Hoja de trabajo No. 3

Realizar: Programa para usar algoritmos de ordenamiento (sort)
Realizarse: en **PAREJAS.**

Objetivos:
a. Comparación de la complejidad de tiempo de corrida de algoritmos de sort.
b. Utilización de Profilers para medición de tiempos de corrida.
c. Control de versiones del programa.
d. Emplear JUnit para casos de prueba.

Programa a realizar:
Implementar los algoritmos de la siguiente lista en su programa.

1. **Insertion sort**
2. **Merge sort**
3. **Quick sort**
4. **Radix Sort**
5. **Bucket Sort**
6. **Un sort que elija la pareja no listado acá (no se permite realizar BubbleSort)**

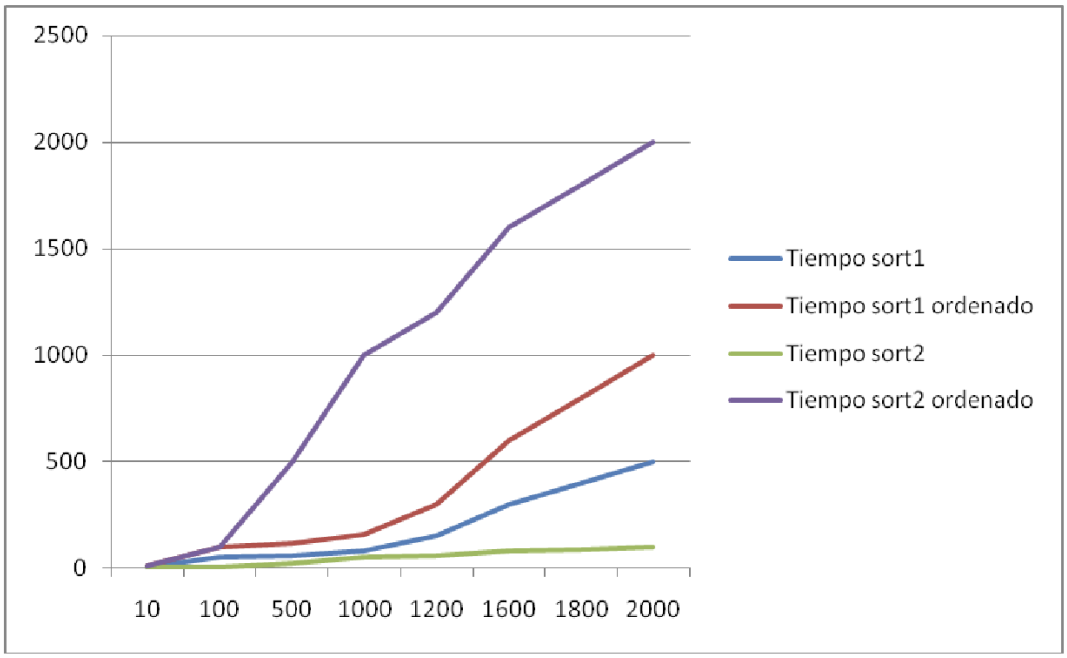
Puede utilizar como referencia el código que se encuentra en el libro Java Structures, de Duane A. Bailey u otro similar. Recuerde siempre incluir la referencia a la fuente que utilizó.
Es interesante también la información sobre sorts del siguiente sitio: http://panthema.net/2013/sound-of-sorting/
Desarrolle un programa que genere al azar y guarde en un archivo, números enteros. Debe generarse hasta 3000 números. Luego utilice su programa para leer los datos de ese archivo, guardarlos en un arreglo y ordenarlos, usando cada uno de los algoritmos de sort.

Utilice el Profiler para medir el tiempo empleado en el ordenamiento para: 10 números hasta 3000 números. El propósito será generar una gráfica con el eje Y mostrando el tiempo y el eje X mostrando la cantidad de números ordenados. Usted puede seleccionar los intervalos que considere adecuados en el eje X.

Deberá correr el programa también para calcular el tiempo que lleva ordenar con su algoritmo una colección de datos YA ordenada. Esto quiere decir que luego de ordenar los datos del arreglo, debe correr nuevamente su programa con esos datos y medir la cantidad de tiempo utilizada para este nuevo ordenamiento.
Recuerde que sus métodos de sort deben recibir como parámetro una estructura de datos que implemente la interfaz Comparable.

Tareas:

a. Realizar una gráfica (en Excel u otro programa similar) de los tiempos de corrida de sus algoritmos. Ver ejemplo de gráfica a continuación:
   NOTA: aunque en la gráfica se muestran solamente dos sorts, en la que su grupo elabore deben estar representados los cuatro algoritmos de sort.
   NOTA: incluya también en la gráfica el rendimiento TEORICO esperado, de acuerdo la complejidad de tiempo del algoritmo usado O( ). Indique cual es la complejidad de tiempo para cada uno de los algoritmos usados en notación big O.
b. Debe dejar evidencia de todo el desarrollo en el repositorio de git (o sistema similar). Indicar como acceder a su repositorio y si es necesario, agregar a su catedrático y auxiliar para que tengan acceso al mismo.
c. Un documento en Word o PDF que explique que profiler utilizó, como lo empleó y los resultados que se obtuvo en cada algoritmo de sort. Incluya la gráfica obtenida.
d. Incluya pruebas unitarias con JUnit para cada uno de los algoritmos de sort implementados.



Debe subir al Canvas todos los productos elaborados en todos los incisos y el enlace a su repositorio de GIT. **Asegúrese de permitir ingreso a su repositorio a su catedrático y auxiliares del curso o que sea un repositorio público.**

Calificación

Aspecto	Puntos
Estilo de codificación: comentarios, identación, nombres de variables significativas	5
Implementación de los sorts	30 (5 puntos cada algoritmo implementado)
Uso del repositorio: existen más de tres versiones guardadas.	5
Medición del tiempo de corrida usando un Profiler.	15
Gráfica con tiempos de corrida.	20
Pruebas JUnit para los sorts.	25
TOTAL:	100