UNIVERSIDAD CATOLICA DEL NORTE

FACULTAD DE INGENIERÍA Y CIENCIAS GEOLÓGICAS

**Taller Numero 1**

**Programación Avanzada.**

**Profesor a cargo:**Tomás Alberto Reiman Beltran.

**Ayudante :**Edgardo Antonio Ortiz Gonzales.

**Integrantes.**

-Ángel Cuevas.

- Amaro Acuña

**Rut:**

**-**21.009.361-3

-20.785.596-0

**Correos:**

**-**[angel.cuevas@alumnos.ucn.cl](mailto:angel.cuevas@alumnos.ucn.cl)

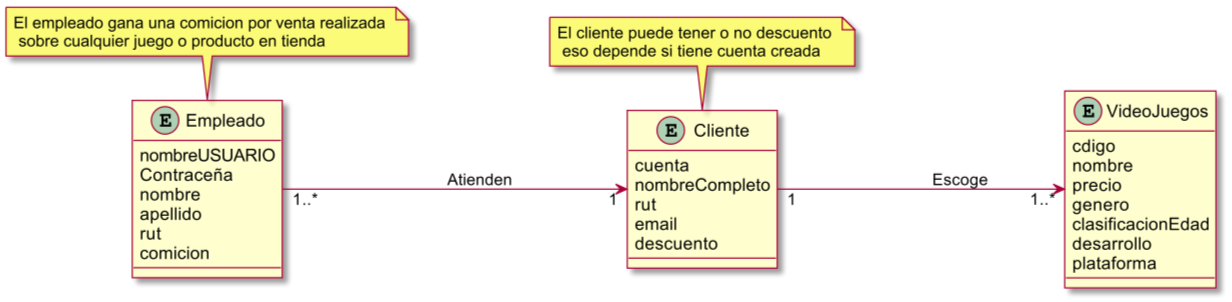
-[amaro.acuna@alumnos.ucn.cl](mailto:angel.cuevas@alumnos.ucn.cl)

**Paralelo.**

**-**C1

Explicacion del codigo:

* Dentro del enunciado del taller número 1 logramos destacar 3 entidades visibles que serían las de **Empleados**,**Clientes** y las de **videojuegos** de las cuales cada una tendría los siguientes parámetros.



Este modelo muestra las tres entidades que reconocimos en el contexto del taller 1, que consiste en una tienda de videojuegos con: Empleado, Cliente y VideoJuegos.

Empleado: Esta entidad representa a los empleados que trabajan en la tienda de videojuegos. Tienen atributos como nombre de usuario (nombreUSUARIO), contraseña, nombre, apellido, rut (identificación personal) y comisión. La comisión indica el porcentaje de ganancia que el empleado recibe por cada venta que realiza en la tienda.

Cliente: Esta entidad representa a los clientes que compran en la tienda de videojuegos. Tiene atributos como cuenta (posiblemente un identificador de cuenta en la tienda), nombre completo, rut, email y descuento. El descuento indica si el cliente tiene algún tipo de descuento especial, lo que puede depender de si tienen una cuenta creada en la tienda.

VideoJuegos: Esta entidad representa los videojuegos que se venden en la tienda. Tiene atributos como código, nombre, precio, género, clasificación de edad, desarrollo (estudio que lo creó) y plataforma en la que se puede jugar.

Las notas asociadas a las relaciones entre las entidades brindan información adicional:

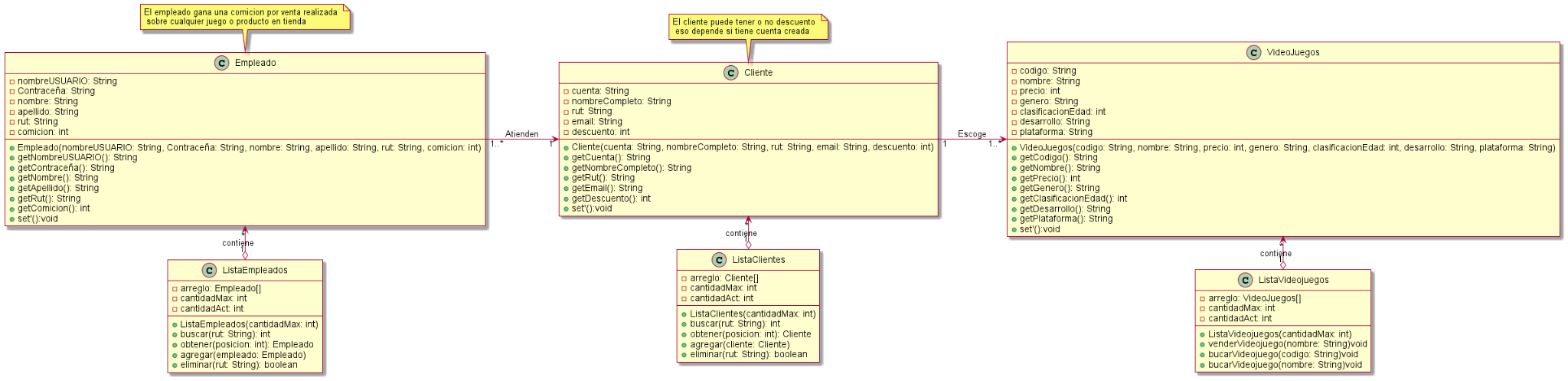
N1: Indica que el cliente puede tener o no descuento, lo que depende de si tienen una cuenta creada en la tienda. Esto sugiere una relación entre la existencia de una cuenta y la posibilidad de tener un descuento.

N2: Indica que los empleados ganan una comisión por cada venta realizada en la tienda, independientemente del producto vendido. Esto resalta la importancia de la relación entre los empleados y las ventas de la tienda.

Las flechas entre las entidades representan las interacciones entre ellas:

Diagrama de Clases.

(Nota se tiene que hacer zoom para apreciar mejor el diagrama)



Explicación: Tenemos 3 clases la de Empleados, la de Clientes y la de Videojuegos las cuales cada una tendrá sus parámetros y sus correspondientes listas.

1. **Empleados:** como atributos estas tendrán **Nombre de usuario, contraseña, nombre, apellido, rut y comisiones**. De los cuales iniciaremos con su propio constructor ingresando los parámetros que definimos anteriormente, también ingresamos sus respectivos get´s() que cada uno pedirá los datos como String exceptuando las comisiones debido a que estos valores serán un dato numérico.

**Lista Empleados:** Definiremos el arreglo que vendría a llamar a la clase Empleados para sacar la información de allí y almacenarla, también definimos la cantidad Máxima y Mínima de la lista y agregaremos las opciones.

* Buscar: Se ingresa el rut del trabajador el cual el programa tendrá que retornar los parámetros definidos.
* obtener: Se listará todos los datos almacenados de los empleados guardados en la lista.
* agregar: se preguntaran por pantalla todos los parámetros del empleado y una vez ingresado se almacena en la lista.
* eliminar: Se preguntará el rut del Empleado para de esa manera eliminar la información que se encuentra en la propia lista.

