Proyecto: **Sistema de gestión CineMas - Loja**

Programación Orientada a Objetos

Docente:

Ing. Pedro Daniel Irene Robalino

UTPL
La Universidad Católica de Loja

Estudiantes:

Joel Alexander Domínguez Ochoa.

Axel Stewart Román Torres.

Carrera:

Computación

Abril-Agosto 2025

El presente proyecto de investigación y desarrollo, denominado "CineMas", se enfoca en la aplicación práctica y teórica de la Programación Orientada a Objetos (POO) para construir un sistema de gestión integral para salas de cine. Desarrollado enteramente en lenguaje Java, "CineMas" emula las operaciones de este entorno, abarcando desde la administración de la cartelera de películas y la programación de funciones, hasta la gestión de clientes y el procesamiento detallado de transacciones de venta, tanto de boletos como de snacks. Una de las directrices fundamentales del desarrollo fue la implementación de una arquitectura Modelo-Vista-Controlador (MVC), buscando una clara separación de responsabilidades y una mayor mantenibilidad del código. El modelado conceptual y estructural del sistema se apoyó en el Lenguaje Unificado de Modelado (UML), utilizando específicamente diagramas de clases elaborados con la herramienta DIA-UML. Entre sus funcionalidades destacadas, el sistema incorpora un módulo para la aplicación de promociones y descuentos de forma dinámica, basado en criterios preestablecidos como el día de la semana. Adicionalmente, se ha implementado un mecanismo de persistencia de datos para todas las transacciones de venta, las cuales son registradas en un archivo de texto plano (.txt) mediante el uso de flujos de caracteres, asegurando la integridad y disponibilidad histórica de la información. La interacción con el usuario se realiza a través de una interfaz de consola intuitiva, guiada por menús. Este proyecto no solo ha servido como un ejercicio práctico para consolidar los conocimientos adquiridos en POO, sino también en el manejo de estructuras de datos, la serialización de archivos y los principios de diseño de software robusto y organizado.

1. Introducción

1.1. Contexto y Justificación

La industria del entretenimiento cinematográfico, a pesar de la emergencia de nuevas plataformas de streaming, sigue siendo un sector relevante que requiere sistemas de gestión eficientes para optimizar sus operaciones diarias. La Programación Orientada a Objetos (POO) ofrece un paradigma idóneo para el desarrollo de dichos sistemas ya que nos permite modelar de forma intuitiva las complejas interacciones y entidades presentes en un cine, como películas, horarios, salas, clientes y transacciones. Este proyecto se enmarca en la necesidad académica de aplicar los conocimientos de POO adquiridos en la Carrera de Computación de la Universidad Técnica Particular de Loja, utilizando el lenguaje Java como herramienta principal. Java, por su robustez, portabilidad y amplio ecosistema, es una elección frecuente para el desarrollo de aplicaciones empresariales y de gestión.

1.2. Planteamiento del Problema

Los sistemas de gestión de cines tradicionales pueden variar en complejidad, pero todos comparten la necesidad de manejar eficientemente la información de la cartelera, la venta de entradas, snacks y un registro de la actividad financiera. Un sistema manual o insuficientemente automatizado puede llevar a errores, ineficiencias y una mala experiencia para el cliente. Nuestro Proyecto busca

abordar estos aspectos mediante la creación de un prototipo de software que, aunque limitado a una interfaz de consola, implemente la lógica fundamental de estas operaciones, sirviendo como una base conceptual y práctica para sistemas más complejos. El proyecto se enfoca en la correcta aplicación de los principios de POO y el patrón MVC para lograr una solución bien estructurada.

1.3. Alcance y Limitaciones

Alcance

- Gestionar una lista precargada de películas, salas y funciones.
- Simular la compra de boletos para funciones específicas, aplicando promociones según el día.
- Permitir la compra de snacks.
- Asignar clientes (de una lista precargada y seleccionados aleatoriamente) a las transacciones.
- Generar facturas simplificadas por cada transacción.
- Mantener un registro de todas las ventas en memoria durante la sesión y persistirlas en un archivo de texto (.txt).
- Ofrecer un menú interactivo para la navegación del usuario.

