PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA
DEPARTAMENTO DE CIENCIA DE LA COMPUTACIÓN

IIC2233 - Programación Avanzada
2° semestre 2015

Actividad 15

Threading básico - Workers!

Problema

Gracias a que usted modeló a la perfección el problema presentado en la AC01, se ha podido calcular, para la gran mayoría de las estrellas en la base de datos, el promedio y la varianza de la magnitud de su brillo. Sin embargo, hay algunas estrellas famosas que a lo largo de la historia poseen millones de observaciones, tales como Arturo, Sirio, Canopus, Vega y la estrella binaria Alpha Centauro. Como los cálculos toman mucho tiempo en realizarse, se le pide nuevamente su ayuda.

Las magnitudes aparentes de los brillos de estas estrellas estarán descritas en un archivo diferente para cada estrella. No te alarmes por los valores de las magnitudes: están en escala logarítmica y pueden ser negativos (de hecho, a menor valor de magnitud, más brillante se ve la estrella).

Hay un archivo de ejemplo en esta carpeta. Para conseguir los demás archivos, corre el programa **getdata.py** que está disponible junto con el enunciado (paciencia, puede tardar unos tres minutos en terminar). **No subas estos archivos a tu repositorio. Habrá descuento de puntaje**.

Requerimientos

En esta actividad deberás escribir un programa que pida comandos a un usuario para trabajar con los archivos. Cada vez que se ejecuta un comando se debe hacer en un thread independiente del programa principal, de modo de poder seguir ingresando inputs sin que se quede pegado. Los distintos comandos que pueden ser ingresados son los siguientes.

- **mean estrella:** calcula el promedio de los datos en el archivo de la estrella.
- **var estrella:** calcula la varianza de los datos que hay en el archivo de la estrella. Solo puede echarse a correr si ya se ha terminado de ejecutar el comando `mean estrella` para dicha estrella.
- **exit:** indica que ya no se ingresarán más comandos.

Se requiere procesar cada comando en un thread distinto. Para esto, debes escribir la clase **Worker**, cuyas instancias se inicializan a partir de una instrucción (**mean** o **var**) y el nombre de una estrella. Además, debes escribir el código del programa principal, el cual debe funcionar como se indica en los puntos a continuación:

- Pide un comando al usuario. Si el comando no es **exit**, se analiza si el comando puede ejecutarse o no. Hay dos razones por las cuales un comando no podrá ejecutarse:

    - No se puede ejecutar **var estrella** si el promedio asociado a dicha estrella no ha sido calculado todavía (o no ha terminado de calcularse). Esto es porque se necesita conocer el promedio para calcular la varianza.
    - No se puede ejecutar un comando si un worker ya lo está ejecutando. Sin embargo, si un worker ya terminó de ejecutarlo, **sí** puede volver a ejecutarse por otro worker.
- Si el comando no puede ejecutarse, se debe imprimir la razón en la consola. Si se puede, se debe crear un worker que lo ejecutará y echarlo a correr.
- Cuando un worker termina de ejecutar un comando, se imprime en la consola qué worker ha terminado, cuál es el comando que ejecutó y cuál es el resultado.
- Se repite lo anterior hasta que se ingrese **exit**. Cuando esto ocurre, si hay workers que no han terminado de ejecutar su comando, deben detenerse. Se debe imprimir en la consola todos los comandos que fueron echados a correr. No es necesario que imprima el resultado de cada comando, pero sí debe indicar cuáles comandos terminaron su ejecución y cuáles no.

To - DO

- (0.5 pts) Revisar que solo se puede calcular `var estrella` si ya se calculó `mean estrella`.
- (1.0 pts) Revisar que no haya un worker ejecutando el comando antes de crear uno que lo ejecute.
- (1.0 pts) Instanciar un worker cuando corresponda indicando qué comando ejecutará, y echarlo a correr.
- (1.0 pts) Cuando se ingresa `exit`, los workers que no han terminado se detienen.
- (1.5 pts) Cada worker ejecuta correctamente su función en el método `run`. Esto considera que se haga un `print` con los elementos especificados.
- (1.0 pts) Luego de ingresar `exit`, se imprime cuáles comandos terminaron su ejecución y cuáles no.

Tips

- Solo por esta vez, no se preocupe de que las variables que utilice sean thread-safe (es decir, que no ocurra que se generen problemas porque varios threads intenten modificarlas de forma casi simultánea). Ya tendrá suficiente de esto en la siguiente actividad.
- Para saber si ya fue calculado el promedio para una determinada estrella, puede usar la información del diccionario estático `mean_data`.
- **Recuerda borrar los archivos** de tu carpeta.

Ejemplo

```
Ingrese siguiente comando:
mean AlphaCentauri
Creando worker para: mean AlphaCentauri
Ingrese siguiente comando:
mean AlphaCentauri
[DENIED] Ya hay un worker ejecutando el comando
Ingrese siguiente comando:
mean Arcturus
Creando worker para: mean Arcturus
Ingrese siguiente comando:
var AlphaCentauri
[DENIED] No se puede calcular varianza sin haber calculado el promedio antes!
Ingrese siguiente comando:
var ArcturuSoy el Thread-2 y termine mi trabajo!
        El resultado de mean Arcturus es -0.04
Ingrese siguiente comando:
s
Creando worker para: var Arcturus
Ingrese siguiente comando:
Soy el Thread-1 y termine mi trabajo!
        El resultado de mean AlphaCentauri es -0.27
Ingrese siguiente comando:
mean AlphaCentauri
Creando worker para: mean AlphaCentauri
Ingrese siguiente comando:
exit
Comandos ingresados por el usuario
Alcanzo a terminar: mean Arcturus
Alcanzo a terminar: mean AlphaCentauri
No alcanzo a terminar: var Arcturus
No alcanzo a terminar: mean AlphaCentauri
```