Progetto Metodologie di Programmazione

“MovieTime -VideoRental”

Realizzato da:

Rossi Tommaso
matricola: 5972246

e-mail: tommaso.rossi8@stud.unifi.it

Da sostenere anche la prova scritta

Descrizione del progetto

Ho scelto di sviluppare la traccia del Rental ossia il videonoleggio di film. L'applicazione si presenta come una sorta di store multimediale dove il cliente, dopo aver effettuato il login con username e password personale, può effettuare il noleggio di uno o più film. Inoltre ogni cliente possiede una tessera con un certo credito all'interno che può usufruire per il pagamento dei noleggi. Queste tessere possono essere di tre tipi:

1- Basic: tessera che permette di avere il 20% di sconto su un ordine effettuato.
2- Children: tessera che permette di noleggiare solo film di genere Animazione, ma su quest'ultimi ha uno sconto sulle novità del 20%, adatta per i bambini.
3- Gold: tessera che permette di avere una collezione di punti Gold: essi sono dei punti cumulabili che si ottengono ogniqualvolta si effettua un certo ordine. Si ottengono 5 punti per ogni film normale e 8 punti per ogni novità. Quando l'utente raggiunge i 50 punti ha diritto ad utilizzare uno sconto del 50% su tutto l'ordine.

Le tessere possono essere ricaricate.

Un cliente può essere anche Amministratore del sistema. Esso, oltre ad avere la possibilità di noleggiare, può anche accedere al database dei film e degli utenti, aggiungendo, modificando o eliminando un certo oggetto dai database.

Scelte di design principali del progetto

Durante la realizzazione del progetto le scelte di design del codice sono state soprattutto la specializzazione delle varie classi: ogni classe ha una propria funzionalità e questo rende il codice estendibile, riusabile e facilmente modificabile. Per questo motivo ho creato dapprima delle classi astratte che vengono estese da altre classi per permettere il riuso del codice già scritto. Inoltre ho creato varie interfacce, che sono state implementate dalle diverse classi che compongono il progetto.

- Ho deciso di utilizzare il design pattern Strategy per quanto riguarda il calcolo dello sconto per le tre diverse tessere. Vengono utilizzate tre diverse strategie per l'applicazione dello sconto e ogni metodo di pagamento viene sviluppato in modo diverso a seconda del tipo di tessera che utilizza l'utente. Per attuare il pattern si utilizza l'interfaccia "RentalCardPriceStrategy" che ha al suo interno il metodo "calculateTotal" che esegue il calcolo dello sconto dell'ordine effettuato e il metodo "executePayment" che, oltre ad effettuare il pagamento settando il nuovo credito, ha anche altre funzioni come per esempio, per le tessere Children, effettua un controllo per non permettere a questo tipo di utente di noleggiare film di genere diverso da quello per bambini o, per le tessere Gold, di calcolare nuovi punti.

- Inoltre ho deciso di utilizzare il design pattern Visitor per la creazione del menu principale a seconda che l'utente effettua l'accesso come cliente o amministratore. Per effettuare questo tipo di operazione viene utilizzata la classe "CreateMenuPanel" che implementa l'interfaccia "Visitor": la classe ha al suo interno un oggetto (di stato) della classe astratta "MenuPanel" che viene creata a run-time a seconda della firma del metodo, attraverso i 2 metodi implementati "CreatePanelAccess(Admin)" e "CreatePanelAccess(Client)". Inoltre le classi "Client" e "Admin" implementatno il metodo "accept(Visitor v)" dell'interfaccia "User": ogni utente che effettua l'accesso richiama questo metodo e accetta un certo oggetto "Visitor" che ha la possibilità di creare due tipi di oggetti concreti:
  1- "MenuClient": in questo contesto verranno visualizzati solo i panelli per il noleggio, ricarica credito, dati dell'utente e il proprio ordine effettuato.
  2- "MenuAdmin": estende la classe "MenuClient" e ha al suo interno i pannelli per operare con i database del sistema.

Esecuzione tipo del programma

Il programma viene eseguito dalla classe "Main", che attraverso il metodo "runApplication()" crea un oggetto della classe "MovieTimeUI". In quest'ultima viene creato attraverso il metodo "initialize()" la finestra principale che ospiterà tutti i pannelli dell'applicazione. Dopodiché il primo componente che viene aggiunto alla finestra è l'oggetto "HomePagePanel". Esso estende la classe "JPanelWithBackground" che ha al suo interno tutti i metodi per settare l'immagine di sfondo del pannello e la dimensione di esso. Dopo aver richiamato tali metodi della super classe, viene inizializzata la pagina principale che ha al suo interno un oggetto di tipo "PanelLogin". Questo oggetto è un pannello che permette, attraverso l'inserimento del proprio "username" e "password", di accedere al menu principale. Oltre al pannello di login la classe ha al suo interno un altro oggetto di tipo "LoginSystem" il quale ha il compito di effettuare il controllo dei dati di accesso e di negarlo se non esiste nel database degli utenti. Per la creazione dei componenti si utilizza la classe "ComponentCreator" che ha il compito di creare i componenti della classe: oltre a varie "JLabel" che migliorano l'interfaccia crea anche una "JTextField" per l'username e una "JPasswordField" per la password. Questi dati vengono poi controllati dal LoginSystem: all'interno della classe viene utilizzato l'oggetto della classe "DBSource" che ha l'unico compito di connettersi al database degli utenti e viene utilizzato ogni volta che c'è bisogno di interrogare il database.

Dopo queste operazioni iniziali viene visualizzata la homepage del sistema.

Di seguito mostro la lista dei possibili accessi al sistema attraverso la seguente tabella.

USERNAME	PASSWORD	TIPO UTENTE	TIPO TESSERA
username0	password0	Client	Basic
username1	password1	Admin	Gold
username2	password2	Client	Children

Dopo aver inserito i dati e premuto nel bottone “Login” viene effettuato il controllo e, se i dati sono stati inseriti correttamente, viene creato un oggetto di tipo “SetAccessUser” che ha al suo interno delle variabili di stato che vengono settate con i dati dell’utente che ha effettuato l’accesso. Questi settaggi vengono fatti prendendo le informazioni dal database degli utenti e delle tessere, rispettivamente oggetti del tipo “DatabaseUser” e “DatabaseCard”. Esse hanno al suo interno delle liste che vengono utilizzate come strutture dati per la conservazione delle informazioni del database.

Dopodiché viene creato l’oggetto di tipo “CreateMenuPanel” che creerà, come detto in precedenza, il menu che richiamerà il metodo astratto della classe “MenuPanel” cioè “SetUserDataComponent”. Esso serve per settare tutti i pannelli della finestra principale: ogni pannello richiama il metodo “SetComponent” prendendo in input i vari dati dell’utente e dei database.

Il menu si presenta con 6 bottoni che visualizzerà ognuno un certo pannello e un altro bottone “LogOut” per uscire dal sistema. Quest’ultimo viene creato perché la classe astratta “MenuPanel” estende la classe astratta “JPanelWithBackgroundAndLogOutBtn” nel quale viene creato il bottone di uscita. Ogni pannello del menu estende la classe “JPanelWithBackgroundAndBackButton” il quale ha al suo interno la creazione di un bottone “back” che permette di tornare al menu principale.

Di seguito vengono descritti i bottoni del menu. Per semplicità si prende come esempio l’accesso come amministratore per permettere di descrivere tutti i pannelli.

- *Rental*: cliccando questo bottone si aprirà il pannello “RentalPanel” dove possiamo aggiungere i film presenti nella lista nel nostro ordine. Il pannello si presenta con una “JScrollPane” con all’interno una “JList” che mostra tutti i titoli disponibili nel catalogo. Quando viene selezionato un certo film, nella parte destra di un altro pannello vengono visualizzate tutte le informazioni del film selezionato (titolo, categoria, regista, prezzo, ecc.). Inoltre nel pannello è presente una “JTextField” dove è possibile cercare il film con ben tre metodi di ricerca (per titolo, per regista, per categoria) attraverso il bottone “Find” oppure resettare la ricerca attraverso il bottone “Reset”. Per aggiungere un film al proprio ordine basta selezionarlo e cliccare sul bottone “Add To Order” sempre che esso non sia già nel nostro ordine, altrimenti lancia un messaggio che ci informa che il film è già nell’ordine.

- *Recharge Credit*: cliccando questo bottone si aprirà il pannello "RechargePanel" per ricaricare il proprio credito. Dopo aver selezionato il numero della carta e l'importo basta cliccare sul bottone "Recharge" per eseguire la ricarica. Un messaggio ci informerà del nuovo credito.
- *My Order*: cliccando questo bottone si aprirà il pannello dell'ordine dell'utente. Esso si presenta con una "JScrollPane" con all'interno una lista dei film che sono stati aggiunti all'ordine attraverso il pannello "RentalPanel". Attraverso questo pannello è possibile modificare il proprio ordine come ben vogliamo. All'interno ci sono tre "JPanel" che sono:
  1 - Un pannello per visualizzare i dati dell'utente (username, numero tessera, tipo di tessera, credito).
  2 - Un pannello per gestire i giorni del proprio noleggio attraverso un "JSpinner". Il prezzo totale dell'ordine aumenterà automaticamente se aumentiamo i giorni del noleggio.
  3 - Un pannello per visualizzare i dati dell'ordine ossia prezzo dei film nell'ordine, sconto applicato e totale.
  Ci sono inoltre 3 bottoni che sono:
  1 - "Pay Order": effettua il pagamento dell'ordine richiamando il metodo "ExecutePayment" della tessera utilizzata. Se non ci sono film nell'ordine un messaggio avviserà di inserirli, altrimenti viene effettuato il pagamento e scalato il proprio credito.
  2 - "Remove All": rimuove tutti i film dall'ordine.
  3 - "Remove Movie": dopo aver selezionato un film nell'ordine e cliccando su esso si può toglierlo dalla lista dell'ordine.
- *User data:* cliccando questo bottone si aprirà il pannello "DataUserPanel" dove si ha la possibilità di visualizzare i dati dell'utente e della propria carta. Inoltre è possibile attraverso il bottone "Modify" e "Save" di modificare i propri dati e salvarli.
- *Edit Database Movie*: cliccando su questo bottone si aprirà il pannello "EditDatabaseMoviePanel" che estende la classe "EditDatabasePanel". All'interno troviamo una "JScrollPane", che viene richiamata dalla super classe, la quale contiene un oggetto "TableMovie". Quest'ultima è una "JTable" che mostra tutti i dettagli e le informazioni dei film nel database. Le righe della tabella possono essere selezionate e attraverso delle "JTextField" all'interno di un altro pannello possono essere visualizzate. Attraverso poi i bottoni "Add Object", "Remove Object" e "Update Object" possono essere rispettivamente aggiunti, rimossi e aggiornati gli elementi della tabella.
- *"Edit Database User":* si comporta come il precedente bottone visualizzando il pannello "EditDatabaseUserPanel" che, anch'esso come il precedente, estende la classe "EditDatabasePanel". L'unica differenza con il precedente è