Las limitaciones de este proyecto incluyen:

- No se implementa la selección de asientos específicos.
- Carece de un sistema de autenticación de usuarios o roles diferenciados (ej. administrador, vendedor).
- Los datos (películas, funciones, etc.) son cargados al inicio y no pueden ser modificados, añadidos o eliminados por el usuario durante la ejecución del programa.
- La persistencia de datos se limita a un archivo de texto para las ventas

2. Objetivos

2.1. Objetivo General

- Diseñar, desarrollar e implementar un prototipo de sistema de gestión de cine basado en consola denominado "CineMas", utilizando el lenguaje de programación Java y aplicando los principios de la Programación Orientada a Objetos y el patrón arquitectónico Modelo-Vista-Controlador, que permita la administración de la cartelera, funciones, clientes, el procesamiento de ventas de boletos y snacks con aplicación de promociones, y la generación de registros persistentes de dichas transacciones en archivos de texto.

2.2. Objetivos Específicos

- Modelar la estructura estática y las relaciones entre las entidades del sistema mediante un diagrama UML, utilizando la herramienta DIA-UML, para guiar el proceso de desarrollo.
- Aplicar el patrón arquitectónico Modelo-Vista-Controlador (MVC) para estructurar el código fuente en Java, garantizando la separación de la lógica de presentación, la lógica de negocio y la gestión de datos.
- Diseñar y desarrollar un mecanismo para la persistencia de los detalles de cada factura (boletos y snacks) en un archivo de texto plano (.txt).

3. Marco Teórico Referencial

3.1. Programación Orientada a Objetos (POO)

La Programación Orientada a Objetos es un paradigma de programación que utiliza "objetos" – instancias de clases – para diseñar aplicaciones y programas informáticos. Se basa en varios principios fundamentales:

- Encapsulamiento: Agrupa en una misma entidad (la clase) los datos (atributos) y las operaciones que los manipulan (métodos), ocultando la complejidad interna y exponiendo solo lo necesario. En "CineMas", cada clase como Pelicula o Cliente encapsula sus datos (título, nombre, etc.) y los métodos para acceder o modificar estos datos de forma controlada (getters/setters).
- Abstracción: Consiste en identificar las características esenciales de un objeto y obviar los detalles irrelevantes. Permite modelar entidades del mundo real (una película, una función) de manera simplificada en el software.

3.2. Lenguaje de Programación Java

Java es un lenguaje de programación de alto nivel, orientado a objetos, concurrente y robusto, desarrollado por Sun Microsystems (ahora Oracle). Sus características principales, como la portabilidad (gracias a la Java Virtual Machine - JVM), su extensa biblioteca estándar (API), y su fuerte tipado, lo convierten en una elección popular para una amplia gama de aplicaciones. Para nuestro proyecto, Java proporciona las herramientas necesarias para implementar la lógica de este programa, las estructuras de datos y el manejo de archivos de manera eficiente.

3.3. Patrón Arquitectónico Modelo-Vista-Controlador (MVC)

MVC es un patrón de diseño arquitectónico que divide una aplicación en tres componentes interconectados:

- **Modelo (Model)**: Es el cerebro del programa. Contiene los datos y la lógica. Es independiente de la interfaz de usuario. En "CineMas", las clases como Pelicula, Funcion, FacturaFuncion, etc., y sus respectivas lógicas de cálculo (como el total de una factura) residen aquí.

- **Vista (View)**: Es la representación visual de los datos del modelo; es decir, la interfaz de usuario. Se encarga de mostrar la información al usuario y de capturar sus acciones. En este proyecto, la clase VistaCineMas cumple este rol mediante la consola.
- **Controlador (Controller)**: Actúa como intermediario entre el Modelo y la Vista. Recibe la entrada del usuario desde la Vista, la interpreta, interactúa con el Modelo para procesar los datos o ejecutar acciones, y luego selecciona la Vista apropiada para mostrar el resultado. Controlador (ej., PeliculaControlador, FuncionControlador) en "CineMas" realizan esta función.

La adopción de MVC promueve la separación de intereses, facilita las pruebas y mejora la mantenibilidad del programa.

3.4. Diagramas de Clases (UML)

Es una forma gráfica para visualizar, especificar, construir y documentar los artefactos de un sistema. Los diagramas de clases son un tipo de diagrama estructural en UML que describe la estructura de un sistema mostrando sus clases, atributos, métodos y las relaciones entre los objetos. En el desarrollo de "CineMas", se utilizó DIA-UML para crear el diagrama de clases, lo que permitió definir claramente las entidades del sistema y sus interconexiones antes de la codificación, sirviendo como un plano para la implementación.

