Ho creato un ControllerGUI la cui unica funzione è collegare tutte le Jframe con la logica ed impostare la navigazione di jframe e jdialogue. Ci saranno poi uno o più altri controllers che non avranno niente a che fare con la GUI e riguarderanno solo la logica.

Ho scelto di creare una classe SuperJFrame che fa da padre a tutte le Jframe del progetto. Serve per raccogliere a fattore comune tutti gli attributi ed i metodi comuni a tutte le jframe. In particolare:

🡪 la SuperJFrame crea automaticamente l'associazione di ogni finestra con la GUI (ogni finestra ha un attributo ControllerGUI)

🡪 usa come icona un'immagine rappresentativa dell'app, che viene ereditata da tutte le jframe dell'app; se la si vuole cambiare si può cambiarla una volta per tutte cambiandola solo nella jframe-padre, anzichè cambiarla ad una ad una per tutte le finestre

🡪 implementa metodi per l'inserimento di immagini scalate su labels e bottoni, utilizzabili quindi in tutto il progetto. Su questi metodi avevo pensato di usare i generics per una implementazione più elegante, ma non esiste un modo semplice per farlo perché non ho trovato alcuna classe-padre comune a Jbutton e Jlabel che implementasse il metodo .setIcon; comunque non è una tragedia e penso di potermi limitare ad un semplice overloading.

Anche per le jDialogue sto pensando di fare una scelta analoga, creando una Jdialogue-padre, ma non so se in quel caso ne vale la pena.

Indirizzo github:

[https://github.com/EmaRomano/progettoCinema.g](https://github.com/EmaRomano/progettoCinema.git)

[it](https://github.com/EmaRomano/progettoCinema.git)

--------------------------------------------------------------------------------------------------

per la gui, sui bottoni si sono evitate scritte descrittive ingombranti ed antiestetiche, lasciando il compito di far capire all’utente la funzione di ciascun bottone con icone significative di sfondo al bottone stesso; a scanso di equivoci, per maggiore chiarezza si sono usate anche le tooltips.

Ho ritenuto opportuno introdurre nel costruttore di alcune finestre un riferimento a sè stesse:

SuperJFrame questaFinestra = this;

questo perché è un riferimento necessario da usare negli action listener: in essi il “this” non si riferisce alla finestra corrente. Questo fatto l’ho usato nel metodo a seguire.

Nel programma, come in quasi tutti i programmi del mondo, ci sono molte finestre con un bottone “indietro”; per evitare di scrivere un metodo “bottoneIndietroPremuto” per ogni finestra che contiene un simile bottone, ne ho fatto solo uno (bottoneIndietroPremutoDallaFinestra) che prende come parametro la finestra dalla quale viene premuto il bottone, fa setVisible(false) su di essa e setVisible(true) sulla finestra precedente (per capire qual è ho usato un semplice costrutto di selezione usando “instanceof”).

--------------------------------------------------------------------------------------------------

Promemoria per i programmatori che metteranno mano al codice per manutenzione/sviluppi:

🡪il datepicker usato conteneva alcuni piccoli bugs. Ad esempio: la finestrella finestraCalendario restava aperta (anche durante la navigazione in finestre successive) ogni volta che l’utente cliccava sul pulsante “scegli data” ma poi non cliccava su alcuna data, ed il problema si poteva ripetere più volte lasciando aperte più di tali finestre. Si è risolto il problema in una maniera non elegante ma efficace: ogni volta che si esce da una finestra contenente il date picker si fa il dispose() di finestraCalendario. Si raccomanda di seguire lo stesso tipo di soluzione (oppure di implementarne una migliore).

--------------------------------------------------------------------------------------------------

Per i dati sull’affluenza in un singolo spettacolo si è scelto di riportarli usando un tasso percentuale t.a. (tasso di affluenza), definito come:

t.a. = (numero di paganti/numero di posti disponibili)\*100 ;

Si è ritenuto infatti che tale stima fosse più significativa rispetto ad una riportata in termini di numero assoluto di spettatori; si pensi infatti ad uno spettacolo con 200 spettatori: in una sala di 250 posti è un ottimo risultato, in una di 500 posti (generalmente più costosa per investimenti sulle tecnologie, manutenzione e costi delle pellicole) è un flop.

Per le statistiche sull’affluenza per sale si è poi usato un tasso di affluenza medio (t.a.m.), calcolato come media aritmetica delle affluenze degli spettacoli tenuti in ciascuna sala. Tali dati sono stati poi riportati in modo visivamente efficace in diagrammi a barre.

Diverso discorso per quanto rigarda le statistiche di affluenza per fasce orarie: in ciascuna fascia si tengono infatti spettacoli in sale con un numero di posti disponibili molto diverso, ed usare quindi un t.a.m. porterebbe a risultati fuorvianti: si pensi, per semplicità, ad un cinema con due sole sale, di 100 e 500 posti rispettivamente, di cui la prima completamente piena e la seconda completamente vuota; ne risulterebbe un t.a.m. del 50%, dato del tutto inaccettabile visto che in realtà il cinema è pieno per solo 1/6. Per questo motivo si è scelto di riportare i dati sull’affluenza per fasce orarie in termini di numero assoluto di spettatori; nei relativi diagrammi a barre la fascia di maggiore affluenza è sempre piena al 100%, le altre sono riportate in proporzione a questa.

I dati relativi agli spettacoli più remunerativi sono riportati semplicemente in termini di incasso, con una scelta sul diagramma a barre analoga a quella appena descritta circa l’affluenza per fasce orarie; in aggiunta, si è riportato un dato del tipo incasso/posti disp. , che descrive quanto è fruttato uno spettacolo per ciascun posto.