Universidad Nacional
Autónoma de México

Facultad de Ciencias

# Modelado y Programación

2024-1

# Práctica final

# Supermercado Concurrente

## Equipo 15


Correa Vázquez Alejandro 316194130
Eduardo vargas Herrrera 421016970

# Introducción

Esta práctica tiene como objetivo determinar la cantidad óptima de cajas que un supermercado debería abrir para gestionar eficientemente las operaciones de venta. Para abordar este problema, se diseñó un ejercicio que emplea patrones de diseño y programación concurrente. Cada caja se representa como un hilo, garantizando que la simulación se ejecute en paralelo y proporcionando un enfoque más realista a la gestión de clientes y cajas.

# Procesos y tecnicas utilizadas

## Formación de Clientes

En el proceso de formación de clientes, aquellos que llegan con un carrito que contiene más de 20 productos se dirigen a las cajas normales. Estas cajas organizan a los clientes de la siguiente manera: primero, los clientes eligen la caja con la menor cantidad de clientes formados. Si todas las cajas tienen 0 clientes, el cliente se forma en la caja 1, el siguiente en la caja 2, y así sucesivamente. Cuando todas las cajas tienen más de un cliente, se forman en la caja con la menor cantidad de clientes formados. En caso de empate, se selecciona la caja que tenga al último cliente con menos productos.

Para las cajas rápidas, los clientes con menos de 20 productos se forman en una fila única. Esperan a que una caja rápida en la zona se desocupe. La primera caja disponible recibe al primer cliente formado en la cola. Si hay varias disponibles, el cliente elige aleatoriamente. Si no hay ninguna disponible, el cliente espera a que se desocupe una.

## Cancelación de Ventas por el Gerente

El gerente cancela el 3% de las ventas por defecto. Cuando una caja alcanza este porcentaje, se activa el método `cancelar()`. La caja pone su hilo en reposo y notifica al gerente, permaneciendo cerrada hasta que el gerente la "despierte". Posteriormente, la caja vuelve a operar normalmente.

## Descanso y Registro de Ventas de las Cajas

Las cajas cuentan con un método de descanso con un tiempo de vida. Al agotarse este tiempo, la caja deja de recibir clientes. Una vez atendidos todos los clientes formados, la caja se apaga por un periodo y luego vuelve a sus operaciones normales. Además, las cajas pueden llamar a `crearTicket()` para registrar la venta en un archivo, generando un ticket con la información de la venta.

## Proceso de Ventas y Control de Stock

Cada venta se considera como el proceso de atender todos los productos en el carrito de un cliente. La suma unitaria y total, junto con la substracción del IVA, se registran en el ticket. Posteriormente, se ajusta la cantidad de productos en el almacén según la venta realizada. Se implementa un contador al inicio de la sesión que hace referencia al almacén, asegurando que no se vendan más productos de los disponibles en stock.

asda

# Uso de Patrones de Diseño en la Práctica

## Introducción

Durante la realización de la práctica, se implementaron diversos patrones de diseño para mejorar la estructura y flexibilidad del sistema. A continuación, se describen los patrones utilizados y sus aplicaciones específicas.

## Patrones de Diseño Utilizados

### MVC (Modelo Vista Controlador)

Se aplicó el patrón Modelo Vista Controlador para la creación de una vista que funciona como un menú interactivo. La comunicación entre la vista y las entidades del sistema, como las cajas, el almacén y los clientes, se llevó a cabo mediante un controlador. Este controlador actuó como un puente facilitando la interacción entre las partes del sistema.

### Composite

El patrón Composite se implementó en el sistema de menús. Las opciones del menú se estructuraron utilizando Composite, permitiendo que las opciones fueran tanto menús que mostraran opciones como menús anidados. De esta manera, al seleccionar una opción, se desplegaría otro menú según la estructura compuesta.

### Strategy

El patrón Strategy se empleó para la creación del sistema que define el comportamiento de los clientes. Se proporcionaron dos estrategias diferentes para cada tipo de caja (rápida o lenta), permitiendo una flexibilidad significativa en el comportamiento del sistema, en la seccion siguiente se detalla el algoritmo con el cual se formaron a los clientes en ambas cajas

### Singleton

Se utilizó el patrón Singleton para garantizar la existencia de un único gerente y un solo almacén en las implementaciones. Esto asegura que solo haya una instancia de estas clases en todo el sistema.

### Observer

El patrón Observer se implementó en el área rápida. Su finalidad fue conocer qué cajas estaban desocupadas. La fila que maneja esta área se modeló como un observador para detectar cambios en el estado de las cajas y tomar decisiones informadas, pues a diferencia de las cajas normales, el area rapida cuenta con una unifila