Relazione per "Libro"
Corso di Programmazione ad Oggetti

Chiara Ceccarini
Alberto Mulazzani

27 Febbraio 2015

Indice

Capitolo 1

Analisi

1.1 Requisiti

Il programma, LIBRO, deve permettere la gestione di una libreria con annesso magazzino, che dovrà quindi gestire sia i libri in negozio sia gli ordini. Il software permetterà al negoziante di monitorare in tempo reale l'attività del suo negozio, con una sezione relativa al suo fatturato, per avere un resoconto finanziario sull'andamento della propria attività, e una sezione relativa alle statistiche, con informazioni sui libri più venduti e gli autori più attivi. Il software permette, infine, la possibilità di gestire carte soci con accumulo di punti in base alla spesa del cliente, punti con cui si potranno ottenere promozioni o sconti proposti dal titolare della libreria.

1.2 Problema

LIBRO dovrà gestire una libreria in cui è possibile aggiungere, rimuovere e modificare i dati dei libri presenti e fare le stesse operazioni per quanto riguarda gli ordini, problematiche di bassa difficoltà. Inoltre, il software, utilizzando i dati delle vendite, dovrà fornire al momento dell'analisi la situazione finanziaria del negozio e le statistiche relative ai libri più o meno venduti e agli autori che hanno scritto più o meno libri. E' data inoltre la possibilità tramite una carta soci di accumulare punti (un punto ogni 10 euro di spesa) per poter ottenere sconti o aderire a promozioni scelte dal titolare del negozio. La problematica più grossa è stata salvare e caricare i dati riguardanti i libri presenti in negozio e quelli in ordine e i dati di ciascuna carta soci da file, in modo da dare all'utente l'opportunità di gestire più librerie e di non perdere i dati di ciascuna di esse.

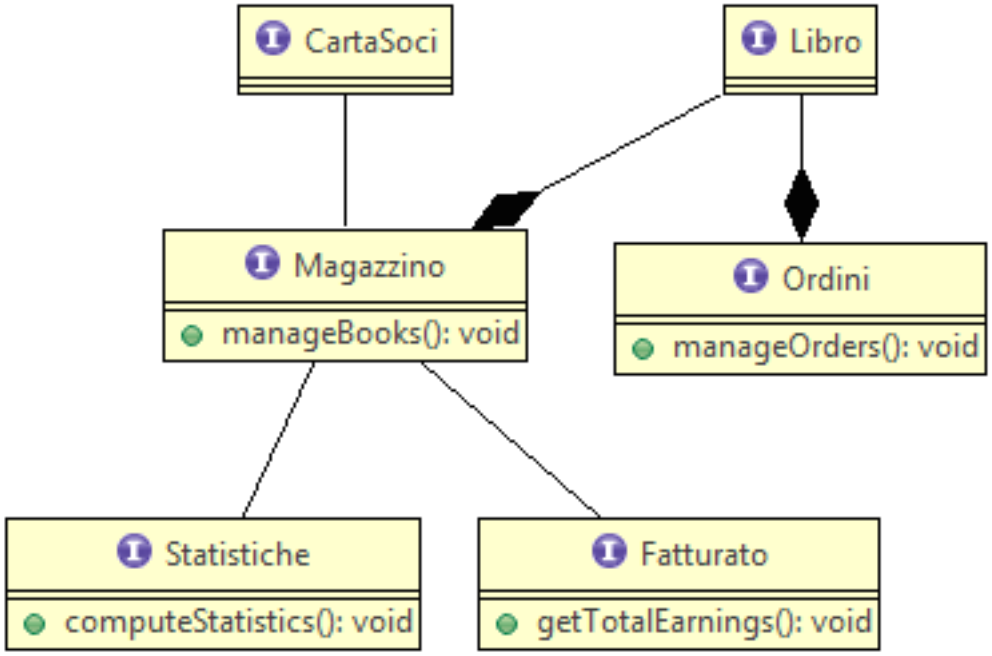

Figura 1.1: Schema UML dell'analisi del problema: L'applicazione deve gestire il Magazzino e gli Ordini, entrambi formati da Libri. Il magazzino, gestendo anche le vendite, sarà collegato alle Carte Soci, così da poter aggiungere punti sulla carta dello User che ha acquistato il Libro. Utilizzando i dati degli acquisti e delle vendite, verranno creati i dati mostrati dalle sezioni Statistiche e Fatturato.

Capitolo 2

Design

2.1 Architettura

Il pattern model-view-controller è stato applicato creando una serie di view che mostrano e prendono dati in input dall'utente. Questi dati vengono poi passati al relativo controller che li elabora e li dirige sui giusti metodi del model. Il controller quindi fa da tramite nella comunicazione tra view e model e in quella tra model e view. Dato che si è cercato di staccare quanto più possibile il concetto della view da quello del controller, sarà possibile utilizzare lo stesso controller per view testuale, grafica o di altro tipo.

2.2 Design dettagliato

Nella parte del Controller si è utilizzato il pattern SINGLETON per meglio gestire i controller, poiché si necessitava di istanziarli un'unica volta all'apertura dell'applicazione. Nella View è stato utilizzato il pattern strutturale Facade per collegare a JOptionPane dei JPanel più complessi e completi così da facilitare l'utente nell'utilizzo e in modo da evitare di appesantire l'applicazione caricando solamente le view necessarie al momento e chiudendole nel momento in cui non siano più necessarie.

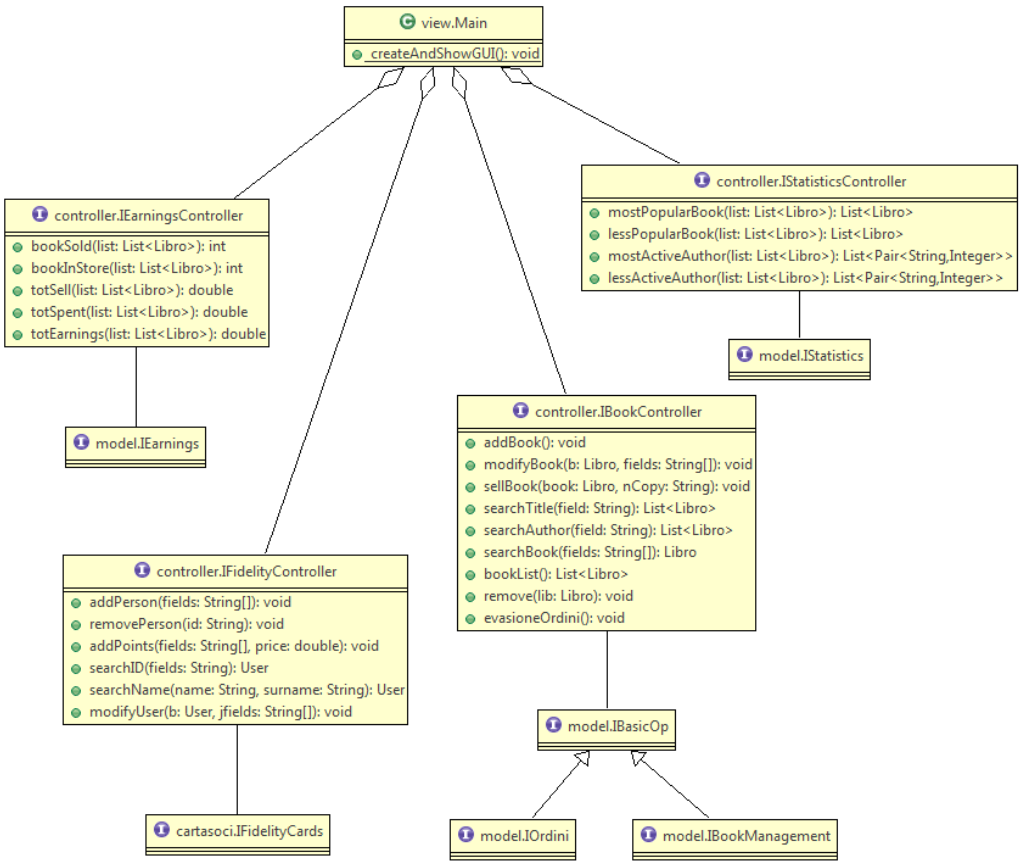

Figura 2.1: Schema UML architetturale. Ad una View vengono collegati i quattro Controller a cui sono assegnate le operazioni descritte nel problema. I quattro controller sono collegati a model diversi. Il controller per le operazioni di Magazzino e Ordini è collegato ad un model composto da tre parti che si dividono le operazioni richieste dal controller. In questo modo, non intercorrono rapporti diretti fra View e Model e i diversi Model hanno come unico punto di accesso il relativo Controller.

