Parque de Diversiones

Programación Concurrente

alumno: Jamiro zuñiga tomas

legajo: FAI-3429

profesora: Silvia Amaro

Año de cursada:2024

Índice

Recursos Compartidos

Tipo	Nombre de la Clase
Atracciones	●**FilaAutos** ●**FilaBarco** ●**FilaComedor** ●**FilaMontaña** ●**FilaRealidadVirtual** ●**FilaTeatro** ●**FilaTren** ●**Premios**
Adicionales Para Parque	●**Parque**

Hilos

Nombre Hilo	Uso
Visitante	**usa todos los recursos compartidos**
Reloj	**usa el recurso Parque**
Asistente	**usa el recurso FilaTeatro y parque**
EncargadoPremios	**usa el recurso Premios y parque**
Encargado	**usa el recurso FilaRV**
AutosChocadores	**usa el recurso FilaAutos**
BarcoPirata	**usa el recurso FilaBarco**
MontañaRusa	**usa el recurso FilaMontaña**

AreaDePremios	**usa el recurso Premios**
Espectaculo	**usa el recurso FilaTeatro**
RealidadVirtual	**usa el recurso FilaRV**
Trenes	**usa el recurso FilaTren**

Mecanismo

RecursoCompartido	Mecanismo
FilaAutos	monitores
FilaBarco	Locks
FilaComedor	CyclicBarrier
FilaMontaña	Semáforos
FilaRealidadVirtual	Locks
FilaTeatro	Semáforos
FilaTren	BlockingQueue
premios	Exchanger
Parque	Monitores

Explicación

Constructor: Es el “main” donde se crean objetos compartidos y los hilos, también donde estos últimos se ejecutan.

Reloj: este simula el tiempo de un dia (24hs) mediante un objeto de la clase AtomicInteger para realizar operaciones atómicas sobre un valor entero y así las actualizaciones al valor sin interferencia de otros hilos.

```
horaActual.addAndGet(delta:1);
System.out.println("hora actual: "+horaActual);
switch (horaActual.get()){
    case apertura:{
        parque.abrirParque();
        Thread.sleep(millis:1000);
        break;
    }
    case cierre:{

       parque.cerrarParque();
       Thread.sleep(millis:1000);
       break;
    }
    case nuevoDia:{
       horaActual.set(newValue:0);
       System.out.println(x:"");
       System.out.println(x:"----------------------");
       System.out.println(x:"comienza un nuevo dia");
       System.out.println(x:"----------------------");
       System.out.println(x:"");
       break;
    }
    default:
        break;
}
```

si el parque abre o cierra llama a un método del objeto parque para que los visitantes puedan entrar o salir.

Parque: Los métodos ingresarParque depende del estado del parque (variable: parqueAbierto) este en true (significa que abrió) así el visitante puede ingresar al parque , el estado del parque lo cambia el “Reloj”.

Visitante: Este tiene todos los objetos compartidos con las Atracciones(hilos) , además del objeto “Parque” que comparte con el hilo “reloj” , mediante un bucle y un método que retorna un booleano:

```
while(this.parque.estadoDelParque() ){//mie
```

puede ingresar al parque si retorna “true” e intenta realizar una atracción.

Atracciones Explicación

Montaña Rusa:tiene capacidad para 5 personas y requiere que todos los asientos estén ocupados para arrancar, si el espacio de espera está llena el visitante se va a otro lado.

-el método subirMontaña()

●Sincroniza el contador de asientos ocupados
●Si hay se ocuparon los 5 asientos , los visitantes no esperan se van a otra atracción
●una vez que se ocuparon los 5 asientos ,libera la montaña rusa para que arranque con el semáforo **montaña.release();**
●usa **espera.release();** para liberar al visitante que espera

-el método arrancar()

● toma el permiso que liberó el último visitante que subió a la montaña rusa
● usa **montaña.acquire();** si se lleno arranca si no espera

-el método parar()

● avisa a los visitantes que bajen con “**visitanteSale.release(5)**” libera al visitante

-el método bajarMontaña()

● si tiene permisos “**visitanteSale.acquire()**” empieza a decrementar el contador hasta 0 y libera a los próximos visitantes con “**asientos.release(5)**”

Autos Chocadores: Una vez lleno los 10 autos con 2 personas cada uno inicia la actividad , los visitante deben esperar a que esté lleno todo para iniciar.

-el método SubirAuto()
●Sincroniza el contador de personas que ya se subieron a los autos, el primero en ingresar cambia el lógico "**finalizó**" a falso para que no salga
●Si se ocuparon los 20 asientos , deben esperar (**this.await()**) y cambia el lógico "inicio" a verdadero para que arranque la atracción.
-el método arrancarAutosChocadores()
●si no se llenaron los 20 asientos de los autos espera "this.await()"
●se llenaron los asientos inicia atracción

-el método detenerAutosChocadores()
●cambia el estado de los lógicos **"finalizó"** a true para que los visitantes bajen de los autos y el **"inicio"** a falso para que no arranque la atracción si no se llenaron los 20 asientos
-el método BajarAuto()
●si el logico "**finalizó**" es verdadero los visitantes bajan de los autos y decrementa el contador a 0 para notificar a los nuevos visitantes que suban

Barco Pirata: *cuando se llena con 20 personas el viaje comienza o cuando pasa determinado tiempo de espera,los demás esperan.*
-el método SubirBarco()
●Sincroniza los contadores de espera y de ingres, booleanos para iniciar viaje y tiempo de espera para iniciar viaje

●Si el barco se llenó cambia el lógico "**finalizó**" a verdadero y notifica al lock (**llegada.signalAll()**) para que inicie el viaje el barco

●si el barco partió o hay gente esperando y el barco está lleno esperan los visitante (salida.await())

-el método arrancarBarco()
●sincroniza el boolean **llenoATiempo** y **partioBarco**
●si el barco está vacío espera ,sino cuenta n tiempo para que se llene o arranca incompleto

-el método detenerBarco()
●sincroniza las variables lógicas:
-**finalizó** a falso, para que no inicie otro viaje hasta que se suban los 20 visitantes
-**espera** a verdadero, para que los visitantes bajen del barco
-**unPasajero** a falso, que espere hasta que suba alguien y vuelva a contar el

tiempo
-**partioBarco** a falso, notifica a los visitantes que no está el barco en viaje

● notifica a los visitantes que bajen del barco con **salida.signallAll()**

-el método BajarBarco()

●sincroniza el contador de espacio (decrementa)
●si ya está vacío el barco cambia el booleano de espera a falso y notifica a los visitantes que suban al barco con la condición **salida.signallAll()**

Comedor:se sientan en grupos 4 en mesas , pero cuando se llena una mesa de 4 recién comienzan a comer al mismo tiempo.

-el método entrar Comedor()

●sincroniza el contador de personas en el comedor
●si está lleno el comedor espera (**this.wait()**)

-el método BuscarMesa()

●sincroniza las mesasOcupadas[],asientosOcupados[] y mesas[], mientras busca una mesa disponible.
●si el asientosOcupados[i]=4 entonces mesasOcupadas[i]=verdadero ,osea si la mesa está llena cambia el estado a true para que puedan comer.
●si se sentó un visitante en alguna mesa espera 5 minutos hasta que se llene para comer si no comen los que estén

-el método DejarMesa()

●sincroniza el asientosOcupados[i] osea la mesa donde se sentó
●cambia el estado de la mesa de llena a con espacio para que se siente otro.

-el método SalirComedor()

●sincroniza la salida de personas (contador) decrementando y notifica que salió.

Realidad Virtual: En esta actividad los visitantes le piden al encargado que les de el equipo(visorRV, 2 manoplas, una base) , una vez con todo el equipo (completo) pueden entrar.

-el método entrarRV()

●sincroniza la salida de personas (contador) decrementando y notifica que salió

-el método SalirRV()
●sincroniza la salida de personas (contador) decrementando y notifica que salió
-el método SalirComedor()
●sincroniza la salida de personas (contador) decrementando y notifica que salió
-el método SalirComedor()
●sincroniza la salida de personas (contador) decrementando y notifica que salió

Tren:los visitantes se colocan en una cola de espera ,el tren tiene capacidad para 10 personas y parte una vez que la fila está llena o pasaron 5 minutos de espera.
-el método SubirTren()
●sincroniza la cola de espera de forma que se suban al tren de forma ordenada
●si se subio al menos un visitante al tren este arranca el conteo de espera(**arrancarTren.take()**)

-el método arrancarTren()
●sincroniza la salida del tren con respecto a visitantes en caso de que no subieron en el tiempo de espera o no entraron por espacio

-el método frenarTren()
●avisa a los 10 pasajeros que puede bajar (esperaPasajero.put(1))

-el método abajarTren()
●sincroniza el descenso de visitantes del tren y avisa a los visitantes de la cola de espera que suban

Premios: El visitante intercambia fichas con el encargado del área para poder jugar, después de jugar obtiene un premio según la cantidad de puntos .
-el método pedirFicha()
●sincroniza los visitantes al pedir una ficha al encargado
-el método darFicha()
●sincroniza al encargado entregando una ficha al visitante
-el método jugarJuegos()
● retorna un número entero random entre 1 y 10000 (puntos)
-el método recibirPremio()
● Según la cantidad de puntos que hizo obtiene un premio

Teatro:de vez en cuando hay un espectáculo cuando este pasa ingresan n asistentes en grupos completos además de ingresar en grupos de 5 visitantes este teatro cuenta con una capacidad máxima de 20.

-el método ingresarAsistente()
● sincroniza la entrada del grupo de asistentes al teatro
-el método ingresaVisitante()
● Sincroniza la entrada de los grupos de 5 visitantes al teatro

-el método inicieLaFuncion()
● si ingresaron los 20 visitantes y el grupo de asistentes inicia la función

-el método bajarTelon()
● avisa a los visitantes y asistentes que pueden salir
-el método saleAsistente()
● los asistentes salen a tomar un descanso hasta la próxima función
-el método saleVisitante()
● la salida de los visitantes hasta que esté vacío el teatro y avisa que pueden entrar nuevos visitantes