Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación
IIC2233 — Programación Avanzada 2018-2

11 de octubre de 2018
Evaluada

Actividad 09

Threading

Introducción

¡Oh no! ¡El malvado Dr. Herny está tan enojado con los estudiantes de *Programación avanzada* por haber visto su desempeño en *"50 Sombras de Hernán"*, que decidió calificar con 3,9 a todos los alumnos del curso. Sin embargo, el benevolente NebiLockbottom ha formulado un plan para rescatarlos de dicha situación. ¡Apúrate y salva las notas del curso!

Instrucciones

Como nadie quiere que el sombrío Dr. Herny se salga con la suya, has decidido ayudar a NebiLockbottom, quien te encargó organizar varios equipos de programadores para llevar a cabo la misión. Cada equipo constará de dos programadores:

- ***Hacker:*** Se encargará de desencriptar un archivo entregado que te permitirá ingresar a la computadora de Dr. Herny.
- ***Cracker:*** Creará una inyección de código que cambia los 3,9 por gloriosos 7.

Misión

La misión constará de tres equipos de programadores formados por sus respectivos *hacker* y *cracker*. Cada equipo trabaja de forma independiente de los otros, por lo que **la misión termina una vez que cualquiera de los tres equipos logre ingresar e inyectar el código exitosamente** en la computadora de Dr. Herny. Se te indicarán distintas actividades que se llevan a cabo para lograr este objetivo, y cada una de ellas toma una cierta cantidad de tiempo. Considera que **un minuto de la simulación equivale a un segundo de la vida real**. Para lograr esto, puedes utilizar la función `time.sleep`.

Para realizar la misión, deberás crear las siguientes entidades:

- **Equipo:** Cada equipo debe tener un nombre y encargarse de la instanciación, esto es, el manejo de sus integrantes (*hacker* y *cracker*) que deben ser modelados como *threads*.
- **Hacker:** Se encargará de desencriptar el archivo entregado. Como este está en un cubo mágico para evitar que caiga en manos enemigas, el *hacker* tardará entre 4 a 12 minutos, con distribución

uniforme discreta, en abrirlo para recién comenzar a desencriptar[1]. Una vez que logre lo anterior, la desencriptación es *instantánea* y deberás hacer un `print` indicando el nombre del equipo y avisar que su *hacker* terminó de realizar la desencriptación.

- **Cracker:** Debe escribir 50 líneas de código para generar el programa que cambia las notas. Tendrá una velocidad aleatoria (uniforme discreta) de escritura que estará en el rango de $5 - 15$ líneas de código por minuto. Una vez que termine de escribir, tiene que hacer un `print` especificando el nombre del equipo y que su *cracker* terminó el código. Además, en cada minuto existe 20 % de probabilidad de que su computador sea víctima de un ciberataque provocado por Dr. Herny. Si esto sucede, el *cracker* no puede seguir trabajando hasta que el problema sea solucionado.
- **Misión:** Corresponderá a la clase que se encargará del manejo del programa e instanciación de los equipos, por lo que deberás implementar el método `run`[2]. Una vez que la misión sea completada, debes imprimir en pantalla el nombre de cada equipo, cuántas líneas escribió el *cracker*, y si el *hacker* logró desencriptar el archivo. También debes indicar el equipo ganador.

Por último, tienes que considerar cómo implementar a NebiLockbottom, para que cumpla con lo siguiente:

- **NebiLockbottom:** Es el encargado de ayudar a los *crackers* en caso de un ciberataque provocado por las defensas del Dr. Herny. Cuando ocurre un ataque, podrá reparar en un tiempo aleatorio uniforme de 1 a 3 minutos, tiempo en el cual el *cracker* deberá esperar a que NebiLockbottom termine para seguir escribiendo. Si el software de Dr. Herny justo ataca a otro *cracker*, este último deberá esperar a que NebiLockbottom se encuentre disponible. Recuerda que deberás avisar en todo momento lo que está ocurriendo, es por esto que cada vez que NebiLockbottom empiece y termine de reparar un computador debes hacer un `print` indicando el nombre de NebiLockbottom, el equipo del *cracker* que ayudó, y si empezó o terminó la reparación.

Hint: NebiLockbottom no es necesariamente una clase. Lo único que debe cumplir es que solo un *thread* lo pueda ocupar a la vez, después de todo, solo puede atender a un *cracker* al mismo tiempo.

Archivo

Los protagonistas de "Buscando a Enzo" y "Lo que Benja se llevó" se infiltraron en la sombra central del Dr. Herny y lograron entregarle a NebiLockbottom un archivo encriptado llamado `pista.txt` con las claves del villano, y una función llamada `desencriptar` que abre un archivo encriptado y entrega un *string* del contenido desencriptado.

Notas

- Modelar los minutos como segundos puede ahorrarte mucho tiempo.
- Puedes importar el módulo `random` para emplear `randint(a, b)` y emular el comportamiento de una distribución discreta uniforme.
- Para que un *thread* espere a otro puedes usar `join`.
- Recuerda que puedes crear clases que hereden de `threading.Thread`.
- Para controlar el uso de recursos simultáneamente, puedes usar *lock* de *threading*.

[1] El *hacker* puede abrir el cubo aunque el Dr. Herny haya atacado al *cracker*.
[2] Por conveniencia llamado así, no es necesario que sea un `thread` porque ya sería el *main thread*.

- Aunque en los requerimientos (a continuación) no se especifique sobre el uso de elementos de *threading*, para la correcta implementación de los `run` se evalúan estos contenidos.

Requerimientos

- (2,5 pts) Crear las entidades correspondientes
  - (0,5 pts) NebiLockbottom
    - (0,5 pts) Reparación de un equipo a la vez
  - (0,6 pts) Cracker
    - (0,2 pts) Velocidad de escritura
    - (0,4 pts) Correcta implementación del `run`
  - (0,8 pts) Hacker
    - (0,2 pts) Decodificación
    - (0,2 pts) Creación de archivo
    - (0,4 pts) Correcta implementación del `run`
  - (0,6 pts) Equipo
    - (0,2 pts) Asignación de nombre y parámetros
    - (0,4 pts) Correcta implementación del `run`
- (1,0 pt) Clase Misión
  - (0,2 pts) Asignación de parámetros
  - (0,8 pts) Correcta implementación del `run`
- (1,5 pts) *Prints* y registro
  - (0,3 pts) `print` correctos para NebiLockbottom
  - (0,3 pts) `print` correctos para *hacker*
  - (0,3 pts) `print` correctos para *cracker*
  - (0,6 pts) Registro final
- (0,5 pts) Poblar el programa con 3 equipos
- (0,5 pts) Manejo del tiempo

Entrega

- **Lugar:** En su repositorio privado de GitHub, en la **carpeta** `Actividades/AC09/`
- **Hora del último *push* válido:** 16:40