estructura de las clases para estudiar el gestor y exponer

1. Jerarquía de herencia:
    - Mostrar Juego como clase abstracta con JuegodeMesa, JuegoderealidadVirtual y JuegoporConsola como clases derivadas
2. Relaciones:
    - Usuario tiene una SalaJuego (composición)
    - Usuario tiene una colección de Juego (agregación)
    - Usuario tiene un diccionario de HistorialDeEstados (agregación)
    - SalaJuego implementa IChequeoEstado
    - SalaJuego contiene una lista de Juego (agregación)
3. Enums:
    - Incluir EstadoJuego y Plataforma como enumeraciones separadas
4. Interfaz:
    - Mostrar IChequeoEstado como interfaz implementada por SalaJuego

Herencia (Juego y sus subclases)

- **Relación**: Generalización/Especialización
- **Cardinalidad**:
    - Juego (abstracta) ——▷ JuegodeMesa
    - Juego (abstracta) ——▷ JuegoderealidadVirtual
    - Juego (abstracta) ——▷ JuegoporConsola
- **Cada subclase hereda todos los atributos y métodos de Juego**

Usuario - Juego (Colección)
Relación: Agregación (Usuario "tiene" Juegos)
Cardinalidad: 1..* (Un usuario puede tener de 1 a muchos juegos en su colección)
Representación: Usuario ◇———1..*——— Juego

2. Usuario - SalaJuego
Relación: Composición (Usuario "posee" una SalaJuego exclusiva)
Cardinalidad: 1..1 (Cada usuario tiene exactamente una sala de juego)
Representación: Usuario ◆———1——— SalaJuego

3. SalaJuego - Juego (Juegos Activos)
Relación: Agregación (Sala "contiene" Juegos activos)
Cardinalidad: 0..* (Una sala puede tener de 0 a muchos juegos activos)
Representación: SalaJuego ◇———0..*——— Juego

4. Usuario - HistorialDeEstados
Relación: Agregación (Usuario "mantiene" historiales)
Cardinalidad: 0..* (Un usuario puede tener de 0 a muchos historiales, uno por cada juego)
Representación: Usuario ◇———0..*——— HistorialDeEstados

5. HistorialDeEstados - Juego
Relación: Asociación (Historial está asociado a un Juego específico)
Cardinalidad: 1..1 (Cada historial corresponde exactamente a un juego)
Representación: HistorialDeEstados ———1——— Juego

6. Juego - EstadoJuego (enum)
Relación: Dependencia (Juego "usa" EstadoJuego)
Cardinalidad: 1..1 (Cada juego tiene exactamente un estado actual)
Representación: Juego - - - -> EstadoJuego

1. Clase Juego (Abstracta)
Atributos:
nombre (string)
año (int)
creador (string)
genero (string)
estadoactual (EstadoJuego)
Métodos:
mostrarDetalles() (abstracto)
MostrarEstado()

Características:
Clase abstracta que sirve como base para otros tipos de juegos
Implementa la funcionalidad común a todos los juegos
Define el método abstracto mostrarDetalles() que deben implementar las clases derivadas

2. Clase JuegodeMesa (Hereda de Juego)
Atributos adicionales:
cantidadJugadores (int)
duracionMinutos (int)
Métodos:
Implementación de mostrarDetalles()
Características:
Representa juegos de mesa físicos
Extiende la clase base Juego con atributos específicos de juegos de mesa

3. Clase JuegoderealidadVirtual (Hereda de Juego)
Atributos adicionales:
plataformaVR (string)
requiereControladores (bool)
Métodos:
Implementación de mostrarDetalles()
Características:
Representa juegos de realidad virtual
Incluye información específica sobre hardware VR

4. Clase JuegoporConsola (Hereda de Juego)
Atributos adicionales:
plataforma (Plataforma - enum)
Métodos:
Implementación de mostrarDetalles()
Características:
Representa juegos para consolas tradicionales
Utiliza el enum Plataforma para especificar la consola

5. Clase Usuario
Atributos:
nombre (string)
correo (string)
contraseña (string)
coleccion (List<Juego>)
sala (SalaJuego)
historiales (Dictionary<string, HistorialDeEstados>)
Métodos:
VerificarContrasena()
Registrarse()
IniciarSesion()
AgregarJuego()
EliminarJuego() (2 sobrecargas)
BuscarJuego()
MostrarColeccion()
MostrarDatos()
CambiarEstadoJuego()
ObtenerHistorialJuego()
MostrarEstadosJuegos()
Características:
Clase central que gestiona la colección de juegos del usuario
Maneja el historial de estados de cada juego
Controla la sala de juegos del usuario

6. Clase SalaJuego (Implementa IChequeoEstado)
Atributos:
propietario (Usuario)
fechaCreacion (DateTime)
juegosActivos (List<Juego>)
Métodos:
AgregarJuego()
EliminarJuego()
BuscarJuego()
CambiarEstado()
Iniciar() (de IChequeoEstado)
Pausar() (de IChequeoEstado)
Finalizar() (de IChequeoEstado)
Características:
Implementa la interfaz IChequeoEstado
Gestiona los juegos activos en una sesión de juego

7. Clase HistorialDeEstados
Atributos:
_nombreJuego (string, readonly)
_cambios (List<Dictionary<string, object>>)
Métodos:
Iniciar()
Pausar()
Finalizar()
ObtenerEstadoActual()
EstaFinalizado()
RegistrarCambio() (privado)
ObtenerDiasEnEstadoActual()
ObtenerHistorialCompleto()

Interfaz IChequeoEstado
Métodos:

- Iniciar()
- Pausar()
- Finalizar()

Características:

- Define la interfaz para controlar el estado de los juegos
- Implementada por SalaJuego

9. Enum EstadoJuego
Valores:

- NoIniciado
- Jugando
- Pausado
- Finalizado

Enum Plataforma
Valores:

- NES
- ARCADE
- SEGA
- AMANDA_PEGASUS (((revisar error en el nombre)))
- PLAYSTATION