Entrega #4: Clase abstracta e interfaces

Nombre: Luis Romero, Heinz Beckers, Sebastian Villalón
Tema: Administrador de Películas-Clientes en un videoclub

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.

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.

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. **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.