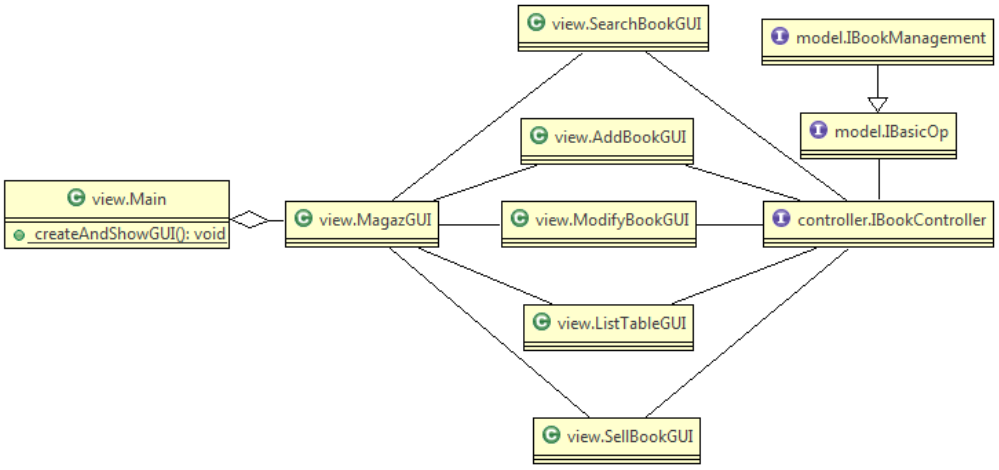

Figura 2.2: UML dettaglio Facade. La figura rappresenta il dettaglio della gestione del Magazzino, analoga alle altre. Una View principale Main ha al suo interno una MagazGUI che, in seguito ad azioni intraprese dall'utente, istanzia dei JOptionPane creati in maniera tale da permettere una complessità maggiore rispetto ad una semplice JDialog. Le azioni intraprese all'interno di queste "SottoGUI" vanno ad agire sul controller nelle maniere descritte e di conseguenza sul Model.

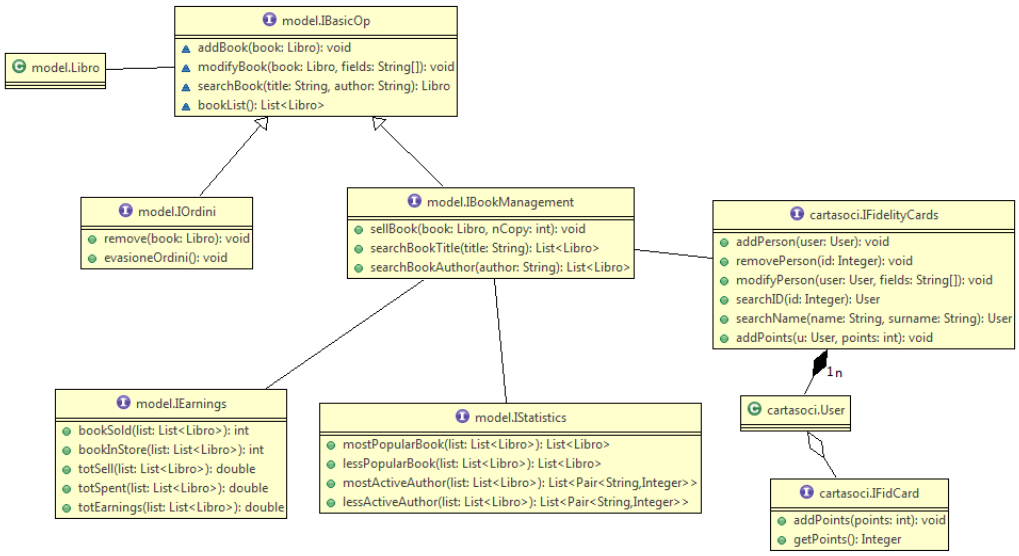

Figura 2.3: Schema UML di dettaglio del model di una applicazione. L'interfaccia IBasicOp è estesa dalle due interfacce IOrdini e IBookManagement, che aggiungono a quest'ultima le funzionalità più specifiche relative ad Ordini e a Magazzino. Il modello lavora con un oggetto di tipo Libro, che è l'unità base di tutta l'applicazione. La parte di Magazzino dell'applicazione è collegata alle altre tre sezioni, quali IEarnings, IFidelityCards e IStatistics, che gestiscono in ordine la sezione Economia, Carte Soci e Statistiche. L'interfaccia IFidelityCards è modellata su un oggetto di tipo User, che mantiene al suo interno un secondo oggetto di tipo FidCard, concetto della Carta associata ad un Utente.

