Introducción a los Sistemas Distribuidos y Paralelos

Trabajo Final

- Simulador de Gestión para Centros Comerciales -

Alumno: Gonzalo Ezequiel Bravo

Docente a Cargo: Walter Fabián Aguero

Ingeniería en Computación. Escuela de Producción y Tecnología UNRN

Año 2025

Indice

1. Introducción

El presente trabajo consiste en la realización de un programa concurrente multihilo. El contexto elegido es el de un Centro Comercial que cuenta con un control de acceso a diferentes comercios, un cine, un patio de comidas y un estacionamiento.

Las diferentes áreas del **Centro Comercial** serán tomadas como recursos compartidos con capacidad limitada, lo que permite la implementación de conceptos de programación concurrente como semáforos, monitores y sincronización:

- **Comercios**: Cada tienda tiene una capacidad máxima de clientes. Los clientes deben esperar para acceder a una tienda si está llena.
- **Cine**: la sala de cine solo inicia la película cuando se llena la capacidad y, cuando esta finaliza, todos los espectadores salen de la sala al mismo tiempo.
- **Patio de Comidas**: Tiene un número limitado de mesas. Los clientes deben esperar a que una mesa esté disponible para sentarse a comer.
- **Estacionamiento**: Con un número limitado de espacios para estacionar, los vehículos deben esperar a que un lugar se desocupe.

- El problema involucra:

  • Múltiples personas independientes (hilos en ejecución) con decisiones propias (se comportan de manera aleatoria), y vehículos (hilos en ejecución).

  • Recursos limitados (capacidad del centro comercial, capacidad de acceso a comercios y sala de cine, número limitado de mesas y espacios para estacionar los vehículos).

  • Coordinación entre hilos (por ejemplo en el caso del acomodador del cine, el cual también será tratado como un hilo en ejecución).

  • Tiempos de espera activos y pasivos (esperar a que se liberen lugares/recursos).

  • Posibles condiciones de carrera (que se pueden evitar mediante sincronización).

2. Funcionamiento del software

El sistema cuenta con las siguientes restricciones:

El **centro comercial** multihilo tendrá una capacidad limitada de 100 personas. Cuando la capacidad esté llena, las personas tendrán que esperar afuera, hasta que salga por lo menos 1 persona para poder entrar.

Dentro del centro comercial, hay cuatro comercios (tienda de ropa, juguetería, tienda de electrónica y tienda deportiva), un cine, un estacionamiento y un patio de comidas.

Cuando una persona (hilo) ingresa al centro comercial, decide aleatoriamente a que lugar va a ir (comercio, cine, patio de comidas) y al finalizar su visita a cada espacio, elige otro lugar al cual va a dirigirse, sin considerar los lugares ya visitados…..o puede retirarse del centro comercial. Si la persona ya visitó todos los lugares, sólo puede retirarse del centro comercial.

Cada **comercio** (tienda de ropa, juguetería, tienda de electrónica y tienda deportiva), tiene una capacidad limitada de 20 personas. Una vez que la capacidad de ingreso al comercio se completa,

las personas que quieran ingresar, deben esperar hasta que por lo menos 1 persona salga del comercio.

El **cine** tiene una capacidad limitada de 30 personas y está presente un acomodador, quien se encargará de verificar cuando la sala esté llena para que se inicie la película, la cual tiene un tiempo de duración establecido. Cuando la película termina, todas las personas salen al mismo tiempo del cine. Se gestiona la lógica de acceso a la sala de cine y el ciclo de películas.

El **patio de comidas** tiene un número limitado de 30 mesas. Los clientes esperan a que una mesa esté disponible para sentarse a comer, el acceso a las mesas estará controlado por semáforos. Las personas se quedarán en el patio de comidas durante un tiempo aleatorio.

El **estacionamiento** cuenta con un número limitado de 50 espacios para vehículos. Los vehículos deben esperar a que un lugar se desocupe para estacionar.

