Relazione JBudget

Studente: Greta Sorritelli

Matricola: 104952

I tre principali concetti di quest'app per la gestione delle spese familiari sono:

- *Conto:* due tipologie di conti: gli *asset* che rappresentano la nostra disponibilità di denaro; le *liabilities* che rappresentano invece dei *debiti* da estinguere.
- *Movimento:* rappresenta una *uscita* o *entrata* da un conto.
- *Transazione:* rappresenta un insieme di movimenti. L'uso della *transazione* è necessario perché alcune spese possono coinvolgere più movimenti.

Modello:

Interfacce usate:

Account: questa interfaccia è implementata dalle classi che hanno la responsabilità di gestire un conto. Permette di accedere e modificare le informazioni del conto: *ID*, *nome*, *descrizione, saldo iniziale, tipologia e lista di movimenti*. Consente inoltre di ottenere il saldo attuale cioè il *balance*.

Il balance verrà aumentato o ridotto a seconda del tipo di account e del tipo di movimento.

Movement: questa interfaccia è implementata dalle classi che hanno la responsabilità di gestire un singolo movimento. Ogni movimento ha: *ID*, *descrizione, tipo, importo, account associato, lista dei tag associati al movimento e transazione di cui fa parte, data.*

Transaction: questa interfaccia è implementata dalle classi che hanno la responsabilità di gestire una transazione. Permette di accedere e modificare le informazioni associate ad una transazione: *ID, lista dei tag, data, lista dei movimenti.* Un tag associato (o rimosso) ad una transazione viene aggiunto (o rimosso) ad ogni movimento della transazione. La transazione ha anche un *saldo* (ottenibile tramite il metodo *getTotalAmount())* che permette di ottenere la variazione totale dei movimenti della transazione.
Nella transazione avviene la creazione di un movimento associato.

Tag: ha un codice ID univoco ed è rappresentato da un nome ed una descrizione.
Un tag si riferisce a un movimento e una transazione.

Ledger: questa interfaccia è implementata dalle classi che hanno la responsabilità di gestire tutti i dati dell'applicazione. È responsabile della creazione dei conti, dell'aggiunta e cancellazione delle transazioni, della creazione e cancellazione dei tag.

Classi:

SimpleAccount: SimpleAccount presenta come parametri: *ID*, *tipologia, nome*, *descrizione, saldo iniziale.* Ha anche un *balance* per le modifiche successive relative al saldo del conto e una *lista di movimenti* associati.

Ogni conto possiede un balance positivo, sia che l'account sia di tipo ASSET che di tipo LIABILITIES.

Ha la responsabilità di modificare il balance secondo i movimenti associati.

Il metodo che realizza queste operazioni è refreshBalance() che aggiunge o sottrae al balance l'amount di un movimento a seconda dei rispettivi tipi.

Nel caso di LIABILITIES il saldo viene visto come un debito da estinguere. Il debito si riduce per ogni movimento di tipo CREDIT fino ad arrivare a zero.

Se invece si fa un movimento di tipo DEBIT verso il conto LIABILITIES allora il debito crescerà.

Il metodo refreshBalance() usa i metodi addAmount(double amount) e subtractAmount(double amount).

Inoltre il metodo revertBalance() si occupa di aggiornare il balance dopo l'eliminazione di un movimento, come se questo non fosse stato mai effettuato.

SimpleMovement: l'oggetto di tipo SimpleMovement ha i seguenti campi: *ID*, *descrizione, tipo, importo, account associato, lista dei tag associati al movimento e transazione di cui fa parte.*

Viene aggiunto tramite una transazione, un movimento fa sempre parte di una transazione.

Ha la responsabilità di impostare la transazione di cui fa parte come parametro *transaction* quando la transazione aggiunge il movimento. Durante questa operazione il movimento ha anche la responsabilità di assegnare al *date,* la data della transazione.

Un movimento si occupa anche di aggiungere o rimuovere dei tag, indipendentemente da quelli della transazione.

SimpleTransaction: Una transazione possiede: *ID, lista dei tag, data, lista dei movimenti.*
Ha le seguenti responsabilità:
Aggiornare i tag della transazione sulla lista dei tag dei movimenti: un tag che fa parte della transazione fa parte anche dei movimenti di questo, se si rimuove un tag dalla transazione, lo si rimuove anche dai movimenti.
Aggiungere un movimento (o rimuoverlo) ed associarsi ad esso per potergli impostare la propria data.
La transazione ha anche un *saldo* (ottenibile tramite il metodo *getTotalAmount())* che permette di ottenere la variazione totale dei movimenti della transazione:
se i movimenti sono di tipo DEBIT allora il saldo diminuisce;
se i movimenti sono di tipo CREDITS allora il saldo aumenta.

SimpleTag: è rappresentato da: *ID, nome e descrizione*.
Un tag è presente sia in un movimento che in una transazione.

