Informe taller 1

Programación Avanzada

Parte 3

Integrantes:

Nombres:	**Rut:**	**Correo:**
• Diego Franco Cortés	21.603.362-0	diego.cortes07@alumnos.ucn.cl
• Rodrigo Aguilera	21.760.570-9	rodigo.aguilera01@alumnos.ucn.cl

Paralelo: 10796 – C1

Fecha: 28 de abril de 2024

Profesor: Tomas Alberto Reimann Beltran

Ayudantes: Edgardo Antonio Ortiz Gonzales - Marcelo Antonio Cespedes Arqueros

Requisitos

Creación y comprensión de modelos de dominio

Diagramas de clase

Implementación aplicación Java

Se nos solicita realizar el desarrollo de una aplicación capaz de procesar lectura de un archivo y almacenamiento de datos. Realizando operaciones para el correcto funcionamiento de una aplicación que será utilizada como “caja”, permitiendo la venta de videojuegos, almacenando datos sobre clientes, empleados y los propios videojuegos. Generando estadísticas según las ventas.

Puntos para considerar:

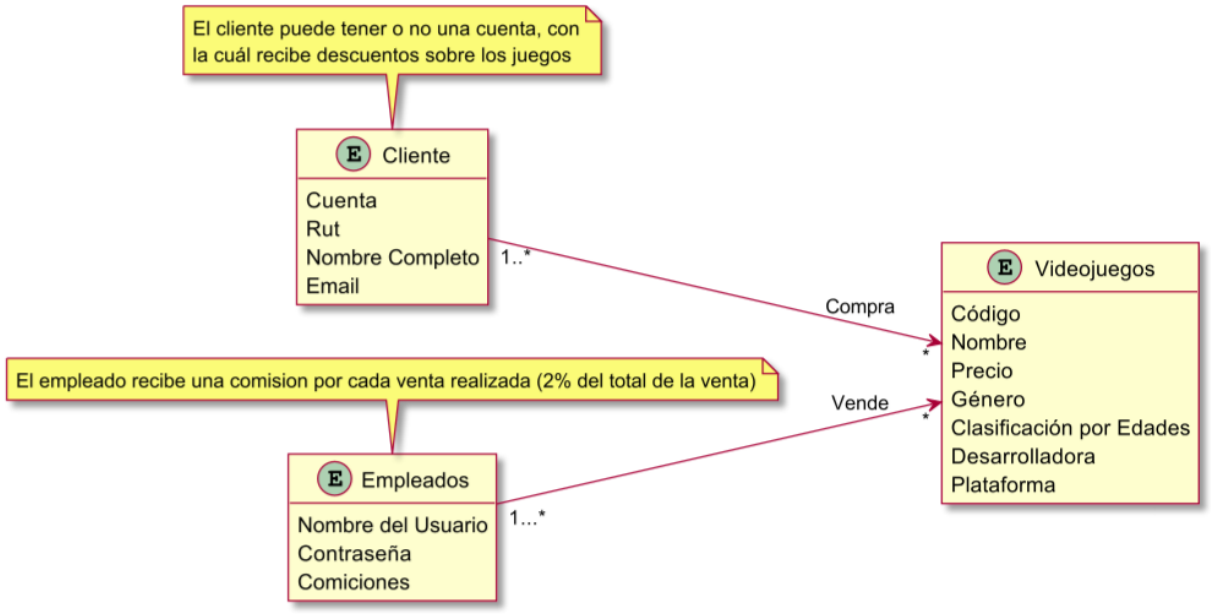
- Lectura de 2 Archivos, lista de los empleados y listado de videojuegos
- Al Inicializar la aplicación, se debe verificar la cuenta de un empleado, leído en la lectura de archivo.
- Un cliente puede poseer una cuenta de **miembro**

Se debe crear un menú donde se puedan realizar las siguientes operaciones:

- ✓ Vender un videojuego
- ✓ Buscar un videojuego
- ✓ menú estadísticas
- ✓ Cerrar la aplicación

Se considerará la creación de un modelo de dominio, diagrama de clases y posteriormente la aplicación en Java.

Modelo de dominio



Para la creación de un modelo de dominio, tomamos en consideración las 3 siguientes instancias importantes.

Empleado: como la entidad sobre la que se regirán las ventas, considerando sus identificadores únicos como nombre de usuario, contraseña. Además, añadiendo las comisiones que obtendrá según la venta.

Cliente: como la entidad que realiza la compra, que puede o no tener una cuenta de miembro. Posee un Rut, nombre completo del cliente y un email.

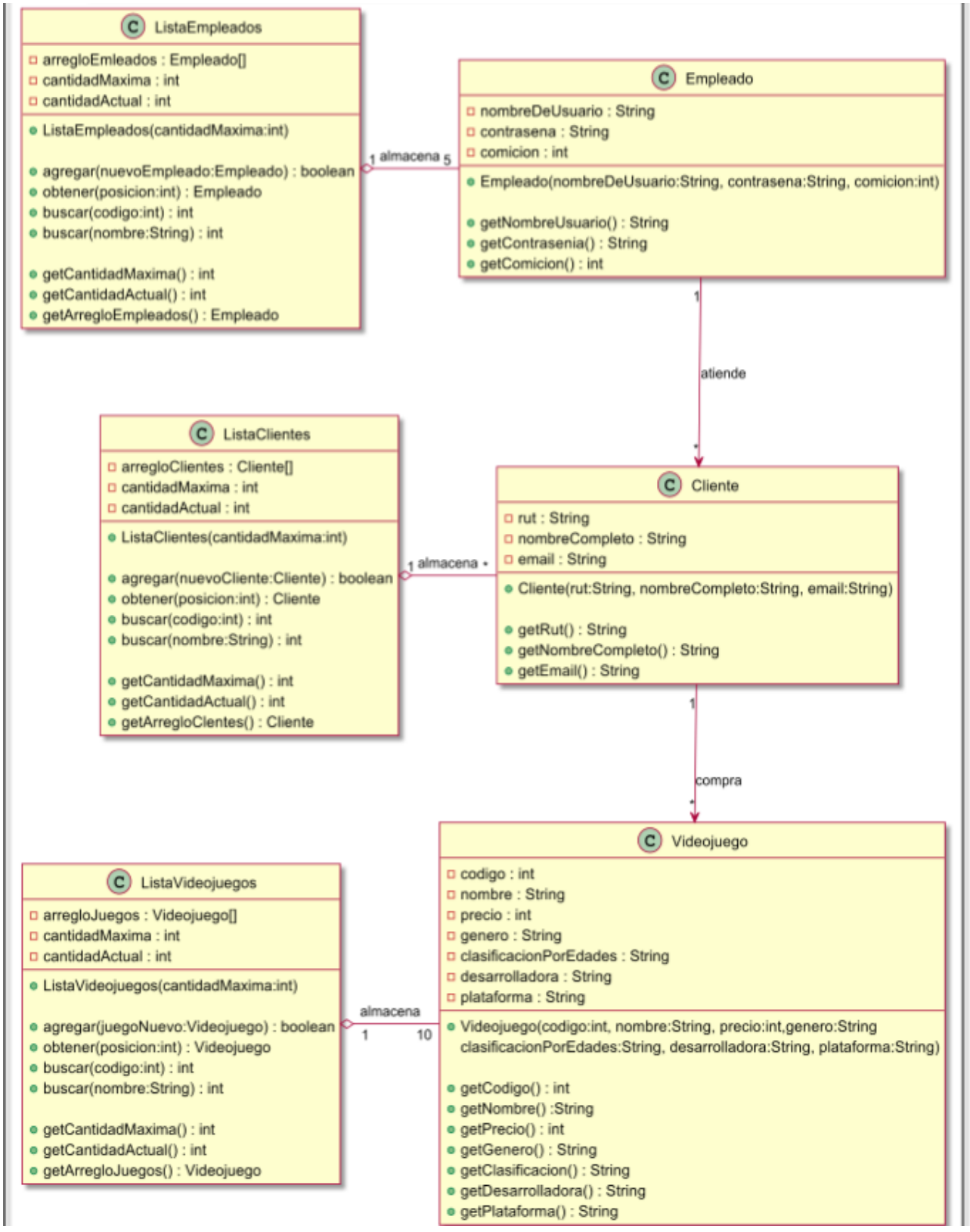
Videojuego: la entidad más importante en relación con las operaciones del programa, que se identifica con un código, nombre, precio, genero, clasificación edades, desarrolladora y plataforma.

Se relaciona de tal manera que:

Uno o muchos empleado administran muchas ventas de videojuegos, uno o muchos clientes compran muchos videojuegos.

Tiempo de dedicación taller (horas)		
Integrantes	Modelo del Dominio	Informe del taller
Rodrigo Aguilera	1 horas	0 hora
Diego Cortes	0 horas	1 hora

Diagrama de clases



Clases

Empleado

Posee nombre de usuario (String); contraseña (String) y las comisiones de venta (int).

Métodos:

En esta clase se encuentra un método constructor (crear una instancia empleado); y métodos get para obtener los datos de la instancia empleado.

ListaEmpleados

Posee un arreglo para almacenar diferentes instancias de empleados ([] Empleado); cantidad máxima de la lista (int); cantidad actual de la lista (int).

Métodos:

Método Constructor (crear el arreglo ListaEmpleados); métodos para buscar un empleado por nombre de usuario o contraseña (String); método para obtener un empleado (Empleado) ; y métodos get para obtener los datos de la lista.

Cliente

Posee rut (String); Nombre (String); Email (String)

Métodos:

Método Constructor (crear instancia de cliente); métodos get para obtener los datos de la instancia cliente.

ListaClientes

Posee un arreglo para almacenar diferentes instancias de clientes ([] Cliente); cantidad máxima de la lista (int); cantidad actual de la lista (int).

Métodos:

Método Constructor (crear el arreglo ListaCliente); método para agregar un cliente (boolean); método para buscar un cliente por rut del usuario (String); método para obtener un cliente (Cliente) ; y métodos get para obtener los datos de la lista.

Videojuego

Posee un código (int); nombre (String); precio (int); genero (String); Clasificación (String); desarrolladora (String); plataforma (String).

Métodos:

Método constructor (crear una instancia de videojuego); y métodos get para obtener los datos de la instancia videojuego.

ListaVideojuegos

Posee un arreglo para almacenar diferentes instancias de videojuegos ([] Videojuego); cantidad máxima de la lista (int); cantidad actual de la lista (int).

Métodos:

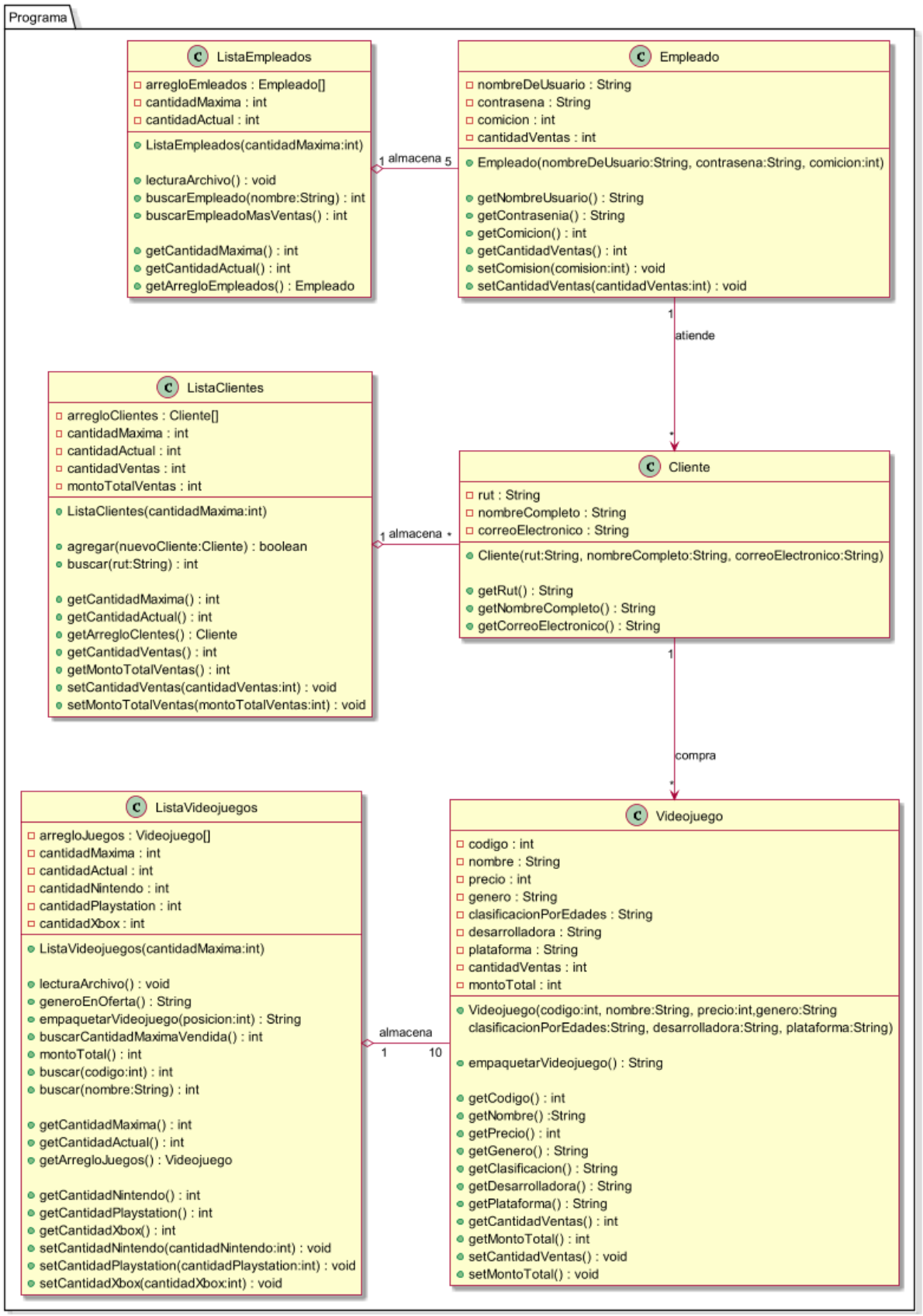
Método Constructor (crear el arreglo ListaVideojuegos); método para agregar un Videojuego (boolean); métodos para buscar un juego por código o nombre (String); método para obtener un videojuego (Videojuego) ; y métodos get para obtener los datos de la lista.

Se relaciona de tal manera que:

Un empleado se encuentra registrado en una lista de empleados, un cliente puede o **no** estar registrado en una lista de clientes y un videojuego se encuentra almacenado en una lista de videojuegos.

Uno o muchos empleados administran muchas ventas de videojuegos, uno o muchos clientes compran muchos videojuegos.

Diagrama de Clases Final



Tiempo de dedicación taller (horas)		
Integrantes	Diagrama de clase	Informe del taller
Rodrigo Aguilera	3 horas	0.5 hora
Diego Cortes	1 horas	1.5 hora

Implementación aplicación en java

Para la realización de la aplicación necesitamos;

Crear las **clases** listas para guardar la información de; Videojuegos, Empleados, Clientes. Luego leer los archivos de texto correspondientes y guardarlos en las listas, para eso usamos una función:

La lectura de archivo se realiza **dentro** de las clases listas, Existente en las listas Videojuegos y Empleados.

```
1 usage   DiegoCortes07 +1
public void LecturaArchivo() throws IOException {

    Videojuego nuevoVideojuego;

    In archivoEntrada = new In( s: "Juegos.txt");

    while (!archivoEntrada.isEmpty()) {

        String[] separacionVideojuego = archivoEntrada.readLine().split( regex: ",");

        int codigo = Integer.parseInt(separacionVideojuego[0]);
        String nombre = separacionVideojuego[1];
        int precio = Integer.parseInt(separacionVideojuego[2]);
        String genero = separacionVideojuego[3];
        String clasificacionPorEdades = separacionVideojuego[4];
        String desarolladora = separacionVideojuego[5];
        String plataforma = separacionVideojuego[6];

        nuevoVideojuego = new Videojuego(codigo,nombre,precio,genero,clasificacionPorEdades,desarolladora,plataforma);

        this.listaVideojuegos[this.cantidadActual] = nuevoVideojuego;
        this.cantidadActual++;
```

En **Main** solo tenemos una función llamada **menú ()**

menú ()

En la función menú se inicializan las listas con un valor exagerado (999). luego se realiza el llamado a la lectura de los archivos en las listas.

Luego de que ocurra lo anterior, se procede a ingresarse un menú por pantalla.

Un **menú inicial** donde el empleado debe ingresar sus datos.

Menú Inicial:

```
// se crea un menu para el inicio de sesion de los empleados
boolean menuinicial = false;

String generoOferta = listaVideojuegos.generoEnOferta();


while (!menuinicial) {


    StdOut.println("\n**** [Bienvenido al sistema de ventas] ****");
    StdOut.println("Por favor, Identifique su identidad");
    StdOut.println("****************************************");
    StdOut.println("ingrese una opcion para continuar:");
    StdOut.println("[1] Iniciar Sesion ");
    StdOut.println("[2] Cerrar programa");

    StdOut.print("\nOpcion: ");
    String opcion = StdIn.readString();

    switch (opcion) {
```

Se ingresa sesión por teclado, se verifica que los datos del empleado al iniciar sesión sean correctos (nombre, contraseña). Si la información ingresada existe, El programa puede proseguir con el MenuPrincipal ();

En el menú inicial, además, se deja inicializado cual será el genero que se encuentra en oferta para las futuras ventas.

MenúPrincipal ();

```
boolean menu = false;

while (menu != true) {

    StdOut.println("El genero en oferta es : "+generoOferta+"!!");

    StdOut.println("*****************************");
    StdOut.println("      [Menu Principal]");
    StdOut.println("*****************************");

    StdOut.println("Ingrese una opcion: ");
    StdOut.println("[1] Vender videojuego");
    StdOut.println("[2] Buscar videojuego");
    StdOut.println("[3] Menú estadisticas");
    StdOut.println("[4] Salir");

    StdOut.print("\nOpcion: ");
    String opcion = StdIn.readString();

    switch (opcion) {
```

En el menú principal, se imprime por pantalla el genero y se procede a preguntar por la opción a realizar

Opciones:

1) Vender un videojuego

Esta opción, la realizamos dentro de otra función para realizarlo de manera más autónoma, función llamada vendervideojuego ();

Esta función se encarga de preguntar por el nombre del videojuego a comprar, verificar que exista en la lista videojuegos, luego se pregunta al cliente si posee una cuenta de miembro.

Si el cliente **posee** una cuenta de miembro, se verifica ingresando su rut, si el rut existe en la lista de los empleados, la cuenta miembro existe.

Si el cliente **no posee** una cuenta de miembro, se pregunta por pantalla si desea ser un miembro.

```
StdOut.println("\nExisten descuentos exclusivos por ser miembro!! ");

StdOut.println("\nDesea ser miembro?");

StdOut.println("[1] SI");
StdOut.println("[2] NO");

String opcionMiembro = StdIn.readString();
```

Si desea ser miembro, se preguntan los datos para crear un miembro (rut,nombre,contraseña) **Si el rut ingresado ya existe en la lista clientes**, no se permite la creación de este nuevo cliente y se interrumpe la compra.

Si no desea ser miembro, se prosigue la compra del videojuego.

Se tiene en cuenta aun el **genero en oferta** que asignamos en el menú inicial.

Finalmente, se pide concretar la compra y aquí ocurren dos situaciones.

Que sea miembro; si el cliente ha creado una cuenta o se ha logrado identificar una cuenta como miembro en la lista, se verifica el género del videojuego a comprar con el genero que se encuentra en descuento y si coinciden, el precio del videojuego se ve reducido un 30%.

Que no sea miembro; si el cliente no ha optado por ser cliente miembro de la tienda, se prosigue a hacer la compra de forma normal, sin existir la posibilidad de (**genero oferta**).

Al concretar la compra, se realizan los cálculos de los acumuladores y contadores, que se utilizaran en las estadísticas.

2) Buscar un videojuego

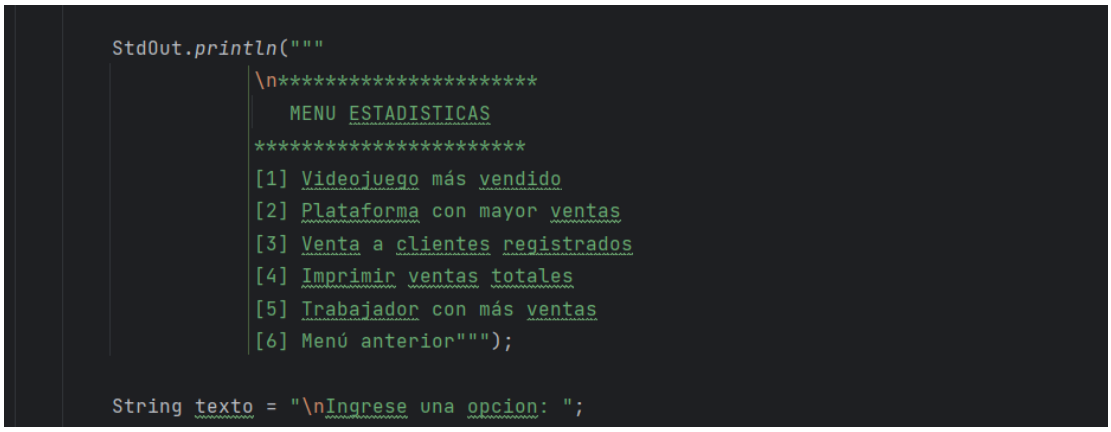
Se pide que se especifique el modo para buscar el videojuego, siendo posible buscar por código(**índice**) del videojuego o por el **nombre**.

En ambos casos se busca verificar que el videojuego buscado exista y se imprima por pantalla todos sus datos.

```
private static void buscarVideojuego(ListaVideojuegos listaVideojuegos) {

    int posicionJuego;

    boolean buscarvideojuegoMenu = false;
    String opcion;

    while (buscarvideojuegoMenu != true) {

        StdOut.println("\neliga una opcion para buscar el videojuego!!");
        StdOut.println("[1] Codigo unico");
        StdOut.println("[2] Nombre Videojuego");

        opcion = StdIn.readString();
```

3) menú estadísticas

Se despliega un menú, que es una función, donde se muestran todas las estadísticas que se pueden imprimir por pantalla



```
StdOut.println("""
            \n**********************
               MENU ESTADISTICAS
            **********************
            [1] Videojuego más vendido
            [2] Plataforma con mayor ventas
            [3] Venta a clientes registrados
            [4] Imprimir ventas totales
            [5] Trabajador con más ventas
            [6] Menú anterior""");

String texto = "\nIngrese una opcion: ";
```

Opción Estadísticas:

1) Videojuego mas vendido; Se busca en la lista videojuegos cual es el videojuego que mas se ha vendido, si es que existe. Y se despliega en pantalla su información.

2) Plataforma con mayores ventas; Con los contadores de ventas que existen en la lista videojuegos, se compara cual es la plataforma con mayores ventas y se despliega.

3) Venta a clientes registrados; Con el acumulador y contador de ventas para los clientes que son **miembros,** se despliega esos datos por pantalla

4) Imprimir ventas totales; Con el acumulador del monto total que se registra al vender un videojuego **sin importar las condiciones**. Se imprime el acumulador por pantalla.

5)Trabajador con mas ventas: con el contador y acumulador que se encuentra en cada empleado y se actualiza cada que el empleado realiza una venta, se busca cual es el empleado que mas ventas ha realizado. Una vez encontrado se imprime su acumulador y contador por pantalla.

6) Menú anterior; se termina de ejecutar la función, se devuelve al menú (MenúPrincipal())

4) Salir

El programa termina y se deja un mensaje de despedida por pantalla.

Validaciones

Tomamos en cuenta las posibles situaciones en el que el programa pueda dar error o caerse por completo, por lo que realizamos validaciones en cada menú, utilizando el condicional switch y también validando el ingreso de palabras en caso de números enteros (**int**)

Además de, verificar los datos ingresados y que existan; ejemplo (inicio sesión)

que las posiciones de los datos buscados sean validas; ejemplo (posiciones en las listas)

Tiempo de dedicación taller (horas)		
Integrantes	Aplicación java	Informe del taller
Rodrigo Aguilera	15 horas	2 hora
Diego Cortes	15 horas	2 hora