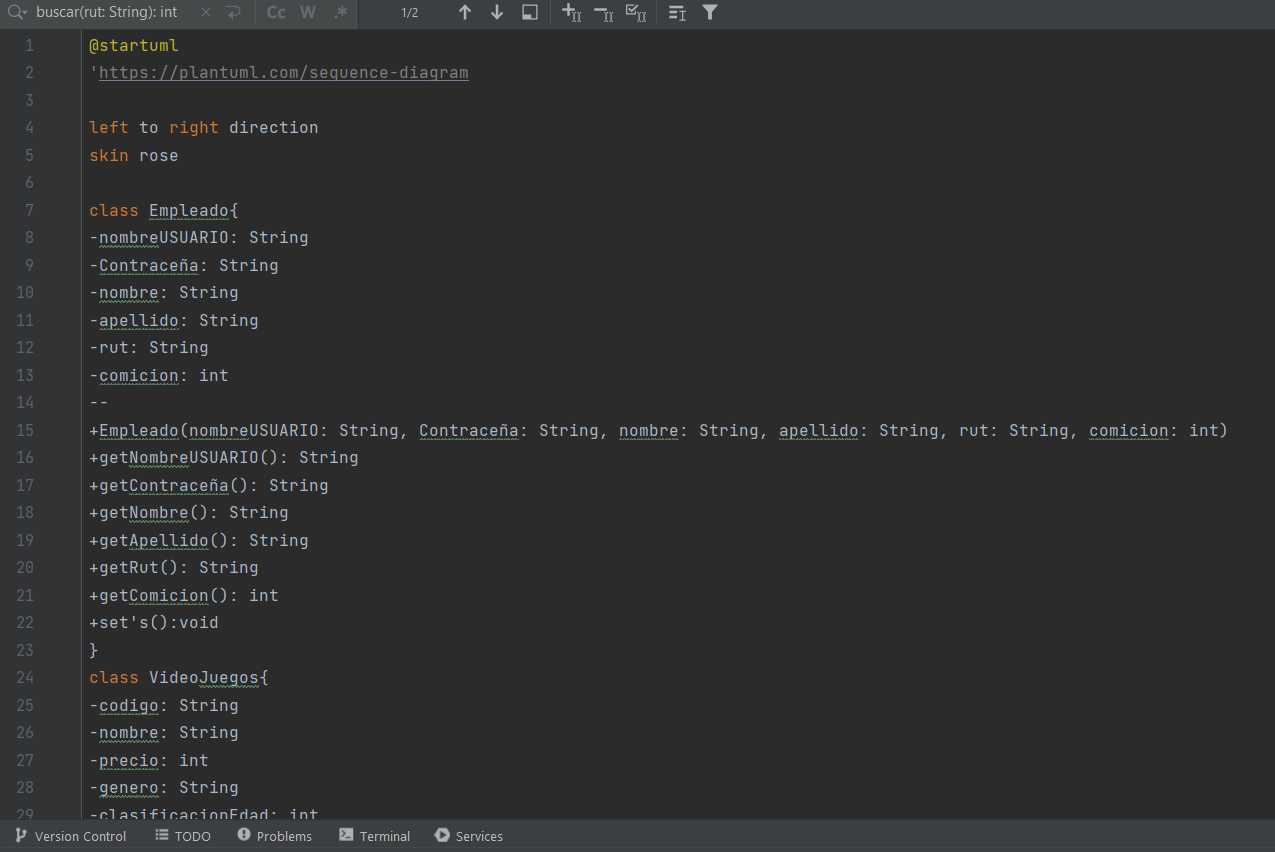
1. **Cliente:** al igual que en la clase empleados se definirán los parámetros del cliente los cuales son **cuenta,Nombre ,rut ,email ,descuento** igualmente se necesitara de un constructor y se pedirán los datos a través de los get´s() los cuales todos serán de tipo String con excepción del descuento que será de tipo entero y en caso de ser necesario se añadieron los set´s() para cambiar algún valor en caso de ser necesario.

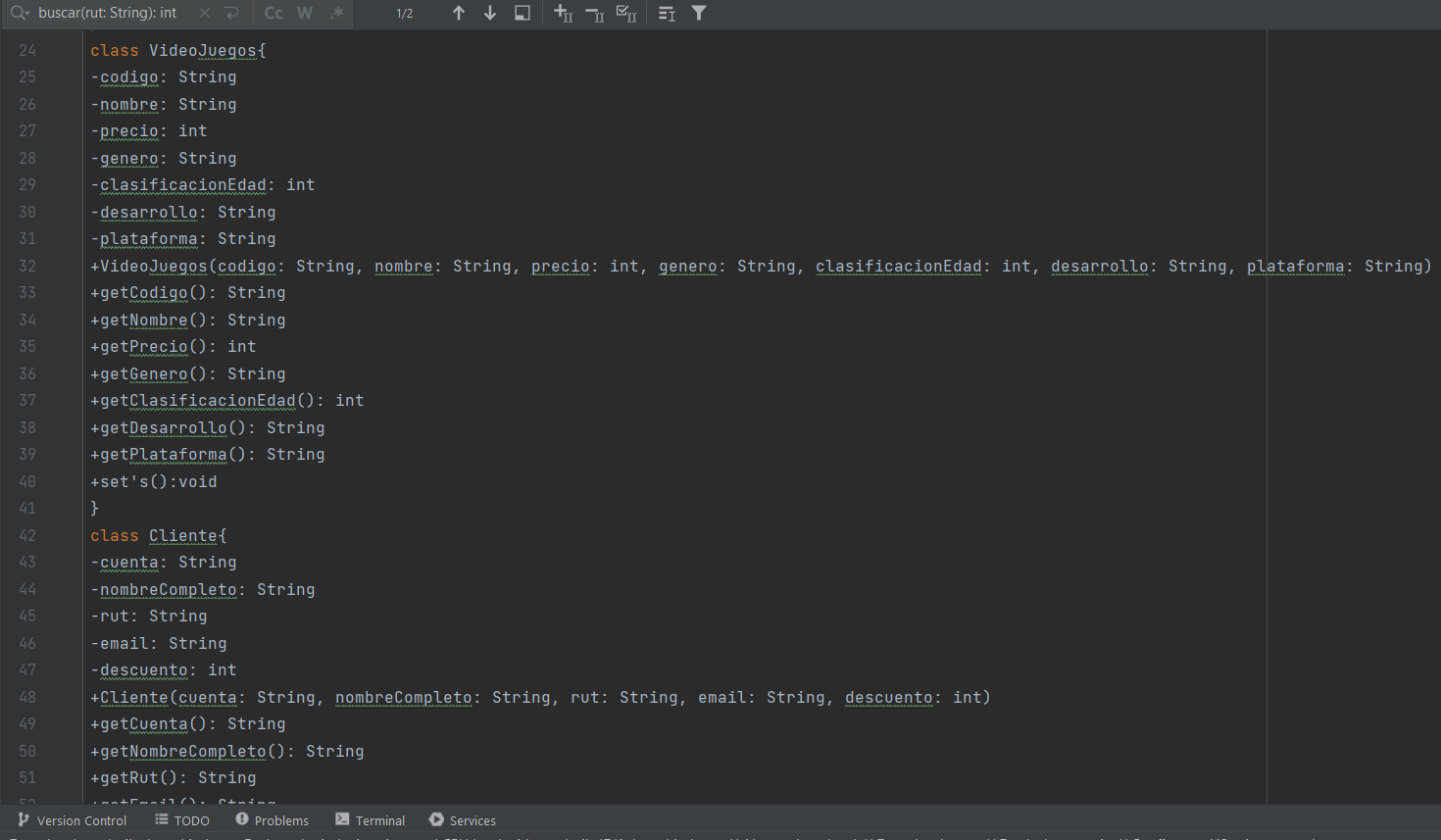
**Lista Cliente:**Tendrá las mismas opciones que la lista Empleado se necesitará un arreglo que llame a los datos de la clase cliente una cantidad máxima, mínima, se podrá buscar, agregar, obtener y eliminar, con la diferencia que ahora se tendrá que pedir el rut del cliente para poder buscar en su respectiva lista.

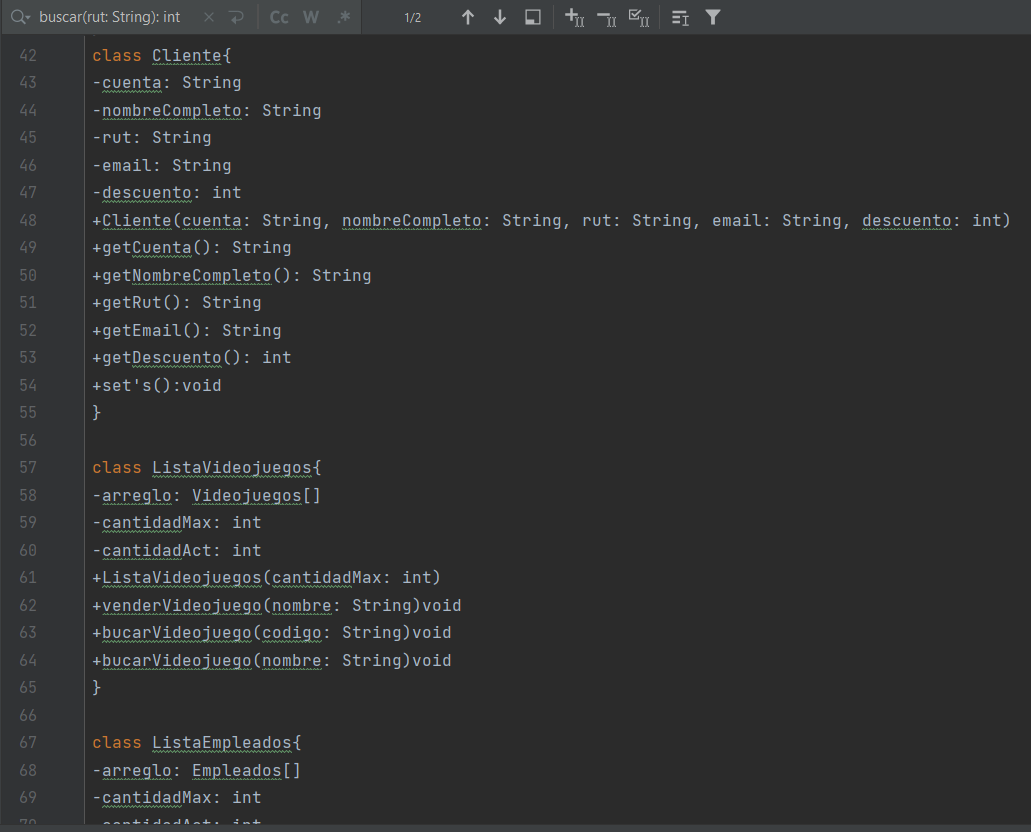
1. **Video Juegos:** Los parámetros de esta clase serán **código, nombre, precio, género, clasificación, desarrollo, plataforma** de los cuales será la misma metodología que las anteriores clases, con constructor, get´s() y set´s() todas de tipo String con excepción del precio y la clasificación.

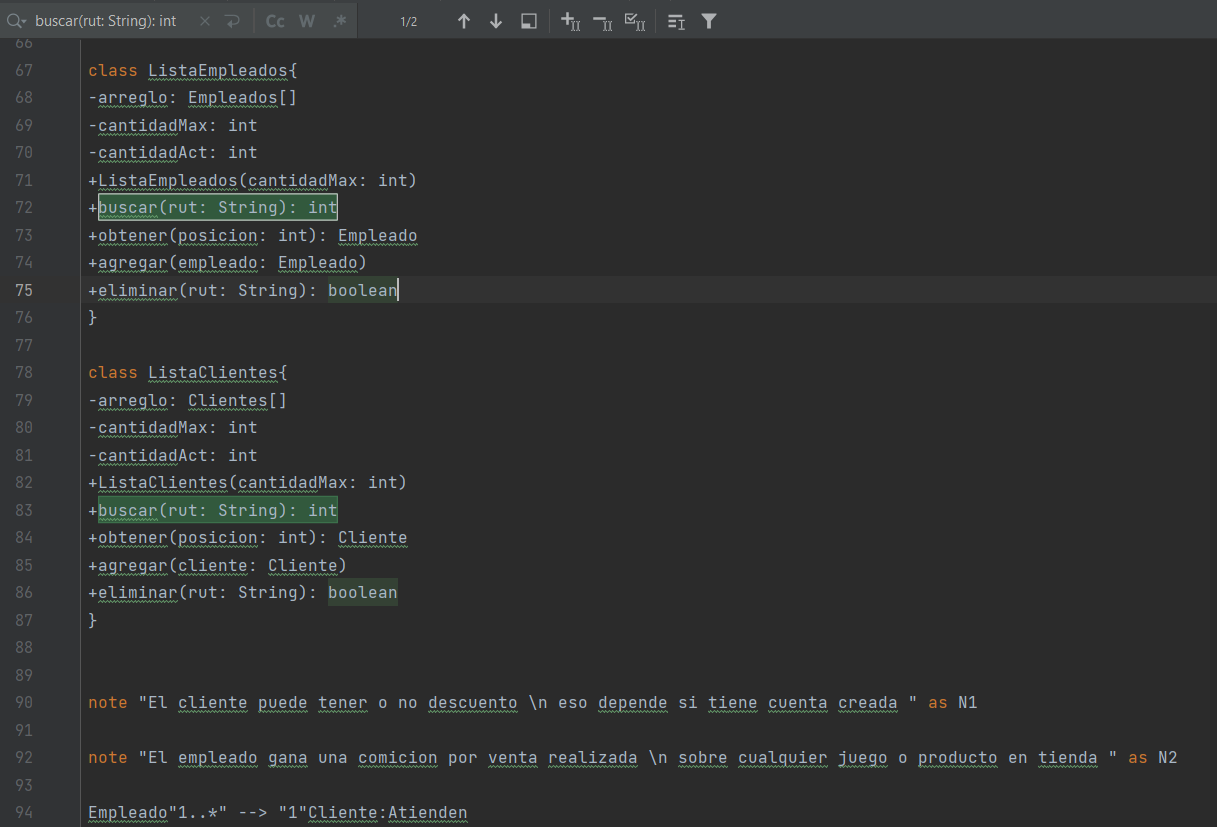
**Lista VideoJuegos:** inicia igual que las otras listas con el arreglo, la cantidad máxima,mínima y se podrá buscar el juego a través de el código o el nombre y ver la opción de venderlo.

