Laboratorio 3 ARSW 2023-2

Johan Sebastian Garcia Martinez

Ejercicio - Programación concurrente, condiciones de carrera y sincronización de hilos

Parte 3. Sincronización y Deadlocks



1. Revise el programa de highlander-simulator

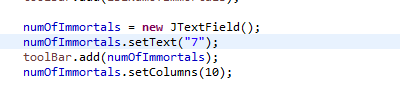
* Toda la configuración de los jugadores inmortales se encuentra en el método “setUpInmortals” de la clase de ControlFrame, donde se toma el valor de las variables las cuales se definen de la siguiente manera:

- DEFAULT\_IMMORTAL\_HEALTH: Salud del jugador inmortal

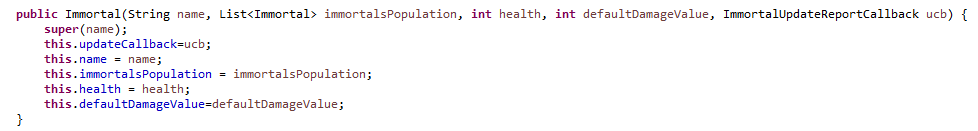
- DEFAULT\_DAMAGE\_VALUE: Daño del ataque de cada jugador.

los cuales se encuentran como variables globales y luego encontramos la cantidad de jugadores dentro de la variable “numOfImmortals” directamente en el constructor de la clase

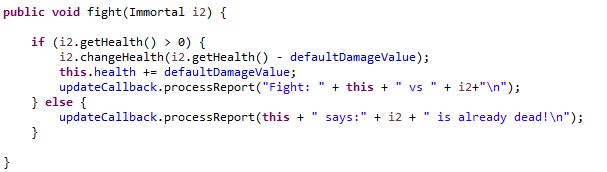
* Se tienen N Jugadores inmortales los cuales se encuentran definidos en la variable “numOfImmortals”



* Cada jugador conoce a los N-1 jugadores restantes, esto se debe a que en el constructor de cada jugador inmortal, se envía como parámetro el LinkedList con la información de todos los jugadores inmortales que le preceden



* cada jugador ataca permanentemente a otro inmortal seleccionado de manera random, con un número que se genera por método random dentro del método run del hilo inmortal y el ataque está definido dentro del método “fight” de la misma clase hilo, al cual le llega como parámetro el jugador al cual va a realizar su ataque, el valor de “defaultDamageValue” se le resta a la vida del jugador atacado y se le suma a la vida al jugador que ataca



1. Dada la implementación del juego se puede determinar una condición de carrera que se puede definir como una colisión de eventos donde varios hilos entran a modificar una misma variable, por lo que no es thread-safe, esto debería controlarse con sincronización para poder cumplir el invariante definido como:

**Invariante**: La sumatoria de la salud de todos los jugadores siempre debe ser la misma ya que el ataque resta tantos puntos de vida a la víctima cómo suman al atacante.

El valor de **N** para el invariante sería el producto de la cantidad de jugadores multiplicado por la salud definida inicialmente para cada uno, en un momento donde no se encuentren haciendo operaciones de incremento/reducción en la variable de salud.

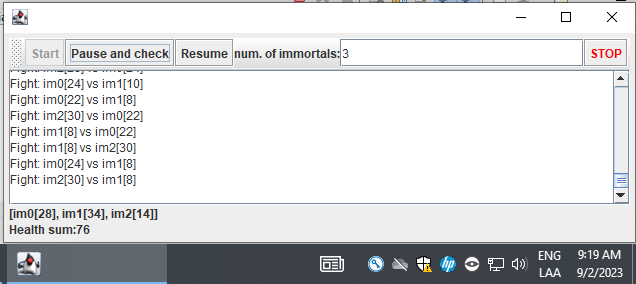
En el código, el valor **N** para la sumatoria de las vidas de todos los jugadores está dado de la siguiente manera:

**numOfInmortals \* DEFAULT\_IMMORTAL\_HEALTH = N**

como la variable “numOfImmortals” es un string, vamos a obtener su valor con el siguiente código:

**Integer.parseInt(numOfImmortals.getText());**

1. Ejecute la aplicación y verifique cómo funciona la opción de “pause and check”:



El valor determinado para las variables de este ejemplo fueron:

- Default\_immortal\_health = 10

- Default\_damage\_value = 2

- numOfImmortals = 3

Por lo tanto el valor del invariante **N**  que se puede ver reflejado en el label como **Health sum:** debería ser 3\*10 , pero como podemos observar en este caso nos muestra 76, esto nos muestra que el invariante no se está cumpliendo.