Capitolo 3

Sviluppo

3.1 Testing automatizzato

Sono stati utilizzati test di tipo JUnit per la parte di model, in particolare si è provato il corretto funzionamento delle funzioni di gestione del magazzino e degli ordini, con relative statistiche e fatturato. Lo stesso tipo di test è stato utilizzato per controllare la parte relativa alla carta soci, verificandone il corretto funzionamento. Un test manuale, invece, è stato eseguito per verificare la correttezza delle funzioni di caricamento e salvataggio di dati e il corretto funzionamento dell'interfaccia grafica su sistemi Windows 7 64bit, Windows 8.1 64 bit e Mac OS X Yosemite.

3.2 Divisione dei compiti e metodologia di lavoro

Ceccarini:

- model con relativi test
- model e controller delle statistiche
- model e controller del fatturato

Mulazzani:

- view
- caricamento e salvataggio dati da/su file
- controller della carta soci e del magazzino

Dopo aver deciso insieme su carta le interfacce da usare e i possibili metodi da implementare, ci siamo divisi il lavoro cercando di dividerlo equamente e poi si è lavorato singolarmente. Abbiamo interagito per le parti comuni e per la fase di debugging, in modo da migliorare il più possibile il codice.

Si è utilizzato giornalmente il tool Mercurial da linea di comando e il repository Bitbucket che ha facilitato molto l'interazione fra le parti e la condivisione del codice.
Il Software è stato sviluppato contemporaneamente su OS Windows e Unix (MacOSX), garantendo una compatibilità completa con tutte le piattaforme. Richiede però l'installazione di Java 1.8 sul dispositivo utilizzato.

3.3 Note di sviluppo

Per il salvataggio dei dati relativi ai libri giacenti in magazzino, in ordine e alla carta soci abbiamo utilizzato la libreria XStream per salvarli in formato .xml (http://xstream.codehaus.org).
Per la parte relativa alle statistiche abbiamo utilizzata la classe Pair, fornita nella soluzione di un esercizio fatto in laboratorio, in modo da tener conto dell'autore e dei suoi libri.
Infine, per inserire un background all'interno dei panel abbiamo usato la classe BackgroundPanel creata da Rob Camick
(https://tips4java.wordpress.com/2008/10/12/background-panel/).

Capitolo 4

Commenti finali

4.1 Conclusioni e lavori futuri

Ci sentiamo abbastanza soddisfatti del software, pensiamo che sia facilmente intuibile l'uso da parte di un possibile cliente e di immediato utilizzo per l'utente finale anche con poche conoscenze informatiche. L'applicazione si potrebbe migliorare nella parte di Aggiunta e Vendita dei libri tramite l'implementazione della lettura di codice ISBN e codice a barre direttamente dal libro, purtroppo impossibile al momento. Tale migliorie potrebbero rendere l'applicazione completamente e facilmente utilizzabile in un ambiente di lavoro professionale.

Appendice A

Guida Utente

Il programma si divide in 5 sezioni principali, accessibili tramite il menù di sinistra.
Il programma si apre sulla sezione di Magazzino.
Nelle due sezioni di Magazzino e Ordini è possibile inserire, rimuovere (o vendere), modificare, ricercare un libro e ottenere la lista dei libri o ordini presenti. Si richiede di inserire tutti i dati necessari per ogni operazione. Si ricorda che il codice ISBN è formato da 13 cifre.
Le sezioni Statistiche e Fatturato saranno vuote fino ad inserimento dei dati nella sezione di Magazzino. Il prezzo di un libro al negoziante è calcolato come il 76 % del valore nominale del libro.
La sezione Carta Soci gestisce le Carte Fedeltà, ogni carta necessita di tutti i dati richiesti. All'aggiunta di ogni carta viene assegnato un Identificativo da comunicare al Cliente (con anche un supporto fisico eventuale a discrezione del Negoziante).