El programa finalizará cuando se reciba la señal TERM, simulando el cierre del centro comercial, en este momento se esperará a que los hilos Persona, Vehículo y Acomodador finalicen sus tareas y a continuación se mostrará un informe por pantalla con las estadísticas finales de visitas, la cantidad de personas que visitó cada espacio del centro comercial y vehículos que ocuparon lugares en el estacionamiento.

3. Diagrama de clases UML

Para el desarrollo de este programa se eligió un modelo orientado a objetos. A continuación se presentan un diagrama de clases UML, que da una visión de cómo las clases se relacionan por asociación, herencia, composición, etc.

- «create» para los constructores

- Relaciones de asociación y generalización (herencia) e interfaces

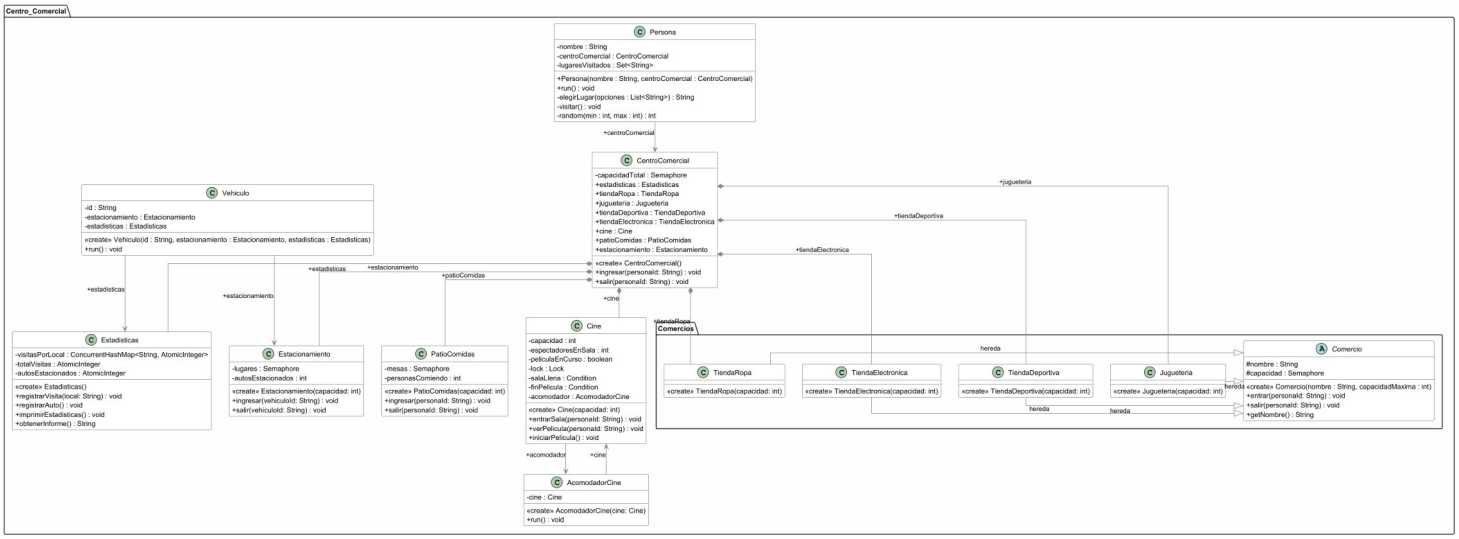
- Roles, navegabilidad

- Visibilidad: (publica(+), protegida(-) o privada(#)

- Asociación entre clases y direcciones de las flechas

- Relaciones de agregación (rombos blancos o negros)

- Relaciones de generalización, composición y asociación



4. Estructura del software – Clases

4.1 Estructura de clases

Centro_Comercial/

├── Main.java

├── CentroComercial.java

├── Persona.java

├── Vehiculo.java

├── Acomodador.java

├── Comercio.java (clase abstracta o interfaz)

├── Cine.java

├── PatioComidas.java

├── Estacionamiento.java

├── Estadisticas.java

└── comercios/

  ├── TiendaRopa.java

  ├── Jugueteria.java

  ├── TiendaElectronica.java

  └── TiendaDeportiva.java

4.2 Diseño de clases

Main.java es el programa principal donde se crea el centro comercial y se inician los hilos de Personas y Vehículos.

