RELAZIONE PROGETTO
Programmazione e Modellazione ad Oggetti

REALIZZATO DA:

[redacted]

[redacted]

Sessione Autunnale 2020/21

1. Specifica del Software

Il progetto consiste nell'implementazione di un'interfaccia grafica che traccia le spese di una famiglia.

Il programma permette all'utente di:

1. Accedere al proprio profilo tramite Password (stabilita di default, sarà "0000");
2. Memorizzare nuove spese;
3. Cancellare una spesa;
4. Calcolare delle statistiche;
5. Visualizzare tutte le spese e le statistiche;

Le statistiche di interesse sono:

- Il totale speso;
- Una media di quando si è speso;
- Il saldo rimanente (all'inizio, il conto parte da 10000 euro)

Ogni prodotto comprato, avrà le seguenti caratteristiche:

- Nome di chi ha eseguito la spesa;
- Cosa si è comprato;
- Il prezzo del prodotto;
- Quante unità sono state comprate;
- Data dell'acquisto;
- Categoria di appartenenza dell'acquisto (Alimentari, Salute...);

Inoltre, il programma salva i dati in formato JSON su disco.

2. Studio del problema

I punti critici del progetto sono i seguenti:

- Accedere al proprio Account
  - Per accedere alla schermata di gestione delle spese, l'utente dovrà inserire una specifica password che, come sopracitato dalla specifica, è già stabilita ("0000") e cliccare sul tasto "Login". Se la password è sbagliata, verrà visualizzato un messaggio di errore.

- Aggiungere una nuova spesa.
  - In questa fase l'utente inserisce i dati associati all'acquisto, ovvero nome di chi ha effettuato la spesa, prodotto, prezzo, quantità, categoria e data. Inoltre, alcuni campi devono essere compilati obbligatoriamente, altrimenti viene visualizzato un messaggio di errore. Una volta che sono stati inseriti tutti i dati il programma li elabora e crea un nuovo oggetto di tipo ListViewItem da inserire nell'apposita ListView.

- Modificare una spesa già memorizzata.
  - Qui, l'utente deve prima cliccare su un elemento della ListView (lstItems) da modificare e successivamente cliccare sull'apposito tasto "MODIFICA". Una volta fatto ciò, i campi della sezione "Nuova spesa" verranno riempiti con le caratteristiche dell'elemento selezionato e, quest'ultimo, scomparirà dalla ListView (Database). Infine, basterà premere di nuovo il tasto "AGGIUNGI".

- Eliminare una spessa memorizzata.
  - Anche qui, l'utente deve prima selezionare una determinata spesa dalla ListView e successivamente cliccare sul tasto "CANCELLA". Comparirà, anche, una finestra di avvertimento per confermare la scelta e l'elemento verrà cancellato dal Database.

- Visualizzare tutte le informazioni associate ai vari acquisti.
  - Tutti le varie spese e le statiche associate a quest'ultime saranno visualizzate nella sezione "Storico e Statistiche" della stessa finestra della gestione delle spese e aggiornate automaticamente ogni volta che si aggiunge, modifica o elimina un acquisto.

- Salvare e caricare il Database.
  - Il programma, per poter lavorare in maniera corretta, utilizza un file in formato JSON per salvare tutti gli acquisti effettuati.

