“””Jose76Tron/Cine\_Proyecto\_UT6\_ETS””” https://github.com/Jose76Tron/Cine\_Proyecto\_UT6\_ETS/tree/master

El patron que usa es de Singleton debido a que solo genera una instancia inmutable del objeto CineBooking el cual se mantiene constante durante la ejecución del programa que ocurre en la clase cinemabookingApplication

* El programa esta diseñado para simular en su forma mas primitiva el diseño de un sistema de booking de un cine cualquiera
* El objeto singleton se encarga tanto en cargar la cantidad de asientos, consultar los asientos disponibles, reservar los asientos y cancelar los asientos reservados
* Una de las ventajas de usar el diseño singleton para esta clase de proyectos es por la simplicidad y facilidad de extrater información sin tenter que recurrir a generar nuevas instancias de otros objetos, es decir: se crea una especie de directorio en el cual se almacena información concreta y que debe permanecer hasta que se termine la aplicación
* El patron singleton no tiene ninguna desventaja notoria, lo peor que podría ser es que si este objeto es un controlador estará sobrecargado de métodos que aunque puedan ser útiles al momento de hacer mantenimiento hay que cuidar que los métodos no se dañen al cambiar otros de la misma clase, es decir si tengo un método que depende de otros 3 pero al cambiar el método original se rompe el funcionamiento de uno de estos métodos, puede generar un problema, por esta razón el patron singleton es el mejor siempre que se mantega sencillo y objetivo (centrado a cosas especificas / poca mutabilidad)
* Una de las formas que se pudieron haber solucionando sin usar este patron de instanciación única es haber creado atributos y métodos estáticos en la clase ejecutora (almacenadora del método main(String[] args), al estilo Python)

A continuación es el diagrama de clases lo cual casi no tienen sentido realizarlo debido al tipo de diseño

A screenshot of a computer

Description automatically generated with medium confidence