Funciona como punto de entrada del sistema del centro comercial. Simula la concurrencia de personas, vehículos y el control del cine con múltiples hilos. Además, maneja un cierre ordenado.

CentroComercial.java representa un centro comercial que gestiona la capacidad total de visitantes y contiene distintas tiendas, un cine, un patio de comidas y un estacionamiento. Se instancian todas las clases y se controla la capacidad total mediante variables semáforo. Contiene métodos que controlan la entrada y salida de personas del Centro Comercial.

Persona.java extiende de Thread y representa la lógica de visita aleatoria al centro comercial. Representa una persona que visita el centro comercial, recorriendo distintos locales de manera aleatoria. Evita repetir lugares ya visitados. Registra visitas en Estadísticas.

Vehiculo.java extiende de Thread y representa un vehículo que ingresa al estacionamiento del centro comercial y permanece allí un tiempo determinado. Controla el comportamiento del vehículo al ser iniciado como hilo y simula el ingreso al estacionamiento, la permanencia por un tiempo aleatorio, y su posterior salida. Registra los vehículos en Estadísticas.

Comercio.java clase abstracta que representa un comercio genérico dentro del centro comercial con control de capacidad. Controla la capacidad máxima permitida dentro del comercio con un semáforo y contiene los métodos que permiten a las personas entrar o salir de las tiendas.

Juguetería.java, TiendaDeportiva.java, TiendaElectrónica.java y TiendaRopa.java, representan las tiendas dentro del centro comercial y heredan sus atributos y métodos de la clase abstracta Comercio.java.

PatioComidas.java representa el patio de comidas del centro comercial, con una cantidad limitada de mesas disponibles. Controla el ingreso y salida de personas utilizando un semáforo para manejar la concurrencia. Controla el ingreso y salida de personas con synchronized.

Cine.java representa una sala de cine en la que se proyectan películas solo cuando la sala está llena. Utiliza locks y condiciones para sincronizar el acceso entre múltiples hilos (personas y acomodador), manejar la sala llena, película en curso, y sincronización general. Implementación de un monitor.

Acomodador.java extiende de Thread. Es el hilo encargado de controlar el ciclo de la película en el cine. Espera que la sala se llene, inicia la película, espera que termine y luego reinicia el proceso.

Estacionamiento.java representa el estacionamiento del centro comercial con una capacidad limitada de lugares. Utiliza un semáforo para controlar el acceso concurrente a los espacios disponibles. Control de concurrencia con Semaphore y contadores protegidos con synchronized.

Estadisticas.java lleva un registro concurrente de las visitas a los locales del centro comercial y de los autos que se estacionan. Maneja los contadores y contiene métodos para mostrar informes finales por pantalla.

5. Desarrollo del software

El objetivo de este trabajo es programar una solución al problema antes planteado, utilizando semáforos y/o monitores en el lenguaje de programación Java, utilizando las librerías y funciones que provee el lenguaje para programación concurrente. La solución de software debe modelar a las personas, los vehículos y el acomodador, como hilos. Sincronizando la operatoria antes descripta, el comportamiento simultáneo de visitantes y recursos, controlar el acceso seguro a recursos compartidos y garantizar restricciones como la capacidad máxima de los comercios (tiendas, cine, patio de comidas y el estacionamiento) o la secuencia de eventos del cine por ejemplo.

5.1 Programación concurrente con Java

Para el desarrollo se utilizaron la librería Tread y la API concurrent de Java. Las clases Persona, Vehículo y AcomodadorCine, heredan de Thread. También se usa la API java.util.concurrent con Semaphore, Lock, Condition para implementar monitores.

5.2 Multihilo, asincrónicos o sincrónicos

Es un programa multihilo, los cuáles se crean directamente. En la mayor parte del programa, los hilos son independientes y operan de forma asincrónica.

Se puede dar sincronismo en algunos casos como en el cine o comercios que simulan sincronismo.

