**Parte III.**

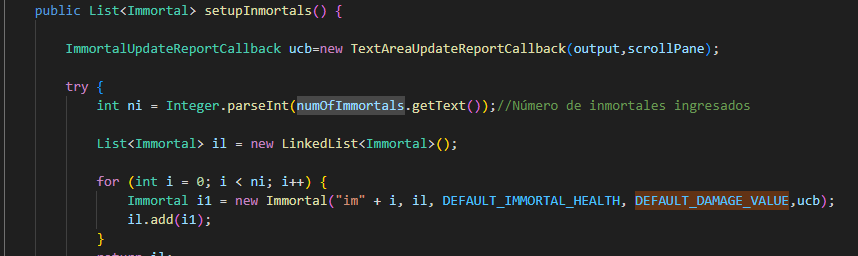
Sincronización y Dead-Locks.



**2**. Revise el código e identifique cómo se implemento la funcionalidad antes indicada.

* Se tienen N jugadores inmortales.

En clase ControlFrame se leen los números de inmortales ingresados por el usuario:



* Cada jugador conoce a los N-1 jugadores restantes

Cada inmortal se crea con la información de todos los inmortales que existen:

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente

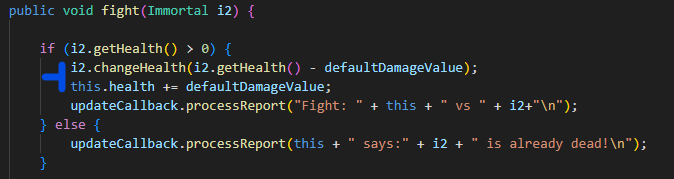
* Cada jugador, permanentemente, ataca a algún otro inmortal. El que primero ataca le resta M puntos de vida a su contrincante, y aumenta en esta misma cantidad sus propios puntos de vida.

Método run de Inmortal contiene un ciclo infinito en donde se llama al método de ataque (fight()).

Texto

Descripción generada automáticamente

Se disminuye el valor del ataque y se aumenta el nivel de vida.



Dada la intención del juego, un invariante debería ser que la sumatoria de los puntos de vida de todos los jugadores siempre sea el mismo(claro está, en un instante de tiempo en el que no esté en proceso una operación de incremento/reducción de tiempo). Para este caso, para N jugadores, cual debería ser este valor?.

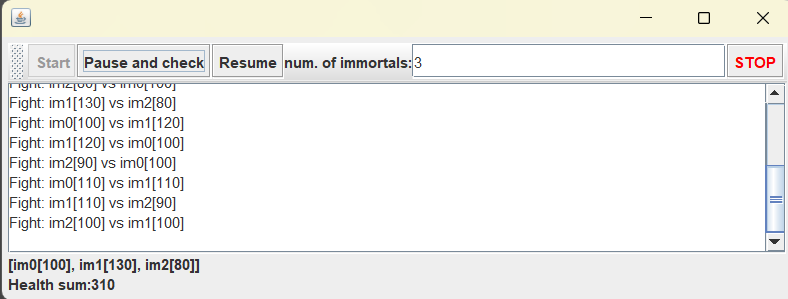
Rta/: Para un juego con N inmortales, donde cada uno comienza con 100 puntos de vida, la sumatoria invariante de los puntos de vida de todos los jugadores debería ser:

Invaritante = N \* 100

**3**. Ejecute la aplicación y verifique cómo funcionan las opción ‘pause and check’. Se cumple el invariante?.

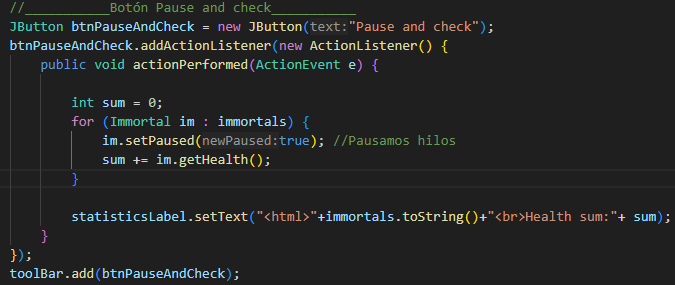
Rta/: En la ejecución de la aplicación podemos ver que no se esta cumpliendo el invariante.

La suma de la salud para este caso con N=3 inmortales, es de 310 lo cual es incorrecto ya que la suma total debe ser 300.



**4**. Una primera hipótesis para que se presente la condición de carrera para dicha función (pause and check), es que el programa consulta la lista cuyos valores va a imprimir, a la vez que otros hilos modifican sus valores. Para corregir esto, haga lo que sea necesario para que efectivamente, antes de imprimir los resultados actuales, se pausen todos los demás hilos. Adicionalmente, implemente la opción ‘resume’.

Pausar todos los hilos



Implementación botón Resume recorriendo todos los inmortales(hilos) y reanudando cada uno:

Pantalla de computadora con letras

Descripción generada automáticamente con confianza media

**5**. Verifique nuevamente el funcionamiento (haga clic muchas veces en el botón). Se cumple o no el invariante?.

Efectivamente se cumple con el invariante: Numero de hilos 3 -> invariante = 300

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

**6**. Identifique posibles regiones críticas en lo que respecta a la pelea de los inmortales. Implemente una estrategia de bloqueo que evite las condiciones de carrera.

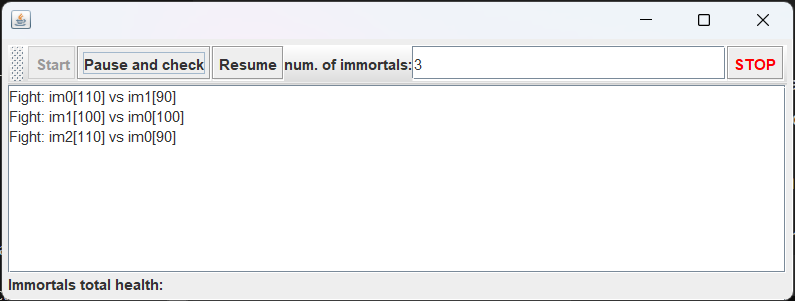
Método fight() se bloquean tanto el hilo que se le suma vida como al que se le resta:

Texto

Descripción generada automáticamente

**7**. Tras implementar su estrategia, ponga a correr su programa, y ponga atención a si éste se llega a detener. Si es así, use los programas jps y jstack para identificar por qué el programa se detuvo.

Ejecutamos nuestro programa y vemos que se detiene:



Abrimos un terminal y ejecutamos el programa jps para identificar el id de muestro proceso Java:

Texto

Descripción generada automáticamente

Una vez identificado nuestro ID de proceso ejecutamos el programa ‘jstack’ para mostrar una traza completa de los hilos que están ejecutándose en ese momento.

Se encuentra un deadlock (bloqueo mutuo), y nos dice lo siguiente:

Texto

Descripción generada automáticamente

El inmortal1(hilo) y el inmortal2(hilo) se están bloqueando mutuamente lo que provoca que el programa no avance.

**8**. Plantee una estrategia para corregir el problema antes identificado (puede revisar de nuevo las páginas 206 y 207 de *Java Concurrency in Practice*).

Estrategia de solución:

1. En cada Inmortal creamos un objeto nuevo para identificar el bloqueo del hilo de manera compartida



Captura de pantalla de computadora

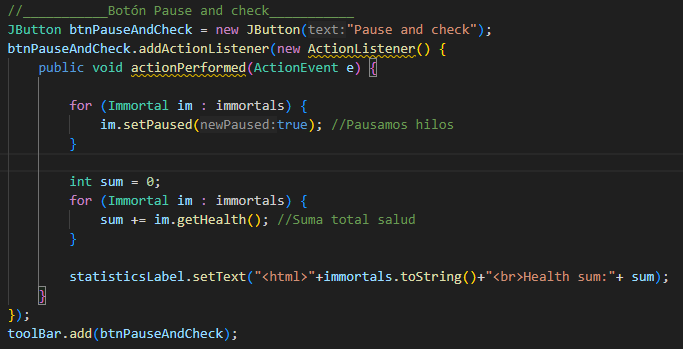
Descripción generada automáticamente

1. En botón ‘Pause and check’ controlamos el pausar de los hilos con ayuda de un método setPaused de cada hilo:

Método que setea la variable ‘paused’ para indicar si el hilo esta pausado o no. Lo llamamos en la ejecución del botón ‘Pause and check’

Texto

Descripción generada automáticamente



1. Controlamos los bloqueos de los hilos

Siempre bloqueamos primero el hilo que fue credo primero por la aplicación para que no hayan deadlocks



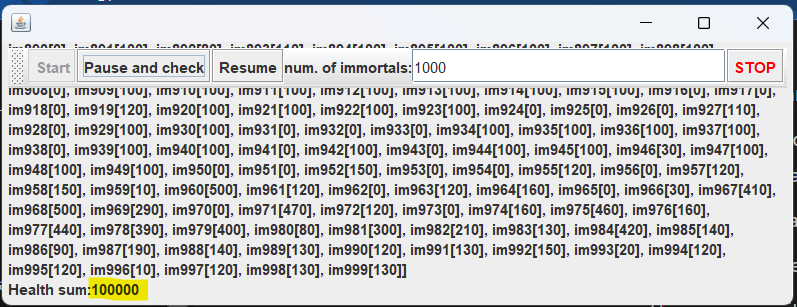
**9**. Una vez corregido el problema, rectifique que el programa siga funcionando de manera consistente cuando se ejecutan 100, 1000 o 10000 inmortales. Si en estos casos grandes se empieza a incumplir de nuevo el invariante, debe analizar lo realizado en el paso 4.

- con 100 hilos ejecución correcta.

Texto

Descripción generada automáticamente

* Con 1000 hilos ejecución correcta:



**11**. Para finalizar, implemente la opción STOP.

Ejecutamos el programa y luego damos a STOP y vemos que detiene todos los hilos:

