

Disegnare il diagramma delle classi di progettazione di Cinema MultiSala. Giustificare le scelte progettuali fatte e i pattern utilizzati.

Il diagramma delle classi di progettazione di Cinema MultiSala è stato svolto nell'ottica di evitare la rigidità del codice, e quindi nell'intenzione di rendere il codice riusabile, facilmente riutilizzabile e senza un eccessivo utilizzo di memoria.

A tal fine sono stati individuati alcuni pattern specifici:

Singleton: classe Cinema.

Supponendo che l'applicazione rimanga limitata alla gestione di un solo cinema multisala (e non ad esempio una catena di cinema multisala) la classe Cinema incarna il pattern Singleton, in quanto può esistere solamente un'istanza di cinema in tutto il sistema.

Cinema inoltre, per garantire l'unicità dell'istanza creata tiene traccia della sua sola istanza e intercetta tutte le richieste di creazione affinché vengano redirette all'istanza creata tramite il metodo GetInstance().

Flyweight (1):

Flyweight: IOorario (interfaccia implementata dal ConcreteFlyweight)

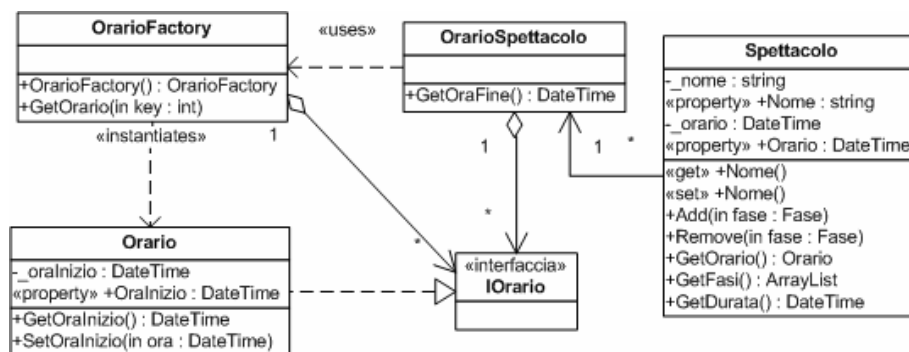
Concrete Flyweight: Orario

Flyweight Factory: OrarioFactory

Cliente: OrarioSpettacolo

Stato Estrinseco: Spettacolo

Si suppone che i film inizino più o meno tutti agli stessi orari. Quindi, al fine di ottenere un migliore utilizzo della memoria, si decide di mantenere l'oggetto "Orario" come condiviso, in modo tale da poter essere utilizzato simultaneamente da differenti clienti evitando conseguentemente che vengano creati svariati oggetti della stessa classe Orario specificanti lo stesso orario di inizio di uno spettacolo.



Flyweight (2):

Flyweight: IFase (interfaccia implementata dal ConcreteFlyweight)