che viene creata un oggetto “TableUser” che visualizza gli oggetti del database degli utenti.

Descrizione delle classi che compongono il progetto divise per pacchetti

pacchetto movietime.core.accesspanel

1- **CreateMenuPanel**
Classe che implementa l’interfaccia “Visitor”. Implementa i metodi “CreatePanelAccess(Admin admin)” e “CreatePanelAccess(Client client)” e viene creato rispettivamente un “MenuClient” e un “MenuAdmin”. Inoltre ha un metodo “getPanel()” per richiamare l’oggetto creato con tali metodi.

2- **JPanelWithBackgroundAndLogoutBtn**
Classe astratta che estende la classe “JPanelWithBackground”. Possiede un metodo privato “drowLogoutButton()” che ha la funzione di creare un bottone per permettere l’uscita dal sistema.

3- MenuPanel
Classe astratta che estende la classe “JPanelWithBackgroundAndLogoutBtn". Ha un metodo astratto setUserDataComponent (User u, DatabaseMovie dbMovie,Database dbUser). Quest’ultimo ha la funzione di settare con i dati del database e dell’utente tutti i pannelli di cui è composto il menu; infatti nel corpo del metodo ci sono tutti i pannelli che richiamano a sua volta ognuno un altro metodo “SetComponent” che settano tali informazioni passate in input.

4- **MenuClient**
Classe concreta che estende la classe astratta “MenuPanel”. Nel costruttore della classe vengono creati gli oggetti che compongono i pannelli del menu (“RentalPanel”,”DataUserPanel”,”RechargePanel” e “OrderPanel”) e aggiunti alla finestra principale. Nella classe ci sono metodi privati che, quando vengono richiamati hanno il compito di creare ognuno i bottoni principali del menu. Essi vengono creati delegando il compito della creazione ad un altro oggetto che è all’interno della classe ossia “ComponentCreator”. Ogni bottone ha un “ActionListener”: infatti quando vengono cliccati verranno settati i pannelli elencati precedentemente e altri metodi ereditati dalla classe “JPanelWithBackground” fra cui:
SetVisiblePanel() => setta il pannello che vogliamo rendere visibile.
SetPanelVisibility() => setta la visibilità degli altri pannelli a false, non facendoli visualizzare dall’utente.
SetBack => setta il pannello che il tasto “Back” farà visualizzare in seguito.

Inoltre eredita il metodo protected PaintComponent() della classe JComponent di Java.

5- **MenuAdmin**
Classe concreta che estende la classe “MenuClient”. Oltre ad ereditare metodi e oggetti della super classe ha in più la creazione di altri due bottoni “Edit Database Movie” e “Edit Database User” che fanno visualizzare i pannelli “EditDatabaseMoviePanel” e “EditDatabaseUserPanel”.

6- **Visitor**
Interfaccia che viene implementata dalla classe “CreateMenuPanel” e crea a seconda della firma del metodo i menu descritti precedentemente.

pacchetto movietime.core.card

1- **RentalCard**
Classe astratta che rappresenta la tessera del noleggio dei film. Essa implementa l’interfaccia “RentalCardPriceStrategy” che ha al suo interno i metodi “ExecutePayment” e “CalculateTotal” che esegue il pagamento e calcola gli sconti rispettivamente. Inoltre nella classe ci sono tutti i metodi per richiamare tutte le variabili della classe (getNumber(), getCredit(),ecc) e metodi per settarle (setDiscount(),setTotalOrder(), ecc.). All’interno della classe ha anche un metodo astratto “SetGoldPoint” che viene implementato soltanto dalle tessere di tipo Gold.

2- **Basic**
Classe concreta che estende la classe astratta RentalCard e tutti i suoi metodi. Questo tipo di tessera ha lo sconto del 20% su tutto l’ordine effettuato.