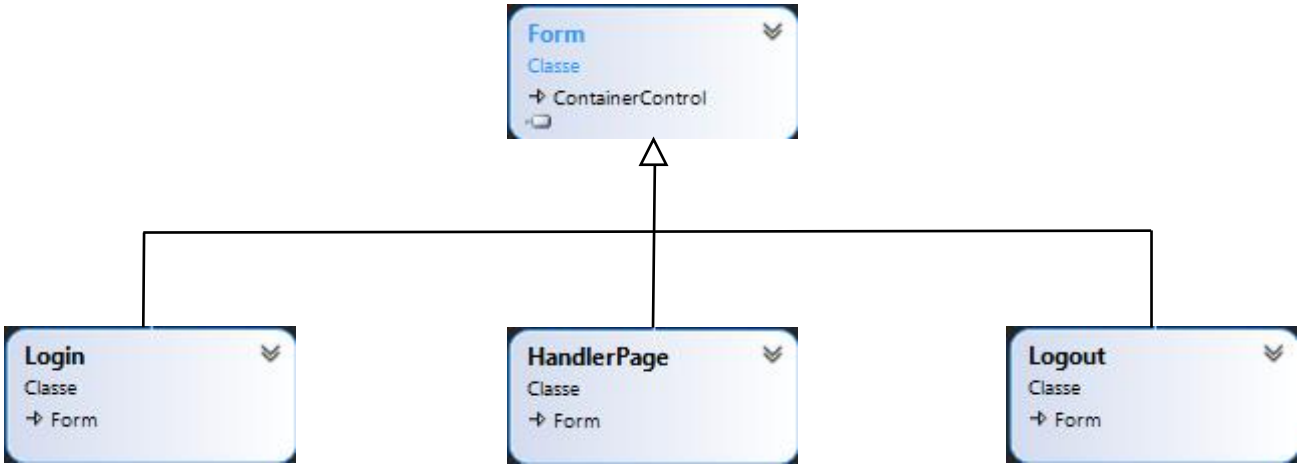
- Uscire dal proprio Account
  - Per effettuare il Log out, l'utente dovrà premere sul tasto "LOG OUT". A seguire comparirà una finestra di avvertimento per confermare tale scelta.

