Entrega Acumulada B

Tema: Administrador de Películas-Clientes en un videoclub

Nombres: **Luis Romero, Heinz Beckers, Sebastian Villalón**

1.1 Análisis de los datos a utilizar y principales funcionalidades a implementar que dan sentido a la realización del proyecto.

El problema que se nos presenta es lograr la implementación de un sistema que sea capaz de administrar un registro de clientes, así como de películas, el cual nos permita mediante el uso del historial de arriendo de un cliente y el stock de cintas disponibles en la tienda, sugerir a este películas que le podrían interesar arrendar y permitir que las arriende.

Dentro de las principales entidades tenemos a Película la cual almacenará los principales datos de un film, la entidad Cliente la cual contendrá datos sobre el arrendador de películas que serán necesarios para la recomendación de estas, además del manejo del negocio,la entidad Arriendo que contiene datos relacionados al arriendo de una película y por último la entidad VideoClub la cual de una manera u otra contendrá a todas las clases anteriormente mencionadas, además de información relacionada a la tienda física.

En cuanto a las principales funcionalidades que nos gustaría implementar tenemos las siguientes:

1. Administración de películas (agregar, actualizar, mostrar, eliminar).
2. Administración de clientes (agregar, actualizar, mostrar, eliminar).
3. Arrendar películas.
4. Recomendar películas.

1.2 Diseño conceptual de clases del Dominio y su código en Java

- **Clase Videoclub**

Esta clase se encarga de almacenar todos los datos relacionados a una tienda/videoclub. Entre los atributos de esta tienda tenemos los String nombre y dirección, además de 3 arraylist y 3 hashmap , uno de clase Película que almacena todos los films (con su respectiva información) que posee la tienda, otro de clase Cliente que contiene todos los clientes que ha tenido la tienda (con su respectiva información) y por último de clase Trabajador, que guarda todos los empleados de la tienda. Además se agrega un hashmap Presentes, que almacena todas Personas que ingresaron a la tienda.

Posee los métodos setter y getter para cada variable privada.

- **Clase Película**

Contiene las variables que almacenan los datos de la película, además de la cantidad de copias existentes, las disponibles y su id única.

Posee los métodos setter y getter para cada variable privada.

- **Clase Persona (abstracta)**

Esta clase es del tipo abstracta y contiene los datos personales de las personas relacionadas al videoclub tales como nombre, rut, deuda (en caso de existir) y los arriendos de la persona. Incluye el método abstracto Identificación() que ha de ser implementado por las subclases.

Posee los métodos setter y getter para cada variable privada.

- **Clase Transacciones (interfaz)**

Interfaz que declara los métodos arrendar(), devovlerArriendo() y pagarDeuda().

- **Clase Cliente**

Esta clase extiende la clase abstracta Persona e implementa la interfaz Transacciones, por lo que hereda los métodos y atributos de persona y se ve obligada a implementar los métodos definidos por la interfaz así como el método abstracto de la superclase. Además tiene los atributos propios historialArriendos, un ArrayList e historialXid, un Hashmap. Su función principal es representar a los clientes que tiene el videoclub.

Posee los métodos setter y getter para cada variable privada.

- **Clase Trabajador**

Esta clase extiende la clase abstracta Persona e implementa la interfaz Transacciones, por lo que hereda los métodos y atributos de persona y se ve obligada a implementar los métodos definidos por la interfaz así como el método abstracto de la superclase. Además tiene los atributos propios cargo, sueldo y vecesArriendoAtrasados. Su función principal es representar a los trabajadores del videoclub quienes tendrán acceso a funciones reservadas solo para ellos.

Posee los métodos setter y getter para cada variable privada.

- **Clase Arriendo**

Esta clase se encarga de almacenar atributos relacionados al arriendo de una película, como por ejemplo, id de la película, fechas de arriendo y devolución, valoración de la película por parte del cliente y veces arrendada por este.

Posee los métodos setter y getter para cada variable privada.

- **Clase MenuTienda**

Esta clase contiene los métodos necesarios para la implementación y ejecución del menú por consola. No posee atributos.

- **Clase Funciones**

Clase que contiene los métodos necesarios para la implementación y ejecución de las funciones principales del proyecto. No posee atributos.

- **Clase FechaInvalidaException**

Clase que se extiende de la clase Exception, creada para el manejo de excepciones cuando se ingresa una fecha con un formato incorrecto.

- **Clase RutInvalidoException**

Clase que se extiende de la clase Exception, creada para el manejo de excepciones cuando se ingresa un rut con un formato incorrecto.

- **Clase InfoMenu**

Clase que guarda la información de los menús para poder mostrarlos nuevamente, posee atributos nameVentana, nameFrame y una lista con los botones del menú.

Posee los métodos setter y getter para cada variable privada.

- **Clase MenuAdmin**

Clase que contiene la interfaz gráfica del menú de los trabajadores, contiene los tributos rutEmpleado, tienda, nombreVentana, nombreFrame, menuActual, menuAnterior, una lista botonesActuales y un hashmap menús.

- **Clase RegistrarCliente**

Clase que contiene la interfaz gráfica en la cual los trabajadores registran nuevos clientes, posee el atributo local.
Además, contiene los métodos y atributos para el manejo de la interfaz gráfica

- **Clase ClienteGUI**

Clase que contiene la interfaz gráfica principal de los clientes, en la cual seleccionan qué es lo que desean hacer, contiene los atributos rut y tienda.

- **Clase Cliente_InformacionGUI**

Clase que contiene la interfaz grafica en la cual lo clientes selecciona que es lo que quieren mostrar por pantalla, contiene los atributos rut y tienda.

- **Clase Cliente_PeliculasGUI**

Clase que contiene la interfaz gráfica en la cual los clientes pueden buscar una película, desplegar catálogos de películas o pedir que se les recomiende una película, posee los atributos rut y tienda.

- **Clase Cliente_ServiciosGUI**

Clase que contiene la interfaz gráfica en la cual los clientes pueden arrendar o devolver una película, además de pagar su deuda, posee los atributos rut y tienda.

- **Clase BuscarPeliculaGUI**

Clase que contiene la interfaz gráfica en la cual se ingresa el nombre de una película a buscar, contiene los atributos rut y tienda.

- **Clase DesplegarMiFichaGUI**

Clase que contiene la interfaz gráfica en la cual se despliega la información de un cliente, contiene los atributos rut, tienda y x que almacena el cliente a buscar.

- **Clase DesplegarPeliculaGUI**

Clase que contiene la interfaz gráfica en la cual se despliega la información de una película, posee los atributos tienda y película.

- **Clase FiltroGUI**

Clase que contiene la interfaz gráfica en la cual se seleccionan los filtros para buscar una película, contiene los atributos tienda, input1 e input2.

- **Clase MostrarDatosGUI**

Clase que contiene la interfaz gráfica en la cual se muestran datos pedidos, contiene los atributos rut, tienda y modo que indica que es lo que se va a mostrar.

1.3 Todos los atributos de todas las clases deben ser privados y poseer sus respectivos métodos de lectura y escritura (getter y setter).

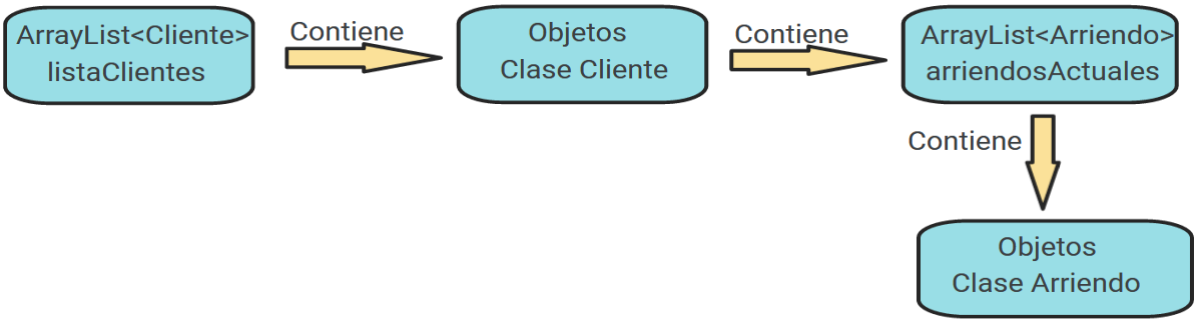
Atributos se hacen privados.