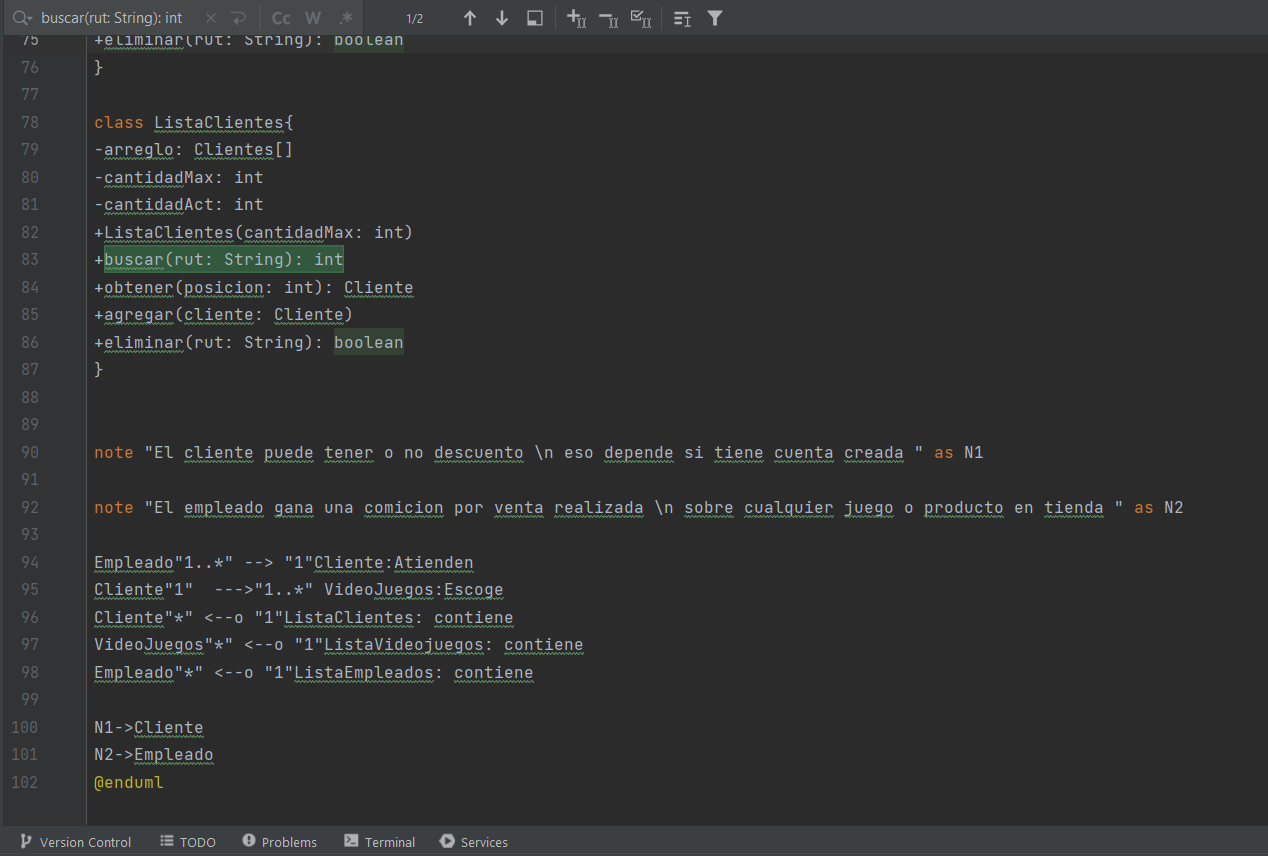
* código del diagrama de clase:











**Explicación general del código :**

* Inicialización de las listas: Se crean tres listas para almacenar empleados, clientes y videojuegos. Cada una se inicializa con una capacidad máxima de 9999 elementos.
* Lectura de archivos de datos: Se leen los datos de archivos de texto (videojuegos.txt y empleados.txt) para cargar información inicial sobre los videojuegos disponibles y los empleados de la tienda en las listas correspondientes.
* Menú principal: Se muestra un menú principal que permite al usuario iniciar sesión o cerrar el programa.
* Segundo menú (iniciar sesión): Si el usuario elige iniciar sesión, se le pide que ingrese su nombre de usuario y contraseña.
* Si las credenciales coinciden con las de un empleado en la lista, se muestra un segundo menú con opciones adicionales.
* Tercer menú (opciones del empleado): Este menú ofrece varias opciones al empleado, como vender un videojuego, buscar un videojuego, ver estadísticas, etc.
* Opciones del menú estadística: Este menú ofrece opciones relacionadas con estadísticas de ventas y rendimiento, como el videojuego más vendido, la plataforma con más ventas, ventas de clientes registrados, imprimir ventas totales y mostrar al trabajador con más ventas.
* Funciones estadísticas: Cada opción del menú estadístico tiene su propia función para realizar la tarea específica, como encontrar el videojuego más vendido, la plataforma con más ventas, etc.
* Retorno al menú anterior: Después de realizar una acción en el menú estadístico, el programa vuelve al mismo menú para permitir al usuario elegir otra opción o volver atrás.
* función número 4 del menú de estadísticas (imprimir ventas totales)

Recibe tres parámetros: contenedorVideojuegos, contenedorEmpleado y contenedorCliente, que son para llamar nuestras clases ListaVideoJuegos, ListaEmpleado y ListaCliente, respectivamente. Estas listas contienen información sobre los videojuegos disponibles, los empleados de la tienda y los clientes registrados.

Itera sobre la lista de clientes (contenedorCliente) utilizando un bucle for. El bucle recorre todos los clientes registrados en la lista.

En cada iteración, obtiene un cliente de la lista usando el método optener(i) de la clase ListaCliente y lo asigna a la variable listar.

Imprime información sobre las ventas totales del cliente actual. La cantidad de ventas se obtiene utilizando el método getVenta() del objeto listar.

Después de imprimir la información de todas las ventas totales de los clientes, llama a la función menuEstadistica, pasando como argumentos las listas de videojuegos, empleados y clientes.

* función número 5 del menú de estadísticas (trabajador con más ventas)

Recibe tres parámetros: contenedorVideojuegos, contenedorEmpleado, y contenedorCliente, que parecen ser instancias de las clases ListaVideoJuegos, ListaEmpleado, y ListaCliente, respectivamente. Estas listas contienen información sobre los videojuegos disponibles, los empleados de la tienda y los clientes registrados.

Itera sobre la lista de empleados (contenedorEmpleado) utilizando un bucle for. El bucle recorre todos los empleados registrados en la lista.

En cada iteración, obtiene un empleado de la lista usando el método optenerEmpleado(i) de la clase ListaEmpleado y lo asigna a la variable listar.

Imprime información sobre el empleado actual, incluyendo su nombre de usuario y la cantidad de comisiones generadas por las ventas de videojuegos. La comisión se obtiene utilizando el método getComicion() del objeto listar.

Después de imprimir la información de todos los empleados, llama a la función menuEstadistica, pasando como argumentos las listas de videojuegos, empleados y clientes.

**Tiempo dedicado al taller 1:**

* 2 horas para el diseño
* 39 horas para codificar
* 2 horas para el informe(entrega 1)
* 2 horas para el informe(entrega 2)
* 3 horas para el informe (ultima entrega)

En resumen el tiempo total que tomó la realización de este primer taller fue de aproximadamente 48 horas para Ángel Cueva y Amaro Acuña en conjunto en la plataforma discord.