5.3 Comunicación y Sincronización

Los conceptos de sincronización y comunicación, se desarrollan con Semaphore para controlar la capacidad. Lock y Condition en el Cine.

5.4 Recursos compartidos y sincronización

Las tiendas y el cine son recursos compartidos. Sincroniza el acceso con Semaphore (comercios) y Lock/Condition (cine).

5.5 Patrones de soluciones utilizados

Productor-consumidor: implícito en el uso del Cine (personas esperan para ver la película cuando se llene la sala).

Cliente-servidor: el modelo del CentroComercial con hilos de personas/vehículos actuando como clientes y los recursos como servidores.

Barreras: el Cine simula una barrera con Condition, esperando a que la sala se llene.

5.6 Comunicación por pase de mensajes

Se usa **memoria compartida** mediante variables y estructuras comunes accedidas por múltiples hilos (estadisticas, CentroComercial, capacidad, etc.).

5.7 Sincronización por exclusión mutua o por condición

Exclusión mutua con Semaphore y Lock.

Condiciones (Condition) en el Cine.

5.8 Riesgos estructurales, dependencia de datos o de control

Dependencia de datos: No es evidente, pero podría haberla si múltiples hilos modificaran estructuras sin protección.

Dependencia de control: Existe cierta lógica secuencial en los pasos que siguen los hilos.

Riesgos estructurales: No se manifiestan directamente en el código actual, pero podrían surgir si se agregaran más tipos de sincronización o comunicación.

5.9 Inanición, overloading, deadlock

Inanición: Podría ocurrir si muchas personas saturan continuamente ciertas tiendas.

Overloading: Podría pasar si se crean más hilos que la capacidad total.

Deadlock: No hay evidencia directa, pero el Cine podría tener riesgo si no se manejan bien las condiciones.

5.10 Secciones críticas y condiciones de carrera

Las secciones protegidas por semáforos o Lock son críticas.

Condiciones de carrera: No se observan directamente, ya que se usan mecanismos de sincronización adecuados. Pero si se accede a Estadísticas sin protección, podrían aparecer.

5.11 Semáforos y Monitores

En la mayor parte del programa se utilizan semáforos para controlar el acceso a comercios y la capacidad total del centro comercial.

En el cine se utiliza Lock y Condition, lo que equivale al uso de monitores.

6. Interfaz Gráfica

La interfaz gráfica simula la gestión de un centro comercial. Fue desarrollada con Java Swing para la GUI. Hay dos clases principales: SimuladorApp (punto de entrada) y Simulador (interfaz y lógica de simulación). Se redirige la salida estándar (System.out) a una JTextArea, lo que facilita mostrar mensajes en pantalla sin consola externa.

Al presionar el botón "Iniciar Aplicación" se inicia el programa.

Al presionar el botón "Finalizar Aplicación" finaliza el programa y se muestran las estadísticas.

7. Análisis en tiempo de ejecución

Ingreso y comportamiento de personas:

Luego de la ejecución del programa principal se obtuvieron las siguientes conclusiones:
-El sistema simula la entrada de personas al centro comercial y su posterior distribución entre distintos sectores (tiendas, cine, patio de comidas).

-Se observa concurrencia en el ingreso: múltiples personas ingresan simultáneamente o en rápida sucesión.

-Algunas personas realizan múltiples actividades (entran, van a tiendas, comen, y salen), lo que indica que los hilos asociados manejan varias etapas del ciclo de vida de cada visitante.

Ocupación de recursos:

-Se registran entradas y salidas del estacionamiento.

-Ocupación y liberación de mesas del patio de comidas

-Entrada progresiva al cine, tiempos de espera y liberación de asientos.

-Distribución de visitas equilibrada en los diferentes comercios, lo que sugiere una correcta sincronización y control de capacidad en cada tienda.

Sincronización y finalización:

-Se recibe la señal TERM que activa una rutina ordenada de cierre del sistema. Todos los hilos realizan una salida limpia.

-El acomodador también finaliza sus tareas, lo que indica que los hilos están correctamente sincronizados.