1.4 Se deben incluir datos iniciales dentro del código.

Datos iniciales incluidos como archivos en la carpeta data, bajo los nombres de clientes.tsv y peliculas.tsv.

2.1 Diseño conceptual y codificación de 2 (dos) niveles de anidación de colecciones de objetos.

Se decidió usar la siguiente anidación de colecciones ubicadas en clase VideoClub y Cliente:



Lo anterior también aplica para el Hashmap<Cliente> clientesXRut, el cual contiene clientes, los cuales contienen dos Arraylist.y Hashmap de objetos Arriendo.

2.2 Diseño conceptual y codificación de 2 (dos) clases que utilicen sobrecarga de métodos.

Se tiene la siguiente sobrecarga en la clase VideoClub.

- **mostrarDatosPeliculas()**:muestra todos los datos de las películas dentro del ArrayList listaPeliculas.
- **mostrarDatosPeliculas(String id):**muestra todos los datos de una película en específico dentro del ArrayList listaPeliculas, que tiene como parámetro un String que equivale a la id de la película cuyos datos queremos obtener.

También en clase Cliente tenemos otra sobrecarga.

- **mostrarHistorial(String id):** Permite ingresar como parámetro el id de una película, verifica que esa película se encuentre en el historial, para posteriormente mostrar por pantalla la información de ese Arriendo.
- **mostrarHistorial(VideoClub tienda):** Permite mostrar por pantalla todo el historial de arriendos del cliente.

2.3 Diseño conceptual y codificación de al menos una clase mapa del JCF.

Se implementan dos HashMap en la clase VideoClub:

- **pelisXid**: Almacena las películas que posee el videoclub, a cada película se le asigna un String llamado id, la cual es única y es usada como key.
- **clientesXrut**: Almacena los clientes que posee el videoclub, el rut propio de cada cliente es usado como key.

Estos dos HashMaps acompañan a sus respectivos ArrayList, listaPeliculas y listaClientes , los cuales se usan para poder recorrer a todos los clientes y películas del videoclub.

3.1 Diseño de diagrama de clases UML

Se encuentra adjuntado en el .zip entregado y en el repositorio de github, este se encuentra dividido en dos parte en la rama mismo proyecto, se llaman "parte1UMLEntregaFinal" y "parte2UMLEntregaFinal, la imagen se encuentra en 2 partes y en ese formato porque al parecer el programa se desconfiguro y no supimos cómo arreglarlo.

3.2 Implementación de menú del sistema para funcionalidades

El acceso a ciertas funcionalidades de la aplicación se ha implementado mediante la creación de dos menús, uno para el cliente del videoclub y otro para el administrador/trabajador de este establecimiento. La implementación del menú se encuentra en la clase "Interface".

En interface nos encontramos con 3 métodos principales *inicioMenu(VideoClub)*, *menuCliente()* y *menuAdmin()*. *inicioMenu(VideoClub)* se encarga de dar la opción a que tipo de menú entrar, menú cliente o administrador y en caso que se desee salir del programa también está la opción disponible.

El método login despliega las siguientes opciones:

"1)Cliente": Accede al menú de clientes.
"2)Trabajador": Accede al menú de trabajador
"0)Finalizar programa": Finaliza la ejecución del programa

El método *menúCliente* despliega las siguientes opciones:

"1)Películas": Da acceso al siguiente submenú.

"1)Desplegar catálogo de películas": Muestra el catálogo completo de películas que tiene disponible la tienda.
"2)Buscar Película": Muestra los datos guardados de una película, dado su nombre.
"2)Recomendar película": Recomienda la película mejor valorada en la tienda.
"2)Recomendar película por género": Recomienda la película mejor valorada en la tienda, según género.
"0)Menú anterior": Retorna al menú anterior.

"2)Servicios para cliente": Da acceso al siguiente submenú.

"1)Arrendar películas": Permite al cliente arrendar películas.
"2)Devolver películas": Permite al cliente devolver las películas arrendadas.
"3)Pagar deuda": Permite al cliente pagar su deuda.
"0)Menú anterior": Retorna al menú anterior.

"3)Información usuario": Da acceso al siguiente submenú.

