Corso di Programmazione ad Oggetti

Bender – Gestionale per Bar

Giacomo Scaparrotti – A.A. 2014/2015

28/02/2015

Capitolo 1 – Analisi

Requisiti

Lo scopo di questo programma è di fornire uno strumento semplice e funzionale per la gestione delle ordinazioni di un bar o di un ristorante. Le principali funzionalità che devono essere fornite sono l'aggiunta e la rimozione di ordini, la possibilità di personalizzare il menù e la stampa del conto. Il numero di tavoli deve poter essere variabile anche ad applicazione già avviata, per poter far fronte a situazioni impreviste (ad esempio, la separazione di un tavolo in due tavoli più piccoli). Inoltre bisogna poter salvare lo stato del bar/ristorante per poi poterlo ricaricare in una nuova istanza dell'applicazione. Il salvataggio deve inoltre lavorare in modo automatico.
In generale, l'applicazione deve essere improntata alla semplicità d'uso, così da poter essere usata nel modo più rapido possibile, fattore cruciale nella frenesia a cui spesso sono sottoposti i camerieri.

Problemi

Il primo problema che si pone nel realizzare un'applicazione con i suddetti requisiti sta nell'organizzare i dati relativi alle ordinazioni nel modo più razionale possibile. Questi dati devono infatti essere accessibili singolarmente, e al contempo essere organizzati in modo organico. Si deve infatti poter conoscere la quantità di piatti di una certa pietanza che ancora devono essere serviti con la stessa facilità e rapidità con cui si calcola il conto totale di un tavolo.
Questi dati, una volta reperiti, devono poi essere presentati in modo chiaro e ben leggibile, potendo inoltre interagirvi in modo intuitivo, ad esempio nel segnare come servito un certo piatto.

Capitolo 2 – Design

Architettura

L'architettura dell'applicazione si basa sul pattern MVC. La separazione tra modello, controller e vista è molto forte, grazie alla ridotta presenza di relazioni tra le classi, ridotte al minimo indispensabile. Inoltre, tutte le classe interagiscono con le altre solo grazie alle rispettive interfacce. Queste caratteristiche dovrebbero garantire un'ottima manutenibilità futura del codice.

Vediamo ora più nel dettaglio come sono stati realizzati questi tre macro-componenti.

Model

Il modello è costituito da due elementi fondamentali, IRestaurant e IDish, che rappresentano rispettivamente l'intero elenco delle ordinazioni e un singolo piatto che può essere ordinato. È presente inoltre IMenu, che rappresenta il menu da cui poter scegliere i piatti, che è funzionale all'interazione con l'utente, e Variation, che è una variazione di un IDish già esistente. Purtroppo per questioni di tempo, come poi verrà spiegato nel capitolo dedicato, quest'ultima classe non è attualmente utilizzata nel programma.

