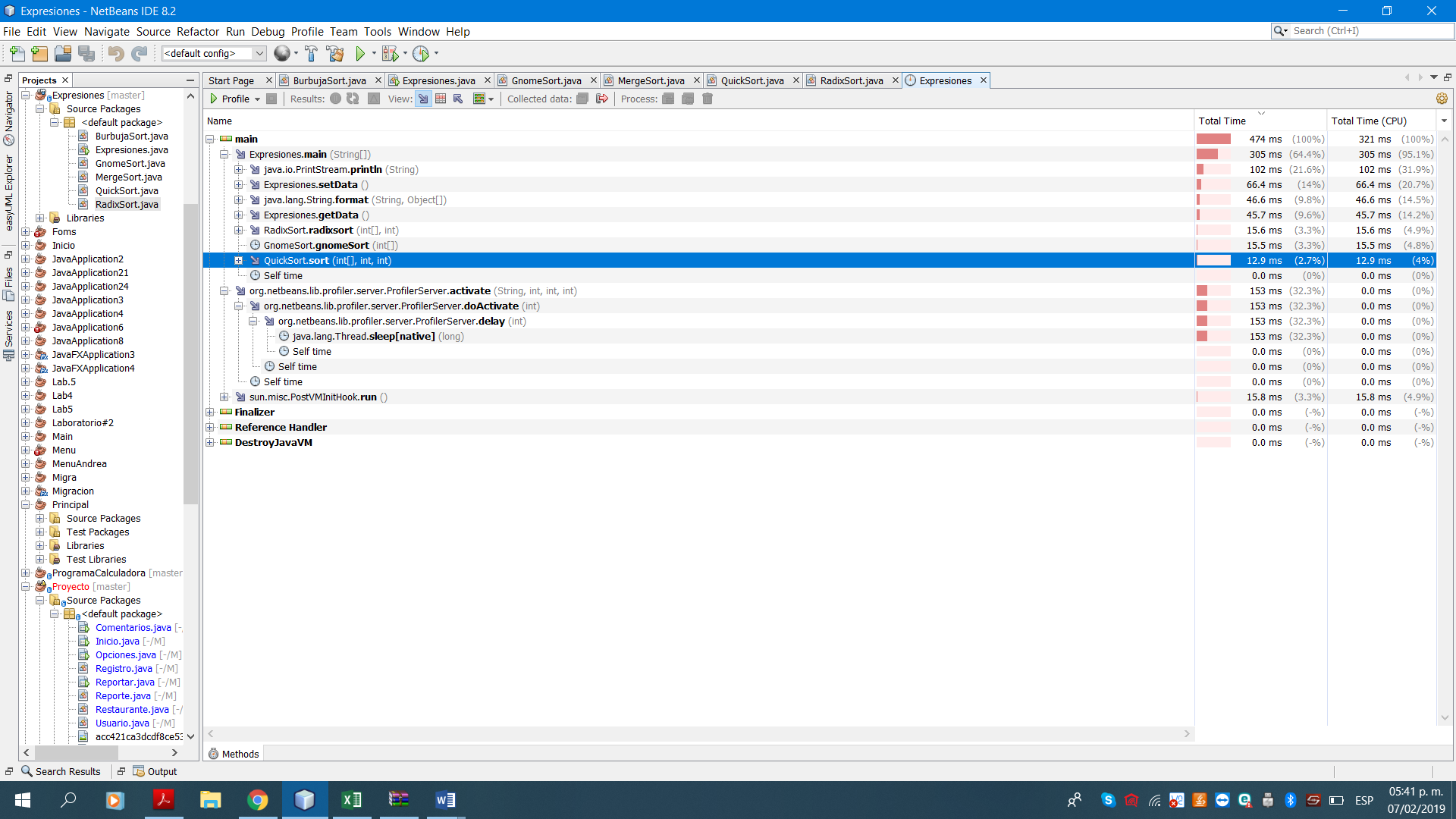
Andrea Elías - 17048

Pablo Sao - 11530

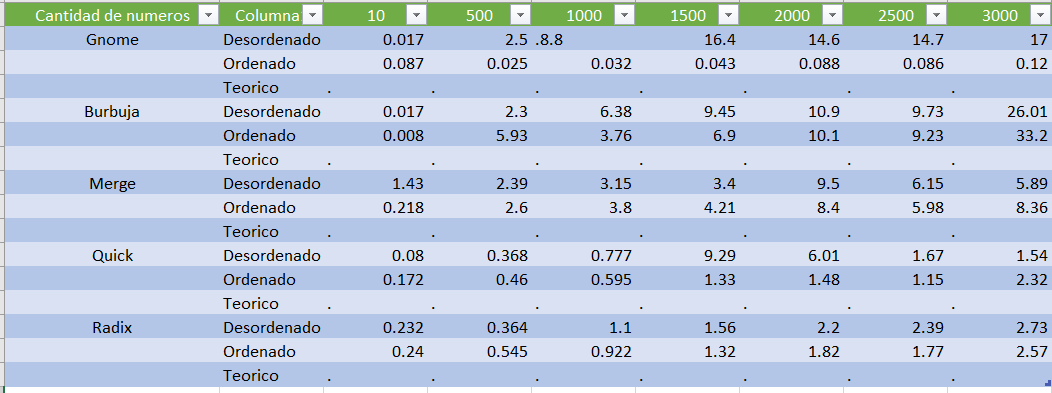
**Profiler Utilizado**

Se utilizó el profiler que ya viene integrado con NetBeans. Este se empleó analizando únicamente la clase main. Entre los resultados nos mostraba cuanto tiempo se había tardado en procesar cada instrucción dentro del main, dichas instrucciones eran llamar a cada sort para que ordenara determinada cantidad de números.

**Ejemplos**



**Resultados**

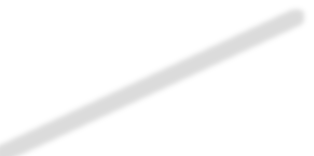


**Gráficos**

|  |  |
| --- | --- |
| Azul | Desordenado |
| Anaranjado | Ordenado |
| Gris | Teórico |

**GnomeSort**

Complejidad O(n)



500 1000 1500 2000 2500 3000 3500

0

35

30

25

20

15

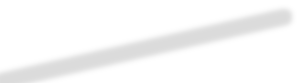
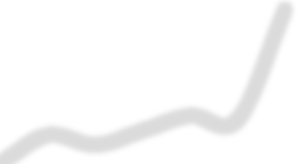
10

5

0

# **BurbujaSort**

Complejidad O(n^2)



500 1000 1500 2000 2500 3000 3500

0

35

30

25

20

15

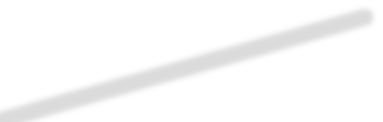
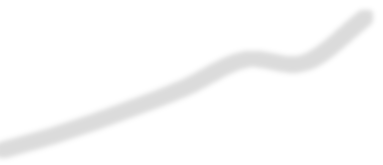
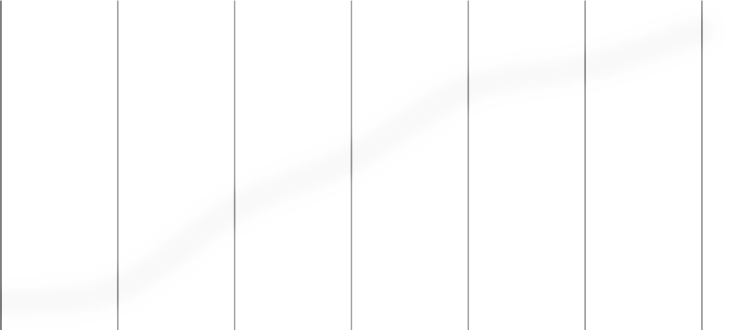
10

5

0

# **RadixSort**

Complejidad O(nk)



3500

3000

2500

2000

1500

1000

500

0

3

2.5

2

1.5

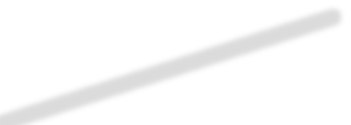
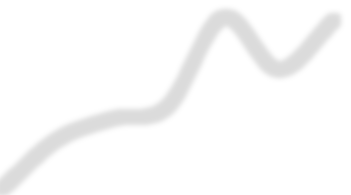
1

0.5

0

# **MergeSort**

Complejidad de O(n log(n))



10

9

8

7

6

5

4

3

2

1

0

0

500

1000

1500

2000

2500

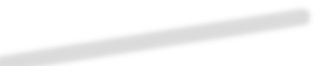
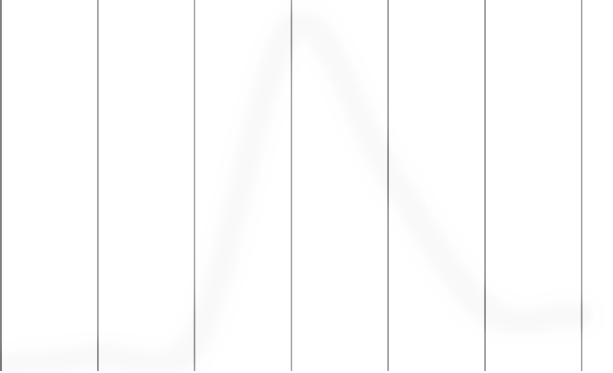
3000

3500

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | |  | |  |
|  |  |  |  |  | |  | |  |
|  |  |  |  |  | |  | |  |
|  |  |  |  |  | |  | |  |
|  |  |  |  |  | |  | |  |
|  |  |  |  |  |  |  |  |  |
|  |  |
|  |  |  |  |  | |  | |  |
|  |  |  |  |  | |  | |  |
|  |  |  |  |  | |  | |  |
|  |  |  |  |  | |  | |  |

# **QuickSort**

Complejidad de O(n log(n))