3.5. Manejo de Archivos en Java y Serialización de Caracteres Java

Proporciona un robusto conjunto de herramientas para el manejo de archivos a través de su API de E/S (Entrada/Salida), ubicada principalmente en el paquete java.io. Los archivos son vistos por Java como flujos secuenciales de bytes. Estos flujos pueden ser basados en bytes (para datos binarios) o basados en caracteres (para texto).

- **Flujos de Caracteres:** Representan datos como secuencias de caracteres. Son ideales para manejar archivos de texto, como es el caso del archivo registro_ventas.txt en "CineMas". Clases como FileReader y FileWriter se utilizan para leer y escribir flujos de caracteres, respectivamente.
- **Clase FileWriter:** Permite escribir datos de caracteres en un archivo. Es fundamental para la persistencia de las facturas, ya que se utilizó en modo append para añadir nuevas transacciones sin sobrescribir las existentes.
- **Clase Formatter:** Proporciona la capacidad de escribir datos formateados en un flujo de salida, que puede ser un archivo. En "CineMas", se utiliza en conjunto con FileWriter para dar un formato estructurado a cada línea de factura guardada en registro_ventas.txt, facilitando su legibilidad y posible importación a otras herramientas.
- **Clase Scanner:** Utilizada tanto para leer la entrada del usuario desde la consola como para leer el contenido de archivos de texto, como se demuestra en la funcionalidad verRegistroDeVentas para mostrar el contenido del archivo de log.

La "serialización de caracteres**"** en este contexto se refiere a la conversión de los datos de las facturas (números, cadenas de texto, fechas) en una representación de cadena formateada que luego se escribe en el archivo de texto. Esto difiere de la serialización de objetos de Java (que utiliza ObjectOutputStream y ObjectInputStream para convertir objetos completos en secuencias de bytes y viceversa), la cual no fue el enfoque principal para este archivo de registro específico, prefiriéndose un formato de texto legible.

4. Metodología y Herramientas

4.1. Fases del Desarrollo

El proyecto "CineMas" se abordó mediante un enfoque de desarrollo incremental, estructurado en las siguientes fases principales:

- **Fase de Conceptualización y Planificación:**
  - Definición inicial del alcance y los objetivos del proyecto, basados en los requisitos de la asignatura.
  - Establecimiento de un plan de trabajo y distribución de tareas entre los integrantes del equipo (Joel Domínguez y Axel Román).
- **Fase de Diseño:**
  - Análisis detallado de los requisitos funcionales y de datos para el sistema de cine.
  - Diseño de la arquitectura del programa, seleccionando el patrón Modelo-Vista-Controlador (MVC) como base estructural**.**
  - Modelado del Diagrama de Clases UML utilizando la herramienta DIA-UML. Este diagrama sirvió como guía para la estructura de las clases en Java.
  - Diseño de la interfaz de usuario por consola, definiendo los menús y flujos de interacción.
- **Fase de Implementación (Codificación):**
  - Configuración del entorno de desarrollo Java.
  - Creación de la estructura de paquetes (Model, View, Controller) en el proyecto.
  - Codificación de las clases del Modelo, implementando sus atributos, constructores y métodos según el diseño UML.
  - Desarrollo de las clases Controladoras, implementando la lógica de la aplicación y la comunicación entre la Vista y el Modelo.
  - Implementación de la clase Vista (VistaCineMas), incluyendo el método main, la lógica de menús, la captura de entrada del usuario y la presentación de resultados.
  - Desarrollo de funcionalidades específicas como la carga inicial de datos, la compra de boletos, la aplicación de promociones, la compra de snacks y la persistencia de facturas en archivo de texto.

- **Fase de Pruebas y Depuración:**
  - Realización de pruebas unitarias informales (pruebas de métodos individuales) y pruebas de integración (verificación de la correcta interacción entre componentes).
  - Identificación y corrección de errores lógicos y de ejecución. Se prestó especial atención a la correcta aplicación de descuentos y al formato y persistencia de los datos en el archivo de registro.
  - Validación del cumplimiento de los requisitos funcionales.
- **Fase de Documentación:**
  - Elaboración del presente informe técnico, detallando todos los aspectos del proyecto.
  - Preparación del material para la presentación del proyecto.
  - Mantenimiento del repositorio GitHub con el código fuente, modelado UML y demás.

