Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación
IIC2233 — Programación Avanzada 2019-1

09 de mayo de 2019
Actividad Formativa

Actividad 06

Threading

Introducción

Se acerca el invierno y los "Trotadores Blancos" amenazan con atacar los siete imperios. La única forma de salvarse es que Johnny Copo de Nieve y sus aliados logren encontrar la forma de acabar con los Trotadores y estar a tiempo en el campo de batalla. ¡Necesitan tu ayuda!

Johnny Copo de Nieve logró encontrar unos pergaminos con información valiosa acerca de como vencer a los Trotadores Blancos, sin embargo, éstos son muy difíciles de descifrar ya que están llenos de polvo. Para optimizar el tiempo, Johnny y sus amigos decidieron descifrar los textos al mismo tiempo que van al campo de batalla, montados sobre dragones. La victoria será de la humanidad si al menos uno de los pergaminos logra ser descifrado y logra llegar al campo de batalla.

Ya que estás bien versado en las antiguas artes de la programación, la Reina Dany te ha encargado la organización de los equipos que ayuden a destruir a los Trotadores Blancos. Para que todo sea lo más eficiente posible, se crean tres equipos capaces de traer la victoria. Cada equipo está formado por: un dragón que se encargará del transporte; y un jinete (Johnny, Gnomo o Jaimito) que deberá descifrar antiguos textos para encontrar la forma de ganar la batalla. Para esto, usarás todos tus conocimientos de *threading* para una mejor modelación del problema.

Carrera contra la extinción

El viaje hacia la batalla se realizará en forma de simulación, la cual comenzará con tres equipos, formados por un dragón montado por su respectivo jinete. Para efectos de la simulación debes considerar que un dragón siempre viaja con su jinete. Ellos se dirigirán al campo de batalla que se encontrará a una distancia de 1000 metros. Tu programa debe terminar cuando el primer dragón y su jinete lleguen al destino con el texto descifrado. En caso de que llegue un dragón y su jinete al campo de batalla sin que el jinete haya descifrado los textos, hay dos opciones para finalizar:

- Cuando un jinete que ya llegó logre descifrar los textos, mientras su dragón espera.
- Si otro dragón y su jinete llegan con los textos descifrados

En todo momento, tu programa debe imprimir en consola los sucesos que ocurren dentro de la simulación. Por ejemplo: cuando parte el viaje, cuando un jinete logra descifrar el texto, cuando un jinete llega al campo de batalla, o cualquier otro evento importante dentro de la simulación.

Para poder realizar la simulación, deberás crear las siguientes clases, que serán los distintos *threads* y Dany, que debe ser un lock:

- `class Dragon`: El dragón tendrá una velocidad inicial que estará en el rango de los 30 - 40 m/s y cada 500 metros disminuye entre 0 - 5 m/s por cansancio, pero siempre manteniendo una velocidad mínima de avance de 15 m/s. Además, a partir de los 500 metros de viaje, el dragón se puede ver tentado a detenerse y saciar su hambre con una probabilidad de 0,1 en cualquier segundo. Cuando esto ocurra, llegará Dany (La madre de los dragones) al rescate, quien hablará telepáticamente con el dragón para persuadirlo a que vuelva a volar. Por último, cada dragón tiene un nombre (Rhaegal, Viserion y Drogon) y está asignado a un jinete.
- `class Dany`: Es la encargada de evitar que los dragones no se distraigan durante el camino. Cuando un dragón se detenga a comer, se demora entre 2 a 5 segundos en convencerlo de volar,mediante su método convencer. Como Dany no puede estar en dos lugares al mismo tiempo, si un dragón se detiene a comer mientras Dany está hablando con otro, éste seguirá comiendo hasta que Dany termine su tarea. Recuerda que debes avisar en todo momento lo que está ocurriendo, es por esto que Dany debe indicar el tiempo en que inicia y termina de convencer a cada dragón, junto con el nombre del jinete que vuelve a emprender su viaje.
- `class Jinete`: El jinete se encarga de descifrar los textos. Corresponde a Johnny, Gnomo o Jaimito. Una vez que logre terminar de descifrar su texto, el jinete deberá presentarse con su nombre y mostrar el texto descifrado. Así, todos podrán ver quien lo resolvió y cuál es la forma de vencer a los Trotadores. Como los textos son tan antiguos y están llenos de polvo, el jinete tardará entre 40 a 60 segundos en limpiarlos, para recién comenzar a descifrarlos (el jinete puede descifrar aunque el dragón este comiendo).
- `class Viaje`: Esta clase es la encargada de manejar la simulación. Es el *thread* principal encargado de iniciar a los demás. Tiene un método `agregar` el cual recibe un nombre de dragón y un nombre de jinete, con los que debes crear los *threads* correspondientes, para luego, en su método `run` poder empezar la simulación.

Descifrado de textos

Una vez que el jinete termine de desempolvar los textos, deberá descifrar el mensaje oculto dentro de este. Estos mensajes están escritos en Valyrio antiguo. Para nuestra fortuna, el maestre **Valdivieso** te ha proporcionado la función desencriptar, que el jinete utilizará para resolver el archivo `pergaminos.txt`. Cuando el jinete termine de descifrar el texto, deberá avisar que lo logró y crear un nuevo archivo llamado `mensaje.txt` que contenga el mensaje descifrado.

¡Atención! Recuerda que cuando trabajas con *threading* debes evitar que dos o más *threads* accedan a un archivo al mismo tiempo, puede traerte problemas.

Registro

Una vez terminada la simulación, deberás imprimir las estadísticas de cada equipo dragón-jinete:

- Nombre del dragón.
- Distancia recorrida por el equipo.
- Nombrar al jinete, quien se presentará y dirá si resolvió o no el problema.
- Y además, se debe mostrar el tiempo en que se finalizó la simulación.

Archivos

En el archivo `main.py` vienen las clases base y sus métodos, los cuales debes completar para poder hacer tu simulación, así como también la función `desencriptar`, que ya está definida dentro del archivo. Además, el archivo `pergamino.txt` contiene el texto cifrado que el jinete deberá descifrar.

Notas

- Para controlar el uso de recursos simultáneamente, puedes usar `Lock` de `threading`.
- Para que un *thread* espere a otro thread, puedes usar `join`.
- Recuerda que puedes crear clases que hereden de `threading`.
- Utiliza la funcion `time.sleep` para simular por segundos.
- Para todo valor aleatorio que debes calcular, puedes utilizar `randint` de `random`.

Requerimientos

- (2,4 pts) Crear los *threads* correspondientes
  - (1,4 pts) Crea correctamente el *thread* `Dragon`
  - (1,0 pt) Crea correctamente el *thread* `Jinete`
- (0,6 pts) Dany solo puede estar convenciendo a un dragón a la vez
- (1,0 pt) Clase `Viaje`
  - (0,2 pts) Implementación del método `agregar`
  - (0,8 pts) Correcta implementación del `run`
- (1,4 pts) Estadísticas
  - (0,2 pts) Presentación del Jinete ganador con la solución
  - (0,2 pts) Tiempo de demora del equipo salvador
  - (0,6 pts) Presentación de Dragones
  - (0,4 pts) Punto de llegada de todos los equipos
- (0,6 pts) Poblar la simulación con los 3 equipos correspondientes

Entrega

- **Lugar:** En su repositorio privado de GitHub, en la **carpeta** `Actividades/AC06/`
- **Hora del *push*:** 16:30

Auto-evaluación

Como esta corresponde a una actividad formativa, te extendemos la instancia de responder, después de terminada la actividad, una auto-evaluación de tu desempeño. Esta se habilitará a las **16:50 de jueves 9 de mayo** y tendrás plazo para responderla hasta las **23:59 del día siguiente, viernes 10 de mayo**. Puedes acceder al formulario mediante el siguiente enlace:

`https://forms.gle/XPVZy7bsMGNdhgBd7`

El asistir, realizar la actividad y responder la auto-evaluación otorgará como bonificación al alumno 2 décimas para sumar en su peor actividad sumativa del semestre.