"1)Desplegar mi ficha cliente": Muestra los datos almacenados del cliente.
"2)Revisar mi historial": Muestra el historial completo de películas arrendadas por el cliente.

"3)Revisar historial de película": Muestra el historial de solo una película.
"0)Menú anterior": Retorna al menú anterior.

"0)Menú anterior": Retorna al menú anterior.

El método menúAdmin despliega las siguientes opciones:

"1)Ingresar datos": Da acceso al siguiente submenú.

"1)Registrar Cliente": Registra un nuevo cliente en el sistema.
"2)Registrar Película": Registra una nueva película en el sistema.
"3)Registrar Historial": Se agrega una objeto arriendo a al historial de arriendos del cliente
"0)Menú anterior": Retorna al menú anterior.

"2)Mostrar datos": Da acceso al siguiente submenú.

"1)Desplegar lista de clientes": Muestra una lista con los datos de todos los clientes.
"2)Desplegar lista de películas": Muestra una lista con los datos de todas las películas que tiene la tienda.
"3)Desplegar historial de cliente": Muestra el historial de arriendos de un solo cliente.
"3)Desplegar datos de trabajador": Muestra los datos de un trabajador el cual su rut es ingresado por consola..
"0)Menú anterior": Retorna al menú anterior.

"3)Buscar datos": Da acceso al siguiente submenú.

"1)Buscar Película": Muestra los datos guardados de una película, dado su nombre.
"2)Buscar Cliente": Muestra los datos de un cliente, dado su nombre.
"0)Menú anterior": Retorna al menú anterior.

"4)Editar datos": Da acceso al siguiente submenú.

"1)Editar Película": Permite cambiar los datos de una película.
"2)Editar Cliente": Permite cambiar los datos de un cliente.
"3)Eliminar Película": Elimina la película indicada de las colecciones correspondientes.
"4)Eliminar Cliente": Elimina el cliente indicado desde las colecciones correspondientes.
"0)Menú anterior": Retorna al menú anterior.

"5)Servicios para cliente": Da acceso al siguiente submenú.

"1)Arrendar películas": Permite al cliente arrendar películas.
"2)Devolver películas": Permite al cliente devolver las películas arrendadas.
"3)Pagar deuda": Permite al cliente pagar su deuda.
"0)Menú anterior": Retorna al menú anterior.

"6)Generar reportes": Genera archivos con reportes que se guardan en carpeta Reportes.

"1)Deudores": Crea archivo que contiene el nombre de todos los clientes deudores
"2)Películas en arriendo": Crea archivo que contiene información del usuario y el nombre de las películas que tiene arrendadas actualmente.

"0)Menú anterior": Retorna al menú anterior.

3.2.1 Funcionalidad básica (1): Inserción manual o agregar elemento

Mediante el menú para administradores se permite el registro de nuevos Clientes y Películas a los ArrayList y hashMap de estas entidades en la clase VideoClub:

En Funciones.java:

public static void **registrarCliente**(VideoClub x);
public static void **registraPelicula**(VideoClub x);

En VideoClub.java:

public void **addPeliToListaPelis**(Pelicula peli);
public void **addClientToListaClients**(Cliente cliente);
public void **addClientToClientXRut**(String rut, Cliente cliente);
public void **addPeliToPelisXId**(String id, Pelicula peli);

También en la clase Cliente y mediante el menú de Administrador, se permite el ingreso de un nuevo objeto de la clase Arriendo:

En Funciones.java:

public static void **registrarArriendo**(VideoClub x);

En Cliente.java:

public void **addToHistorialXid**(Arriendo arriendo);
public void **addToHistorial**(Arriendo arriendo)
public void **addToArriendosActuales**(Arriendo arriendo)
public void **addToArriendosXid**(Arriendo arriendo)

3.2.2 Funcionalidad básica (2): Mostrar por pantalla listado de elementos

Para mostrar los elementos de la listaClientes debemos ir al menú de administrador codificado en la clase MenuTienda, donde se accede a la opción **"2)Mostrar datos"** y **"1)Desplegar lista de clientes"** la cual llama al método listaCliente(VideoClub) implementado en la clase Funciones, la cual llama a otro método implementado en la clase VideoClub llamado mostrarDatosPeliculas() la cual se encarga de mostrar por pantalla cada elemento de la lista con su s datos más importantes.