Concrete Flyweight: Fase

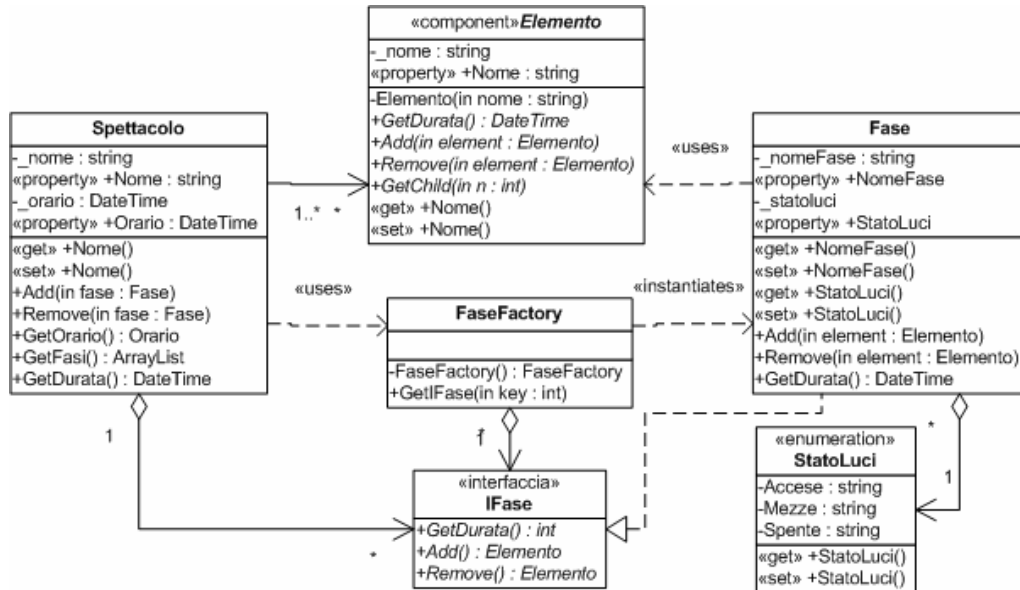
Flyweight Factory: FaseFactory

Cliente: Spettacolo

Stato Estrinseco: Elemento

Stato Intrinseco: StatoLuci

Si suppone che a regime esistano solo tre tipi di fase, caratterizzate dal proprio stato delle luci (accese, mezze, spente). L'elemento visualizzato associato allo stato delle luci invece dipende dal contesto di utilizzo e deve essere memorizzato dal cliente, che lo passa al flyweight (per mezzo della factory) al momento dell'invocazione di un'operazione.



Composite (1):

Component: Elemento

Composite: GruppoDiElementi

Leaf: ElementoSingolo

Attraverso questo pattern viene reso possibile comporre diversi oggetti di tipo Elemento in una struttura più complessa che ne contiene un numero indefinito.

Al fine di ottenere un buon compromesso tra trasparenza e sicurezza, tutte i metodi che il cliente può invocare sono dichiarati nella classe astratta Elemento e le operazioni insensate sulle foglie, quali Add o Remove, genereranno un'opportuna eccezione nel caso in cui il cliente provi ad invocarli sulle foglie.

Composite (2):

Component: Film

Composite: FilmComposite

Leaf: ParteDiFilm

Questo pattern permette di aggregare diverse parti di film per ottenere un film completo.

Al pari dell'utilizzo precedente, in questo pattern si è privilegiata la trasparenza piuttosto che la sicurezza, impostando determinati comportamenti di default che scagliano delle eccezioni nei casi in cui si cerchi di effettuare operazioni senza senso.

Nell'implementazione delle classi e nell'inserimento dei patterns si è fatta particolare attenzione al rispetto dei principi di design, in particolare al rispetto del *Single Responsibility Principle*, il quale raccomanda che ogni classe abbia solamente una responsabilità e che la soddisfi adeguatamente.

- Cinema: tiene traccia delle sale create
- Sala: tiene traccia delle programmazioni della sala, controllando che non ci siano sovrapposizioni di programmazioni, in particolar riferimento alle date di inizio e fine di ogni programmazione.
- ProgrammazioneGiornaliera: tiene traccia degli spettacoli in programma, controllando che non vi siano sovrapposizioni tra gli spettacoli o parti di essi.
- Spettacolo: tiene traccia delle fasi che compongono lo spettacolo, memorizzando il proprio orario di inizio e di fine.
- OrarioSpettacolo: ha il compito di calcolare la durata dello spettacolo.
- OrarioFactory, FaseFactory: crea e condivide i flyweight.
- IOrario, IFase: dichiara l'interfaccia attraverso la quale i flyweight possono ottenere lo stato estrinseco dal cliente e fare operazioni.
- Orario: implementa l'interfaccia IOrario e determina l'orario di inizio di uno spettacolo.
- Fase: implementa IFase ed associa un elemento ad un determinato stato delle luci in sala.
- StatoLuci: riporta lo stato delle luci che possono essere settate in sala.
- Elemento, Film: dichiara i comportamenti di default delle classi che la implementano.
- GruppoDiElementi, FilmComposite: descrive gli oggetti che hanno figli e implementa le operazioni della classe Component.
- ElementoSingolo, ParteDiFilm: descrive gli oggetti senza figli ed implementa le operazioni di default della classe Component.
- Diapositiva, Pausa, Pellicola: specializza la classe ElementoSingolo
- SpotPubblicitario, Trailer, Film: specializza la classe Pellicola.

Note:

Nel disegno del diagramma delle classi di progettazione sono stati sottointesi i costruttori delle classi banali e le operazioni standard.