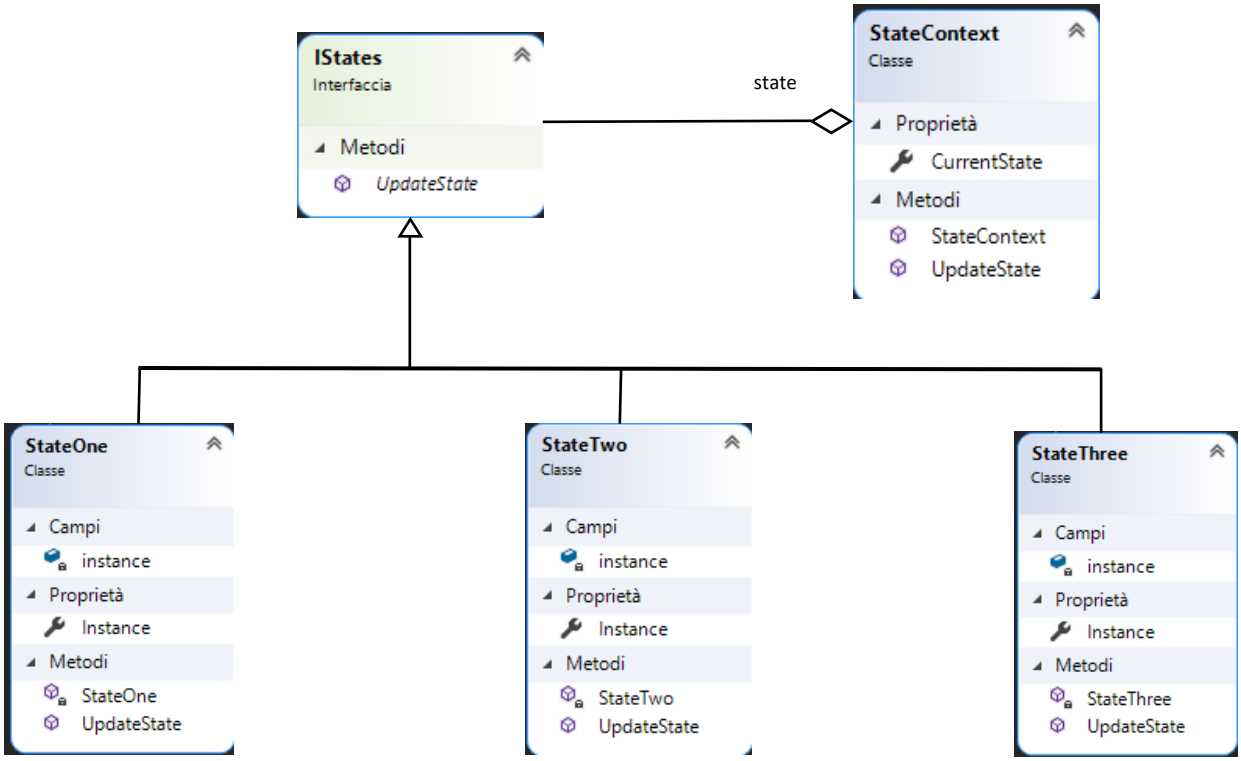
3. Scelte architetturali

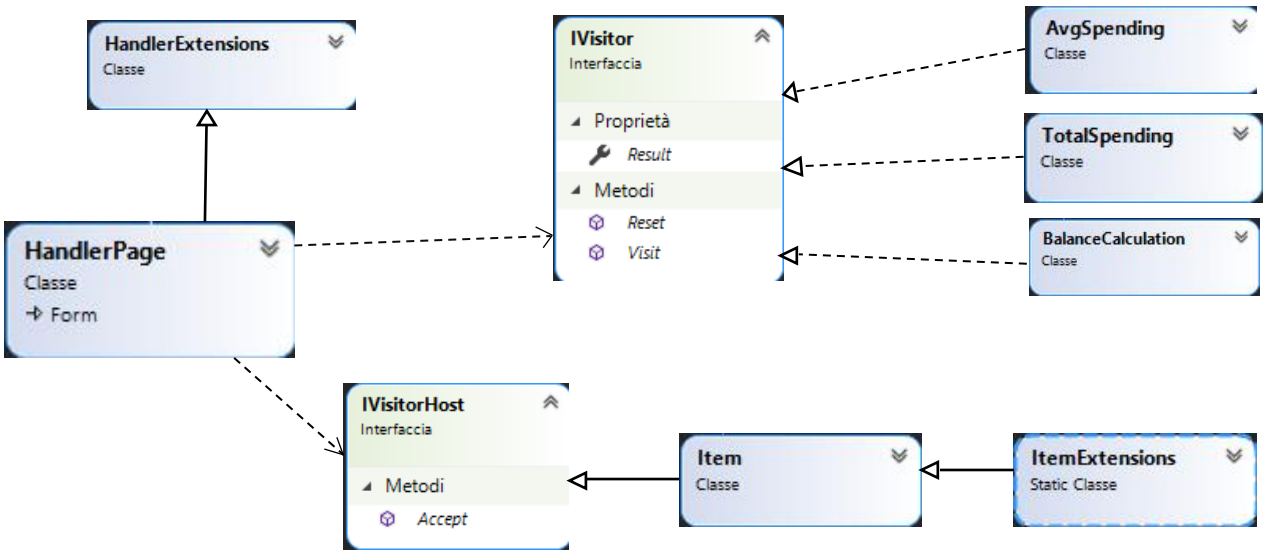
Il programma conta tre moduli (Login.cs, HandlerPage.cs e Logout.cs) in cui il principale è quello in cui avviene la gestione vera e propria delle spese, ovvero HandlerPage.cs. Le varie Form si apriranno in successione, una volta premuto uno specifico tasto, rispettivamente Login.cs, HandlerPage.cs e Logout.cs.

Inoltre, all'interno del programma, sono presenti varie cartelle che raggruppano classi utilizzate per specifici compiti. Esse sono: Calculations (IVisitor.cs, IVisitorHost.cs, TotalSpending.cs, AvgSpending, BalanceCalculation.cs), Database (LocalDB.cs), Extensions (HandlerExtensions.cs, ItemExtensions.cs), Models (Item.cs) e States (IStates.cs, StateContext.cs, StateOne.cs, StateTwo.cs, StateThree.cs).

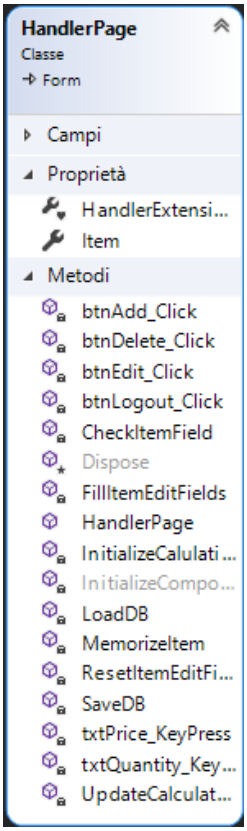
Di seguito, i vari schemi delle classi UML, divisi per Pattern e Form.