1000 1500 2000 2500 3000 3500

500

0

10

9

8

7

6

5

4

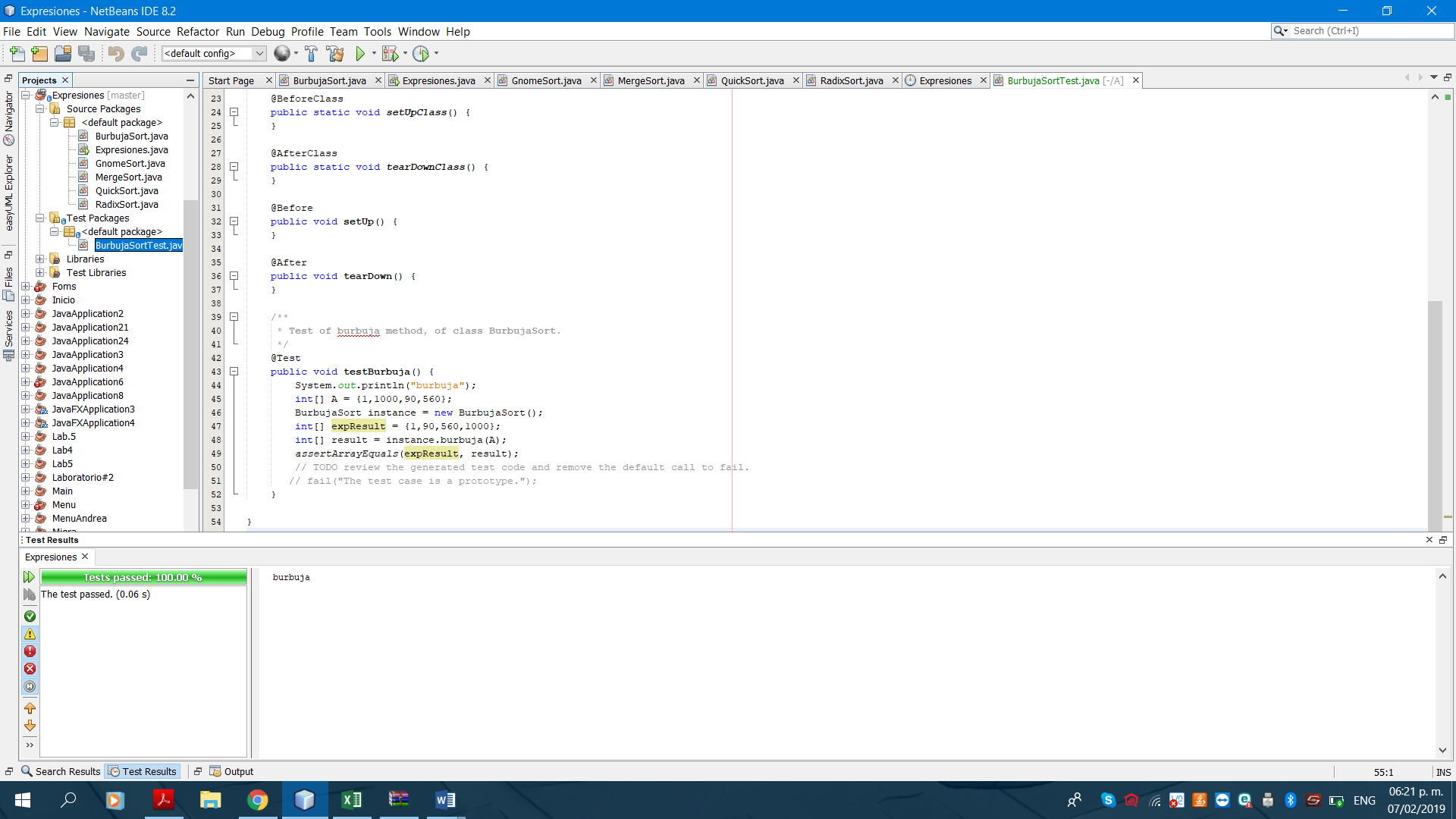
3

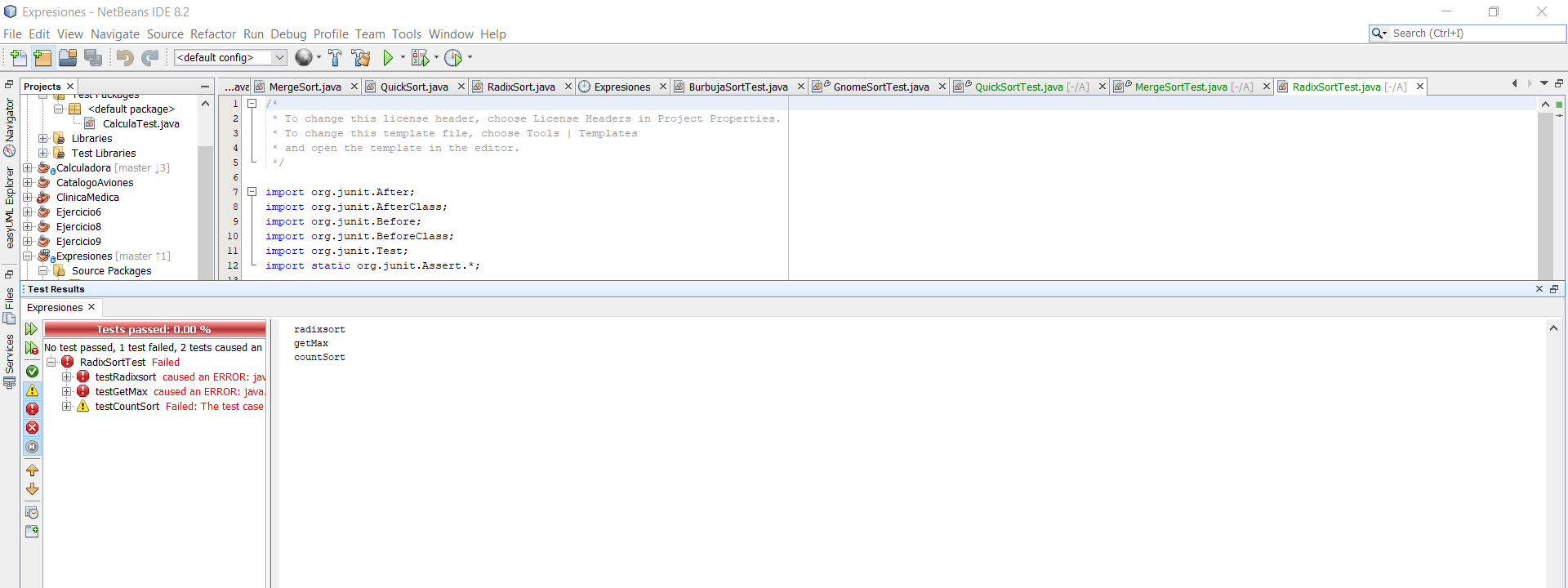
2

1

0

**Pruebas Junit**





**Referencias**

* Bailey, D. (2007). Java Structures. Williams College.
* Codigo extraidp de <http://algs4.cs.princeton.edu/14analysis/Mergesort.java.html>