Para mostrar el historial de arriendo de un cliente, se puede acceder de dos formas dependiendo de si es un cliente o un administrador el que desea ver el array historialArriendos.

Caso de cliente:

Mediante la interfaz se accede al menú cliente ingresando el rut correspondiente, y selecciona la opción **"3)Información usuario"** y luego **"2)Revisar mi historial"** , la cual llama la función implementada en Funciones mostrarHistorialCliente(VideoClub x, String rut), la cual invoca a la función

mostrarHistorial() la cual se encuentra en la clase Cliente e imprime la lista de arriendos del cliente por pantalla.

Caso administrador:

Desde el menú administrador en la clase MenuTienda, está disponible la opción **"2)Mostrar datos"** y luego **"3)Desplegar historial de cliente"** la cual llama a un método ubicado en la clase Funciones llamado mostrarHistorialCliente(VideoClub x), la cual accede a otro método llamado mostrarHistorial() en la clase Cliente, el cual se encarga de mostrar por pantalla una lista con todos los arriendos del cliente.

A.2 Implementación de las siguientes funcionalidades en el menú:

A.2.1 Funcionalidad básica (1): Edición/modificación de elemento

En menú administrador opción **4** se puede acceder a la edición de ciertos elementos**:**

"1)Editar Película": Permite cambiar los datos de una película.
"2)Editar Cliente": Permite cambiar los datos de un cliente.

En menú administrador opción **5->2** y menú cliente opción **2->2** se puede acceder a la función ***delvolverArriendo()***, esta función al devolverse un arriendo, en caso de que no haya arrendado la película antes se agrega el objeto arriendo a las colecciones correspondientes, de lo contrario se ***editan/modifican*** los objetos ya almacenados en las colecciones correspondientes, las cuales se ubican en el segundo nivel de anidación.

A.2.2 Funcionalidad básica (2): Eliminación del elemento para las 2 colecciones anidadas.

En menú administrador opción **5->2** y menú cliente opción **2->2** se puede acceder a la función ***delvolverArriendo()***, esta función al devolverse un arriendo, elimina el objeto arriendo de las colecciones correspondientes, las cuales se ubican en el segundo nivel de anidación..

En menú administrador opción **4** se puede acceder a la eliminación de ciertos elementos.

"3)Eliminar Película": Elimina la película indicada de las colecciones correspondientes.
"4)Eliminar Cliente": Elimina el cliente indicado desde las colecciones correspondientes.

A.3 Se debe generar un reporte en archivo txt/csv que considere mostrar datos de las 2 colecciones anidadas

Las funciones que se encargan de generar reportes se encuentran en la clase VideoClub y son las siguientes:

escribirArchivoDeudores(): Crea archivo que contiene el nombre de todos los clientes deudores.(Primer nivel de anidación)
escribirArchivoArriendosActuales(): Crea archivo que contiene información del usuario y el nombre de las películas que tiene arrendadas actualmente.(Segundo nivel de anidación)

Se pueden acceder desde menú administrador opción **6**.

A.5 Todas las funcionalidades pueden ser implementadas mediante consola.

Las funcionalidades más importantes para el desarrollo del proyecto se pueden acceder directamente mediante el uso del menú por consola.

A.6 Link repositorio github

https://github.com/luis922/Proyecto-POO-2S-2021/tree/Mismo-proyecto

La rama principal del proyecto sobre la cual se trabaja es “Mismo-proyecto”

La rama correspondiente al. zip entregado es ***Mismo-Proyecto.***

4.1 Se deben incluir al menos 2 funcionalidades propias que sean de utilidad para el negocio

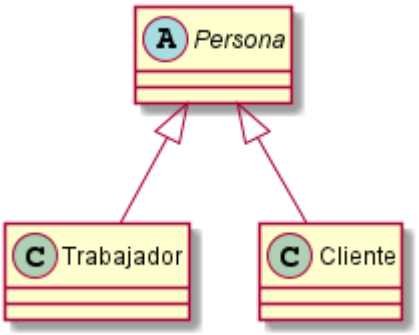
Seleccionar un objeto por criterio:

peliMejorEvaluada(): Muestra la película con mejor valoración.
peliMejorEvaluada(opción): Muestra la película con mejor valoración dado un género.