Se si rimuove un tag dal Ledger lo si rimuove anche dalle transazioni e dai movimenti che lo contenevano.

SimpleLedger: gestisce tutti i dati dell'applicazione.
Contiene l'insieme degli account e delle transazioni, da qui è possibile aggiungerne di nuovi o rimuoverli.
Ha la responsabilità di aggiungere all'account la lista di movimenti ad esso associati durante l'aggiunta delle transazioni. Durante l'operazione viene aggiornato il balance degli account relativi alla transazione.
Ha anche una lista di tag che è possibile modificare.
I tag presenti saranno poi disponibili per essere aggiunti alle transazioni e ai movimenti.
Può ritornare la lista di account e transazioni che rispettano un determinato predicato.

Enumerazioni:

AccountType: un conto può essere di tipo ASSET o LIABILITIES.

Il primo rappresenta una disponibilità di denaro, il secondo un debito da estinguere come ad esempio un prestito.

MovementsType: un movimento può essere di tipo DEBIT o CREDITS.

Un debit va a scalare dei soldi dal conto asset e aumenta il valore del balance del liabilities poiché fa aumentare il valore del debito da estinguere.

Un credit invece aumenta il balance di un asset e scala una somma da un debito.

Controller:

Interfaccia:

LedgerController: interfaccia per il controller del Ledger. Ha la responsabilità di stabilire le operazioni principali per gestire i dati e le operazioni principali. Ha la responsabilità di fornire dei metodi per esportare ed importare i dati su/da file.

Classe:

SimpleLedgerController: ha la responsabilità di stabilire le operazioni principali per gestire i dati e le operazioni principali. Ha un metodo di controllo per i movimenti da inserire.

Fornisce dei metodi per importare ed esportare su file txt, quali per le conversioni di stringhe in oggetti e viceversa.

Una volta riconvertite le stringhe in oggetti si occupa di aggiungerli nel Ledger e reimpostare le relazioni tra i diversi oggetti che erano correlati.

View:

Interfacce:

JavaFXAddTag: Interfaccia per la creazione di una finestra di aggiunta per Tag.

JavaFXRemoveTag: Interfaccia per la creazione di una finestra di rimozione per Tag.

JavaFXFilter: è implementata da JavaFXAccountMovements e JavaFXFilterTransactions per filtrare i movimenti e le transazioni.

Classi:

JavaFXJBudget: Classe principale per la GUI JavaFX. Apre la finestra principale.

JavaFXLedgerController: è il controller della finestra principale. Gestisce le operazioni principali.

Mostra gli Account, le Transaction e i Tag presenti nel Ledger. Inoltre mostra tutti i Movement.

Apre nuove finestre per l'aggiunta di Account, Transaction e Tag.

Apre nuove finestre per aggiungere o rimuovere i Tag dalle Transaction e dai Movement.

Apre una finestra per mostrare i Movement di un Account scelto ed eventualmente filtrarli.

Apre una finestra per filtrare le Transaction.

Inoltre, permette di scegliere una cartella su cui salvare i dati, ed anche importarli, in formato txt.

JavaFXAddAccount: Questa classe serve per aggiungere un Account al Ledger. Controller della finestra di aggiunta.

JavaFXAddTransaction: Classe per la creazione di una Transaction e i relativi Movement presenti in esso. Controller della finestra per la creazione di Transaction e Movement.

JavaFXAddTagLedger: Aggiunta di un Tag al Ledger. Gestisce la finestra per aggiungere un Tag.

JavaFXAddTagTransaction: Aggiunta di un Tag ad una Transaction. Gestisce la finestra per aggiungere un Tag.

JavaFXAddTagMovement: Aggiunta di un Tag ad un Movement. Gestisce la finestra per aggiungere un Tag.

JavaFXRemoveTagTransaction: Rimozione di un Tag da una Transaction. Gestisce la finestra per la rimozione.

JavaFXRemoveTagMovement: Rimozione di un Tag da un Movement. Gestisce la finestra per la rimozione.

JavaFXAccountMovements: Gestisce la finestra che mostra i Movement di un Account scelto dalla finestra principale. Si possono filtrare i movimenti in base a: *ID, data, tag, transazione, tipo*.

JavaFXFilterTransactions: Controllore della finestra per filtrare tutte le Transaction.

Queste sono filtrate in base a: *ID, data, tag e amount complessivo(positivo o negativo).*

Persistence:

Interfacce:

Saver: interfaccia per il salvataggio su file, fornisce un metodo generale per salvare tutto.

L'utente deve poter salvare in modi diversi.

Importer: interfaccia per l'importazione da file, fornisce un metodo generale per importare da file.

Si deve poter garantire il salvataggio su diversi tipi di file.

Classi:

SaverTxt: Classe per salvare i dati presenti nel Ledger nei file txt, creati nel percorso scelto dall'utente. In particolare si occupa di salvare in diversi file: Account, Transaction, Movement e Tag.

Converte gli oggetti in stringhe da scrivere nei file di testo.

ImporterTxt: Classe per recuperare le informazioni dai file soprastanti, in formato txt, e riconvertire le stringhe in oggetti. Si occupa poi di aggiungere gli oggetti al Ledger e ristabilisce le relazioni tra oggetti diversi, quali ad esempio un movimento associato ad un account.

Test:

SimpleAccountTest: controlla che il costruttore della classe generi le giuste eccezioni per i parametri da controllare con il metodo testSimpleAccount(), si accerta che un account contenga un movimento creato in seguito ad una transazione e che il balance dell'account vari in modo adeguato, attraverso i metodi testAddAmount() e testSubtractAmount().

Un metodo inoltre controlla che getMovements(Predicate<Movements > p) ritorni la lista giusta di movimenti attraverso testGetMovements().

SimpleMovementTest: verifica che i tag vengano aggiunti e rimossi in modo giusto con testAddTag() e testRemoveTag(), un tag non può essere aggiunto più volte.

SimpleTransactionTest: verifica la correttezza del metodo getTotalAmount() e i metodi testAddTag() e testRemoveTag() per aggiungere e rimuovere i tag, sia dalla transazione che dai metodi presenti in questa transazione.

Durante l'aggiunta e la rimozione di un movimento verifica che il movimento sia aggiunto/rimosso dalla lista con i metodi testAddMovement() e testRemoveMovement().

SimpleLedgerTest: con questo test si verifica la correttezza del metodo addAccount() e removeAccount() rispetto alla lista di account del Ledger, verifica che addTransaction() aggiunga una transazione e che non sia già presente. Verifica che removeTransaction() rimuova anche i movimenti interni alla transazione sia dal Ledger che dall'account. Verifica anche l'aggiunta e la rimozione dei tag dalla lista relativa presente nel Ledger con i metodi testAddTag() e testRemoveTag().

Un test inoltre controlla che getTransactions(Predicate<Transactions> p) ritorni la lista giusta di transazioni.

SimpleLedgerController: esegue il test del metodo control(Movement movement).

Il metodo si accerta che un movimento possa essere effettuato in base al balance dell'account.

Estendibilità:

Per poter rendere l'applicazione estendibile vengono usate le interfacce come modello da poter implementare da diverse classi.

Funzionalità implementate per l'utente:

Visualizzazione di un Ledger (libro mastro) contenente tutti i dati: l'utente può visualizzare tutti gli account, le transazioni, i movimenti e i tag nella finestra principale "JBudget". I dati vengono prelevati da un oggetto SimpleLedger.

Questa finestra è gestita dal controllore JavaFXLedgerController. Da qui si possono eliminare gli account, le transazioni e i tag tramite la selezione (anche multipla).

Dalla finestra si possono aprire altre finestre per le seguenti operazioni:

Aggiunta di un nuovo account: si apre una nuova finestra per l'aggiunta di un oggetto SimpleAccount. La finestra "Add a new account" è gestita dal controllore JavaFXAddAccount.

Aggiunta di una nuova transazione e relativi movimenti: nella finestra "Add a new transaction" gestita da JavaFXAddTransaction, è possibile aggiungere una transazione e dei movimenti, questi possono essere anche rimossi dalla lista prima di completare l'operazione.

Aggiunta di un nuovo tag: si apre la finestra "Add a new tag" dove è possibile aggiungere un nuovo tag al Ledger. La finestra è gestita da JavaFXAddTagLedger.

Aggiunta di un tag ad una transazione o un movimento: per aggiungere i tag alle transazioni e ai movimenti si utilizzano le finestre "Add a new tag to a transaction" e "Add tags to the movement". I loro controllori sono JavaFXAddTagTransaction e JavaFXAddTagMovement.

Rimozione di un tag da una transazione o da un movimento: i tag vengono rimossi sia dalle transazioni che dai movimenti attraverso le finestre "Remove a tag from a transaction" e "Remove a tag from a movement" gestite rispettivamente da JavaFXRemoveTagTransaction e JavaFXRemoveTagMovement.

Visualizzazione dei movimenti relativi a un account e filtri: facendo doppio click su un account dalla lista della finestra principale è possibile visualizzare i movimenti associati (se presenti) nella finestra "Filter Movements" gestita da JavaFXAccountMovements. Da qui si possono impostare vari filtri per ottenere i movimenti che soddisfano tali condizioni.

Filtrare le transazioni: attraverso la finestra "Filter transactions", gestita da JavaFXFilterTransactions, è possibile filtrare tutte le transazioni secondo certi parametri inseriti dall'utente.

Esportare i contenuti in un file di testo: è possibile esportare le informazioni che l'utente inserisce in file txt attraverso un bottone che farà scegliere all'utente dove salvare i file, precisamente in quale cartella.

Importare i contenuti da un file di testo: è possibile importare le informazioni dai file txt selezionati nella cartella scelta dall'utente grazie ad un bottone per l'import.