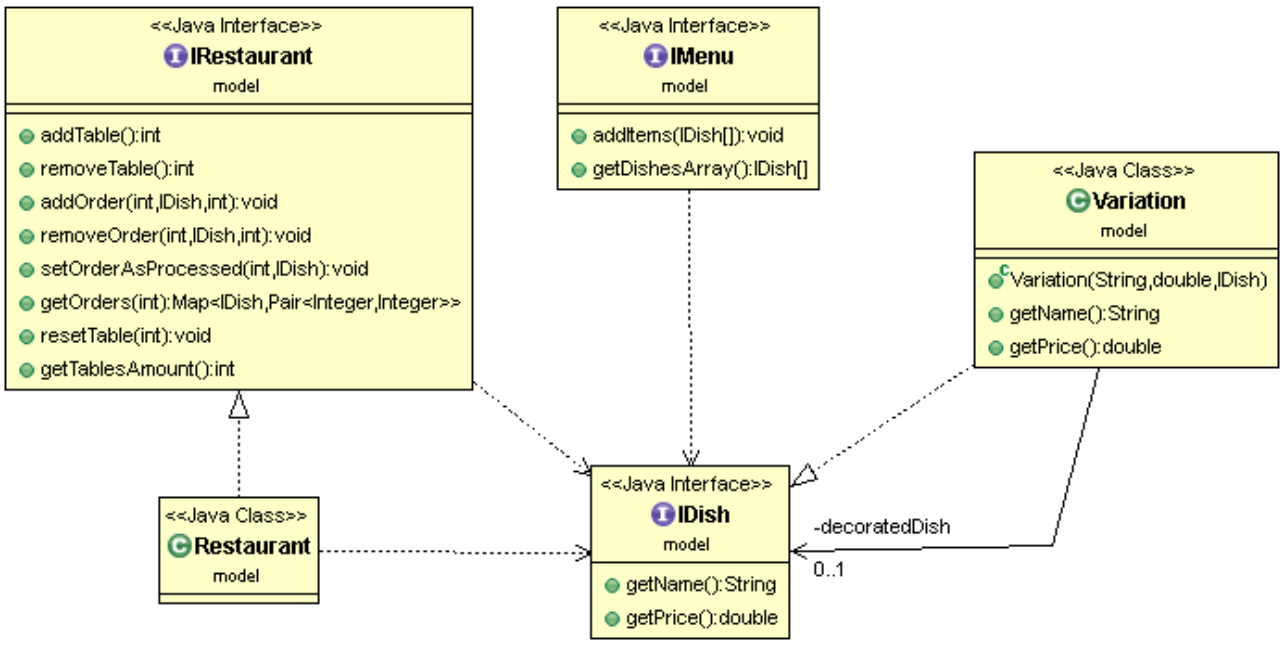

Figura 1 - Diagramma UML del modello di Bender. Si notano le relazioni tra IRestaurant e IDish

View e Controller

La “vista” dell’applicazione si compone di due elementi: una schermata principale, che mostra i tavoli e i comandi di basi, e una finestra di dialogo che permette di interagire con i singoli tavoli.

All’interno di Bender sono poi presenti tre controller: uno gestisce l’interfaccia principale, uno le finestre di dialogo che mostrano lo stato dei vari tavoli, e un terzo gli aspetti trasversali ai vari elementi del software, come il salvataggio e il caricamento da file.

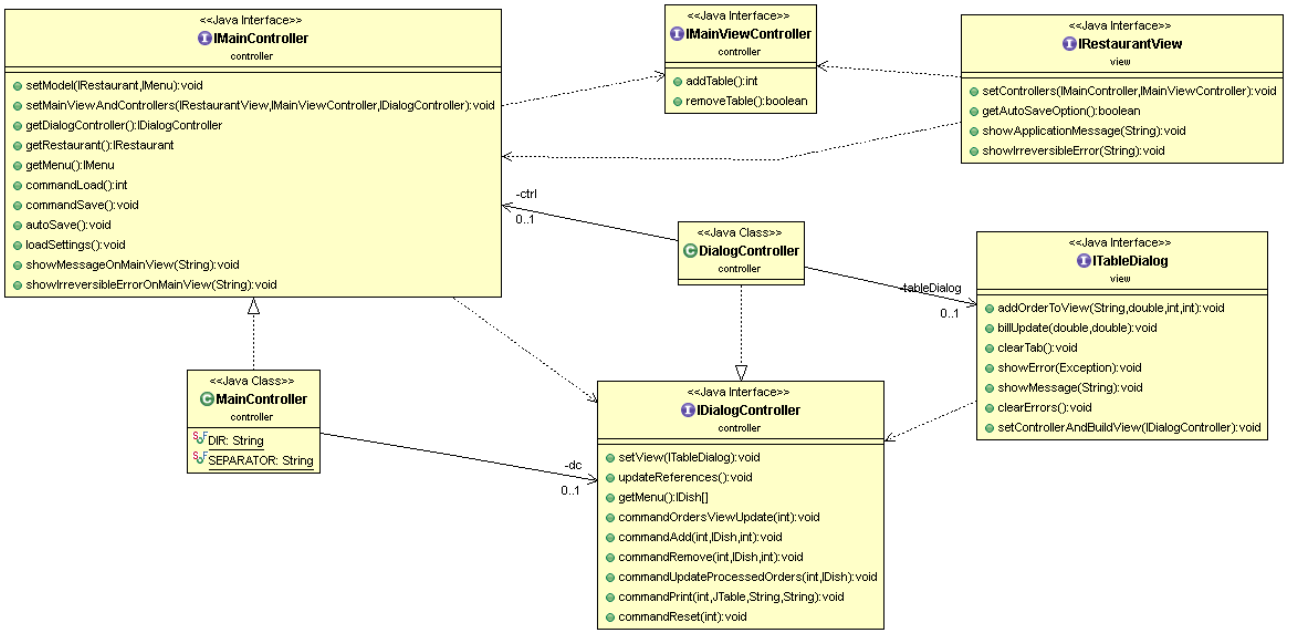

Figura 2 - Dettaglio sulla relazione tra i tre controller e i due elementi della view

Come si può vedere dal diagramma, i due elementi della view agiscono indipendentemente l’uno dall’altro, e interagiscono con il modello tramite i rispettivi controller. Il main controller agisce da “dispatcher”[1], nella misura in cui fornisce agli altri due controller gli elementi del modello su cui andare ad agire. Si può quindi vedere il controller come diviso su due livelli, uno rivolto più verso il modello (IMainController) e uno verso l’interfaccia utente (IMainViewController e IDialogController). Ciò rafforza la separazione tra modello e vista, e rende più flessibile la modifica del software. Questo aspetto è ulteriormente chiarificato dal diagramma in Figura 3.

[1] Dispatcher è inteso come un oggetto che fornisce altri oggetti a quelli che ne hanno necessità, ad esempio viene fornito l’oggetto contenente il menù al controller della finestra di dialogo dei tavoli, così da poterlo poi mostrare a video

Figura 3 - Relazione tra controller e modello. Il diagramma non comprende le classi concrete per rendere più leggera la rappresentazione

Design Dettagliato

Per quanto riguarda i dettagli progettuali, si possono individuare alcuni punti salienti.

Innanzitutto, è stato applicato in modo pervasivo il pattern MVC. Questo aspetto è già stato precedentemente discusso, quindi non verrà ripetuto.

Tra le altre caratteristiche, è da notare il modo in cui si determina quali classi concrete verranno utilizzate nell'esecuzione del programma: nel metodo "main" vengono richiesti infatti degli oggetti che aderiscano alle interfacce presenti nel programma, in particolar modo nel determinare i controller e il modello. Questo permette di cambiare in modo modulare i vari elementi del programma andando semplicemente ad agire sul solo metodo "main", favorendo la modifica anche radicale di intere porzioni di programma, senza dover minimamente intaccare ciò che invece è ritenuto già adeguato.
Si potrebbe definire questo *modus operandi* come una declinazione del pattern creazionale Builder, in quanto i vari oggetti che vengono creati all'avvio dell'applicazione vengono man mano popolati nei loro campi dall'invocazione di opportuni metodi, funzionali all'implementazione della costruzione modulare sopra citata.
Questo aspetto progettuale si lega inoltre alla funzione di "dispatcher" del main controller, in quanto in esso sono state definite diverse classi concrete, che verranno poi fornite man mano agli altri oggetti, proprio grazie ai suoi metodi chiamati nel "main".

Per quanto riguarda la veste grafica del programma, si può notare che solamente l'interfaccia principale è presente per tutto il ciclo di vita dell'applicazione, mentre le finestre di dialogo che mostrano i dettagli dei singoli tavoli esistono solo finché sono necessarie: esse sono infatti create da una Simple Factory presente nella classe concreta dell'interfaccia principale (RestaurantView).
Al contrario, il suo controller, così come gli altri due, esistono in una sola istanza per tutto il ciclo di vita del programma. Si potrebbero quindi considerare dei Singleton, anche se, per questioni

implementative, manca un vero controllo sull'effettiva presenza di una sola loro istanza per volta, quindi non si può affermare che queste classi aderiscano perfettamente a tale pattern.
Per quanto riguarda la parte di modello, è già stata individuata l'applicazione del pattern Decorator alla classe Variation, anche se questa, come già detto in precedenza, non è attualmente utilizzata nel programma, anche se una sua applicazione è una delle priorità nello sviluppo futuro di Bender.

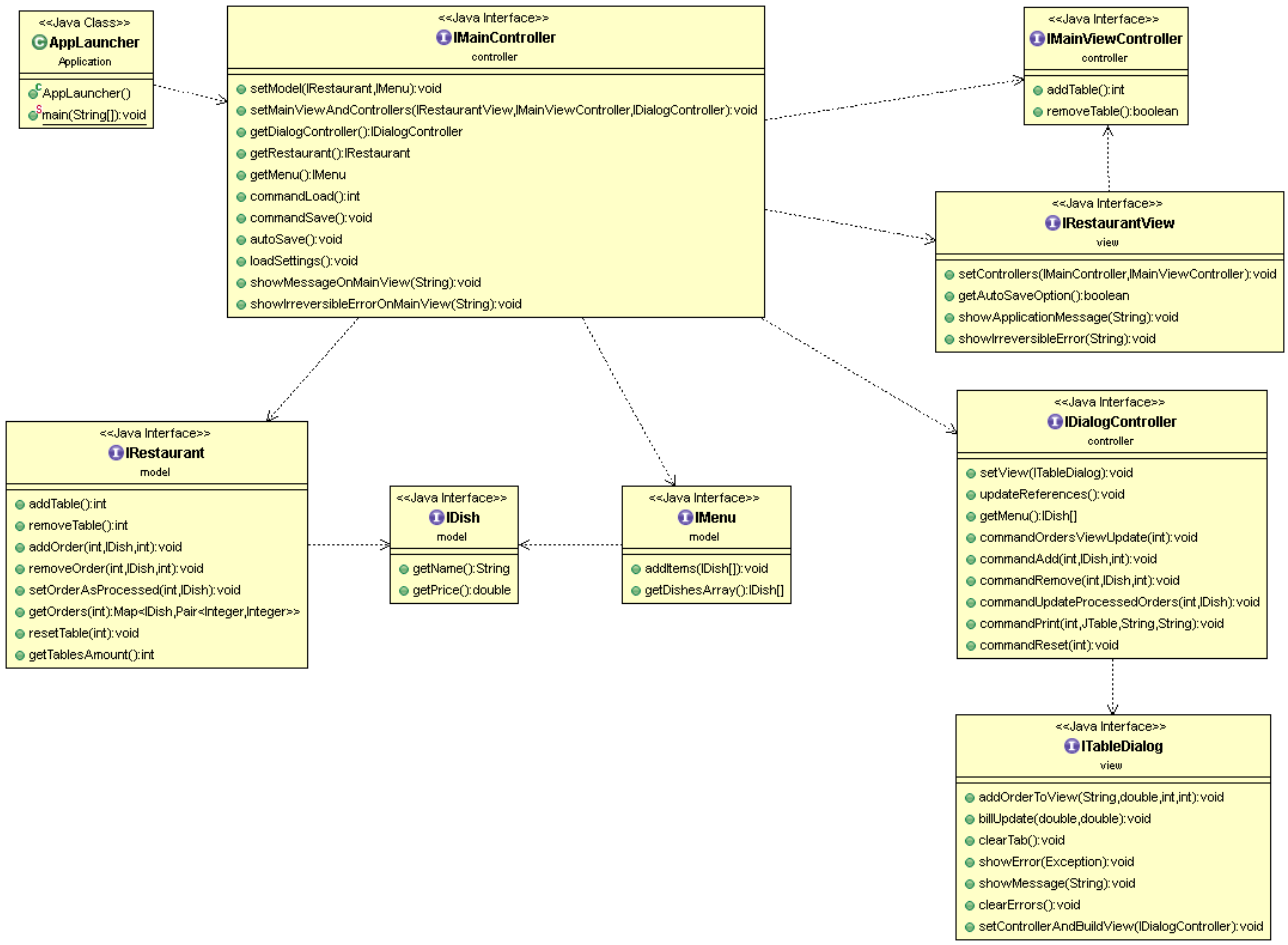

Figura 4 - Diagramma UML delle principali interfacce. Si nota il ruolo centralissimo di IMainController

Capitolo 3 – Sviluppo

Testing

La parte di modello di Bender è stata testata tramite JUnit. È stato verificato il corretto funzionamento delle varie funzioni del modello, come l'aggiunta e la rimozione di ordini e tavoli, e la consultazione degli ordini.
La parte grafica è stata invece testata manualmente, a causa dell'impossibilità di automatizzare un test del genere. Anche i controller della vista sono stati testati manualmente, perché il loro funzionamento presuppone la presenza di un qualche tipo di output (non necessariamente grafico).

L'applicazione è stata sviluppata su piattaforma Windows 8.1, ma, ad applicazione terminata, sono stati fatti dei test di utilizzo anche in ambiente Linux, nello specifico Linux Mint 17 Qiana, principalmente per verificare che l'aspetto grafico non subisse variazioni significative passando da un sistema operativo all'altro.

Sviluppo

L'applicazione è stata sviluppata individualmente.

Note di Sviluppo

La grande maggioranza del codice è stato scritto ex-novo, ma alcuni frammenti sono stati invece reperiti in rete, in particolare le tre classi che gestiscono la visualizzazione di un logo nella schermata principale [1].

Per la realizzazione della finestra di dialogo dei tavoli, si è partiti da una base realizzata grazie al tool Swing Designer, anche se poi il codice così generato è stato molto rimaneggiato, lasciando intatta solo la parte relativa all'uso dei Layout Manager.

La classe Pair, utilizzata nel modello, è stata ripresa con piccole modifiche dalla omonima classe presente in diversi package dei laboratori del corso di Programmazione ad Oggetti dello A.A. 2014/2015.

