Trabajo practico final

Sistema de aeropuerto

Programación concurrente – 2 ° año - LSI

Alexis Leonel, Bustamante Hecht – FAI-2355

Sistema de aeropuerto

El siguiente trabajo tiene por objetivo simular el funcionamiento del Aeropuerto "VIAJE BONITO". El cual inicia en el momento en que el pasajero llega al Aeropuerto para tomar su vuelo y termina en cuanto sube al avión. El aeropuerto tiene siempre sus puertas abiertas, pero atiende a los pasajeros solo de 6.00 a 22.00 hs.

Hay n aerolíneas, coexistiendo en dicho aeropuerto (por ejemplo, Aerolíneas Argentinas, LAN, etc.), cada una de ellas con su puesto de atención a pasajeros.

Los pasajeros tienen reserva en un vuelo de una de las aerolíneas del aeropuerto. Cuando un pasajero ingresa al aeropuerto, este llega a un puesto de informes donde es atendido y desde allí es derivado al puesto de atención que corresponda según su vuelo y aerolínea para hacer el check-in. Cada puesto de atención tiene lugar para una cantidad máxima de pasajeros. Los cuales son atendidos por orden de llegada, aquellos que lleguen después de alcanzado el límite de personas, esperan en un hall central hasta que se haga lugar. Además, en los puestos, hay un guardia que se encarga de dar paso a los pasajeros que llegan a medida que se va desocupando el lugar. No interesa simular el tema de las reservas en una aerolínea particular por lo que la misma, se generará de forma aleatoria al llegar al aeropuerto.

Además, en el aeropuerto hay varias terminales. Cada terminal tiene varios puestos de embarque y una sala de embarque compartida. Cuando un pasajero hace el check-in se le indica la terminal y puesto de embarque que corresponde a su vuelo. Entonces, los pasajeros que ya hicieron el check-in son trasladados a la terminal que les corresponde y permanecen allí hasta que se haga el llamado para embarcar. En este caso, el aeropuerto "VIAJE BONITO" tiene 3 terminales: A, B y C. En la terminal A están los puestos de embarque 1 a 7, en la terminal B los puestos del 8 al 15 y en la terminal C del 16 al 20.

Por otro lado, en cada terminal hay un free-shop en el que los pasajeros que esperan para embarcar pueden hacer compras o solo mirar los productos que se ofrecen. Por una cuestión de organización cada free-shop tiene una capacidad limitada de personas, es decir que, si ya se alcanzó el límite, no podrán ingresar más personas hasta que alguna abandone el free shop y deje el lugar. Además, hay una puerta de ingreso y una puerta de egreso y 2 cajas cercanas a la puerta de egreso.

Al llegar a la terminal, los pasajeros pueden ir al free-shop o simplemente sentarse a esperar el llamado para embarcar en la sala de embarque general. Es importante tener en cuenta que solo pueden ir al free-shop mientras tengan el tiempo suficiente antes de su hora de embarque.

En el aeropuerto hay un “people mover/transporteATerminal”, el cual es similar a un pequeño tren interno que se utiliza para trasladar a los pasajeros hasta las distintas terminales. Este se mueve entre terminales a través de una vía. El tren se detiene en cada terminal si es que algún pasajero lo solicita. Al llegar a la última terminal el tren debe quedar vacío y vuelve al punto de inicio de su trayecto (o sea el ingreso al aeropuerto), allí espera hasta tener su capacidad completa para comenzar un nuevo recorrido.

Debe resolverse utilizando los mecanismos de sincronización vistos en la materia y provistos por el lenguaje: semáforos y monitores (obligatoriamente), Locks, CyclicBarrier, CountDownLatch, Exchanger., implementacione s de BlockingQueue.

# Desarrollo

Enlace a repositorio de GitHub para descargar el sistema: <https://github.com/LeonelBustamante/Trabajo-practico-final-PC>

El desarrollo de este proyecto se realizó mediante un programa en Java controlando la concurrencia de los procesos que fueron parte del sistema. Donde se utilizan métodos de sincronización vistos durante el cursado de la materia, como lo son los que se ven a continuación:

Sincronización carrera:

* Métodos y bloques sincronizados: son métodos o bloques de código en los que solo puede haber un hilo ejecutándose a la vez. Esto se logra mediante la adquisición de una "cerradura" antes de entrar en el método y la liberación de la misma después de salir.
  + La ventaja de esto es que se garantiza la sincronización en el acceso a recursos compartidos, evitando conflictos y errores.
  + Como desventaja puede aumentar la latencia y disminuir la eficiencia si hay una gran cantidad de hilos esperando para acceder al método o bloque sincronizado.
* Semáforos binarios (mutex): un tipo de semáforo que solo puede tener dos valores: bloqueado o desbloqueado. Durante la utilización de este solamente un proceso puede tomar el control del semáforo.
  + La ventaja es que es una forma sencilla y eficiente de sincronizar el acceso a un recurso compartido.

Sincronización por cooperación:

* Semáforos genéricos: pueden tener cualquier número de permisos y se utilizan para controlar el acceso a múltiples recursos compartidos.
  + La ventaja es que permiten un control más específico para la concurrencia, pudiendo tratar con más de un proceso a la vez.
* Locks: mecanismo de sincronización que permite a un hilo bloquear el acceso a un recurso compartido hasta que se libere.
  + La ventaja es que proporcionan un control preciso sobre el acceso a los recursos compartidos.
* Monitores: proporciona un mecanismo de sincronización a través de la adquisición y liberación de cerrojos. Además, incluyen un conjunto de variables condicionales que permiten a los hilos esperar hasta que se cumplan ciertas condiciones antes de continuar.
  + La ventaja de los monitores es que proporcionan un control más preciso sobre la sincronización que los cerrojos solos.
* Condición: objetos que permiten a los hilos esperar y ser notificados cuando se cumplan ciertas condiciones. Están asociados con cerrojos y se utilizan en combinación con los monitores para proporcionar un mecanismo de sincronización más avanzado.
  + La ventaja es que permiten a los hilos esperar hasta que se cumplan ciertas condiciones específicas antes de continuar.

