New Generation Accounting

Relazione per “Programmazione ad Oggetti”

Federico Prati, Diego Lai, Nicole Liliana Bassi

31 maggio 2016

Indice

Capitolo 1

Analisi

Il software, sviluppato da NGA Team, è stato ideato per venire incontro alle necessità primarie delle aziende che si presentano sul mercato, dandogli una moderna e veloce soluzione alle più comuni esigenze che esse affrontano quotidianamente.

1. 1 Requisiti concordati

- New Generation Accounting dovrà occuparsi di gestire una o più aziende indipendenti, munendo ognuna di essa con una password personalizzata che assicuri privacy e sicurezza.

- Tra i punti importanti posti dal NGA Team come requisito si trova quello di rendere il software più user-friendly possibile.

- NGA permette la creazione di un proprio catalogo prodotti, di una propria lista clienti e l'evasione di una fattura a quest'ultimi, comportando movimenti economici, movimenti in debiti e crediti, tutto consultabile alla fine in una situazione economica aggiornata.

1.2 Analisi e modello del dominio

NGA dovrà essere in grado di far gestire a uno o più utenti la propria azienda.

L'azienda disporrà di un proprio bilancio, composto da un numero variabile di conti, un'anagrafica conti e un'anagrafica movimenti, per monitorare i vari conti registrati. L'utente avrà inoltre la possibilità:

- di creare fatture
- di gestire le scorte di magazzino
- di gestire un elenco di clienti e fornitori
- registrare nuovi conti e nuovi movimenti tramite partita doppia
- accedere a più aziende
- di disporre di uno Stato Patrimoniale ed un Conto Economico aggiornato
- di avere un'analisi della situazione aziendale calcolata in base a indici e margini
- gestire un'anagrafica di crediti e debiti

Capitolo 2

Design

2.1 Architettura

Il pattern utilizzato per creare l'applicazione è l'MVC (model-view-controller), l'utente interagisce con la view tramite schermate popup, il controler prenderà gli input dalla View e chiamerà le funzioni del model. L'applicazione è stata divisa in sottoparti logiche: dataModel,dataEnum,Controller,Model,View (controller e view sono costituiti da + sottoparti). Maggiori dettagli sulle scelte fatte:

- dataModel contiene le classi e le interfacce utilizzate per la creazione degli oggetti utili al programma
- dataEnum gestisce invece le enumerazioni utilizzate durante la programmazione
- Controller e controller.* gestiscono le varie parti del controller
- View e view* gestiscono le varie parti della view
- Model contiene le classi e le interfacce per la modellazione dei dati

Per View e Controller, avendo molte classi da gestire, si attua una divisione in più package per un discorso di ordine all'interno del programma, questo rende più semplice gestire e cercare le classi durante la programmazione.

2.1 Design dettagliato

2.2.1 Model ()

Il model modella i dati in base alle richieste del controller. E' indipendente dalla view ma collegato al controller. In questo progetto il model è diviso in due parti connesse tra loro solo tramite una Interfaccia.