Concurrencia:

-La salida refleja ejecución concurrente y simultánea de muchos hilos, lo que es esperable en un entorno simulado de este tipo. No se notan errores ni colisiones en el acceso a recursos, lo que indica el funcionamiento de semáforos y monitores.

Salida controlada:

-El proceso termina con código 130, lo cual es estándar para terminación por SIGINT o SIGTERM. Esto sugiere que el programa maneja adecuadamente señales externas de cierre y finaliza de manera segura.

8. Conclusiones finales

El sistema simulado del centro comercial parece estar funcionando correctamente. Maneja de forma segura:

- El ingreso/egreso concurrente de múltiples personas (hilos).
- El acceso a recursos compartidos como el estacionamiento, las tiendas, el cine y las mesas del patio de comidas.
- Realiza un cierre limpio al recibir una señal de terminación.

9. Justificación para su uso en un entorno real o comercialización

Este programa podría evolucionar en un **Simulador de Gestión para Centros Comerciales**, implementado en un entorno real**.** Realizaría las siguientes tareas:

- **Analizar flujos de clientes y congestión** en zonas clave (cine, patio de comidas, tiendas).
- **Probar configuraciones** (cambios en capacidad, tiempos de espera, asignación de recursos).
- **Evaluar el impacto de decisiones de infraestructura**, como aumentar el estacionamiento o redistribuir locales.

El sistema puede ser la base o prototipo para un futuro **software de control y monitoreo de afluencia** en tiempo real, como aplicación útil en centros comerciales inteligentes, eventos masivos o aeropuertos.

Integrando sensores (IoT), cámaras o validadores de acceso, podría **gestionar el acceso de personas** a distintas zonas.

- Podría **alertar sobre sobrecargas**, ayudar a prevenir emergencias y **optimizar recursos** como personal de limpieza o seguridad.

Podría ser útil para el entrenamiento de personal y simulación de situaciones y escenarios de alta concurrencia para:

- Entrenar al personal en **respuestas ante aglomeraciones**.
- Ensayar **protocolos de evacuación** o gestión de flujos.
- **Simular horarios pico**, eventos especiales o días de ofertas.

El programa podría utilizarse para optimización de recursos:

- **Optimizar la asignación de recursos**: por ejemplo, saber cuántos empleados necesita el cine por hora.
- **Evaluar necesidades logísticas**, como el número óptimo de vehículos permitidos en el estacionamiento y luego que las personas realizan sus tareas en los distintos puntos del centro comercial.

10. Bibliografía

Documentación técnica y páginas web:

Object Management Group. (2017). *OMG Unified Modeling Language (OMG UML), version 2.5.1*. https://www.omg.org/spec/UML/2.5.1

Oracle. (2023). *Java SE Documentation (Java 21)*. https://docs.oracle.com/en/java/javase/21/

Oracle. (2023). *Java Platform, Standard Edition 21 API Specification*. https://docs.oracle.com/en/java/javase/21/docs/api/index.html

Oracle. (2023). *Concurrency — The Java™ Tutorials*. https://docs.oracle.com/javase/tutorial/essential/concurrency/

Baeldung. (s.f.). *Java Tutorials and Guides*. https://www.baeldung.com/

Baeldung. (s.f.). *Guide to JVM Shutdown Hooks in Java*. https://www.baeldung.com/jvm-shutdown-hooks

Libros consultados:

Andrews, G. R. (2000). *Foundations of multithreaded, parallel, and distributed programming*. Addison-Wesley.

Ben-Ari, M. (2006). *Principles of concurrent and distributed programming* (2nd ed.). Addison-Wesley.

Grama, A., Gupta, A., Karypis, G., & Kumar, V. (2003). *An introduction to parallel computing: Design and analysis of algorithms* (2nd ed.). Pearson Addison Wesley.

Downey, A. B. (s.f.). *The little book of semaphores*. http://greenteapress.com/semaphores/

Stallings, W. (2005). *Sistemas operativos: Aspectos internos y principios de diseño* (5ª ed.). Pearson Educación.