Capitolo 4 – Commenti Finali

Cercando di giudicare in modo critico il mio lavoro, posso dire di essere generalmente soddisfatto del prodotto finale, nonostante riconosca di aver fatti diversi errori durante lo sviluppo, principalmente in fase di progettazione, in cui non avevo valutato bene l'importanza di alcune funzioni, come il poter segnare come servito un certo piatto. Rivalutazioni di questo tipo, avvenute dopo aver già scritto del codice, hanno portato ad un dilatamento dei tempi di sviluppo, anche se le funzioni che mi ero prefissato di realizzare sono comunque presenti.

Devo dire di essere particolarmente soddisfatto del lavoro svolto sul fronte della qualità del codice, in particolare per quanto riguarda la realizzazione di MVC e la riduzione della dipendenza tra le classi.

Ho sicuramente intenzione di continuare lo sviluppo di questo software, anche a causa del motivo che mi ha portato a scegliere di realizzarlo, ossia il mio lavorare in un bar durante la stagione estiva, conoscendo quindi bene le difficoltà che si incontrano nella organizzazione degli ordini, specie se gestiti da diverse persone. Un software di questo tipo può dare un contributo determinante nel rendere efficiente l'intero servizio, e voglio quindi continuare a migliorarlo per renderlo il più funzionale possibile, ad esempio rendendo i controlli più adeguati all'uso con schermi touchscreen.

Commenti per i Docenti

Come nota a margine, vorrei portare i docenti a conoscenza del fatto che ho avuto dei problemi con la mia installazione del tool Mercurial Eclipse, risolto grazie all'aiuto dell'Ingegner Pianini. Purtroppo, una volta reinstallato, non veniva più riconosciuto il progresso fatto fino a quel momento, e nel forzare un push è stato sovrascritto ciò che avevo precedentemente caricato su BitBucket. Dal mio punto di vista questo non è stato un problema, ma come effetto collaterale non è più visibile lo stato del progetto precedente al 19 febbraio 2015.

Appendice A – Guida utente

All'avvio del programma, si presenta una schermata vuota, con dei pulsanti sulla destra. Premendo "Aggiungi tavolo", la schermata centrale inizierà a popolarsi di pulsanti. Premendo su uno dei pulsanti appena creati, si aprirà una finestra di dialogo con cui interagire con il tavolo selezionato. È possibile selezionare un piatto dal menu a tendina, indicarne una quantità e aggiungerlo all'elenco dei piatti con il pulsante "Aggiungi". Una volta aggiunto un piatto, questo risulterà ancora non evaso. Per indicarlo come tale, è sufficiente fare un singolo clic sul suo nome nell'elenco dei piatti. Facendo invece un singolo clic con il tasto destri del mouse, una unità del piatto cliccato verrà eliminata.
Con i pulsanti in basso è possibile stampare il conto e resettare il tavolo, ossia eliminare tutti i piatti precedentemente aggiunti.

Nella schermata principale, oltre ad "Aggiungi tavolo", è presente anche il tasto per eliminarne uno. Per poter eliminare un tavolo, è necessario che l'ultimo tavolo che è stato aggiunto non abbia dei piatti segnati; deve essere quindi stato resettato con l'apposito pulsante presente nella sua finestra di dialogo.

Sono infine presenti i tasti per caricare uno stato precedente del programma da file e per salvare quello attuale, oltre ad un checkbox con cui indicare se si vuole che il programma salvi automaticamente le modifiche effettuate sui tavoli. Il salvataggio automatico inizia non appena si aggiunge un piatto ad un nuovo tavolo, quindi se all'avvio del programma si dovesse premere per errore il tasto "Aggiungi tavolo", la modifica non verrebbe salvata. Se si vuole comunque forzare il salvataggio, si può comunque premere l'apposito pulsante.
I file vengono salvati nella *working directory* dell'utente.

È opzionalmente possibile caricare un file "menu.txt", contenente un menù differente da quello integrato. Tale file è da collocare in una cartella di nome "data", che deve trovarsi nella propria *working directory.*

I dati in questo file devono essere formattati nel seguente modo:

```
[nome_piatto] + [ -- ] + [costo]
```

La parte centrale della stringa deve presentare l'esatto numero di spazi e di trattini.
I decimali nel costo del piatto vanno separati con il punto e non con la virgola.
I dati formattati nel modo sbagliato non verranno caricati; se in un file ci dovessero essere dei dati formattati erroneamente insieme a dei dati corretti, questi ultimi verranno comunque caricati.

Bibliografia

[1] – Darryl Burke (sviluppatore) - https://tips4java.wordpress.com/2012/04/14/thumbnail-icon/