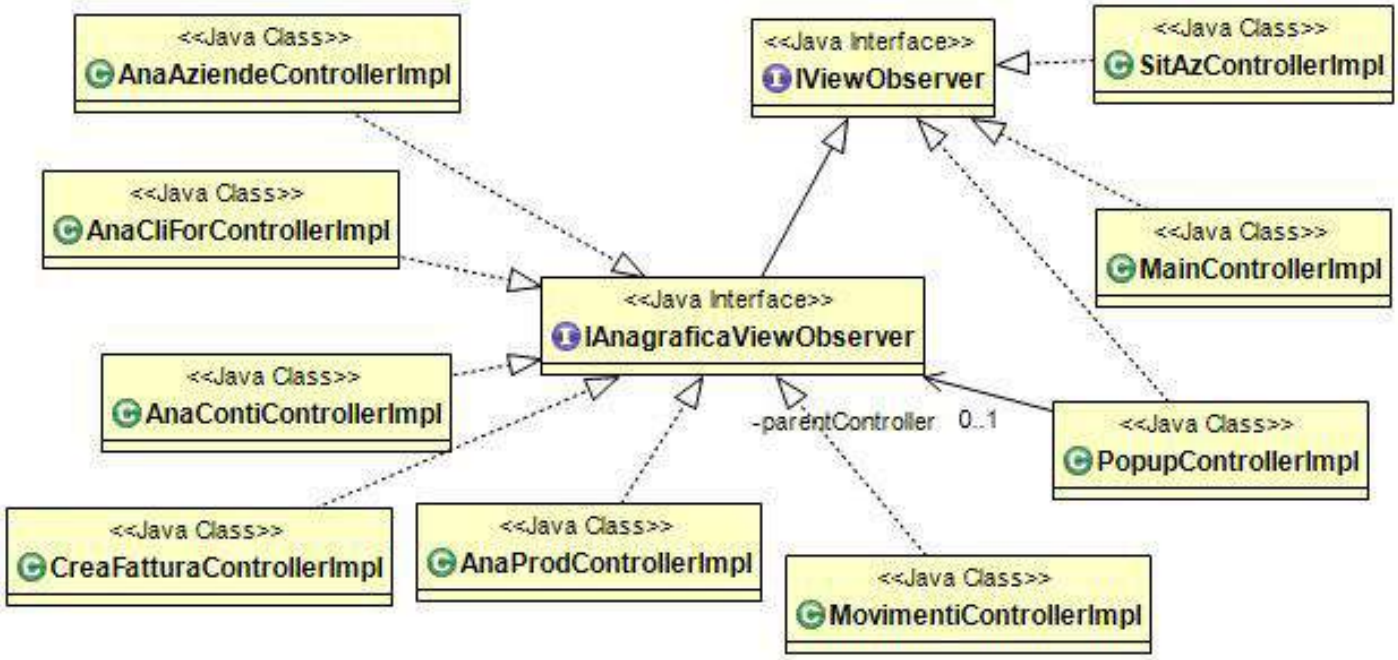
Una parte del Model si occupa di:

- creare e gestire l'interfaccia che collega la classi di anagrafica
- implementare la classe inerente ai movimenti
- implementare l'interfaccia e la classe inerenti alla situazione finanziaria e relativo commento;
- implementare la classe inerente ai conti di mastro

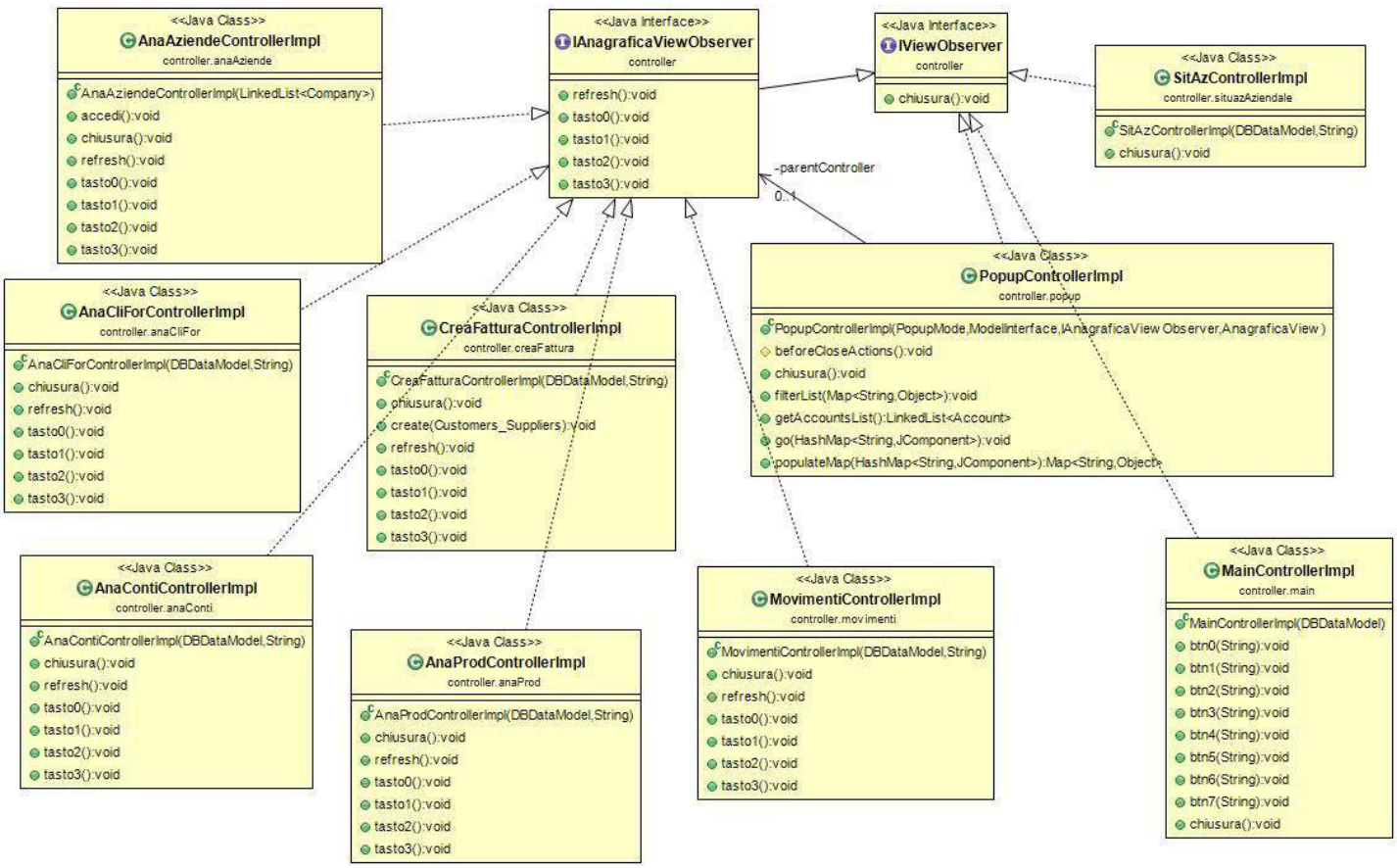
e l'altra si occupa di:

- implementare la gestione delle scorte di magazzino
- gestire l'elenco di clienti e fornitori
- gestire l'elenco di crediti e debiti
- implementare la classe per la creazione di fatture
- implementare le funzioni inerenti alla gestione di più azienda

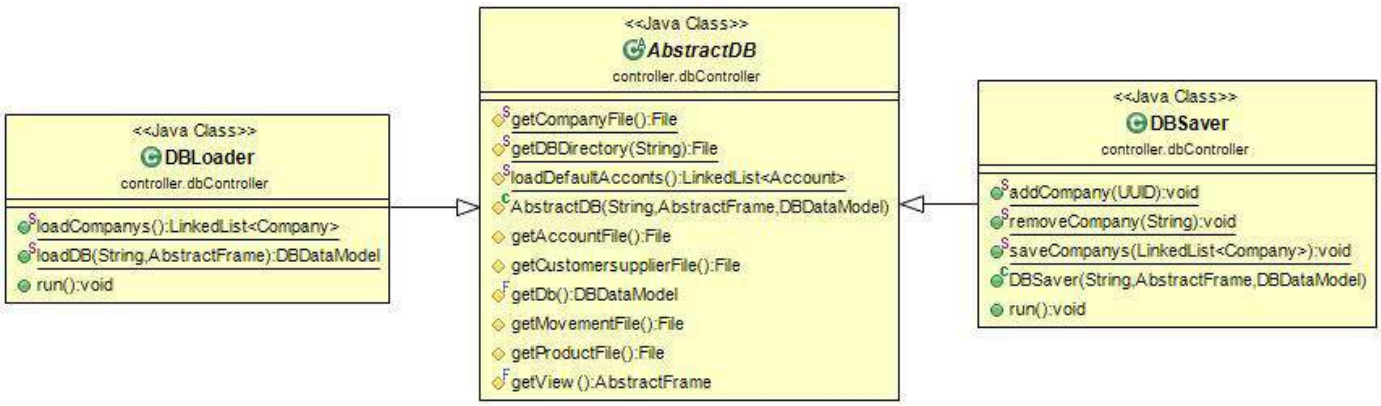
2.2.2 Controller



Il controller di NGA esprime il concetto di controller del modello MVC al meglio. Non esistendo alcun tipo di collegamento tra DB, View e Model il controller ha l'arduo compito di coordinare e gestire questi tre macro componenti. Federico (Pentolo) Prati si è occupato di questa parte, decidendo l'utilizzo di un pattern OBSERVER. Inizialmente implementato in maniera "pura" è stato poi adattato al meglio alle esigenze del programma adattandosi alle necessità del programma.

