

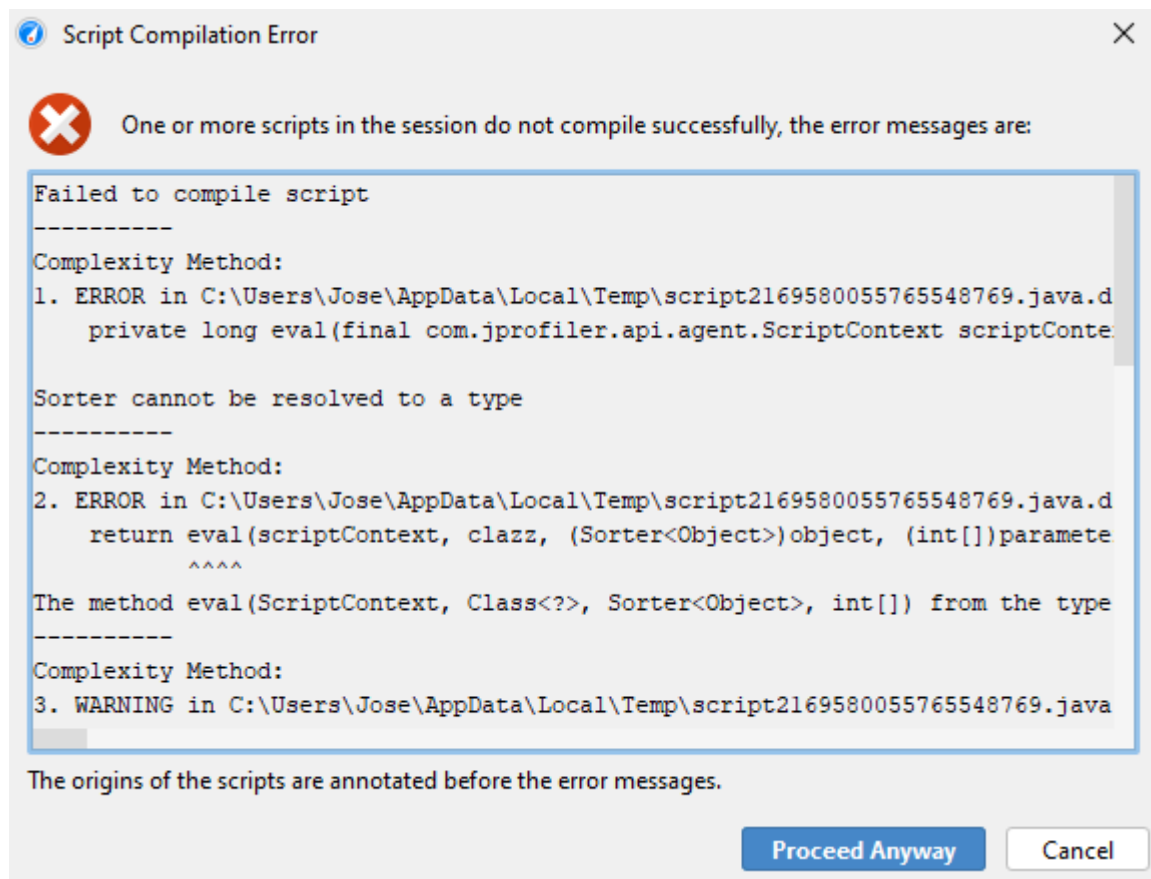
Adrian Lopez
Jose Merida

Hoja de Trabajo 3

Profiler:

JProfiler estaba correctamente configurado y se había utilizado para calcular la complejidad de gnome Sort (la primera implementación) correctamente. Sin embargo, ahora ocurre un error donde no se puede resolver la clase “Sorter”. Este no ocurría anteriormente y se hizo lo posible por solucionarlo sin embargo no se encontro solucion :c

Todos los métodos de sorting tienen un input n que se refiere a la longitud del array, este se utiliza en JProfiler para obtener la gráfica de complejidad.



Utilizando el profiler de IntelliJ IDEA, nos podemos dar una idea de los algoritmos más eficientes en cuanto a tiempo de ejecución.

Gnome Sort: 5,310ms

44 ms		<pre>for (int i = 10; i <= 3000; i++) { Integer[] currentArray = generateRandomArray(i); sorter.gnomeSort(currentArray, i); System.out.println("Gnome " + i + " elementos"); }</pre>
5,310 ms		
11 ms		

Merge Sort: 352ms

67 ms		<pre>for (int i = 10; i <= 3000; i++){ Integer[] currentArray = generateRandomArray(i); sorter.mergeSort(currentArray, i); System.out.println("Merge " + i + " elementos"); }</pre>
352 ms		
22 ms		

Quick Sort: 176ms

33 ms		<pre>for (int i = 10; i <= 3000; i++){ Integer[] currentArray = generateRandomArray(i); sorter.quickSort(currentArray, i); System.out.println("Quick " + i + " elementos"); }</pre>
176 ms		
22 ms		

Radix Sort: 892ms

44 ms		<pre>for (int i = 10; i <= 3000; i++){ Integer[] currentArray = generateRandomArray(i); sorter.radixSort(currentArray, i); System.out.println("Radix: " + i + " elementos"); }</pre>
892 ms		

Bubble Sort: 13,688ms

22 ms		<pre>for (int i = 10; i <= 3000; i++){ Integer[] currentArray = generateRandomArray(i); sorter.bubbleSort(currentArray, i); System.out.println("Bubble " + i + " elementos"); }</pre>
🔥 13,688 ms		
22 ms		

JUnit

✓	SorterTest	8 ms
✓	radixSort()	8 ms
✓	gnomeSort()	
✓	bubbleSort()	
✓	quickSort()	
✓	mergeSort()	

Clase SorterTest en Github c: