Diseño y Programación O.O.

Taller 4

Descripción del Análisis

Nombre: Jose Luis Almonacid Peñaranda **Código:** 202320415

1. Cine

Descripción: Representa un cine dentro de la cadena CineMax. Es el nivel más alto de la jerarquía de entidades y **contiene varias salas**.

- **Relaciones:**
  - Tiene una relación de **composición** con Sala (1 .. *), ya que un cine no puede existir sin sus salas.
- **Atributos:**
  - nombre: String → El nombre del cine, para identificación.
  - dirección: String → La ubicación física del cine.
- **Métodos:**
  - + obtenerSalas(): List<Sala> → Devuelve la lista de salas del cine.

2. Sala

Descripción: Representa una sala dentro del cine. **Puede contener múltiples asientos y estar asignada a funciones de películas**.

- **Relaciones:**
  - **Composición** con Asiento (1 .. *), ya que cada sala tiene asientos.
  - **Asociación** con Función (1 .. *), ya que una sala tiene múltiples funciones programadas.
- **Atributos:**
  - número: int → Identificador único de la sala.
  - capacidad: int → Número máximo de espectadores.
  - tipoTecnología: String → Describe la tecnología de proyección (IMAX, 3D, 2D).
- **Métodos:**
  - + obtenerAsientosDisponibles(): List<Asiento> → Devuelve los asientos disponibles en la sala.

3. Asiento

Descripción: Representa un asiento dentro de una sala. **Cada asiento pertenece a una sala y puede ser reservado**.

- **Relaciones:**
  - **Asociación** con Reserva, ya que los clientes seleccionan asientos al comprar boletos.
- **Atributos:**
  - número: int → Número de asiento dentro de la fila.
  - fila: String → Indica en qué fila está el asiento (A, B, C, etc.).
  - tipo: String → Puede ser regular o premium, lo que afecta el precio.
- **Métodos:**
  - + marcarComoReservado(): void → Cambia el estado del asiento a reservado.

4. Función

Descripción: Representa una proyección de una película en una fecha y hora específicas dentro de una sala.

- **Relaciones:**
  - **Asociación** con Sala (1 .. 1), ya que cada función se proyecta en una sola sala.
  - **Asociación** con Película (1 .. 1), ya que cada función tiene una sola película.
  - **Asociación** con Reserva (1 .. *), ya que una función puede tener múltiples reservas.
- **Atributos:**
  - fecha: Date → Día de la proyección.
  - hora: Time → Hora de la función.
- **Métodos:**
  - + verificarDisponibilidad(): Boolean → Indica si hay asientos disponibles.

5. Película

Descripción: Es la clase base para cualquier película en el sistema. **Se proyecta en funciones y puede tener distintas subclases**.

- **Relaciones:**
  - **Asociación** con Función (1 .. *), ya que cada película puede tener múltiples funciones.
  - **Generalización (herencia)** con Largometraje, Documental y Animada.
- **Atributos:**
  - título: String → Nombre de la película.
  - género: String → Ejemplo: acción, comedia, drama.
  - duración: int → Duración en minutos.
  - clasificaciónEdad: int → Edad mínima recomendada.
  - fechaEstreno: Date → Fecha en que se lanza la película.
- **Métodos:**
  - + obtenerFuncionesDisponibles(): List< Función> → Devuelve las funciones de la película.

6. Cliente

Descripción: Representa a los usuarios que compran boletos en el sistema.

- **Relaciones:**
  - **Asociación** con Reserva, ya que los clientes realizan reservas.
  - **Asociación** con ProgramaLealtad, ya que los clientes pueden acumular puntos.
- **Atributos:**
  - ID: int → Identificador único.
  - correo: String → Contacto del cliente.
  - contraseña: String → Credenciales de acceso.
  - historialCompras: List<Reserva> → Lista de reservas realizadas.
  - puntosLealtad: int → Puntos acumulados.
- **Métodos:**
  - + realizarReserva(Función función, List<Asiento> asientos): Reserva
  - + consultarHistorialCompras(): List<Reserva>

5. Reserva

Descripción: Representa una compra de boletos realizada por un cliente.

- **Relaciones:**
  - **Asociación** con Cliente, Función y Asiento, ya que cada reserva involucra a un cliente, una función y los asientos seleccionados.
- **Atributos:**
  - monto: double → Precio total de la compra.
  - fechaTransacción: Date → Fecha de la compra.
  - asientosReservados: List<Asiento> → Asientos seleccionados en la reserva.
- **Métodos:**
  - + calcularMonto(): double → Calcula el monto total a pagar por la reserva.
  - + confirmarPago(Pago pago): Boolean → Confirma si el pago de la reserva fue exitoso.

6. Pago

Descripción: Representa el pago de una reserva realizada por el cliente.

- **Relaciones:**
  - **Asociación** con Reserva, ya que cada pago pertenece a una reserva.
- **Atributos:**
  - tipo: String → Puede ser "tarjeta débito" o "tarjeta crédito".
  - monto: double → Valor del pago.
  - fechaTransacción: Date → Fecha en que se realizó el pago.
- **Métodos:**
  - + procesarPago(): Boolean → Ejecuta el pago y devuelve si fue exitoso o no.

7. Empleado

Descripción: Representa a los trabajadores del cine que gestionan el sistema y programan las funciones.

- **Relaciones:**
  - **Asociación** con Película, Función y Sala, ya que los empleados pueden gestionar estas entidades.
- **Atributos:**
  - correo: String → Email del empleado.
  - contraseña: String → Clave para acceder al sistema.
- **Métodos:**
  - + programarFunción(Función función): void → Permite programar funciones en el sistema.
  - + gestionarPelículas(Película película): void → Agrega o edita información de películas.

8. ProgramaLealtad

Descripción: Representa el sistema de lealtad de los clientes, donde pueden acumular puntos y obtener beneficios.

- **Relaciones:**
  - **Asociación** con Cliente, ya que cada cliente puede estar inscrito en el programa de lealtad.
- **Atributos:**
  - niveles: List<String> → Niveles de membresía (Bronce, Plata, Oro).
  - beneficios: Map<String, String> → Beneficios según el nivel de membresía.
- **Métodos:**
  - + calcularPuntos(int montoCompra): int → Calcula puntos obtenidos por una compra.
  - + obtenerBeneficios(): String → Devuelve los beneficios disponibles según el nivel del cliente.