4.2. Herramientas de Software Utilizadas

Lenguaje de Programación: Java (JDK versión 23).

Entorno de Desarrollo Integrado (IDE): (Apache NetBeans IDE 23). Se seleccionó por su robusto conjunto de herramientas para el desarrollo del proyecto

Herramienta de Modelado UML: DIA-UML. Utilizada para la creación del diagrama de clases, facilitando la visualización y el diseño de la estructura del software antes de la codificación.

Sistema de Control de Versiones: Git. Utilizado para el control de versiones del código fuente, permitiendo el trabajo colaborativo y el seguimiento de cambios.

Plataforma de Alojamiento de Repositorios: GitHub (github.com/POO-B-AA25/aab1-proyecto-g3). Utilizada para alojar el código fuente, la documentación y facilitar la colaboración y entrega del proyecto.

5. Diseño del Sistema

Arquitectura del programa: MVC en "CineMas"

La arquitectura Modelo-Vista-Controlador (MVC) fue seleccionada para el desarrollo de "CineMas" con el fin de promover una clara separación de responsabilidades, mejorar la cohesión dentro de los componentes y reducir el acoplamiento entre ellos. Esto facilita la comprensión del sistema, su mantenimiento y futuras extensiones.

5.1. Componente Modelo

El Modelo es el núcleo del sistema, tiene todos los datos y la lógica del programa fundamental. Es independiente de la interfaz de usuario y de la lógica de control. En "CineMas", el Modelo está constituido por un conjunto de clases Java que representan las entidades del dominio y sus operaciones:

Entidades de Datos: Clases como Pelicula, Funcion, Sala, Cliente, Snack, Boleto, Horario, y Promocion definen la estructura de los datos manejados por el sistema. Por ejemplo, la clase Pelicula almacena el título, sinopsis, género y clasificación.

Lógica de programa: Lógica como el cálculo del subtotal y el valor total de una factura (en FacturaFuncion y FacturaSnack), o la determinación de si una promoción es aplicable (en Promocion.aplicaDescuento()), reside en el Modelo. Este componente no tiene conocimiento directo de la Vista ni del Controlador.

5.2. Componente Vista

La Vista es responsable de presentar la información del Modelo al usuario y de capturar las interacciones de este. En "CineMas", este rol es desempeñado por la clase VistaCineMas.java.

Presentación de Datos: Muestra el menú, listas de películas, detalles de funciones, facturas generadas y registros de ventas en la consola.

Captura de Entrada: Utiliza java.util.Scanner para leer las selecciones y datos ingresados por el usuario. La Vista no contiene lógica de negocio; su función es puramente de interfaz. Delega las acciones del usuario al Controlador.

5.3. Componente Controlador

El Controlador actúa como el intermediario entre la Vista y el Modelo. Interpreta las acciones del usuario provenientes de la Vista, invoca los métodos apropiados en el Modelo para procesar dichas acciones y, finalmente, selecciona la Vista adecuada para mostrar los resultados.

Manejo de Solicitudes: Cada clase Controlador (PeliculaControlador, FuncionControlador, FacturaControlador, PromocionControlador) gestiona un aspecto específico del Modelo. Por ejemplo, FuncionControlador maneja la lista de objetos Funcion.

Coordinación: Cuando un usuario solicita comprar un boleto, VistaCineMas notifica al controlador correspondiente (implícitamente a través de sus propios métodos que luego usan los controladores de datos), el cual recupera la información de funciones, interactúa con el PromocionControlador, y finalmente con el FacturaControlador y la clase FacturaFuncion para generar la transacción. En "CineMas", dado que la clase VistaCineMas se ha estructurado con métodos estáticos y contiene la lógica de flujo principal, actúa en gran medida como un controlador de aplicación o coordinador, utilizando a su vez los otros controladores (PeliculaControlador, etc.) que son más bien gestores de datos o servicios del modelo.

Flujo de Interacción MVC en una Transacción Típica (Ej: Comprar Boleto)

Usuario (Vista): El usuario selecciona la opción "Comprar Boletos" en el menú mostrado por VistaCineMas.

Vista al Coordinador/Controlador (en VistaCineMas): El método comprarBoletos() en VistaCineMas se activa.

Coordinador/Controlador usa otros Controladores de Datos:

comprarBoletos() solicita la lista de funciones al FuncionControlador.

La Vista muestra estas funciones. El usuario selecciona una.

comprarBoletos() solicita las promociones al PromocionControlador para verificar si alguna aplica a la fechaDeLaFuncion.

Coordinador/Controlador interactúa con el Modelo:

Se crea un objeto Boleto.

Se crea un objeto FacturaFuncion (clase del Modelo), pasándole el boleto y la promoción (si existe). La clase FacturaFuncion calcula internamente su subtotal y valor total (aplicando el descuento).

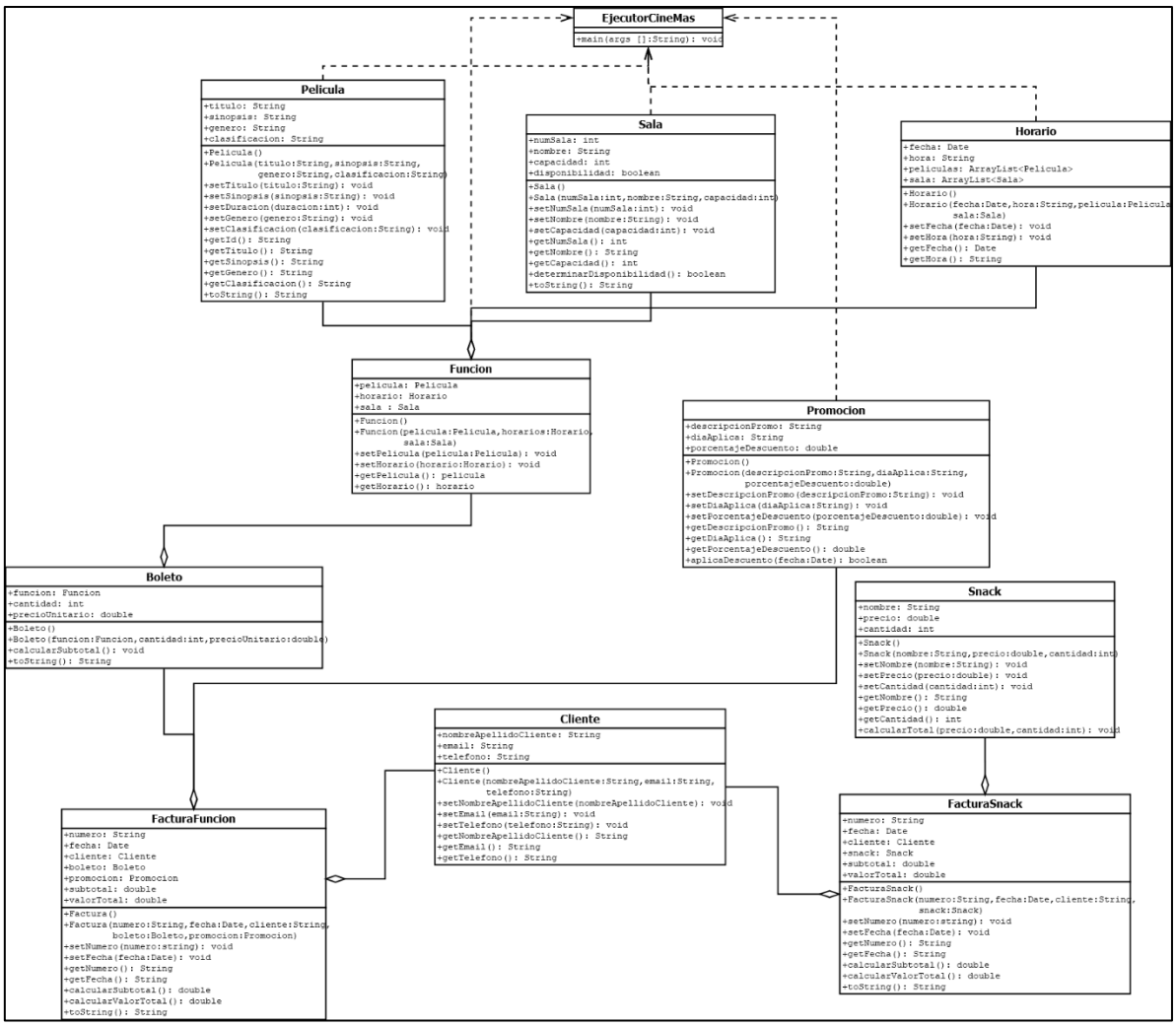
La nueva factura se añade al FacturaControlador.