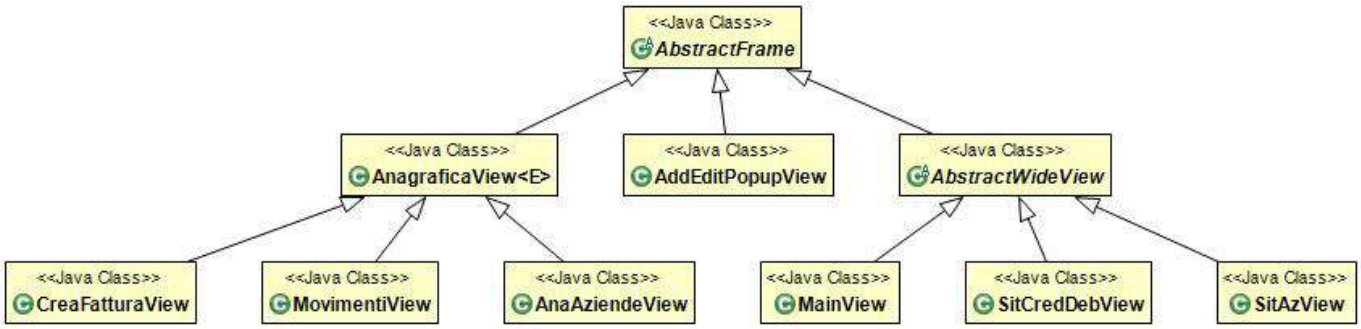


Come si può vedere dal modello UML il cuore del controller è IViewObserver, un'interfaccia implementata da ogni classe facente parte del controller. IAnagraficaViewObserver è la sua diretta figlia, con lo scopo di gestire tutte le schermate che rispettano il layout "anagrafica" ossia con una tabella centrale e 5 tasti nella parte inferiore.
Le classi del controller ricalcano fedelmente le rispettive view, per riuscire a sfruttare al massimo l'ereditarietà offerta dal linguaggio di programmazione Java.



La gestione del DataBase è uno dei compiti principali del controller specialmente per un'applicazione di tipo gestionale, dove la memorizzazione dei dati è un punto estremamente importante e critico. Con un Pattern SINGLETON per gestire l'istanza del DB il controller si fa in tre! AbstractDB è il magazzino delle costanti e dei metodi che accomunano lettura e scrittura, mentre DBLoader e DBSaver si occupano rispettivamente dell'Input e dell'Output.
Il pattern Singleton essendo di natura estremamente flessibile e con decine di varianti è stata una scelta perfetta, una sola istanza del DB "gira" per l'intero programma, viene ovviamente restituita una copia difensiva delle varie liste tuttavia l'istanza da cui queste vengono clonate e alla quale poi queste torneranno dopo le varie aggiunte, modifiche e rimozioni è una.

2.2.3 View



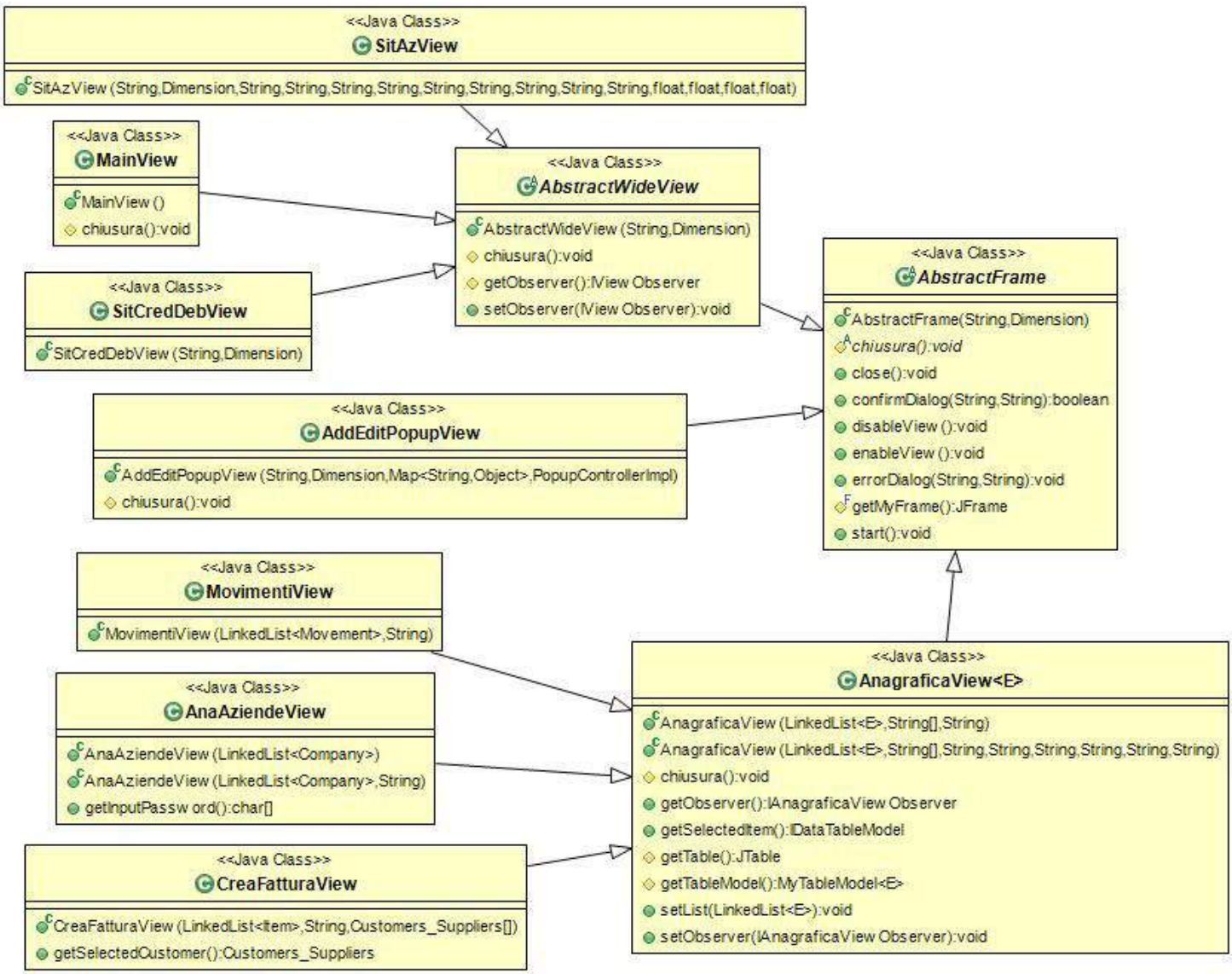
Come una piramide la gestione della View ha nella punta il suo punto di forza, AbstractFrame, una classe astratta che implementa già dentro di sè tutti i principali aspetti delle view di NGA.
Nessuna view è indipendente da AbstractFrame.
Da questa importantissima classe astratta come chiaramente visibile dal modello UML si distaccano tre "rami":

- AnagraficaView: classe generica implementata che gestisce tramite tipo generico tutte le view che rispettano il layout "anagrafica". Che da sola consente a sè e alle classi che la estendono di avere una JTable creata ad-hoc per contenere e gestire oggetti e 5 tasti nella parte inferiore;
- AbstractWideView: classe astratta per le view di tipo "wide" ossia view con un pannello centrale e un tasto per la chiusura nella parte inferiore;
- AddEditPopupView: la classe più complessa del programma, io, Pentolo, ho provato a impedire la nascita di una God-Class tuttavia non mi è stato possibile, poichè questa classe ha il difficilissimo compito di adempiere a tutti i compiti di aggiunta, modifica e filtro per TUTTE le varie anagrafiche, quindi 6 anagrafiche e 3 modalità: 18 utilizzi separati per la stessa classe. Non vado molto fiero di come ho gestito questo componente, tuttavia è estremamente flessibile, adattabile e in caso di future espansioni è già pronto a gestire, senza modifiche, una grossa fetta di casistiche possibili.

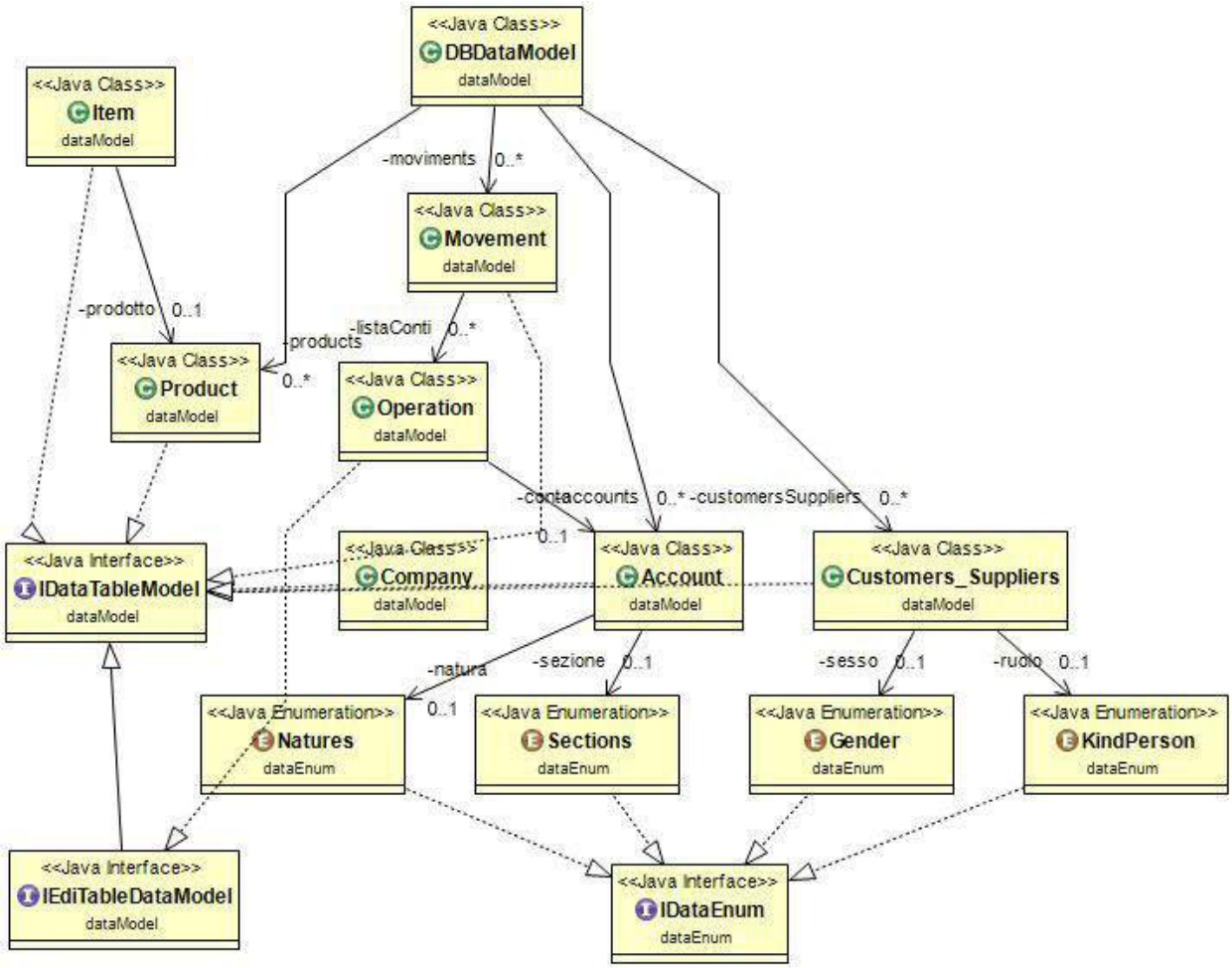
Per quanto riguarda il layout “anagrafica”, per dare continuità nell’uso all’utente e mantenere un’esperienza di uso semplice e intuitiva le seguenti classi seguono il medesimo layout grafico:

- Anagrafica aziende (estende)
- Anagrafica clienti e fornitori
- Anagrafica conti
- Anagrafica prodotti
- Movimenti (estende)
- Crea fattura (estende)

Solamente metà di queste sono classi effettivamente implementate, per necessità speciali:
Campo password e tasto accedi per l’anagrafica aziende, celle su più righe per movimenti e ComboBox per selezione del cliente e tasto “crea fattura” per crea fattura.

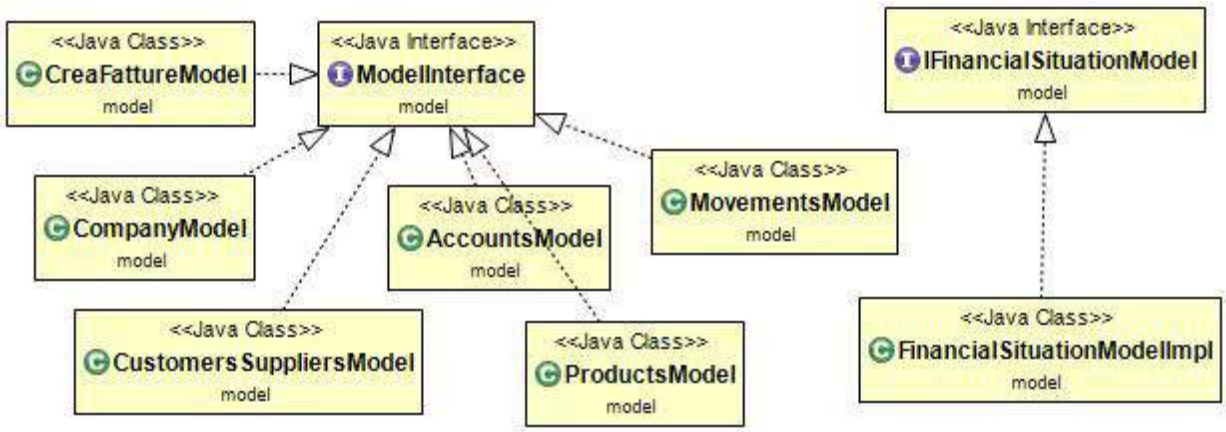


2.2.1 dataModel e dataEnum



In questo schema UML vediamo come sono tra di loro collegate le sottoparti dataModel e dataEnum, tutto gira intorno alle interfacce IDataEnum, IEditTableDataModel e IDataTabelModel. Le classi più piccole come Account e Product utilizzano le enumerazioni e implementano le varie interfacce. Le enumerazioni cosi come le classi base vengono utilizzate anche nelle varie classi di Model Controller e View

2.2.1 Model - Bassi Nicole Liliana + Diego Lai



(Bassi Nicole Liliana)

In questo schema UML viene gestita la parte di partita doppia del programma. Si divide in due parti fondamentali, guidate dalle interfacce IFinancialSituationModel e ModelInterface. La prima delle due dichiara i metodi utilizzati dalla classe figlia per esporre la situazione finanziaria dell'azienda. La classe figlia, FinancialSituationModelImpl, utilizza oggetto gestite da dataModel ed enumerazioni gestite da dataEnum. E' collegata alla seconda parte di questo schema perché entrambe utilizzano un oggetto gestito dalla classe DBDataModel, nel prospetto precedente. ModelInterface è l'interfaccia che dichiara tutti i metodi che saranno implementati dalle classi figlie. Per quanto riguarda le classi di partita doppia, MovementsModel e AccountsModel, entrambe implementeranno tutti i metodi di ModelInterface, e ognuna avrà poi dei suoi metodi privati. Movements utilizzerà oggetti di Movement ,Operations, Accounts, Sections, Natures e DBDataModel. AccountModel userà oggetti di Account, Sections,Natures e DBDataModel

(Diego Lai)

La figura riguardante il Model descrive tutte le classi di cui l'azienda necessita ed elabora i dati della stessa.

Permette quindi di: gestire uno o più utenti, i quali tramite la loro password personalizzata, ottengono il diritto all'utilizzo delle varie classi a loro dedicate; permette la creazione,la modifica e l'eliminazione di una o più aziende; permette la gestione delle lista clienti; permette la gestione di un proprio listino prodotti, il quale può essere uniformato alle esigenze di ogni singola azienda; infine permette l'emissione di una fattura, permettendo l'acquisto di un prodotto dal listino ed addebitando il subtotale direttamente nel conto debiti.

3 Sviluppo

3.1 Divisione dei compiti

Federico(Pentolo) Prati si è occupato della View, gestendo finestre popup per l'interazione tra utente e applicazione. Un altro compito a lui assegnato è quello del Controller, gestendo quindi il collegamento tra utente, view e model, attraverso classi e interfacce che, presi gli input dalla view, chiameranno i vari metodi del model

Bassi Nicole Liliana si è occupata della gestione della partita doppia, gestendo in particolare aggiunta, rimozione, modifica di conti e movimenti,con relativi aggiornamenti. Inoltre si è occupata di calcolare margini, indici, Stato Patrimoniale e Conto Economico per la redazione di un'analisi finanziaria basata sui dati precedentemente citati.

Diego Lai si è occupato della gestione clienti e fornitori, crediti, debiti e scorte in magazzino. Ha inoltre implementato funzioni che creassero fatture ed altre per la gestione della possibilità di avere più aziende che utilizzano questa applicazione

Le tre parti sono state in seguito integrate.

3.2 Difficoltà incontrate durante la programmazione

(Bassi Nicole Liliana) in partenza non avevo ben capito come utilizzare il DVCS, e ho creato qualche danno, ma poi con il passare del tempo sono riuscita a non creare altri pasticci. Altre difficoltà le ho trovate nella programmazione vera e propria, non mi sono mai trovata a lavorare in gruppo e non ho troppe basi di programmazione, ma con l'aiuto di Federico sono riuscita a risolvere qualche dubbio e a procedere.

(Federico Prati) La difficoltà è stata la grande mole di lavoro, in sè semplice ma forse troppo corposa. Ho passato un 25% del mio tempo nell'aiutare i miei compagni di

progetto. Le uniche due classi complesse sono state la JTable modificabile e la AddEditPopupView.

Appendice A

Guida per l'utente

All'avvio del programma la prima finestra che apparirà è l'Anagrafica Aziende, tutto quello che l'utente dovrà fare è cliccare sul pulsante AGGIUNGI e compilare i vari campi, una volta fatto questo si seleziona l'azienda con la quale si vuole operare e, una volta digitata la password corretta, siamo pronti per cominciare. Effettuato l'accesso apparirà un menù che porteranno l'utente, con un click, alla sezione dell'applicazione desiderata. Le strade saranno:

- "Crea fattura"
- "Disconnetti/Connetti Azienda" per tornare all'anagrafica delle aziende
- "Anagrafica Clienti e Fornitori" per avere la lista di Clienti e Fornitori dell'azienda
- "Gestione Conti" per visualizzare l'anagrafica dei conti registrati fino a quel momento
- "Visualizza Movimenti" per visualizzare in ordine di data i vari movimenti, visti in partita doppia
- "Anagrafica Prod e Scorte di Magazzino" per visualizzare l'anagrafica dei prodotti che l'azienda gestisce
- "Dettaglio Crediti e Debiti" per avere la visuale sui crediti e i debiti dell'azienda
- "Visualizza Situazione Aziendale" per avere Stato Patrimoniale, Conto Economi e analisi per margini e indici dell'azienda
- "Chiudi" che chiude l'applicazione

In ogni finestra di anagrafica oltre a visualizzare i dati memorizzati è possibile applicare delle modifiche a questi ultimi, in basso si troveranno 5 pulsanti per aggiungere, cercare, modificare ed eliminare un dato. Il quinto bottone di porta alla schermata del menù principale.

Autovalutazione:

Tutto il team ha fatto del suo meglio per portare a termine il progetto, nonostante alcune features, per mancanza di tempo, non sono ancora state implementate/perfezionate, la valutazione che ci auto assegneremmo personalmente è la seguente:

- Prati Federico: Considerando l'ingente mole di lavoro e il tempo che ho dedicato alla progettazione e all'aiuto dei miei compagni, mi valuto poco più del voto che già ho ottenuto alla prova pratica (28). Sinceramente autovaluto il mio apporto al progetto 30/30.
- Nicole Bassi: 23/30.
- Diego Lai: 20/30.