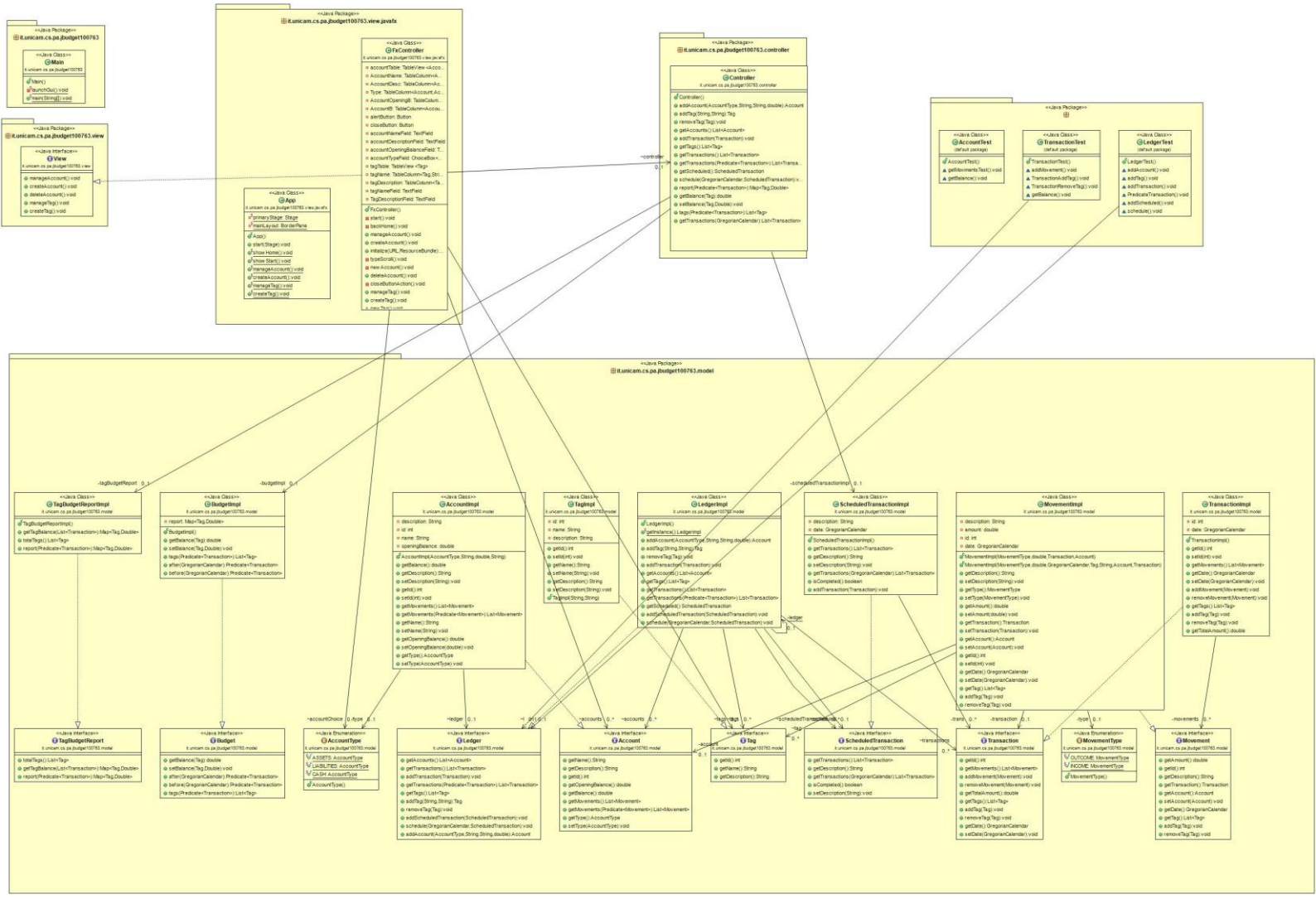
JBudget – PA2020



Per una più agevole visualizzazione del sistema e del funzionamento dei metodi implementati si consiglia la lettura della documentazione Javadoc.

VITTORIO RINALDI 100763

Package it.unicam.cs.pa.jbudget100763.model

Interface Summary

Interface	Description
Account	questa interfaccia è implementata dalle classi che hanno la responsabilità di gestire un conto.
Budget	ha la responsabilità di rappresentare e gestire un particolare budget, ovvero la previsione di spesa/guadagno per uno o più Tag.
Ledger	questa interfaccia ha 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, dell'interazione con le transazioni schedulate.
Movement	questa interfaccia è implementata dalle classi che hanno la responsabilità di gestire un singolo movimento.
ScheduledTransaction	indica una transazione o una serie di transazioni schedulate (previste) ad una certa data.
Tag	Indica la categoria di un movimento e della sua transazione
TagBudgetReport	
Transaction	questa interfaccia è implementata dalle classi che hanno la responsabilità di gestire una transazione.

Class Summary

Class	Description
AccountImpl	Permette di accedere e modificare le informazioni del conto: descrizione, saldo iniziale, tipologia. Consente inoltre di ottenere il saldo attuale a runtime. Inoltre, è possibile accedere alla lista dei movimenti associati all'account e quelli che soddisfano un determinato predicato.
BudgetImpl	Un budget associa ad ogni tag un importo che indica l'ammontare di spesa/guadagno previsto per il particolare tag. Inoltre mostra i tag utilizzati nelle transazioni che rispettano una certa condizione (predicato) e può operare su questi.

Class Summary

Class	Description
LedgerImpl	Tramite il pattern Singletone è prevista una sola istanza di tale classe, questo al fine di concentrare i dati in un unico oggetto. Infatti raccoglie tramite liste tutte le istanze delle Transazioni, degli Account, dei Tag e le ScheduledTransaction presenti nell'applicazione, con allegata gestione di esse. L'oggetto di questa classe fa' da tramite per aggiungere transazioni ad una Scheduled Transaction, nel caso fosse corrispondente alla data desiderata. Tra le altre operazioni, permette inoltre di ricavare le transazioni che rispettano un predicato dato.
MovementImpl	Permette di accedere e modificare le informazioni associate al movimento: descrizione, importo, account associato, lista dei tag associati al movimento. Il movimento è associato ad una transazione da cui ne deriva la data. I tag inseriti nei movimenti vengono raccolti senza ripetersi nella transazione, mentre i tag aggiunti alla transazione vengono distribuiti a tutti i movimenti a lei associati.
ScheduledTransactionImpl	Indica una transazione o una serie di transazioni schedulate (previste) ad una certa data. La serie di transazioni termina quando il metodo isCompleted() restituisce TRUE.
TagBudgetReportImpl	Ha la responsabilità di mostrare il saldo di positivo/negativo di uno o più Tag, sia la totalità di essi che quelli utilizzati solo in determinati predicati
TagImpl	Ha la responsabilità di definire una categoria di spesa/guadagno.
TransactionImpl	Permette di accedere e modificare la informazioni associate ad una transazione: lista dei tag, data, movimenti associati ad essa. Un tag aggiunto (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 il saldo totale di essa ricavato da tutti i movimenti interni alla transazione.

Enum Summary

Enum	Description
AccountType	Ha la responsabilità di definire una categoria di conto.
MovementType	La tipologia di movimento determina l'effetto di un movimento su un conto

Package it.unicam.cs.pa.jbudget100763.controller

Class Summary	
Class	**Description**
Controller	Ha la responsabilità di ricevere i comandi dell'utente e di attuarli modificando lo stato degli altri due componenti del MVC

Package it.unicam.cs.pa.jbudget100763.view

Interface Summary	
Interface	**Description**
View	Ha la responsabilità di indicare le direttive principali riguardo l'interazione dell'utente con l'applicazione

Package it.unicam.cs.pa.jbudget100763.view.javafx

Class Summary	
Class	**Description**
App	Applicazione principale dell'implementazione tramite javaFX.
FxController	Ha la responsabilità di caricare le schermate da avviare e raccogliere tutte le interazioni dell'utente tramite la javaFX GUI e di inoltrarle al controller dell'applicazione principale.

Architettura

Il progetto Gradle creato ha rispettato fin'ora i seguenti principi:

1. Singola responsabilità delle classi verificabile tramite il loro comportamento
2. Il comportamento di un elemento può essere ampliato ed esteso, senza modificare il suo codice.
3. Gli oggetti possono essere sostituiti con dei loro sottotipi senza alterare il comportamento del programma.
4. Sono presenti adeguate interfacce, specifiche per ogni elemento
5. Le classi dipendono dalle astrazioni e non da altre classi concrete

L'architettura delle classi si basa sul Design Pattern MVC (Model View Controller), ed il progetto è suddiviso in package a seconda delle funzionalità di ogni componente di tale design.

Il package **it.unicam.cs.pa.jbudget100763.model**, ha il compito di gestire i dati dell'applicazione, fornisce i metodi per accedere ad essi.

Il package **it.unicam.cs.pa.jbudget100763.view**, stabilisce i comportamenti ritenuti fondamentali per visualizzare i dati contenuti nel model ed occuparsi dell'interazione con gli utenti.

Il package **it.unicam.cs.pa.jbudget100763.controller** include la classe che ha il compito di ricevere i comandi della vista e quindi dell'utente, infine di attuarli modificando lo stato degli altri due componenti del MVC.

Il package **it.unicam.cs.pa.jbudget100763.view.javafx** include il percorso di implementazione della GUI tramite il framework javafx, seguendo le direttive del View. I file FXML sono stati separati dal controller per una questione di pulizia della directory e locati nella cartella Resources.

Il package principale contiene la classe **Main** del progetto, eseguibile.

Note dello sviluppatore

Al termine di questa iterazione si considerano raggiunti i seguenti obiettivi:

- Completo funzionamento del core business del progetto, si intende la gestione degli elementi ritenuti fondamentali e ricavati dall'analisi dei requisiti (gestione degli elementi del model tra cui la Account, Transazioni, Movimenti, possibilità di esaminare statistiche dei movimenti in base a categorie e date, schedulazione di transazioni).
- Raggiuntele aspettative riguardanti sia le funzionalità che le prestazioni medie del sistema.
- Inizio dello sviluppo dell'interfaccia grafica basata su JavaFX: possibilità per l'utente di creare e rimuovere account e tag (disclaimer: una volta creati gli elementi per il momento è necessario aggiornare la loro tabella, riaprendola, problematica individuata e legata all'uso di ObservableList, nella rimozione dell'oggetto il problema non sussiste).
- Test jUnit delle funzionalità base (gestione Account, gestione di Transazioni e Movimenti).
- Controllo sulla forma dei valori in input (eccezioni) gestiti direttamente tramite interfaccia View.

Prossimi obiettivi:

- Sviluppare la permanenza dei dati, in caricamento e salvataggio, per garantire la sincronizzazione tra dispositivi.
- Ampliamento della GUI.
- Implementazione di test specifici per le funzionalità più avanzate del progetto per ridurre il grado di rischio.