Subconjunto filtrado por criterio:

filtradoPorAño(tienda): Muestra un conjunto de películas estrenadas dentro de un rango de años.
filtradoPorValuación(tienda):Muestra un conjunto de películas cuya valoración se encuentra dentro de un rango dado.
filtradoPorGénero(tienda): Muestra un conjunto de películas que son del género ingresado.
filtradoDisponibles(tienda): Muestra un conjunto de películas disponibles para ser arrendadas.
filtradoDirector(tienda): Muestra un conjunto de películas dirigidas por la persona ingresada.
filtradoCalidad(tienda): Muestra un conjunto de películas que poseen una calidad igual a la solicitada.
filtradoDuración(tienda): Muestra un conjunto de películas cuya duración está entre un rango dado._

4.2 Diseño y codificación de 1 (una) clase abstracta que sea padre de al menos 2 (dos) clases. La clase abstracta debe ser utilizada por alguna otra clase (contexto)



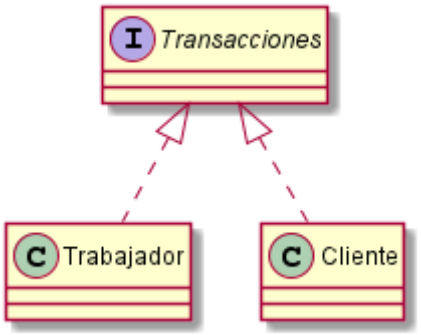
Se crea una clase abstracta de nombre Persona la cual contiene atributos nombre, rut, deuda y dos colecciones que contienen las películas que actualmente se han arrendado. En cuanto a métodos, tiene sus respectivos setters y getters,los relacionados a su función y el método abstracto de identificación que se implementa en sus subclases respectivas, las cuales son la clase Cliente y la clase Trabajador.

La clase Cliente tiene dos atributos que contienen una colección del historial de películas arrendadas por una persona, además de la implementación del método abstracto identificación, el cual nos muestra el nombre y rut de la persona y si esta es un cliente nuevo o antiguo.

La clase Trabajador tiene los atributos cargo, sueldo y vecesArriendosAtrasados, además de la implementación del método abstracto identificación, el cual nos muestra el nombre, rut de la persona y el cargo que esta ostenta dentro de la tienda.

La clase abstracta Persona se utiliza para llevar registro de la personas que han ingresado a la tienda, las cuales se almacenan en un Hashmap llamado presentesXRut y las cuales se muestran siempre al salir del menú inicial (antes de terminar el programa) a través de la función mostrarPresentes en clase VideoClub la cual también hace uso de la implementación del método abstracto identificación presente en las subclases.

4.3 Diseño y codificación de 1 (una) interfaz que sea implementada por al menos 2 (dos) clases. La interfaz debe ser utilizada por alguna otra clase (contexto)



La clase Transacciones es la interfaz que se ha creado para este negocio, la cual consta de tres métodos no implementados: arrendar(), devolverArriendo() y pagarDeuda().

La clase Cliente y Trabajador son las encargadas de implementar estos métodos dentro de su estructura, los cuales son accesibles a través de la opción “Sevicios para cliente” en los menús respectivos.

4.4 Diseño y codificación de 2 (dos) clases que utilicen sobreescritura de métodos.

Por lo hablado anteriormente durante la ayudantía de esta semana hemos considerado como sobreescritura, la implementación de los métodos abstractos de la superclase y los métodos vacíos de la interfaz.

La clase **Cliente** se encarga de sobrescribir 4 métodos:

- **identificación()** : Método abstracto proveniente de la superclase, que imprime por pantalla datos del cliente.
- **arrendar()**: Método que permite al Cliente arrendar una película.
- **devolverArriendo()**: Método que permite al Cliente devolver un arriendo.
- **pagarDeuda()**: Método que permite al Cliente pagar su deuda.

La clase **Trabajador** se encarga de sobrescribir 4 métodos:

- **identificación()** : Método abstracto proveniente de la superclase, que imprime por pantalla datos del trabajador.
- **arrendar()**: Método que permite al trabajador ingresar un arriendo de un cliente.
- **devolverArriendo()**: Método que permite al trabajador devolver un arriendo de un cliente.
- **pagarDeuda()**: Método que permite al trabajador ingresar el pago de una deuda de un cliente.