Coordinador/Controlador actualiza la Vista:

comprarBoletos() formatea y muestra los detalles de la factura (incluyendo el descuento y el total final) en la consola.

Los datos de la factura también se envían al método escribirFacturaEnArchivo() para su persistencia.

5.4 Diagrama de Clases (UML)



En esta imagen se presenta el Diagrama de Clases del sistema CineMas, el cual ilustra la estructura estática del programa, las clases que lo componen, sus atributos principales, métodos más relevantes y las relaciones entre ellas (asociación, agregación, etc.). Este diagrama fue elaborado utilizando la herramienta DIA-UML y sirvió como guía fundamental durante la fase de implementación.

Se pueden observar las clases centrales del paquete Model como Pelicula, Funcion, Sala, Cliente, Snack, Boleto, Horario, Promocion, FacturaFuncion y FacturaSnack. La clase Funcion, por ejemplo, muestra una relación con Pelicula y Sala, indicando que una función específica corresponde a una película y se proyecta en una sala. También se relaciona con Horario. Las clases FacturaFuncion y FacturaSnack se asocian con Cliente y, respectivamente, con Boleto (que a su vez se asocia con Funcion) y Snack. La clase FacturaFuncion también tiene una asociación opcional con Promocion.

6. Implementación

6.1. Configuración del Entorno y Estructura del Proyecto

El desarrollo se llevó a cabo utilizando Java (JDK versión 23).

El proyecto fue gestionado utilizando (Apache NetBeans IDE 23), que facilitó la edición del código, la compilación y la depuración. La estructura del proyecto en el IDE se organizó en los paquetes View, Model, y Controller, reflejando la arquitectura MVC adoptada. El control de versiones se manejó con Git, y el repositorio se alojó en GitHub (POO-B-AA25/aab1-proyecto-g3).

6.2. Desarrollo de Funcionalidades Clave

- Carga de Datos Iniciales y Preconfiguración del Sistema La inicialización de datos se realiza en el método cargarDatosIniciales() de VistaCineMas. En este método:
- Se crean múltiples instancias de Cliente con datos ficticios y se añaden al ClienteControlador.
- Se define una lista de objetos Pelicula, cada uno con título, sinopsis, género (ej. "Animacion", "Accion", "Ciencia Ficcion") y clasificación (ej. "ATP", "Mayores de 13"). Estos se añaden al PeliculaControlador.
- Se instancian objetos Sala (ej. "Sala VIP", "Sala Standard") y se registran en SalaControlador.
- Se crean objetos Promocion especificando su descripción, el día de la semana aplicable (ej. "martes", "jueves", "sabado" - sin tildes para consistencia con la entrada de datos) y el porcentaje de descuento. Estos se añaden al PromocionControlador.
- Se define un catálogo de Snack (ej. "Popcorn Grande", "Gaseosa Mediana") que se agrega al SnackControlador.
- Se utiliza java.util.Calendar para establecer fechas específicas (ej. un martes, un jueves, un sábado del año 2025) para las funciones, lo que permite probar la lógica de promociones. Los objetos Date resultantes se usan, junto con una hora en formato String, para crear objetos Horario.
- Finalmente, se crean múltiples objetos Funcion, asociando una Pelicula del PeliculaControlador, un Horario y una Sala del SalaControlador, y se añaden al FuncionControlador. Esta carga de datos es fundamental para que el sistema tenga un estado inicial con el que el usuario pueda interactuar inmediatamente.
- Gestión y Visualización de la Cartelera de Películas La funcionalidad verPeliculasEnCartelera() obtiene la lista de películas desde peliculaControlador.obtenerPeliculas(). Luego, itera sobre esta lista y, para cada objeto Pelicula, imprime en consola su título, clasificación, género y sinopsis de forma formateada, permitiendo al usuario conocer la oferta cinematográfica.

Proceso Detallado de **Compra de Boletos** El método comprarBoletos(Scanner scanner) orquesta esta compleja interacción:

- Selección de Cliente: Se obtiene la lista de clientes del clienteControlador. Si existen clientes, se selecciona uno aleatoriamente utilizando java.util.Random para asignarlo a la transacción.
- Visualización de Funciones: Se recuperan todas las funciones disponibles del funcionControlador.obtenerFunciones(). Cada función se presenta al usuario mostrando el título de la película, nombre de la sala, y el horario (fecha y hora) formateado por sdfDiaHora y sdfFechaCompleta.
- Selección del Usuario: El usuario ingresa el número correspondiente a la función deseada y la cantidad de boletos. Se realizan validaciones para asegurar que la selección sea válida y la cantidad sea positiva.
- Creación del Boleto: Se instancia un objeto Boleto, asociándolo con la Funcion seleccionada, la cantidad y un precio unitario predefinido (ej. 8.50).

