UNIVERSIDAD DEL VALLE DE GUATEMALA
CC 2016 -Estructura de Datos

Sección 10

Ing.Michaelle Alexander Pérez Riz

Hoja de Trabajo #3

June Herrera - 231038
Jonathan Zacarías - 231104

Guatemala, 8 de febrero de 2024

Análisis:

- Requerimiento
    - ¿Qué acciones debe poder hacer su programa?
        1. Guardar y Leer Arrays desde/archivo
        2. Generar Arrays Aleatorios
        3. Ordenamiento
    - ¿Con qué datos va a trabajar?
        1. Arrays de Enteros
        2. Archivos de Texto
    - Descripción de clases:Métodos y Atributos de clases:
        - Clase ArrayFile
            - Métodos:
              saveArrayToFile(int[] array, String filename): Guarda el array de enteros en un archivo de texto.
              readArrayFromFile(String filename): Lee un array de enteros desde un archivo de texto.
        - Clase Main
            - Métodos:
              generateArray(int n): Genera un array de enteros aleatorios de longitud n.
              main(String[] args): Utiliza la clase Sorts para aplicar varios algoritmos de ordenamiento a arrays generados aleatoriamente de diferentes tamaños.

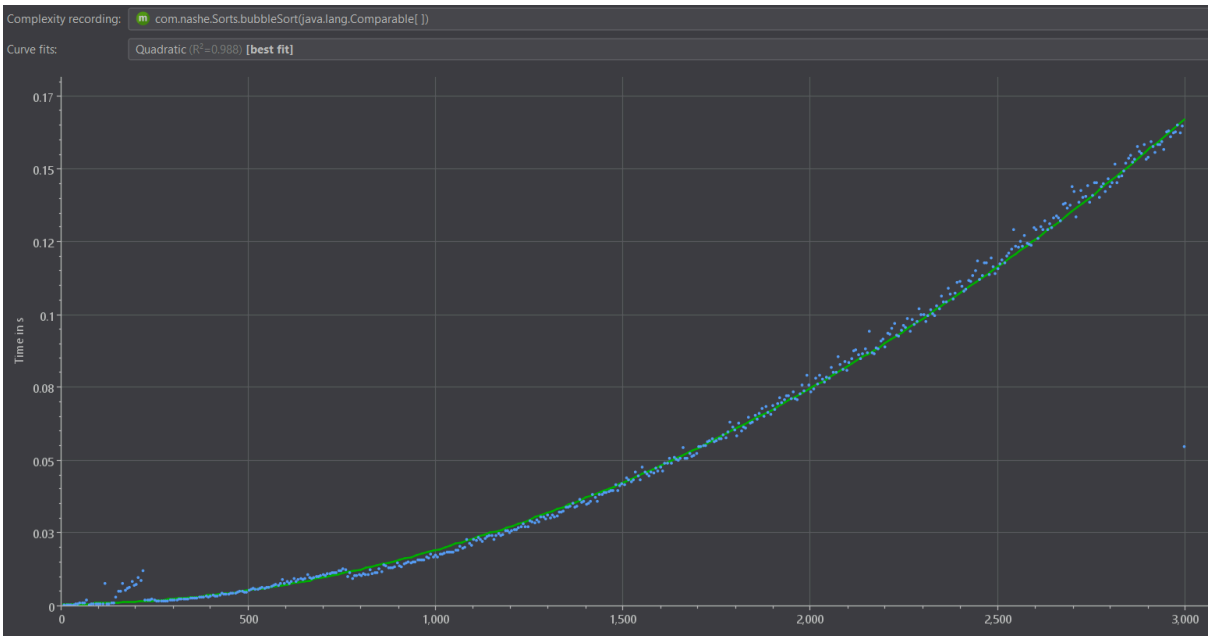
        - Clase Sorts
            - Métodos:
              bubbleSort(T[] arr): Implementa el algoritmo de Bubble Sort.
              gnomeSort(T[] arr): Implementa el algoritmo de Gnome Sort.
              mergeSort(T[] arr): Implementa el algoritmo de Merge Sort.
              quickSort(T[] arr, int low, int high): Implementa el algoritmo de Quick Sort.
              radixSort(T[] arr): Implementa el algoritmo de Radix Sort.
              countSort(T[] arr, int exp): Utilizado en Radix Sort para ordenar según los dígitos.
              getMax(T[] arr): Devuelve el valor máximo en un array (utilizado en Radix Sort).
              merge(T[] arr, T[] left, T[] right): Utilizado en Merge Sort para combinar dos arrays ordenados.

Profiler Utilizado:

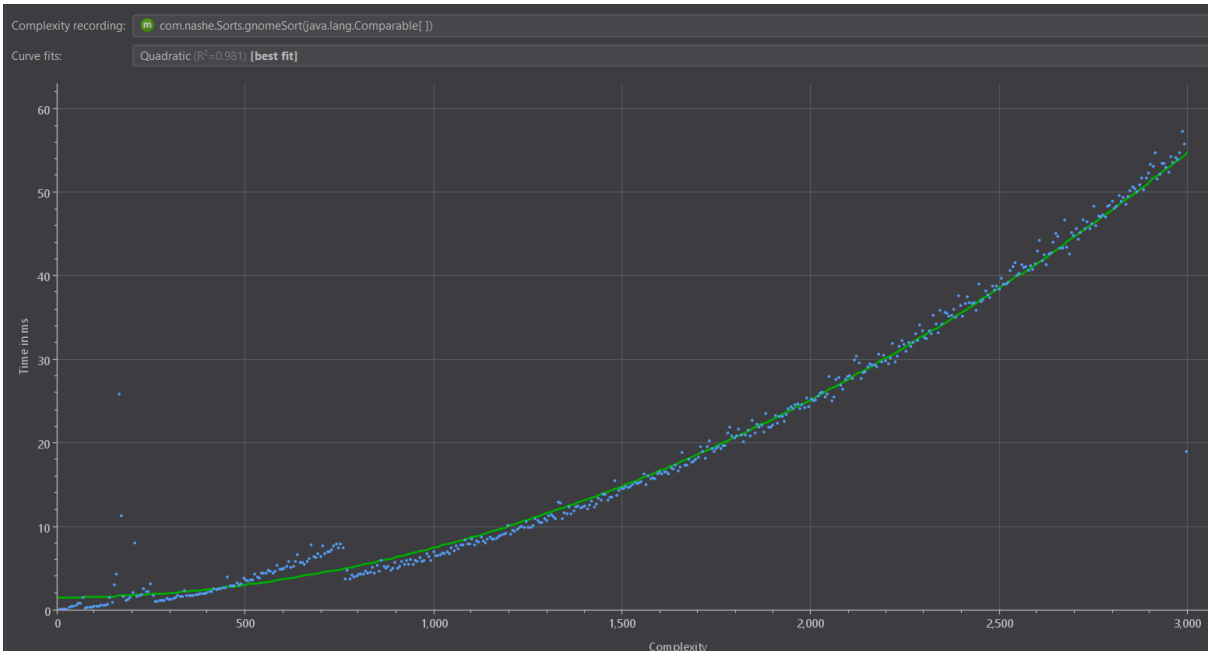
- JProfiler:
  Se utilizó JProfiler para el análisis y perfilado de aplicaciones Java con el objetivo de mejorar su rendimiento y eficiencia este nos ayudó a tener visión detallada del comportamiento de nuestra aplicación Java durante su ejecución.

Gráficos de tiempo de ejecución vs cantidad de elementos del array en JProfiler

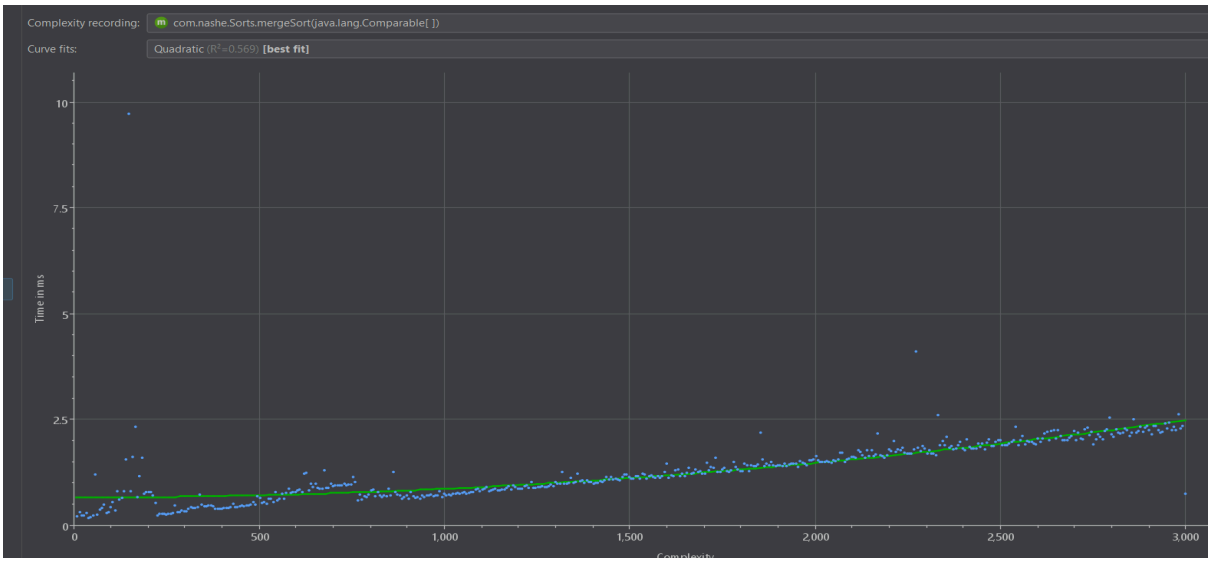
BubbleSort:



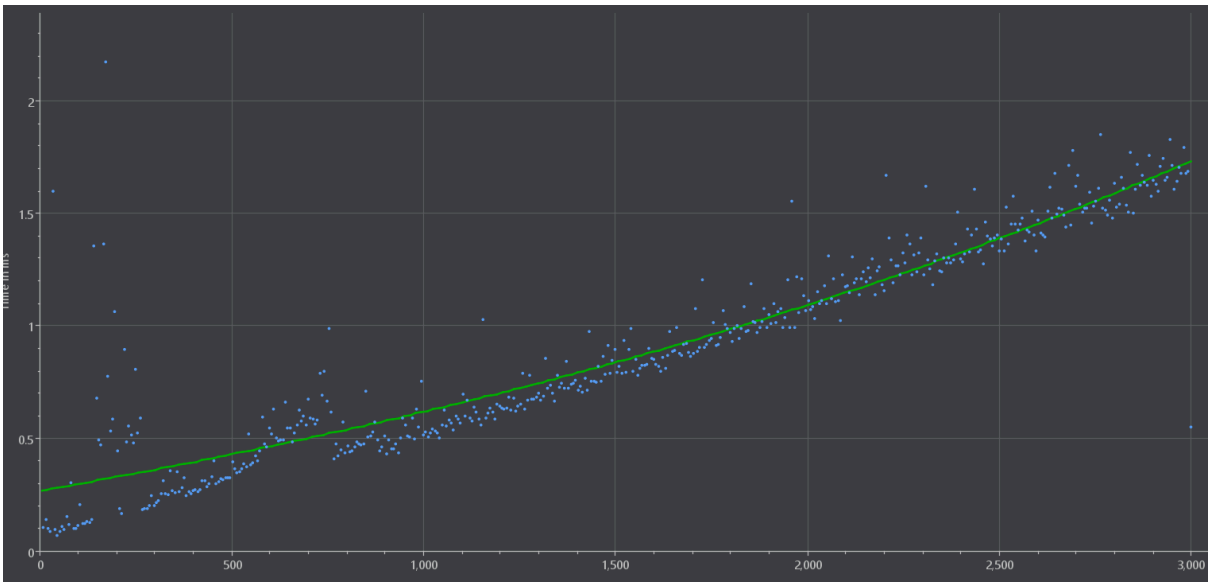
GnomeSort:



MergeSort:



QuickSort:



RadixSort:

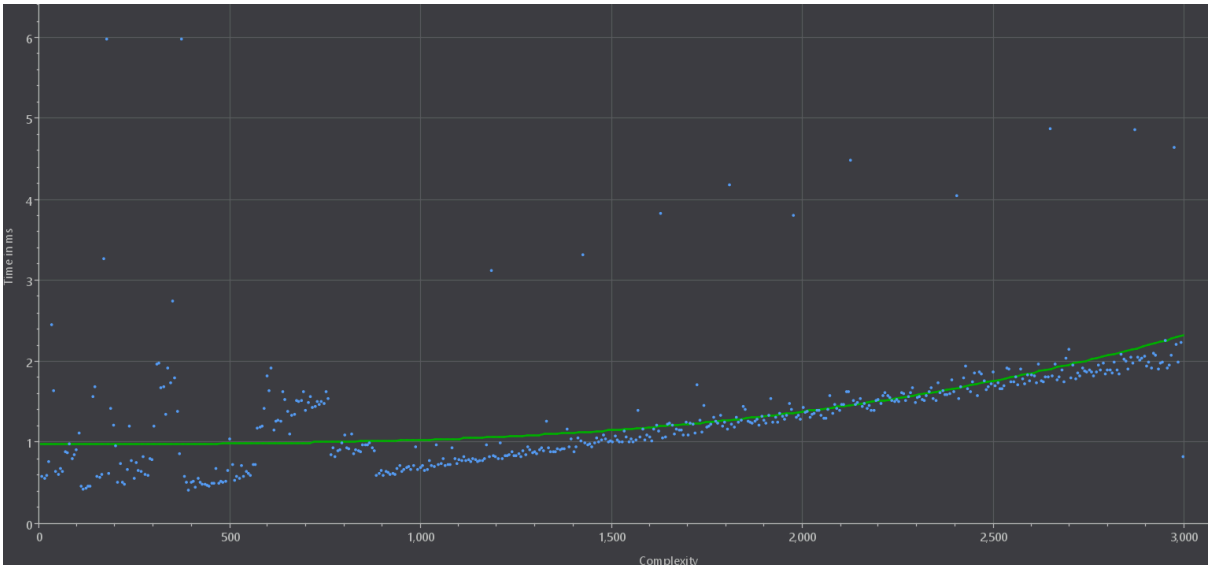


Diagrama de Secuencia UML

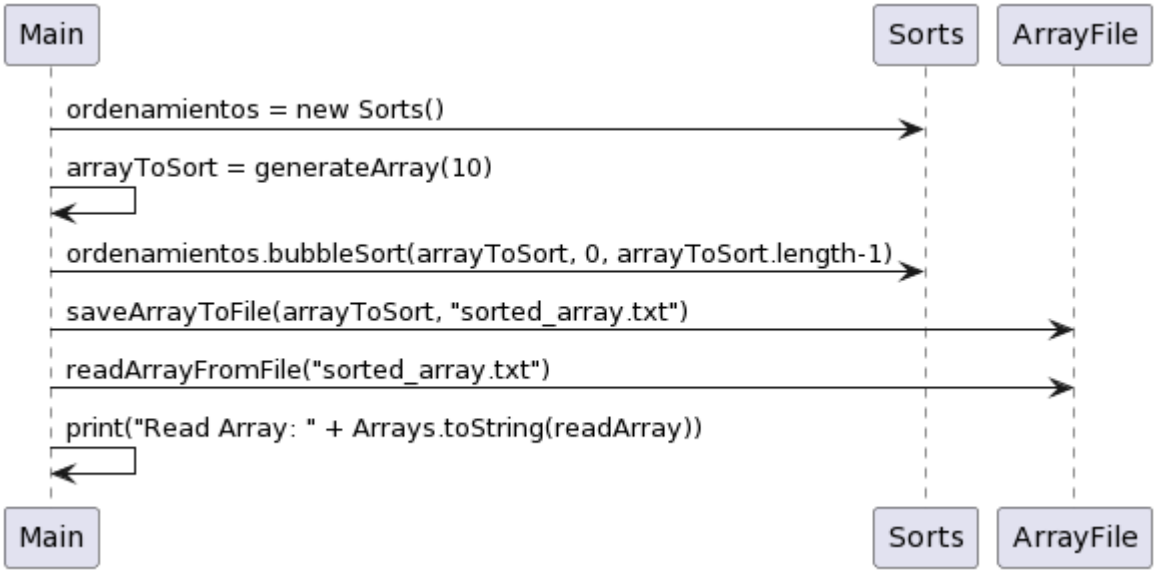


Diagrama de Clases UML