Guia #0

Repaso

Objetivo: *Desarrollar un sistema para una biblioteca de videojuegos, priorizando la flexibilidad para futuros tipos de media.*

Principios Clave:

- ***Programación Orientada a Interfaces:*** *Manipular objetos a través de interfaces, no de clases concretas.*
- ***Repositorios Genéricos:*** *Usar repositorios para gestionar la persistencia y el acceso a datos.*

Escenario

Has sido contratado por un nuevo cliente que solicita un software para la gestión de sus colecciones de videojuegos. Tu tarea es desarrollar un sistema para administrar juegos y expansiones con funcionalidades robustas y buen manejo de excepciones.

Requisitos Funcionales

Definición de Entidades

- La colección debe incluir juegos y expansiones.
- Cada juego y expansión debe contener los siguientes atributos:
    - **Título**, **Creador**, **Género**, **Identificador único**.
    - **Juego:** número de versión.
    - **Expansión:** fecha de lanzamiento.
- Aplicar herencia y abstracción para definir la estructura base.
- Implementar la interfaz `Comparable` para permitir la comparación entre objetos basados en el identificador único.
- Nuestro cliente nos pide que a futuro quiera volver a contratarnos para manejar no solo sus videojuegos, sino otros tipos de media como películas o series. Pensar la mejor manera de implementar esto a futuro.

Validación de Datos

- Antes de agregar un nuevo juego o expansión, verificar que el identificador único no esté repetido.
- Utilizar una colección adecuada de Java para evitar duplicados.
- En caso de identificador duplicado, lanzar una excepción personalizada `IdentificadorDuplicadoException`.

- Validar que el número de versión de un juego sea positivo y la fecha de lanzamiento de una expansión no sea nula.
- Capturar y manejar adecuadamente las excepciones en la interacción con el usuario.

Manejo de la Colección

El sistema debe permitir las siguientes operaciones:

1. **Agregar** un nuevo juego o expansión a la colección.
2. **Eliminar** un objeto específico por su identificador.
3. **Mostrar** todos los objetos de la colección de forma ordenada por título.
4. **Filtrar** los objetos por género.
5. **Modificar** un único atributo de un objeto sin afectar el resto de sus datos.

Requerimiento:

- Implementar el manejo de colecciones utilizando **genéricos**, para facilitar la escalabilidad del sistema en el futuro.
- Utilizar `Comparator` para ordenar los objetos por título al mostrarlos.
- Pensar como permitir al usuario buscar por ID
- Pensar como poder incluir cualquier tipo de media a futuro.

Interacción con el Usuario

- Crear una clase separada para manejar el menú y la interacción con el usuario.
- La clase de interfaz debe incluir validaciones adecuadas y manejar excepciones sin interrumpir la ejecución del programa.
- La funcionalidad del sistema debe ser iniciada desde el método `main`.

Organización del Proyecto

Organizar las clases correctamente en paquetes, de la siguiente manera.

- Modelo: Contendrá la definición de entidades. Separar entre interfaces e implementaciones.
- Excepciones: Contendrá las excepciones propias del sistema
- UI : Contendrá el menú de usuario
- Repositorios: Contendrá los repositorios genéricos. Separar entre interfaces e implementaciones.

La clase `Main` debe estar fuera de cualquier paquete.

Principios de Programación Orientada a Objetos

- Aplicar conceptos de POO como encapsulamiento, herencia, polimorfismo y abstracción.
- Código modular, limpio y bien documentado.
- Implementar buenas prácticas en el manejo de excepciones y colecciones.