Hoja de trabajo No. 8

Realizar: Un sistema de colas con prioridad para que el CPU atienda procesos en un sistema operativo Linux.
Realizarse: Parejas.

Objetivos:
a. Utilización del ADT Priority Queue.
b. Implementación del Priority Queue empleando un **heap**.
c. Utilización de Java Collection Framework para colas con prioridad

Programa a realizar:

```
top - 10:38:39 up 33 min,  2 users,  load average: 0.00, 0.01, 0.05
Tasks:  85 total,   2 running,  83 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.0 us,  0.0 sy,  0.0 ni,100.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.
KiB Mem:   1017936 total,   294112 used,   723824 free,    11744 buffers
KiB Swap:  2826236 total,        0 used,  2826236 free.   117820 cached Mem

  PID USER      PR  NI    VIRT    RES    SHR S %CPU %MEM     TIME+ COMMAND
 1055 teamspe+   8 -12  763192  18292   7320 S  0.7  1.8   0:07.86 ts3ser+
 2148 root      20   0  123528   1496   1096 S  0.3  0.1   0:04.03 top
    1 root      20   0   46144   6652   3932 S  0.0  0.7   0:01.25 systemd
    2 root      20   0       0      0      0 S  0.0  0.0   0:00.00 kthrea+
    3 root      20   0       0      0      0 S  0.0  0.0   0:00.04 ksofti+
    5 root       0 -20       0      0      0 S  0.0  0.0   0:00.00 kworke+
    6 root      20   0       0      0      0 S  0.0  0.0   0:00.00 kworke+
    7 root      rt   0       0      0      0 S  0.0  0.0   0:00.00 migrat+
    8 root      20   0       0      0      0 S  0.0  0.0   0:00.00 rcu_bh
    9 root      20   0       0      0      0 S  0.0  0.0   0:00.00 rcuob/0
   10 root      20   0       0      0      0 S  0.0  0.0   0:00.24 rcu_sc+
```

(imagen tomada de: https://www.tecmint.com/set-linux-process-priority-using-nice-and-renice-commands/)

Usted fue contratado como parte de un equipo de programadores para desarrollar una nueva distribución de Linux que será la sensación en el mundo de los *geek* de tecnología. Se llamará ChichicasteLinux, por su rudeza y defensa contra los ciberataques. Le han asignado la parte de manejar la prioridad de los procesos para ser atendidos por el CPU.

Le han indicado que todos los Linux emplean un atributo del proceso que se llama "nice" que indica como calcular la prioridad. A continuación, una explicación de como funciona este atributo:

Nice[1] value is a user-space and priority PR is the process's actual priority that use by Linux kernel. In linux system priorities are 0 to 139 in which 0 to 99 for real time and 100 to 139 for users. nice value range is -20 to +19 where -20 is highest, 0 default and +19 is lowest. relation between nice value and priority is :

PR = 20 + NI
so , the value of PR = 20 + (-20 to +19) is 0 to 39 that maps 100 to 139.

[1] https://askubuntu.com/questions/656771/process-niceness-vs-priority

Usted decide hacer el sistema basado en una Priority Queue en la que se ingresan los datos del proceso y se retira de esa cola al proceso que tenga la prioridad de atención más rápida.

Las datos del proceso vendrán en un archivo de texto (debe llamarse **procesos.txt**) con los campos separados por comas. El primer dato es el nombre del proceso, el segundo es el nombre del usuario y el tercero es el valor nice. Por ejemplo:

```
vi,juan02,0
ls,maria30,-20
firefox,rosa20,5
cat,juan02,5
```

El sistema permite indicar cual proceso debe ser atendido por el CPU. Así, que cuando el CPU pide el siguiente proceso, se retira de la cola y se muestra el nombre del proceso, nombre del usuario, prioridad (PR) y el valor nice. El orden en que serán retirados los procesos, uno a la vez:

```
ls,maria30,-20,PR = 100
vi,juan02,0,PR= 120
firefox,rosa20,5,PR=125
cat,juan02,5,PR=125
```

Tareas:

a. Codifique la clase Proceso que contendrá los datos del proceso. Notar que esta clase debe implementar la interfaz comparable. De esta forma se sabrá cual proceso debe ser atendido antes, dependiendo de su prioridad (calculada de acuerdo al valor "nice" del proceso).
b. Implemente la clase VectorHeap<E extends Comparable<E>> implements PriorityQueue<E> para manejar una cola con prioridad basada en un Heap.
c. Haga pruebas unitarias para los métodos que insertan y retiran un elemento del VectorHeap.
d. Haga otra versión del programa, pero ahora use el Java Collection Framework para el PriorityQueue: http://docs.oracle.com/javase/7/docs/api/java/util/PriorityQueue.html
Los métodos no necesariamente se llaman como se especificaron en la interface PriorityQueue.

Debe subir a Canvas todos los productos elaborados y los enlaces a su repositorio de github (o equivalente).

Calificación:

Aspecto	Puntos
Uso del repositorio: existen más de tres versiones guardadas, la última versión es igual a la colocada en el Canvas.	10
Implementación de clase VectorHeap (que implementa un PriorityQueue)	10
Documentación Javadoc de VectorHeap	10
Pruebas unitarias de su implementación de VectorHeap	10
Programa de atención de atención de los procesos que requieren CPU que use su implementación de VectorHeap.	30
Programa de atención de de los procesos que requieren CPU que use la implementación proporcionada por el Java Collection Framework.	30

TOTAL:	100