3- **Children**
Classe concreta che estende la classe RentalCard. Questo tipo di tessera può noleggiare solo film di genere Animazione, ma ha uno sconto del 20% sulle novità. Per effettuar il controllo sui film di Animazione la classe ha il metodo “ParentalControl” che restituisce un booleano a seconda che i film siano accettati o no. Se il controllo non passa, il pagamento non viene effettuato e un messaggio ci segnalerà il divieto.

4- **Gold**
Classe concreta che estende la classe RentalCard. Questo tipo di tessera ha un oggetto di tipo “CollectionGoldPoint” che rappresenta la raccolta punti che l’utente può usufruire per avere lo sconto del 50%. All’interno della classe ci sono inoltre i metodi “ControlDiscount” che effettua il controllo se si ha diritto allo sconto o no, “CalculateNewGoldPoint” che calcola i punti acquisiti su un certo ordine. Inoltre abbiamo i metodi “getCollectionGoldPoint” e “SetGoldPoint” usati per richiamare la collezione di punti o settarla.

5- **CollectionGoldPoint**

Classe concreta che implementa l'interfaccia "GoldPointInterface" che ha al suo interno i metodi implementati "getGoldPoint","SetGoldPoint" che richiamano e settano i punti di una certa collezione.

6- **GoldPointInterface**
Interfaccia che implementa i metodi "getGoldPoint" e "SetGoldPoint".

pacchetto movietime.core.creator

1- **ComponentCreator**
Classe concreta che implementa l'interfaccia "ComponentCreatorInterface" che ha al suo interno i metodi per la creazione di tutti i componenti che compongono il progetto (CreateLabel(),CreateButton(),ecc.). Nel costruttore vengono inizializzati tutti i componenti che vengono assegnati ad una variabile di stato ogniqualvolta viene creato tale oggetto. Infatti prima di tutto quando si crea un componente l'oggetto "ComponentCreatorInterface" invoca il metodo "CreateNewComponent" che creerà al suo interno una variabile di stato a seconda del tipo di componente. Dopo averlo creato richiameremo il metodo "SetUpComponent" per settare il font del componente e con il metodo "SetUpComponentProp" invece setteremo la dimensione del componente creato. Inoltre dentro questa classe ci saranno vari metodi getter() che richiameranno i componenti creati.

2- **ComponentCreatorInterface**
Interfaccia che ha tutti i metodi di creazione dei componenti implementati dalla classe "ComponentCreator" descritti al numero 1 del pacchetto.

3- **CreateCard**
Classe concreta che ha l'unico compito di creare un istanza degli oggetti di tipo "Basic", "Children" e "Gold". Crea i seguenti oggetti attraverso il metodo "create" e, a seconda del tipo di firma del metodo, vengono creati tali oggetti.

4- **CreateMovie**
Classe concreta che ha l'unico compito di creare un'istanza degli oggetti di tipo "NormalMovie", "NewRelease". Crea i seguenti oggetti attraverso il metodo "create" e a seconda del tipo di firma del metodo vengono creati tali oggetti.

5- **CreateUser**
Classe concreta che ha l'unico compito di creare un'istanza degli oggetti di tipo "Client", "Admin". Crea i seguenti oggetti attraverso il metodo "create" e, a seconda del tipo di firma del metodo vengono creati tali oggetti.

pacchetto movietime.core.database

1- **DatabaseMovieInterface**
   Interfaccia che ha al suo interno i metodi per la gestione del database dei film. Tali metodi sono "createListMovieFromDatabase()" che interroga il database e restituisce una lista di oggetti di tipo "Movie", "AddMovie" che aggiunge un oggetto di tipo "Movie" alla lista e altri metodi di aggiornamento e rimozione degli elementi della lista.
2- **DatabaseMovie**
   Classe che implementa l'interfaccia "DatabaseMovieInterface" e tutti i metodi. Dentro questa classe viene creata la lista dei film che viene gestita attraverso i metodi implementati.
3- **DatabaseUserInterface**
   Interfaccia simile a "DatabaseMovieInterface" ma a differenza di essa implementa metodi che gestiscono una lista di tipo "AbsUser".
4- **DatabaseUser**
   Classe che implementa l'interfaccia "DatabaseUserInterface" e tutti i metodi. Dentro questa classe viene creata la lista degli utenti che viene gestita attraverso i metodi implementati.
5- **DatabaseCardInterface**
   Interfaccia simile a "DatabaseMovieInterface" ma a differenza di essa implementa metodi che gestiscono una lista di tipo "RentalCard".
6- **DatabaseCard**
   Classe che implementa l'interfaccia "DatabaseCardInterface" e tutti i metodi. Dentro questa classe viene creata la lista delle tessere che viene gestita attraverso i metodi implementati.
7- **DataSource**
   Classe concreta che ha il compito di connettersi al database e interrogarlo attraverso i metodi "ConnectDB" che crea la connessione, poi con il metodo "CreateStatement" esegue la creazione di un oggetto di tipo "Statement" che serve per estrapolare le informazioni e assegnarla ad una variabile. Infine con il metodo "ExecuteQuery(String sql)" esegue una certa Query nel seguente database. Tutti questi metodi fanno parte del pachetto java.sql.*.

pacchetto movietime.core.finder

1- **ResearchMovie**
   Classe astratta che ha al suo interno una certa lista di film temporanea che viene creata quando si effettua la ricerca di un certo record nella lista dei film che si possono noleggiare. All'interno della classe ci sono due variabili di stato

ossia la lista temporanea dei film, il record da cercare e il metodo astratto "research()" che viene specializzato dalle sottoclassi.

2- **ResearchForTitle**
   Classe concreta che estende la classe "ResearchMovie". Eredita il metodo "research" e implementa il corpo del metodo effettuando la ricerca nella lista per titolo del film.
3- **ResearchForCategory**
   Classe concreta che estende la classe "ResearchMovie". Eredita il metodo "research" e implementa il corpo del metodo effettuando la ricerca nella lista per categoria del film.
4- **ResearchForDirector**
   Classe concreta che estende la classe "ResearchMovie". Eredita il metodo "research" e implementa il corpo del metodo effettuando la ricerca nella lista per regista del film.

pacchetto movietime.core.loginsystem

1- **LoginPanel**
   Classe concreta che estende la classe JPanel di java e ha il compito di creare il pannello del login iniziale.
2- **LoginSystem**
   Classe concreta che ha la funzione di controllare il corretto inserimento dell'username e della password da parte dell'utente e lo fa interrogando il database con un istanza dell'oggetto "DBSource" e attraverso il metodo "controlAccess()" setta la variabile booleana di stato "access" a true, se l'inserimento è corretto, richiamando il metodo "SetAccess()".
3- **SetAccessUser**
   Classe concreta che ha il compito di settare tutti i dati dell'utente che ha effettuato l'accesso. Possiede i metodi checkUser() che, attraverso l'username e la password in input, può settare una variabile di tipo User dentro questa classe, oltre alla tessera che utilizza l'utente attraverso il metodo "CheckRentalCard". Infine ha un metodo "getUserAccess" che restituisce l'istanza dell'oggetto User. Servirà in seguito per la creazione del MenuPanel.

pacchetto movietime.core.main

1- **Main**
   Classe concreta che lancia in esecuzione l'applicazione attraverso il metodo statico runApplication() richiamato dal metodo main().Esso crea un istanza di un oggetto di tipo "MovieTimeUI".
2- **MovieTimeUI**

Classe concreta che estende JPanel e rappresenta la User Interface dell'applicazione.

pacchetto movietime.core.movie

1- **MovieInterface**
Interfaccia che ha al suo interno tutti i metodi getter che richiamano le variabili della classe astratta "AbsMovie" che sono tutti i dati di un certo film quali l'id, il titolo,il regista ecc.
2- **Movie**
Classe astratta che implementa i metodi dell'interfaccia "MovieInterface" che ha al suo interno i metodi getter e setter per i dati dei film. Inoltre ha anche al suo interno un metodo astratto getPrice() che viene implementato dalle sottoclassi.
3- **NewRelease**
Classe concreta che estende la classe astratta "Movie". Al suo interno ha una variabile statica costante che è il prezzo del film di questo tipo di oggetto. Un oggetto di questo tipo viene considerato una novità.
4- **NormalMovie**
Classe concreta che estende la classe astratta "Movie". Al suo interno ha una variabile statica costante che è il prezzo del film di questo tipo di oggetto. Un oggetto di questo tipo viene considerato un film normale e non novità.

pacchetto movietime.core.ordermanager

1- **OrderManager**
Interfaccia che viene implementata dalla classe "Order". Dichiara i metodi per gestire un certo ordine fra cui add(Movie m) che aggiunge un film alla lista dei miei ordini, remove(Movie m) che rimuove un film e getPriceListMovie() che restituisce il prezzo totale dei film nella lista.
2- **Order**
Classe che implementa l'interfaccia "OrderManager". Nel costruttore crea una lista di oggetti di tipo "Movie", la quale viene gestita con i metodi implementati dall'interfaccia. Inoltre ha delle variabili che sono: "dayOfRental", che sono il numero dei giorni di noleggio, "priceOrder", che indica il prezzo totale dei film nell'ordine, "discount" che indica lo sconto applicato e "total" che rappresenta il totale dell'ordine con relativo sconto applicato.

pacchetto movietime.core.tabledatabase

1- **CreateTableDatabase**
   Classe concreta che estende la classe "OperationDatabase". Dichiara solo un metodo "selectAllRow()" che restituisce una matrice di oggetti della classe Object che mi serviranno per la costruzione delle tabelle contenute nei pannelli di modifica dei database. Questa classe eredita dalla super classe gli oggetti del pacchetto java.sql che sono "Connection" e "Statement" e gli altri metodi getter e setter.
2- **OperationDatabase**
   Classe concreta che dichiara i metodi di connessione al database del sistema.
3- **TableInterface**
   Interfaccia che dichiara i metodi per la gestione delle tabelle dei pannelli "EditDatabaseMovie" e "EditDatabaseUser". Il metodo "CreateTable" che prende in input un array di oggetti di tipo String per determinare il nome delle colonne della tabella e ne crea un istanza.
   Dopodiché abbiamo i metodi "addRow()","updateRow()" e "removeRow" per aggiungere, modificare o rimuovere una riga dalla tabella creata.
4- **TableMovie**
   Classe concreta che estende "JTable" e implementa l'interfaccia "TableInterface" dal quale ne eredita i metodi. In questa classe viene creata la tabella che viene aggiunta alla "JScrollPane" del pannello "EditDatabaseMovie".
5- **TableUser**
   Classe concreta che estende "JTable" e implementa l'interfaccia "TableInterface" dal quale ne eredita i metodi. In questa classe viene creata la tabella che viene aggiunta alla "JScrollPane" del pannello "EditDatabaseUser".

pacchetto movietime.core.user

1- **User**
   Interfaccia che dichiara i metodi getter e setter che utilizza la classe astratta "AbsUser" i quali servono per richiamare i dati degli utenti (nome,cognome,indirizzo,ecc.) e settarli in caso di modifica. Inoltre dichiara il metodo "accept(Visitor v)" che viene utilizzato per creare il pannello del menu principale.
2- **AbsUser**
   Classe astratta che implementa l'interfaccia "User" che ne eredita i metodi e include tutte le variabili per la costruzione dell'oggetto (id_utente,nome,cognome,indirizzo,ecc.). Il tipo di oggetto non può essere

istanziato e la costruzione viene specializzata dalle sottoclassi “Client” e “Admin”.

3- **Client**
   Classe concreta che estende la classe astratta “AbsUser”. Questo tipo di utente ha la possibilità di noleggiare i film.
4- **Admin**
   Classe concreta che estende la classe astratta “AbsUser”. Questo tipo di utente ha la possibilità di noleggiare i film e di gestire il database del sistema.

pacchetto movietime.gui.component

1- **ComponentGetter**
   Classe che implementa l'interfaccia “ComponenteGetterInterface”, implementa quindi il metodo "getComponents", che viene richiamato finche' non restituisce una lista di componenti.
2- **ComponentGetterInterface**
   Interfaccia implementata dalla classe "ComponentGetter". Dichiara un unico metodo il quale deve restituire una lista di componenti presenti all'interno di un "container" specificato come parametro in input.
3- **ComponentSetter**
   Classe concreta che dichiara al suo interno tutti i metodi per settare i componenti creati, ossia le dimensioni, il formato del testo e lo fa settando una variabile di classe di tipo “StandardComponentInterface”.
4- **StandardComponentInterface**
   Interfaccia che dichiara i metodi che vengono utilizzati dai componenti Standard (StandardButton,StandardLabel,ecc.). Servono per la creazione di un nuovo componente e il settaggio delle dimensioni di esso. L’oggetto creato può essere richiamato attraverso il metodo getStandardComponent().
5- **StandardButton**
   Classe concreta che estende la classe “JButton” e implementa l’interfaccia “StandardComponentInterface”. Eredita i metodi di creazione e settaggio del componente che viene inizializzato dentro la classe ed è di tipo “JButton”.
6- **StandardComboBox**
   Classe concreta che estende la classe “JComboBox” e implementa l’interfaccia “StandardComponentInterface”. Eredita i metodi di creazione e settaggio del componente che viene inizializzato dentro la classe ed è di tipo “JComboBox”.
7- **StandardJList**
   Classe concreta che estende la classe “JList” e implementa l’interfaccia “StandardComponentInterface”. Eredita i metodi di creazione e settaggio del componente che viene inizializzato dentro la classe ed è di tipo “JList”.
8- **StandardJSpinner**

Classe concreta che estende la classe “JSpinner” e implementa l’interfaccia “StandardComponentInterface”. Eredita i metodi di creazione e settaggio del componente che viene inizializzato dentro la classe ed è di tipo “JSpinner”

9- **StandardLabel**
Classe concreta che estende la classe “JLabel” e implementa l’interfaccia “StandardComponentInterface”. Eredita i metodi di creazione e settaggio del componente che viene inizializzato dentro la classe ed è di tipo “JLabel”

10- **StandardPasswordField**
Classe concreta che estende la classe “JPasswordField” e implementa l’interfaccia “StandardComponentInterface”. Eredita i metodi di creazione e settaggio del componente che viene inizializzato dentro la classe ed è di tipo “JPasswordField”.

11- **StandardTextField**
Classe concreta che estende la classe “JTextField” e implementa l’interfaccia “StandardComponentInterface”. Eredita i metodi di creazione e settaggio del componente che viene inizializzato dentro la classe ed è di tipo “JTextField”

pacchetto movietime.gui.panel

1- **HompagePanel**
Classe concreta che estende “JPanelWithBackground” che ne eredita tutti i metodi. La classe rappresenta il pannello iniziale e al suo interno ha un oggetto di tipo “PanelLogin” dove è possibile effettuare il login e anche un oggetto “SetAccessUser” per settare i dati dell’utente che ha effettuato l’accesso.

2- **JPanelWithBackground**
Classe astratta che estende la classe “JPanel” e implementa l’interfaccia “PanelInterface”. Da quest’ultima implementa i metodi “setPanelVisibility()” che setta la visibilità dei pannelli che vogliamo rendere invisibili,”SetVisiblePanel()” che rende un pannello visibile e “getFrame()” che restitusice l’oggetto “JFrame” utilizzato come parametro per la creazione di altri pannelli dello stesso tipo. Inoltre la classe dichiara i metodi “setSize()” che setta la dimensione della finestra principale e dei panelli attraverso un oggetto di tipo “Dimension”, “setImage()” che mi permette di settare una certa immagine di sfondo del pannello e “loadImage()” che utilizza l’oggetto “MediaTracker” per caricare l’immagine settata.
Oltre a tutti questi metodi dichiara anche dei metodi getter e setter: molto utile è il getUser() che restituisce l’utente che ha effettuato l’accesso di modo che possano essere estrapolate tutti i dati di quest’ultimo.

3- **Panel interface**
Interfaccia che viene implementata dalla classe “JPanelWithBackground”.

4- **SetterComponentUserData**

Interfaccia che viene implementata dai pannelli “RentalPanel”, “DataUserPanel”,”OrderPanel”,”RechargePanel”,”EditDatabaseMoviePanel” e “EditDatabaseUserPanel”.Dichiara un unico metodo “setComponent()” che prende in input un utente e i due database e setta attraverso essi tutti i componenti del pannello che richiama tale metodo.

pacchetto movietime.gui.panelwithbackbutton

1- **RentalPanel**
Classe concreta che estende la classe astratta “JPanelwithBackgroundandBackButton” e implementa le interfacce “SetterComponentUserData”e “InitializeComponentRentalPanel”. Quest’ultima interfaccia dichiara tutti i metodi che vengono richiamati nel costruttore ogni volta che si crea un oggetto del tipo “RentaPanel” e servono per inizializzare i componenti che formano tutto il pannello. Questa classe serve per scegliere i film nel catalogo e aggiungerli al nostro ordine.

2- **RechargePanel**
Classe concreta che estende la classe astratta “JPanelwithBackgroundandBackButton” e implementa le interfacce “SetterComponentUserData”e “InitializeComponentRechargePanel”. Quest’ultima interfaccia dichiara tutti i metodi che vengono richiamati nel costruttore ogni volta che si crea un oggetto del tipo “RechargePanel” e servono per inizializzare i componenti che formano tutto il pannello. Questa classe serve per ricaricare il proprio credito.

3- **OrderPanel**
Classe concreta che estende la classe astratta “JPanelwithBackgroundandBackButton” e implementa le interfacce “SetterComponentUserData”e “InitializeComponentOrderPanel”. Quest’ultima interfaccia dichiara tutti i metodi che vengono richiamati nel costruttore ogni volta che si crea un oggetto del tipo “OrderPanel” e servono per inizializzare i componenti che formano tutto il pannello. Questa classe serve per gestire il nostro ordine ed effettuare il pagamento.

4- **DataUserPanel**
Classe concreta che estende la classe astratta “JPanelwithBackgroundandBackButton” e implementa le interfacce “SetterComponentUserData”e “InitializeComponentDataUserPanel”. Quest’ultima interfaccia dichiara tutti i metodi che vengono richiamati nel costruttore ogni volta che si crea un oggetto del tipo “DataUserPanel” e

servono per inizializzare i componenti che formano tutto il pannello. Questa classe serve per gestire i propri dati e modificarli.

5- **EditDatabaseMoviePanel**
Classe concreta che estende la classe “EditDatabasePanel” e implementa le interfacce “SetterComponentUserData”e “InitializeComponentEditDatabasePanel”. Quest’ultima interfaccia dichiara tutti i metodi che vengono richiamati nel costruttore ogni volta che si crea un oggetto del tipo “EditDatabasePanel” e servono per inizializzare i componenti che formano tutto il pannello. Questa classe serve per gestire il database dei film.

6- **EditDatabaseUserPanel**
Classe concreta che estende la classe “EditDatabasePanel” e implementa le interfacce “SetterComponentUserData”e “InitializeComponentEditDatabasePanel”. Quest’ultima interfaccia dichiara tutti i metodi che vengono richiamati nel costruttore ogni volta che si crea un oggetto del tipo “EditDatabasePanel” e servono per inizializzare i componenti che formano tutto il pannello. Questa classe serve per gestire il database degli utenti.

7- **EditDatabasePanel**
Classe concreta che estende la classe “JPanelwithBackgroundandBackButton” ed essa viene estesa dalle classi “EditDatabaseMoviePanel” e “EditDatabaseUserPanel”. Nella classe si dichiara dei metodi che servono per la costruzione dei bottoni “Add Object”,”Remove Object” e “Update Object” e la creazione di una “JScrollPane” per permettere di visualizzare la tabella del database corrispondente. Le “ActionListener” che vengono attivate alla pressione dei bottoni di modifica del database vengono sviluppate nelle sottoclassi.

8- **JPanelWithBackgroundAndBackButton**
Classe astratta che estende la classe “JPanelWithBackground”. La funzionalità di questa classe è quella di creare un bottone che serve per tornare al pannello precedente.

9- **InitializeComponentRentalPanel**
Interfaccia che viene implementata dalla classe “RentalPanel” e implementa tutti i metodi per la costruzione dei componenti.

10- **InitializeComponentRechargePanel**
Interfaccia che viene implementata dalla classe “RechargePanel” e implementa tutti i metodi per la costruzione dei componenti.

11- **InitializeComponentDataOrderPanel**

Interfaccia che viene implementata dalla classe “OrderPanel” e implementa tutti i metodi per la costruzione dei componenti.

12- **InitializeComponentDataUserPanel**
Interfaccia che viene implementata dalla classe “DataUserPanel” e implementa tutti i metodi per la costruzione dei componenti.

13- **InitializeComponentEditPanel**
Interfaccia che viene implementata dalla classe “EditUserPanel”e “EditMoviePanel” e implementa tutti i metodi per la costruzione dei componenti.

Descrizione dei test effettuati

pacchetto movietime.tests

1- **BasicCardTest**
Classe di test che controlla il corretto funzionamento del pagamento senza e con l’aggiunta dello sconto mediante la tessera di tipo “Basic”. Per questo test creo un’istanza di tipo “Basic”. Vengono eseguiti vari test fra cui “calculateTotalPriceWithoutDiscountTest()” che testa se viene effettuato il giusto calcolo nel totale dell’ordine senza lo sconto applicato, “executePaymentWithDiscountTest()” che testa il giusto sconto a tutto l’ordine e infine “executePaymentWithDiscountTestWrong()” che effettua il confronto con un credito sbagliato e uno giusto utilizzando il metodo “AssertNotEquals()”.

2- **ChildrenCardTest**
Classe di test che controlla il corretto funzionamento del pagamento senza e con l’aggiunta dello sconto mediante la tessera di tipo “Children” similmente alla classe precedente. Inoltre viene effettuato il test sul parentalControl attraverso il metodo test “parentalControlDangerMovieTest”.

3- **GoldCardTest**
Classe di test che controlla il corretto funzionamento del pagamento senza e con l’aggiunta dello sconto mediante la tessera di tipo “Gold”. In questo caso vengono effettuati due test, uno quando un messaggio ci chiede di applicare lo sconto del 50% e noi cliccheremo SI (“executePaymentWithYesClickForDiscountTest()”) e un altro dove cliccheremo NO (executePaymentWithNoClickForDiscountTest()). Inoltre

viene effettuato un altro test “calculateNewGoldPointTest()” che testa il funzionamento del metodo “calculateNewGoldPoint()”.

4- **GetComponentTest**
   Classe che controlla le funzionalità della classe "ComponentsGetter". Viene quindi testato il metodo utilizzato per restituire un determinato componente, nello specifico, si testa la possibilità di restituire un pulsante o un'etichetta dal frame all'interno del quale sono stati inseriti.
5- **CreateMenuPanelTest**
   Classe che controlla il metodo “createPanelAccess()” della classe “CreateMenuPanel” e testa se effettivamente viene creato il giusto oggetto di tipo “MenuClient” o “MenuAdmin”. Per fare questo utilizzo un oggetto “ComponentGetter” che mi crea una lista dei componenti della JFrame che gli passo come parametro. Dopodiché controllo con il metodo “checkComponent()” se il pannello creato con l’oggetto “CreateMenuPanel” è effettivamente creato del tipo desiderato
6- **OrderTest**
   Si testano tutti i metodi della classe "Order", attraverso la rimozione e l’aggiunta di oggetti di tipo “Movie” dalla lista dell’ordine.
   Viene verificato che il prezzo sia giusto prima o dopo la rimozione e anche quando si ha il caso di una lista vuota.
7- **ResearchMovieForTitleTest**
   Viene testato il funzionamento del metodo astratto “research()” della classe “ResearchMovie” , nel caso specifico, la ricerca dei film per titolo. Per fare questo creo due record uno buono ed uno falso e un oggetto di tipo “NormalMovie” con il titolo giusto.
   Attraverso il metodo “checkMovieIntoList()” controllo se l’oggetto è presente sia nella lista della classe test, sia nella lista temporanea che la classe “ResearchMovie” crea. Confronto le due variabili con un “AssertEquals”. Lo stesso procedimento viene fatto con il falso record.