Se controló que las propiedades de la programación concurrente se cumplan para que todos los procesos progresen hasta su finalización (liveness) y que las secciones donde la información debe ser trabajada controladamente (sección crítica) esté siendo tratada con mecanismos de sincronización acorde.

En el programa se trabajó de la siguiente manera:

ControlTren:

Este código representa la clase ControlTren que es utilizada para controlar el funcionamiento del tren, la clase tiene una capacidad máxima y una cantidad de terminales y utiliza semáforos para controlar el acceso de los pasajeros y el tren.

Métodos para los pasajeros:

* subirATren y bajarDeTren

métodos para el tren como “comenzarViaje” y “terminarViaje” cada método utiliza los semáforos para controlar el acceso a los recursos y garantizar la sincronización.

El tipo de sincronización utilizada en este código es la sincronización por cooperación basada en semáforos genéricos. Los semáforos se utilizan para controlar el acceso a los recursos compartidos y garantizar que los pasajeros y el tren tengan acceso exclusivo a los recursos cuando es necesario.

PuestoAtencion:  
El código representa un puesto de atención en un aeropuerto en el que los pasajeros pueden entrar a una fila y esperar a ser atendidos por un/a recepcionista. El puesto de atención solo puede atender a 2 pasajeros a la vez.

La clase PuestoAtencion tiene los siguientes métodos para los pasajeros:

* entrarFilaPuestoAtencion: permite que el pasajero entre a la fila del puesto de atención, pero si la cantidad de pasajeros en la fila es igual a 2, el pasajero debe esperar en la sala de espera.
* entrarPuestoAtencion: permite que el pasajero entre al puesto de atención.
* salirPuestoAtencion: permite que el pasajero salga del puesto de atención.

Y tiene los siguientes métodos para el recepcionista:

* hacerPasarPasajero: permite que el recepcionista haga pasar al primer pasajero de la fila.
* atenderPasajero: permite que la recepcionista atienda al pasajero.

Se utiliza la clase ReentrantLock y Condition para garantizar la exclusión mutua y la sincronización entre los threads de los pasajeros y el recepcionista.

Reloj:

Clase que representa el reloj de un aeropuerto. El reloj controla el avance del tiempo y notifica a las terminales y al repositor sobre cambios en la hora actual. Además, utiliza la clase AtomicInteger para almacenar la hora actual, lo que significa que se asegura de que las operaciones de lectura y escritura en la hora sean atómicas, es decir, seguras para ser utilizadas en un entorno concurrente.

Tren:

Es un hilo que simula el funcionamiento de un tren que viaja entre diferentes terminales. La clase se implementa con la interfaz Runnable.

Utiliza la sincronización por Semáforos (genéricos y binarios), los cuales son utilizados para controlar el acceso a los recursos compartidos y la detención del tren en cada terminal. Cada terminal es representada por un semáforo, que se inicia con 0 permisos y acumula uno por cada vez que el pasajero le indique que desea bajar en esta terminal.

La elección de utilizar semáforos se debe a que se desea controlar el acceso de los hilos (pasajeros) a un recurso compartido (el tren), para así garantizar que sólo un hilo a la vez pueda alterar los valores que controlan este recurso.

Tienda:

Esta representa una tienda en un aeropuerto. La tienda tiene un máximo de personas que pueden entrar al mismo tiempo (definido por MAX\_PERSONAS) y una lista de n productos (productos) y cajas.

* obtenerCaja: permite a un pasajero obtener una caja disponible de forma aleatoria
* entrarTienda: permite a un pasajero entrar a la tienda, pero si esta supera el máximo permitido de personas, el pasajero tendrá que esperar hasta que se libere un lugar.
* seleccionarProductos: permite a un pasajero seleccionar una cantidad aleatoria de productos y agregarlos a su carrito. El acceso a los productos está dentro de un bloque sincronizado para que no se altere este valor por otros pasajeros mientras realizan sus compras.

Terminal:

La clase Terminal contiene varios métodos para interactuar con Pasajeros y un Reloj.

El método "esperarVuelo" se utiliza para que un Pasajero espere en la Terminal hasta que llegue la hora de su vuelo, esto lo vemos por el metodo await. El método "pasarHora" permite notificar a todos los Pasajeros que están esperando en la Terminal mediante notifyAll.

Ambos métodos son monitores ya que es utilizada la palabra reservada "synchronized" y a su vez wait y notify dentro de estos, lo que significa que sólo un hilo puede acceder a ellos a la vez y bloquearse.

Aeropuerto:

Incluye métodos para el ingreso de pasajeros al aeropuerto, el comienzo y el fin del horario de atención del aeropuerto.

El método "ingresarAeropuerto" es sincronizado, lo que significa que solo un pasajero puede ingresar al aeropuerto a la vez. Si el aeropuerto aún no está abierto, el pasajero esperará hasta que el reloj notifique que el aeropuerto abrió, si el aeropuerto está abierto el pasajero se dirige al puesto de atención.

Los métodos "comenzarHorarioAtencion" y "terminarHorarioAtencion" también están sincronizados y son responsables de notificar a todos los pasajeros cuando el aeropuerto comienza y termina su horario de atención.