UM UNIVERSIDAD DE MONTEVIDEO

Entregable 2 - Concurrencia

Fecha límite de entrega: 30/09 por tarea de Moodle.
Grupo de 2 personas como máximo.: Avisar al docente en clase si cambia la conformación del grupo.
Fecha de presentación en clase: 30/09 - Deben estar los dos integrantes presentes.

Forma de entrega: Se debe subir un zip, con el nombre Entregable2Apellido1Apellido2.zip, ejemplo (Entregable2GarciaPerez.zip) que contenga los apellidos de los integrantes del grupo.
El zip debe contener:

- ☐ Código fuente
- ☐ Documentación técnica de tipo informe PDF.

Requerimientos funcionales:

Desarrolle un juego que utilice hilos de manera concurrente en lenguaje Java. El juego se desarrollará en un tablero de NxN (sugerimos N=10).
Utilizar la técnica de visualización que considere mejor para mostrar el tablero y la simulación.
A los efectos de este proyecto, **solo se jugará en modo simulación**.
Los personajes de este juego son algunos jugadores, 2 robots amigables y un robot malo.

Comienzo del juego y objetivo:

La partida no comienza hasta que haya M jugadores (sugerimos M=3 o 4 en principio), y al completarse los M jugadores, comenzará la partida. No habrán nuevas partidas hasta que termine la actual.

Los jugadores se posicionan en algún lugar aleatorio del tablero y parten con un número inicial de 2 vidas cada uno. No hay turnos, cada jugador realiza una jugada, y espera un tiempo Z aleatorio (entre Zmin y Zmax) para volver a jugar.

El objetivo del juego es obtener la mayor cantidad de monedas posible y permanecer con vida en el tiempo límite que durará la partida (T). El jugador que permanezca con vida y con más monedas al finalizar el tiempo límite ganará la partida (puede haber empates). Podría pasar también que la partida termine antes del tiempo límite T, si solo un jugador queda con vida.

En cada jugada los jugadores tiran un dado, definen el camino a recorrer por el tablero, se mueven de casilla en casilla y aplican las acciones que tenga cada casilla. Si hay casillas ocupadas por otros jugadores, no pueden pasar por ellas.

Se puede recolectar monedas o vidas, solo pasando por esa casilla, en su recorrido, no necesariamente tiene que finalizar en ella. Para ocupar una casilla deben bloquearla y si tienen un premio lo recogen.

Las casillas pueden estar ocupadas solamente por un jugador a la vez.

Robots amigables:

Existen dos robots amigables, uno coloca vidas en casillas libres del tablero, aleatoriamente. Se puede colocar hasta un máximo de X vidas en total, cuando llega a colocar las X, queda esperando a que los jugadores tomen al menos una de ellas. Cuando eso ocurre vuelve a generar vidas y las coloca en otros lugares aleatorios que estén libres.
Este robot, entre una vida y otra que coloca, espera un tiempo aleatorio entre Xmin y Xmax.

El otro robot amigable coloca monedas en lugares aleatorios que se encuentren libres en el tablero. Por cada casilla pueden haber 1, 2, 5 o 10 monedas. El máximo de casillas con monedas debe ser 10 % del total. Cuando hay 10% de casillas con monedas, el robot de las monedas descansa hasta que tenga casillas libres para volver a colocar. Cuando pasa esto, genera 1, 2, 5 o 10 monedas, y las coloca en alguna casilla aleatoria libre.
Este robot coloca monedas de forma aleatoria, esperando entre una colocación y otra, un tiempo aleatorio entre Ymin e Ymax. La cantidad de monedas colocadas se genera de forma aleatoria entre 1,2,5,10.

Robot malo:

Existe un proceso robot malo, que coloca trampas en casillas aleatoriamente.
Cuando una casilla tiene una trampa, el jugador que pase por ella perderá una vida.
El robot malo podrá poner trampas en hasta un máximo de 10% de la cantidad total de casillas del tablero. Cuando llega a este máximo termina su labor hasta el fin de la partida.
Cuando el robot malo va a colocar trampa en una casilla, la misma debe estar libre (de jugadores y de premios). Si no está libre, debe desistir e intentar colocarla en otra. Los jugadores en su estrategia, no saben dónde están las trampas.
Este robot coloca trampas, esperando entre una y otra un tiempo entre Wmin y Wmax.

Display del tablero en pantalla:

Se deberá proveer un mecanismo para actualizar la pantalla. Para implementar esto, sugerimos un hilo encargado de esta tarea, que reciba las notificaciones de las actualizaciones de movimientos de cada jugador y actualizaciones de los robots, en una cola y los vaya procesando en orden.

Terminación de la partida

Al terminar una partida, ya sea por tiempo expirado o porque ganó alguien, se deben mostrar los resultados. Es posible que hayan quedado jugadores esperando para jugar (porque se iniciaron más de M jugadores). Se podrá elegir comenzar una nueva partida con

estos jugadores o terminar la simulación. Defina lo que considere mejor respecto a este punto y justifíquelo.

Mejoras en la solución:

Cumpliendo con los requisitos pedidos, tómense la libertad de agregar todas las herramientas y mejoras que consideren al juego, tanto para hacerlo más eficiente, como más interesante y desafiante de jugar.

Requerimientos técnicos:

1. La solución debe ser modular y seguir las mejores prácticas de concurrencia en Java, incluyendo la gestión de excepciones y el uso de estructuras de datos seguras para hilos. También debe utilizar las herramientas que considere para asegurar la correcta sincronización del juego y evitar deadlocks.
2. Utilizar un mecanismo para esperar que sean M jugadores para comenzar la partida.
3. Indicar los patrones de diseño concurrente usados y justificar su elección en la documentación.
4. Debe guardarse un log de cada partida en un archivo de texto.
5. La documentación PDF debe contener como mínimo:
    a. Breve detalle de las herramientas utilizadas y justificación para la elección de las mismas (seguridad de hilos, evitación de deadlocks, etc).
    b. Decisiones de funcionamiento del juego, de diseño e implementación tomadas.
    c. Breve conclusión del producto desarrollado y del aprendizaje obtenido.