Programación Concurrente 2024

Trabajo Práctico Nº 1

Condiciones

- El trabajo es grupal, de 4-5 alumnos (3 o 6 es una excepción)
- La defensa del trabajo es con el grupo completo
- La evaluación es individual (hay una calificación particular para cada integrante)
- Solo se corrigen los trabajos que hayan sido subidos al aula virtual (LEV)
- Los problemas de concurrencia deben estar correctamente resueltos y explicados.
- El trabajo debe implementarse en lenguaje Java.
- Se evaluará la utilización de objetos y colecciones, como así también la explicación de los conceptos relacionados a la programación concurrente.

Enunciado

En un sistema de reservas de vuelos, se necesita implementar una funcionalidad para gestionar las reservas de asientos de manera concurrente. Esta funcionalidad consta de varios procesos que deben ejecutarse de forma simultánea, teniendo en cuenta los siguientes lineamientos de diseño.

El sistema debe manejar la reserva de asientos de un avión, representado por una matriz de asientos donde cada asiento puede estar ocupado, libre o descartado.
Además, se deben mantener cuatro registros distintos para las reservas:

- Reservas pendientes de pago.
- Reservas confirmadas.
- Reservas canceladas.
- Reservas verificadas.

El funcionamiento del sistema posee cuatro etapas::
Proceso de Reserva: Este proceso se encarga de recibir las solicitudes de reserva de los usuarios. Se tienen tres hilos que ejecutan este proceso. Cada hilo intenta reservar un asiento aleatorio en la matriz, verificando que esté disponible. Si el asiento no está libre, el hilo debe buscar otro asiento que sí lo esté. Una vez reservado el asiento, el mismo se marca como ocupado y se registra la reserva pendiente en el registro de **reservas pendientes**.

Proceso de Pago: Este proceso es ejecutado por dos hilos, y se encarga de verificar el pago de las reservas pendientes. Cada hilo toma una reserva aleatoria del registro de reservas pendientes y realiza una verificación de pago. Se establece una probabilidad del 90% de que el pago sea aprobado y un 10% de que sea rechazado. Si el pago es aprobado,

la reserva se elimina del registro de pendientes, y se agrega al registro de **reservas confirmadas**; de lo contrario, el asiento pasa a estado descartado mientras que la reserva se marca como cancelada, se elimina del registro de pendientes y se agrega al registro de **reservas canceladas**.

Proceso de Cancelación/Validación: Si un usuario decide cancelar su reserva, se pasa al proceso de cancelación. Tres hilos se encargan de cancelar las reservas. Cada hilo selecciona una reserva aleatoria de las reservas confirmadas y la cancela con una probabilidad del 10%. Si la reserva es cancelada, se elimina del registro de reservas confirmadas y se agrega al registro de **reservas canceladas** mientras que el asiento pasa a estado descartado. Si la reserva no es cancelada, la misma se marca como "checked".

Proceso de Verificación: Al finalizar la ejecución, se debe verificar el estado final de la matriz de asientos y los registros de reservas para asegurar que las operaciones se hayan realizado correctamente. Este proceso selecciona de manera aleatoria una reserva del registro de reservas confirmadas. Para cada reserva marcada como “checked”, se debe eliminar del registro de reservas confirmadas y se debe insertar en el registro de **reservas verificadas**. Este proceso es ejecutado por dos hilos.

Consideraciones:

- ☐ Cada proceso tiene una demora fija por proceso (por cada iteración), pero distinta entre procesos, a ser definida por el equipo.
- ☐ Cada asiento debe ser accesible por un solo hilo a la vez para evitar conflictos de reserva simultánea.
- ☐ Cada reserva debe ser revisada por un solo hilo a la vez (independientemente del proceso).
- ☐ Cada reserva puede ser aprobada, confirmada, cancelada o verificada solo una vez.
- ☐ Los procesos de reserva, pago, cancelación y verificación deben ejecutarse de forma concurrente para simular un entorno de reservas realista.
- ☐ Los tiempos de espera para realizar cada operación deben ser aleatorios y configurables por el grupo.
- ☐ Al iniciar el programa, todos los hilos deben ser lanzados para que comiencen su ejecución.

El sistema debe contar con un LOG con fines estadísticos, el cual registre cada 200 milisegundos en un archivo:

- Cantidad de reservas canceladas.
- Cantidad de reservas aprobadas (verificadas).

Además, al finalizar, el log debe imprimir la ocupación final del vuelo y el tiempo total que demoró el programa.

Ejercicios

1) Hacer un diagrama de clases que modele el sistema de datos con TODOS los actores y partes.
2) Hacer un diagrama de secuencias que modele las interacciones del sistema.
3) El vuelo a modelar tiene una capacidad de 186 pasajeros.
4) Se deben configurar los tiempos de los procesos de modo tal que el programa no demore más de 45 segundos.
5) Hacer un análisis analítico detallado de los tiempos que el programa demora. Luego contrastarlo con múltiples ejecuciones obteniendo las conclusiones pertinentes.
6) El grupo debe poder explicar los motivos de los resultados obtenidos. Y los tiempos del sistema.
7) Debe haber una clase Main que al correrla, inicie los hilos.

Entregar:

a) 1 archivo de imágen con el diagrama de clases en buena calidad.
b) 1 archivo de imágen con el diagrama de secuencias en buena calidad.
c) El código fuente Java (proyecto) de la resolución del ejercicio. El proyecto debe incluir librerías y extensiones necesarias, y debe poder correr en cualquier máquina independientemente del sistema operativo o IDE utilizada.
d) 1 log y la explicación de los resultados.
e) **Un informe obligatorio**, en formato pdf, con estructura formal (carátula, integrantes, desarrollo, etc.), donde se detalle:
    i) Todo el trabajo realizado.
    ii) Decisiones de diseño tomadas y su justificación (tiempos, etc)
    iii) Las conclusiones obtenidas en base a los resultados
    iv) Pueden incluir cualquier otra explicación que crean pertinente para el trabajo

Subir al LEV el trabajo **TODOS** los participantes del grupo.

Fecha de entrega

Hasta Lunes 22 de Abril de 2024 - 23:59 hs.

Fecha de defensa

A coordinar por grupo desde el Martes 23 de Abril de 2024.