Javier Delgado del Cerro
Javier López Cano

PRACTICA 3

Decisiones de diseño

Prevención de errores

Una de las principales decisiones de diseño que tuvimos que hacer fue, a la hora de evitar posibles errores en todas las clases como *Sala*, *Sesion* o *Pelicula*, crear un método *validar* que nos permita saber si todos sus atributos son correctos. De esta forma, al trabajar en la clase cine se llama a dicho método, y nos aseguramos de que el cine no contenga elementos erróneos.

Relacion entre Sala y Sesion

Aunque estuvimos dudando en como hacerlo, finalmente optamos por incluir en *Sala* una lista con todas las sesiones que tienen lugar en dicha sala, y en *Sesion* la sala en la tiene lugar. Así podemos saber el número exacto de butacas que tiene cada sesión disponible, aunque una butaca de una sala se rompa, por ejemplo.
Para gestionar esto, hemos desarrollado el método *crearSesion* en *Cine*, que dada una *Pelicula* y una *Sala* inicializa la *Sesion* con ambos elementos y añade dicha sesión a la lista de sala, de forma que para el usuario es completamente transparente, y evitamos cualquier tipo de error.

Entradas

Respecto a las entradas, tenemos una clase principal *Entrada*, sin ningún tipo de descuento, y una clase *EntradaDiaEspectador* que hereda de *Entrada*, en la que podemos definir un descuento al crearla.
Hemos decidido que el descuento no sea un valor fijo ya inicializado, sino que se establezca al crear el objeto, pues de esta forma las películas más nuevas pueden tener menos descuento, como ocurre en la realidad.

Nos hemos encargado también de devolver el precio de las entradas con dos decimales para asemejarlo lo máximo posible a la realidad.

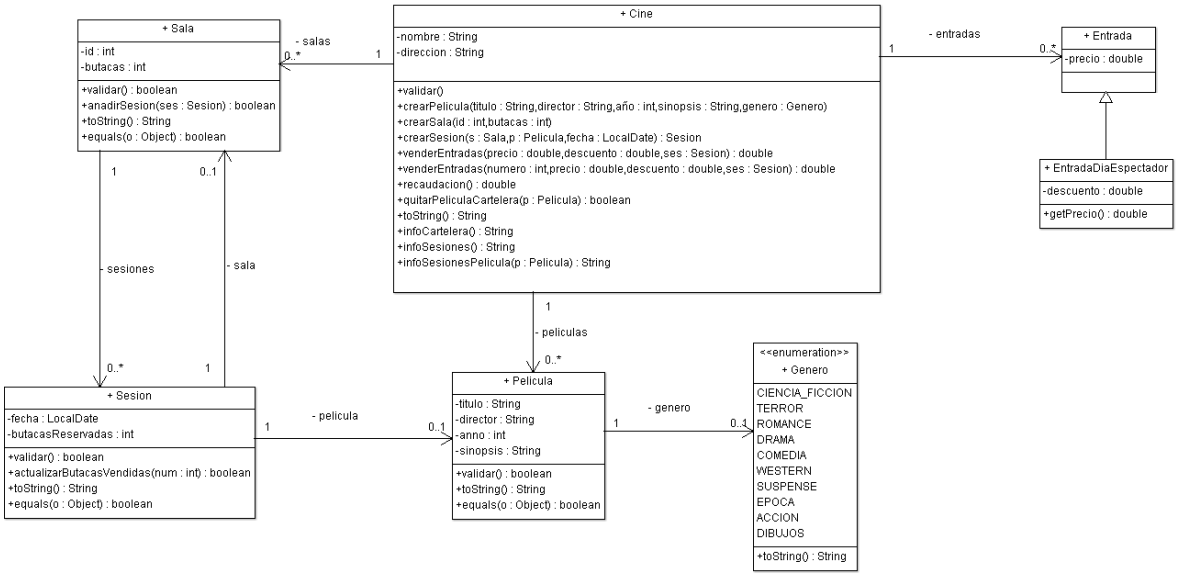
Funcionamiento de la clase Cine

El cine es la clase principal que se encarga de integrar todo, por esto mismo, hemos decidido que en las funciones de *crearSala*, *crearPelicula* y *crearSesion* se pasen todos los atributos necesarios para inicializar los objetos, en vez de los objetos en sí.
Así, comprobamos los objetos, los añadimos a las listas del cine, y devolvemos dicho objeto si todo ha sido correcto, de forma que se puede usar posteriormente, o bien *null* si ha habido algún problema en la validación del objeto, o al añadirlo a las listas (por ejemplo, que ya haya una sesión añadida esa hora).