Aplicación de Promoción:

- Generación de Factura**:** Se crea un número de factura secuencial (ej. "FF-X"). Se instancia FacturaFuncion con el número, la fecha actual (new Date()), el cliente seleccionado, el objeto Boleto y la promocionAplicada (que puede ser null).
- Registro de Factura: La nueva factura se añade al facturaControlador (facturaControlador.agregarFacturaFuncion(factura)).
- Persistencia en Archivo: Se formatea una cadena con los detalles de la factura y se llama a escribirFacturaEnArchivo() para guardarla en registro_ventas.txt.
- Presentación al Usuario: Se imprime en consola un resumen detallado de la factura generada, incluyendo todos los datos relevantes y, si aplica, el descuento.

Lógica de Aplicación de Promociones y Descuentos Dentro de comprarBoletos(), una vez seleccionada la función, se obtiene su fechaDeLaFuncion. El sistema itera sobre la lista de promociones disponibles (obtenidas de promocionControlador.obtenerPromociones()):

- Para cada objeto Promocion (promo), se invoca su método promo.aplicaDescuento(fechaDeLaFuncion).
- Se asume que el método aplicaDescuento dentro de la clase Promocion contiene la lógica para:
- Obtener el nombre del día de la semana de fechaDeLaFuncion (probablemente usando sdfDiaSemana.format(fechaDeLaFuncion).toLowerCase()).
- Comparar este nombre de día con el diaAplica almacenado en el objeto Promocion (ej., "martes", "sabado"). Es crucial que esta comparación sea insensible a las tildes para que funcione correctamente, dado que los días almacenados en Promocion se definieron sin tildes en cargarDatosIniciales() mientras que sdfDiaSemana con Locale("es", "ES") puede producirlos con tildes.

- Si aplicaDescuento() devuelve true, esa Promocion se asigna a promocionAplicada y se interrumpe el bucle.
- Al crear el objeto FacturaFuncion, se le pasa promocionAplicada. La clase FacturaFuncion es la que debe usar el porcentajeDescuento de esta promoción para calcular el valorDescuento y restarlo del subtotal para obtener el valorTotal.

Proceso de Compra de Snacks El método comprarSnacks(Scanner scanner) sigue un flujo similar al de boletos, pero más simplificado:

- Selección aleatoria de un cliente.
- Visualización del catálogo de snacks (nombre y precio) obtenido del snackControlador.
- Selección del snack y cantidad por parte del usuario.
- Creación de un objeto Snack para la factura con la cantidad comprada.
- Generación de un número de factura (ej. "FS-X") y creación de un objeto FacturaSnack.
- Adición de la factura al facturaControlador y persistencia en registro_ventas.txt.
- Presentación de la factura de snacks en consola. Generalmente, no se aplican promociones complejas a los snacks en este modelo.

Sistema de Registro de Ventas y Persistencia en Archivo de Texto

- La persistencia de las transacciones es una característica clave. El método escribirFacturaEnArchivo(String datosFactura) centraliza la escritura:
- Utiliza new FileWriter(NOMBRE_ARCHIVO_REGISTRO, true) para abrir el archivo en modo de adición (append), lo que asegura que cada nueva factura se añada al final del archivo sin borrar las anteriores.
- Envuelve el FileWriter en un Formatter para facilitar la escritura de la cadena datosFactura (que ya viene formateada) seguida de un salto de línea (%n).
- La cadena datosFactura se construye en los métodos comprarBoletos y comprarSnacks usando String.format(Locale.US, "TIPO;NUM;CLIENTE;...", ...). Se incluye el tipo de factura ("BOLETO" o "SNACK"), número, nombre del cliente, detalle del producto, cantidad, subtotal, monto del descuento (0.00 para snacks), valor total y la fecha/hora de la transacción. Se utiliza Locale.US para asegurar el punto como separador decimal. Los campos de texto que podrían contener punto y coma (el delimitador) se sanean con .replace(";", ","). El método verRegistroDeVentas() en VistaCineMas no solo muestra los datos de las facturas almacenadas en memoria por los controladores, sino que también lee y muestra el contenido completo del archivo registro_ventas.txt utilizando un Scanner para que el usuario pueda ver el historial persistente.

- Desarrollo de la Interfaz de Usuario por Consola La clase VistaCineMas es la única responsable de la interacción con el usuario.
- El método main inicia el ciclo principal de la aplicación.
- mostrarMenuPrincipal() imprime un menú numerado con las opciones disponibles.
- Se utiliza un bucle while (!salir) para mantener la aplicación activa hasta que el usuario elija la opción de salir.
- Un switch (opcion) dirige el flujo a los métodos correspondientes según la selección del usuario.
- Se utiliza java.util.Scanner para leer las entradas numéricas y de texto. Se incluye manejo de InputMismatchException para entradas no numéricas en la selección del menú y otras entradas numéricas, solicitando al usuario reintentar.
- Los mensajes al usuario se han diseñado para ser claros e informativos, guiándolo a través de los diferentes procesos. Se ha procurado que los textos en consola no contengan tildes para una visualización consistente en diversos entornos, como fue solicitado.

7. Conclusiones del Proyecto "CineMas"

El desarrollo del proyecto "CineMas" ha permitido la aplicación integral de los conocimientos teóricos y prácticos de la Programación Orientada a Objetos (POO) en la construcción de un sistema de gestión para salas de cine. A continuación, se presentan las conclusiones derivadas de este trabajo:

- **Cumplimiento del Objetivo Principal:** Se logró diseñar, desarrollar e implementar exitosamente un prototipo de sistema de gestión de cine basado en consola, denominado "CineMas". Este sistema utiliza el lenguaje de programación Java y aplica los principios fundamentales de la POO y el patrón arquitectónico Modelo-Vista-Controlador (MVC). Permite la administración de una cartelera de películas, la programación de funciones, la gestión de clientes (de forma precargada), el procesamiento de ventas de boletos y snacks con aplicación dinámica de promociones, y la generación de registros persistentes de dichas transacciones en archivos de texto.
- **Modelado y Diseño Estructural:** El uso del Lenguaje Unificado de Modelado (UML), específicamente a través de diagramas de clases elaborados con la herramienta DIA-UML, fue crucial en la fase de diseño. Este modelado proporcionó una representación visual clara de la estructura estática del sistema, las clases, sus atributos, métodos y relaciones, sirviendo como un plano guía para la implementación en Java.
- **Gestión de Datos y Persistencia:** El sistema gestiona listas precargadas de películas, salas, funciones y clientes. Una funcionalidad destacada es la persistencia de todas las transacciones de venta (boletos y snacks) en

un archivo de texto plano (registro_ventas.txt) mediante flujos de caracteres y la clase FileWriter en modo de adición. Se utilizó Formatter para dar una estructura legible a los datos guardados, asegurando la integridad y disponibilidad histórica de la información.

- **Interfaz de Usuario y Funcionalidades Implementadas:** La interacción con el usuario se realiza a través de una interfaz de consola intuitiva guiada por menús. El sistema permite simular la compra de boletos aplicando promociones según el día de la semana, la compra de snacks, la asignación aleatoria de clientes a las transacciones y la generación de facturas simplificadas.

7.1. Recomendaciones y Trabajo Futuro:

Si bien nuestro proyecto "CineMas" cumple con los objetivos planteados, presenta limitaciones inherentes a un prototipo académico, como la ausencia de selección de asientos, autenticación de usuarios y la imposibilidad de modificar datos en tiempo de ejecución. Como trabajo futuro, se podría considerar:

- Implementar una interfaz gráfica de usuario, para mejorar la experiencia del usuario.
- Desarrollar un sistema de gestión de usuarios con roles (administrador, vendedor).
- Permitir la gestión dinámica de películas, funciones y salas.
- Integrar una base de datos relacional para una persistencia de datos más robusta y escalable, superando las limitaciones del archivo de texto.
- Implementar la selección de asientos en las salas.

En definitiva, el proyecto "CineMas" representa una base conceptual y práctica sólida. La correcta aplicación de los principios de POO y el patrón MVC ha resultado en una solución bien estructurada que sienta las bases para el desarrollo de sistemas de gestión de cines más complejos y completos.