B.1 Implementación de al menos tres ventanas gráficas.

Las funcionalidades del proyecto ahora son accesibles a través de menús con interfaces gráficas implementadas usando principalmente la librería Swing.

No todas las funcionalidades han sido implementadas gráficamente, por lo que se incluye una imagen del diseño de menú empleado y cliente para facilitar la identificación de

estos métodos en la carpeta del proyecto (Diagrama menu empleado y Diagrama menu cliente), para que se pueda visualizar con facilidad.

- **1 Ventana de menú:** Se han implementado ventanas gráficas para todos los menús del proyecto (Menú de Login, Empleado y Cliente). La implementación del menú login y cliente son diferentes de la de menú empleado, ya que las primeras fueron creadas a través del ayudante gráfico de netbeans, mientras que el menú empleado fue implementado mediante la escritura directa de código. Dicho esto, todos los menús se encuentran en un estado funcional.

- **1 Ventana de agregar elementos:** En ***menú empleado -> gestión de datos-> Ingresar datos***, se encuentra la ventana gráfica ***Registrar nuevo cliente***, la cual permite ingresar los datos de un nuevo cliente y registrarlos en el sistema.

- **1 Ventana de mostrar elementos:** En ***menú cliente -> Película***, se encuentra la ventana gráfica ***Desplegar catálogo de películas***, la cual se encarga de mostrar el catálogo de películas que la tienda posee, junto a la información relevante respectiva.

Entrega final

E.F.1 Crear 2 clases que extiendan de una Excepción y que se utilicen en el programa

Se crea la clase FechaInvalidaException la cual se lanza cuando la fecha de arriendo y/o entrega no cumplen con el formato deseado, desde el método formatoCorrectoFecha(String fecha) ubicado en la clase Funciones. Este método es usado en la función nuevoArriendo() incluido también en la clase Funciones, el error es “atrapado” mediante las sentencias try catch incluida en este método. Se puede verificar su uso al arrendar una película e ingresar fechas con formato erróneo.

Además, se agrega la clase RutInvalidoException que se lanza cuando el rut ingresado no cumple con el formato pedido, desde el método formatoCorrectoRut(String rut) ubicado en la clase Funciones, el error es atrapado usando las sentencias try-catch en las clases RegistrarCliente, Funciones y LoginGUI.

E.F.2 Implementar el manejo de excepciones capturando errores potenciales específicos mediante Try-catch

Como se mencionó anteriormente, se capturan errores mediante sentencias Try-catch para poder decidir qué hacer en caso de que se detecte una fecha, rut o monto ingresado con formato invalido, específicamente en los siguientes métodos:

nuevoArriendo(): En clase funciones, app línea 550 (*FechaInvalidaExcpetion*)

pagarDeuda(): En clase funciones, cliente y trabajador, app línea 629, 213, 143 respectivamente (*InputMismatchException*)

Se captura la Excepción RutInvalidoException con la sentencia try-catch, al ingreso de un rut para hacer un login en la aplicación o para registrar un cliente, específicamente en los métodos:

Usos mediante aplicación gráfica:

buttonAceptarActionPerformed(): En clase LoginGUI, función que se ejecuta cuando se genera una acción sobre el botón aceptar. Aproximadamente en la línea 155 de la clase.

actionPerformed(): En la clase RegistrarCliente, función que se ejecuta cuando se genera un evento, se valida que el rut ingresado para hacer el registro del cliente sea válido. Aproximadamente en la línea 77 de la clase.

Usos mediante consola:

loginTrabajadores(): En clase Funciones, verifica que el rut sea válido al ingresar como trabajador en el menú por consola. Aproximadamente en la línea 42.

loginClientes(): En clase Funciones, verifica que el rut sea válido al ingresar como cliente en el menú por consola. Aproximadamente en la línea 61.

registrarCliente(): En clase Funciones, verifica que el rut sea válido al registrar un nuevo cliente por consola. Aproximadamente en la línea 218.

E.F.3 Aplicación del patrón de diseño Strategy (Estrategia)

No se implementan patrones de diseño