Descrizione delle varie classi utilizzate:



<u>Metodi</u>

btnAdd_Click(): metodo privato che permette l'aggiunta la memorizzazione di una nuova spesa all'interno del Database.

btnDelelte_Click(): metodo privato che permette la cancellazione di un acquisto già memorizzato.

btnEdit_Click(): metodo privato che permette la modifica di una spesa già memorizzata.

btnLogout(): metodo privato che permette l'uscita dalla proprio account di gestione spese.

Dipose(): metodo protetto che rilascia ogni risorsa utilizzata dal Form. Viene utilizzato per la chiusura di quest'ultimo.

FillItemEditFields(): metodo privato che permette il riempimento dei campi dell'elemento (ovvero della spesa).

HandlerPage(): costruttore della classe.

InitializeCalulations(): metodo che inizializza i calcoli per le statistiche.

InitializeComponent(): metodo che inizializza i componenti del Form.

LoadDB(): metodo che carica i vari dati nella ListView.

MemorizeItem(): metodo che memorizza la spesa con tutti i campi associati.

ResetItemEditFields(): metodo che permette il reset dei campi associati alla spesa.

SaveDB(): metodo utilizzato per salvare i vari la spesa sul Database.

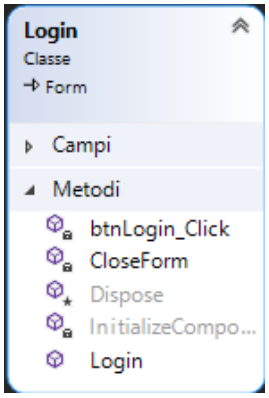
txtPrice_KeyPress(): metodo utilizzato per la validazione dell'input del prezzo.

txtQuantity_KeyPress(): metodo utilizzato per la validazione dell'input.

UpdateCalculations(): metodo utilizzato per aggiornare i calcoli.

Descrizione

Form che permette la gestione delle spese, tramite la memorizzazione, cancellazione e modifica di quest'ultime. Inoltre, è possibile visualizzare delle statistiche legate al complessivo speso.



Metodi

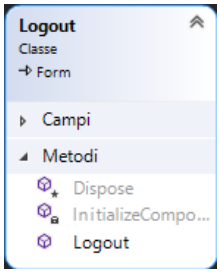
btnLogin_Click(): metodo che permette di accedere al proprio account.

CloseForm(): metodo che controlla la password inserita dall'utente e chiude il Form.

Login(): costruttore della classe.

Descrizione

Prima Form visualizzata dall'utente che permette l'accesso al proprio Account (La password già stabilita, è "0000").

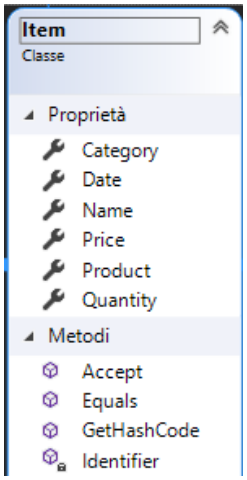


Metodi

Logout(): costruttore della classe.

Descrizione

Ultima Form visualizzata dall'utente che mostra un messaggio di log out avvenuto.



Metodi

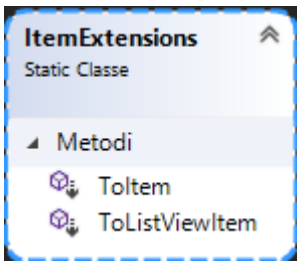
Accept(): metodo che invoca il corrispondente metodo Visit per l'oggetto.

Equals(): metodo che tratta come una sola spesa, due prodotti uguali.

Identifier(): metodo che restituisce il prodotto dei una spesa.

Descrizione

Classe che specifica i campi associati alla spesa da memorizzare.



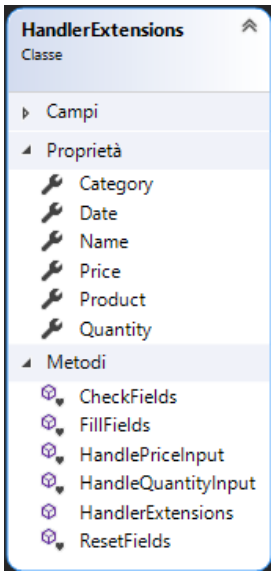
Metodi

ToListViewItem(): metodo che passa l'elemento alla ListView.

ToItem(): metodo che esegue l'azione contraria.

Descrizione

Classe di supporto a Item.cs



Metodi

CheckFields(): metodo che controlla che determinati campi siano riempiti.

FillFields(): metodo che riempi i campi associati alla spesa.

HandlePriceInput(): metodo che permette la validazione del prezzo.

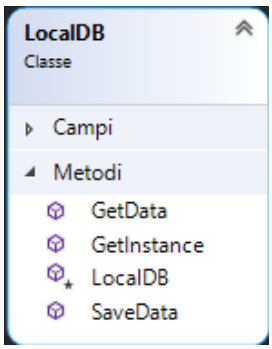
HandleQuantityInput(): metodo che permette la validazione della quantità.

HandlerExtensions(): costruttore della classe.

ResetFields(): metodo che permette il reset della classe.

Descrizione

Classe di supporto a HandlerPage.cs



Metodi

GetData(): metodo che prende i dati dal file json.

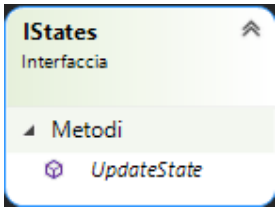
GetInstance(): implementazione dello State Pattern.

LocalDB(): metodo che copia l'indirizzo del file json.

SaveData(): metodo che salva i dati all'interno del file json.

Descrizione

Classe che gestisce il file json che funge da Database.

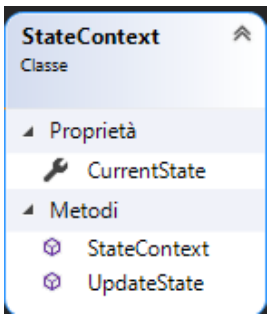


Metodi

UpdateState(): metodo che aggiorna lo stato.

Descrizione

Interfaccia utilizzata nell'implementazione dello State Pattern.



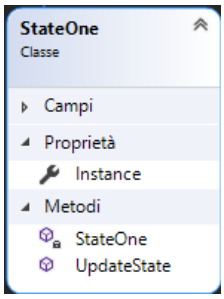
Metodi

StateContext(): costruttore della classe che inizializza il Pattern con il primo stato.

UpdateState(): metodo che richiama lo stato corrente.

Descrizione

Classe utilizzata per l'implementazione dello State Pattern.



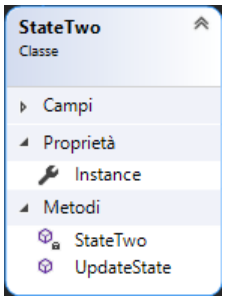
Metodi

StateOne(): implementazione del Singleton Pattern.

UpdateState(): metodo aggiorna lo stato a quello successivo.

Descrizione

Classe utilizzata per l'implementazione dello State Pattern e definisce il primo stato del programma.



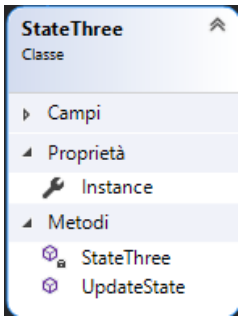
Metodi

StateTwo(): implementazione del Singleton Pattern.

UpdateState(): metodo aggiorna lo stato a quello successivo.

Descrizione

Classe utilizzata per l'implementazione dello State Pattern e definisce il secondo stato del programma.



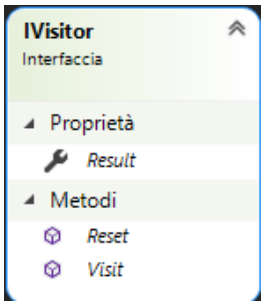
Metodi

StateThree(): implementazione del Singleton Pattern.

UpdateState(): metodo aggiorna lo stato a quello successivo.

Descrizione

Classe utilizzata per l'implementazione dello State Pattern e definisce il terzo stato del programma.



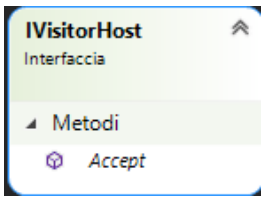
Metodi

Reset(): metodo che esegue il reset dello stato del Visitor.

Visit(): metodo che prende un elemento come parametro.

Descrizione

Interfaccia utilizzata per l'implementazione del Visitor Pattern.

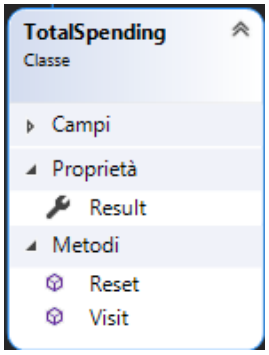


Metodi

Accept(): metodo che accetta un Visitor.

Descrizione

Interfaccia utilizzata per l'implementazione del Visitor Pattern.



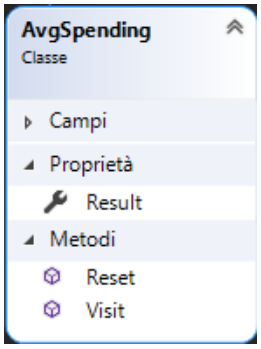
Metodi

Reset(): metodo che esegue il reset dello stato del Visitor.

Visit(): metodo che prende un elemento come parametro.

Descrizione

Classe utilizzata per l'implementazione del Visitor Pattern e che calcola il totale speso dall'utente.



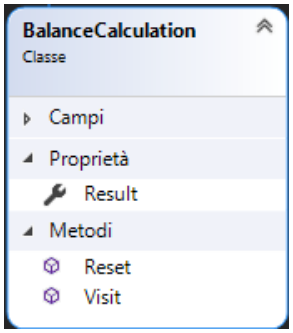
Metodi

Reset(): metodo che esegue il reset dello stato del Visitor.

Visit(): metodo che prende un elemento come parametro.

Descrizione

Classe utilizzata per l'implementazione del Visitor Pattern e che calcola la media spesa dall'utente.



Metodi

Reset(): metodo che esegue il reset dello stato del Visitor.

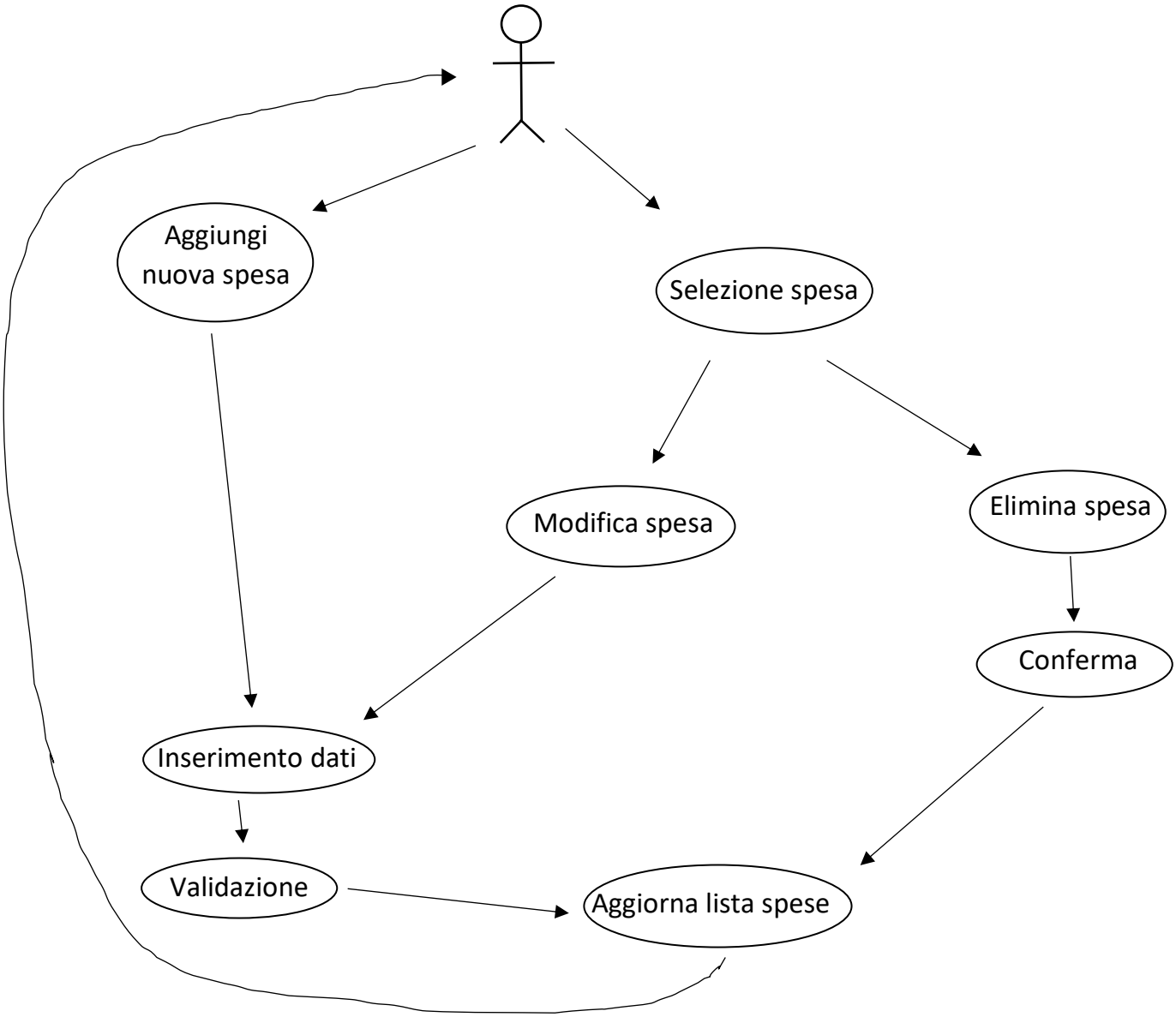
Visit(): metodo che prende un elemento come parametro.

Descrizione

Classe utilizzata per l'implementazione del Visitor Pattern e che calcola il saldo finale dell'utente (che già stabilito, parte da 10000 €).

5. Use Cases con relativo schema UML.

Verrà visualizzato lo schema dei casi d'uso più significativi del programma, ovvero quello relativo alla gestione delle spese.



Descrizione dei casi d'uso:

Use Case: Nuova Spesa	ID: UC1	Attore: Utente
Precondizione: ///		
Corso degli eventi: L'utente clicca sul bottone "AGGIUNGI".		
Postcondizione: UC6		
Alternativa: UC2		

Use Case: Selezione Spese	ID: UC2	Attore: Utente
Precondizione: ///		
Corso degli eventi: L'utente selezione una spesa.		
Postcondizione: UC3 o UC4		
Alternativa: UC1		

Use Case: Modifica Spesa	ID: UC3	Attore: Utente
Precondizione: UC2		
Corso degli eventi: L'utente clicca sul bottone "MODIFICA".		
Postcondizione: UC6		
Alternativa: UC4		

Use Case: Elimina Spesa	ID: UC4	Attore: Utente
Precondizione: UC2		
Corso degli eventi: L'utente clicca il pulsante "ELIMINA".		
Postcondizione: UC5		
Alternativa: UC3		

Use Case: Conferma	ID: UC5	Attore: Utente
Precondizione: UC5		
Corso degli eventi: Compare un messaggio che chiede all'utente di confermare l'eliminazione.		
Postcondizione: UC8		
Alternativa: ///		

Use Case: Inserimento Dati	ID: UC6	Attore: Utente
Precondizione: UC1 o UC3		
Corso degli eventi: L'utente riempie i campi associati alla spesa.		
Postcondizione: UC7		
Alternativa: ///		

Use Case: Validazione	ID: UC7	Attore: Utente
Precondizione: UC6		
Corso degli eventi: Il riempimento e la correttezza dei campi associati alla spesa vengono controllati.		
Postcondizione: UC6 o UC8		
Alternativa: ///		

Use Case: Aggiorna lista spese	ID: UC8	Attore: Utente
Precondizione: UC7		
Corso degli eventi: Viene aggiornato il Database che contiene tutte le spese memorizzate e le statistiche relative al conto.		
Postcondizione: ///		
Alternativa: ///		