Hemos decidido además implementar la lista de las entradas del cine como una lista donde están todas las entradas ya vendidas. De esta forma, creamos todos los tipos de entradas con la funcion *venderEntradas* que, gracias a la sobrecarga de métodos, nos permite crear entradas normales y del dia del espectador, añadirlas a la lista de entradas del cine gracias al polimorfismo, y devolver el importe total de dichas entradas, gracias a la ligadura estática de Java.
Esto nos permite además tener un método recaudacion que nos devuelve el dinero total que ha ganado el cine con la venta de entradas.

Javier Delgado del Cerro
Javier López Cano

Diagrama de clases



Pruebas individuales

Para las pruebas individuales de las clases *Pelicula*, *Sala* y *Sesion*, creamos en primer lugar dos películas, una válida y otra no, y llamamamos a los métodos de validar, toString y equals, para comprobar que funcionan correctamente.
Creamos también dos salas, con número de butacas positivo y negativo, y de nuevo, comprobamos los métodos de validar, toString y equals.

Una vez hemos probado las clases *Pelicula* y *Sala*, tenemos que probar la clase *Sesion*, que utiliza las dos anteriores, y el método *anadirSesion* de Sala, que necesita la clase Sesion.
Para esto, creamos dos sesiones, una válida y otra no válida, y además de probar los métodos validar, toString y equals, comprobamos que el método de comprar entradas funciona correctamente, devolviendo *false* cuando no hay suficientes entradas y *true* en caso contrario.
Por último, para probar el método *anadirSesion* basta con ver que no deja añadir dos sesiones en la misma fecha, pero sí en fechas distintas.

Pruebas de integración

Las pruebas de integración consisten básicamente en comprobar el funcionamiento de la clase Cine.
Para esto, creamos dos cines, uno válido y otro no, y probamos los métodos validar y toString.
Para probar los métodos *crearPelicula*, usamos dos películas distintas y vemos que, efectivamente, una película se puede añadir una única vez, y que no permite añadir películas no válidas.
Usamos este mismo prodemiento, cambiando las películas por salas o sesiones, para comprobar el funcionamiento de *crearSala* y *crearSesion.*
Nos aseguramos también de que el cine permita crear distintas sesiones de una misma película en distintas salas pero misma fecha, o bien en la misma sala pero en distinta fecha, y que no nos permita crear dos sesiones de una o distintas películas en una misma sala y fecha, entre otras cosas, es decir, que sea consistente y funcione correctamente.

Por último, nos queda probar el funcionamiento de los métodos *venderEntradas*, para lo que vendemos entradas de distintas películas y salas, con descuento del día del espectador (u otros descuentos en la opcional). Así, vemos que sí que deja vender entradas cuano hay suficientes butacas libres, y no deja cuando no hay suficientes butacas, como era de esperar.

Javier Delgado del Cerro
Javier López Cano

Ejercicio opcional

Para el desarrollo del ejercicio opcional, simplemente eliminamos la lista de películas del cine, y creamos con ella una clase nueva *Cartelera*, con lo que bastó con cambiar la lista de películas por una cartelera y modificar ligeramente las funciones de *crearPelicula* y *quitarPeliculaCine.*
Respecto a las entradas, optamos por tener una clase *Entrada* normal, y otra case *EntradaDescuento* que heredase de dicha clase (así podemos aplicar el polimorfismo), con un atributo *Descuento* que fuese una enumeración, con lo que tenemos todos los tipos de descuentos aplicables.
Podríamos haber optado también por crear una clase nueva que heredase de *Entrada* para cada tipo de descuento, sin embargo nos parecía que era más dificil añadir nuevos tipos de oferta en este caso, y no aprovechavamos las enumeraciones de Java.

El diagra de clases resultante es el siguiente: