CLASS SCHEDULES MANAGER

Filippo Ghinelli, Martina Magnani

Indice

Capitolo 1

Analisi

1.1 Requisiti

Il software , chiamato “Class Schedules Manager” commissionato dal Professore Mirko Viroli, mira alla costruzione di un gestionale per le lezioni del corso universitario di “Ingegneria e Scienze Informatiche” appartenente all’Università di Bologna, con sede a Cesena.

Class Schedules Manager permette di gestire l’inserimento, la memorizzazione, la cancellazione e la visualizzazione delle lezioni attive nel corso di laurea, per un determinato anno accademico.

Requisiti concordati:

- Il software aiuterà a coordinare l’inserimento delle lezioni settimanali del primo e del secondo semestre, di uno specifico anno accademico.
- La visualizzazione delle lezioni potrà essere totale (visualizzare tutte le lezioni aggiunte, senza filtri). Oppure potranno essere specificate ricerche più selettive:
  - Ricerca per docente
  - Ricerca per insegnamento
  - Ricerca per anno universitario (I anno, II anno, III anno)
  - Ricerca per aula
  - Ricerca per corte

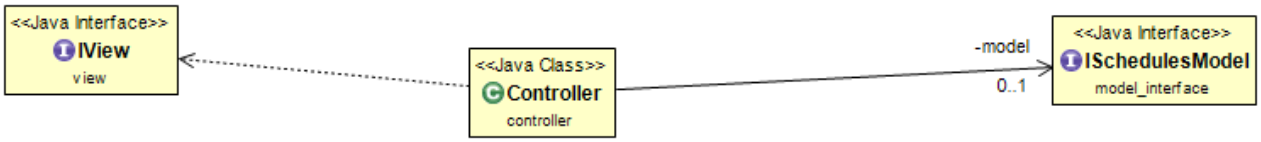
  Il corso di “Ingegneria e Scienze informatiche” di Bologna prevede al II anno la scelta fra due indirizzi: indirizzo di Scienze, oppure indirizzo di Ingegneria.
  La ricerca tramite “corte” permette di visualizzare tutte le lezioni comuni ad entrambi gli indirizzi, tutte le lezioni dell’indirizzo di Scienze oppure tutte le lezioni dell’indirizzo di Ingegneria.
- Il software da la possibilità di cambiare gli orari delle lezioni settimanalmente
- Permette di inserire diversi docenti per diverse ore di una determinata materia
- Class Schedules Manager permette anche ana di esportare l'orario in formato excel

1.2 Analisi e modello del dominio

Class Schedules Manager modella una tabella contenente tutte le lezioni salvate all'interno di esso, con varie funzioni dedite a modificarla o a cambiarne la vista. Inoltre il software permette la memorizzazione di un archivio di professori e insegnamenti.

La difficoltà principale dell'intero programma sta appunto nel fatto di riuscire a far comunicare efficacemente le modifiche effettuate visivamente dall'utente al software (come per esempio la modifica dell'orario delle lezioni aggiungendone altre o spostando quelle già presenti).
Altro aspetto impegnativo è far comprendere all'utente gli elementi di cui è composta una lezione, nel modo più semplice ed espressivo possibile.

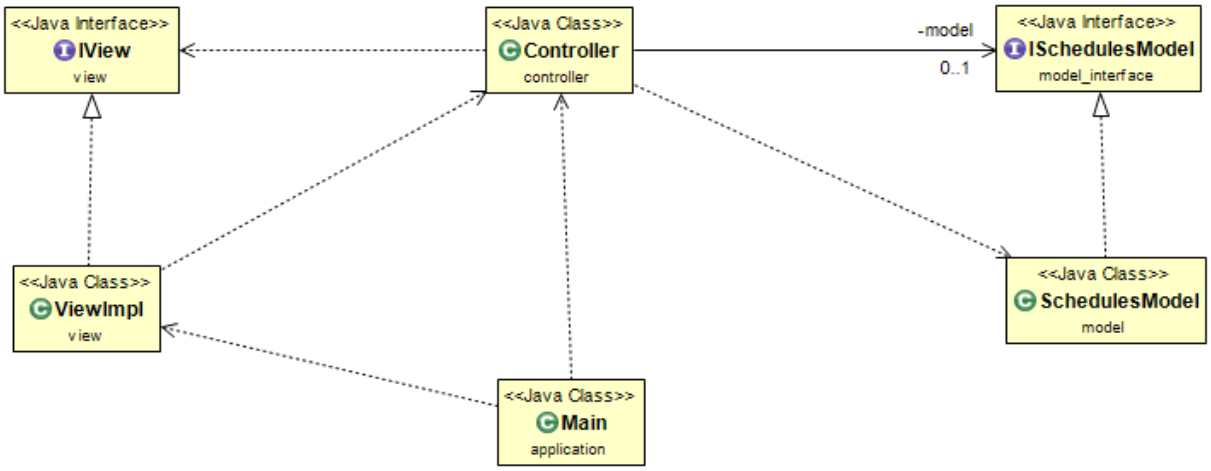
L' applicazione dovrà essere funzionante con ogni sistema operativo.



Capitolo 2

Design

2.1 Architettura



Il software interagisce seguendo il pattern MVC, difatti, abbiamo 3 sezioni principali: il Model, la View e il Controller. Il Controller fa da tramite tra l'interfaccia ISchedulesModel (che è l'entry point del Model) e l'interfaccia IView della View a cui rimanda le informazioni carpite dal Model per poterle visualizzare, inoltre è l'osservatore di alcune azioni della View. Teoricamente se si sostituisse in blocco la View il Controller funzionerebbe lo stesso e non subirebbe modifiche a parte (si ha il dubbio) la funzione di aggiunta di corsi o lezioni, ma sicuramente non va ad influire sul Model in alcun modo.

2.2 Design dettagliato

Il programma è stato suddiviso nei tre package principali che identificano il pattern MVC. Infatti si hanno il package model, il package controller e il package view.

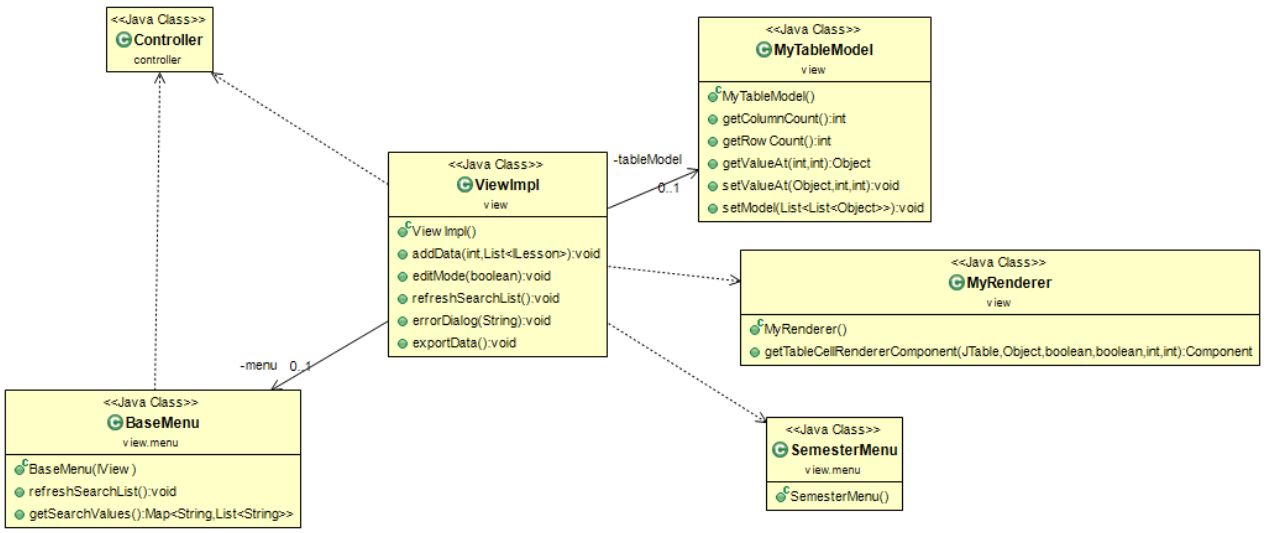
Parte view

Nella view sono presenti oltre al package principale “view” altri tre sottopakage chiamati:
view.menu : package in cui sono presenti tutte le classi dei menu della view.
view.dialog : package in cui sono presenti tutte le classi dialog e affine ad esse della view.
view.utility : package in cui sono presenti tutte le classi usate dalla view per funzioni di utility e che non sono compontenti visibili della view.

Elementi volti a risolvere i problemi riscontrati nell'analisi

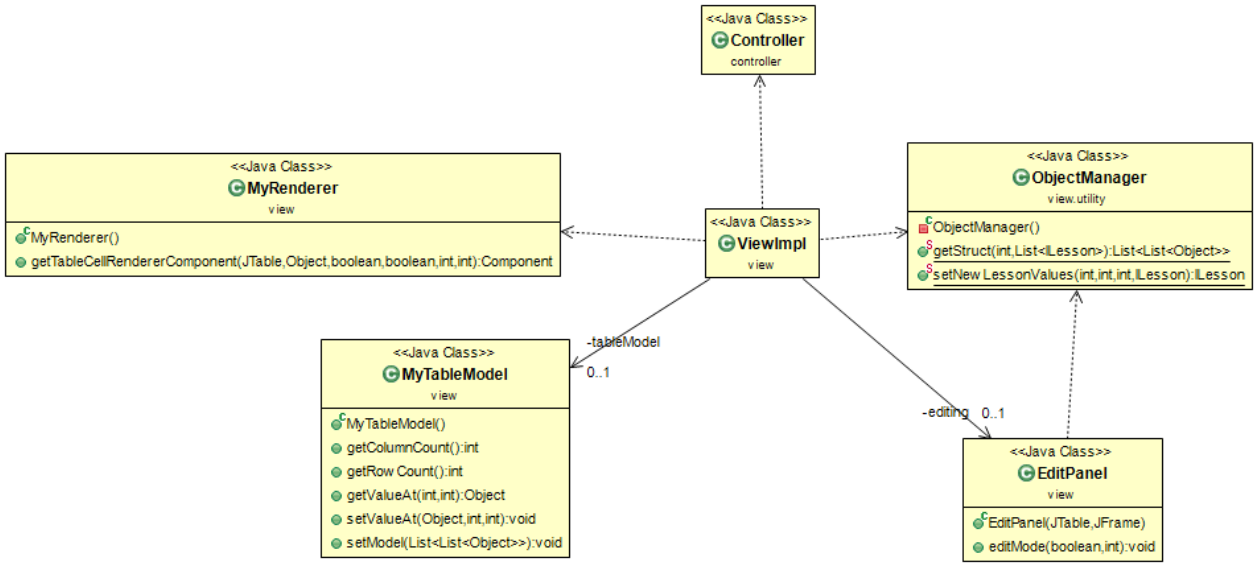
Per la visualizzazione dei dati in modo comodo all'utente:

Queste sono le classi responsabili alla gestione delle lezioni sulla tabella, la classe MyTableModel è responsabile della gestione degli oggetti in tabella, la classe MyRenderer responsabile di come appaiono gli oggetti nella tabella, aiutata dalla classe ColorUtility per assegnare i giusti colori. Infine va menzionata anche la classe ObjectManager che passa alla classe ViewImpl la struttura dati contentente gli oggetti da passare alla tabella formattata secondo le esigenze.

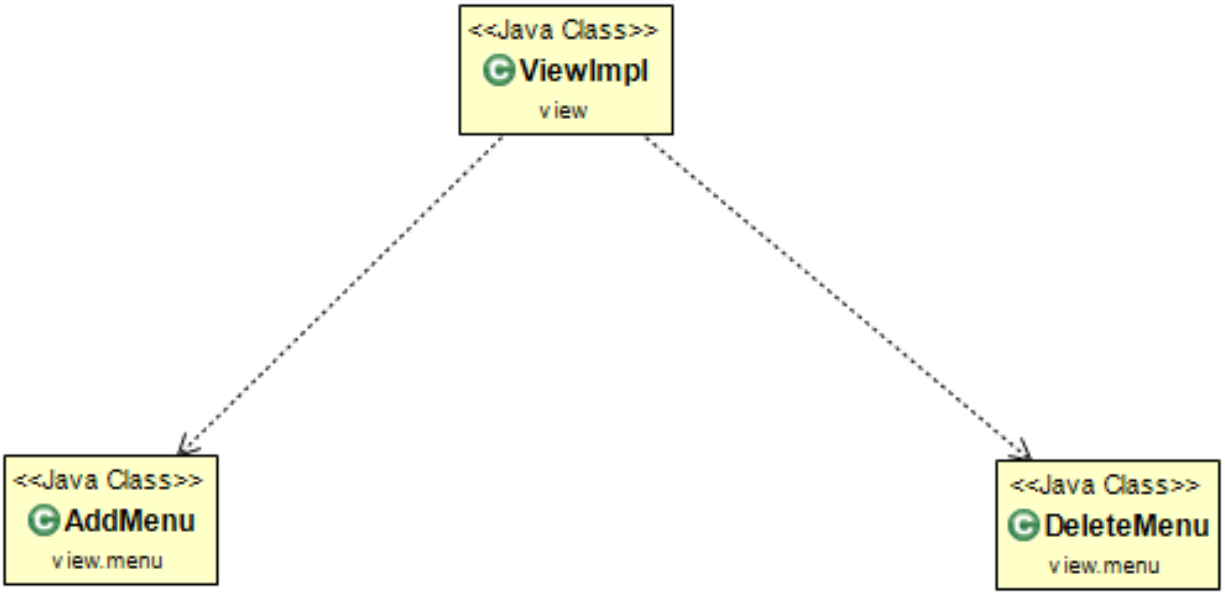


In questo schema si vedono le classi responsabili dei cambiamenti delle varie viste richieste: la classe principale della view (ViewImpl), utilizzando le due classi BaseMenu e SemesterMenu, chiede al Controller gli oggetti della corrispondente vista selezionata e di seguito li passa alla tabella che li riorganizza utilizzando le classi MyTableModel e MyRenderer.

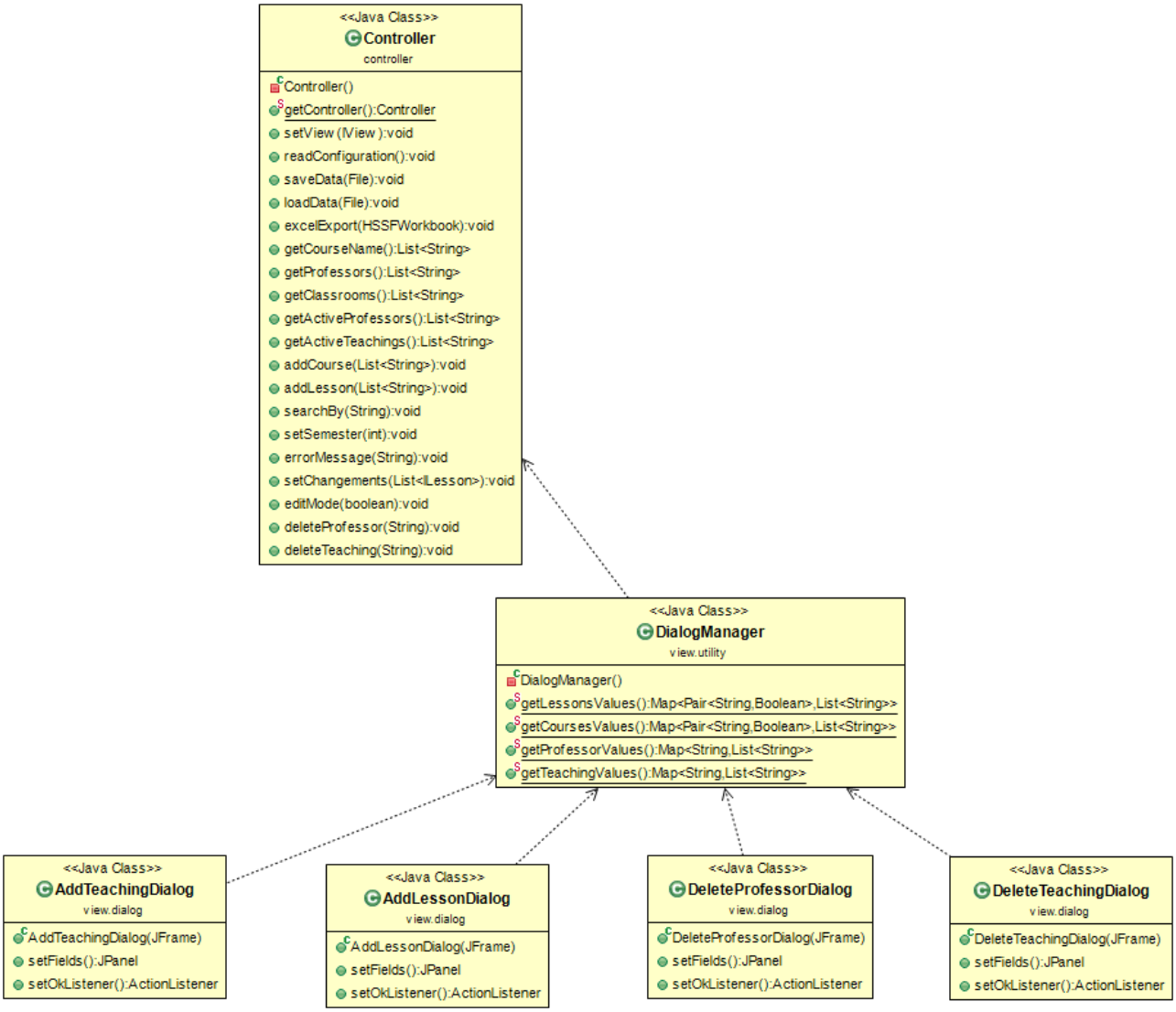
Per le modifiche dei dati da parte dell'utente:



Per gestire le modifiche delle lezioni settimanalmente la view sfrutta completamente la classe EditPanel (che a sua volta usa la classe ObjectManager) che agisce sulla tabella tramite la classe ViewImpl e in seguito le modifiche vengono passate al controller che le passa al model.

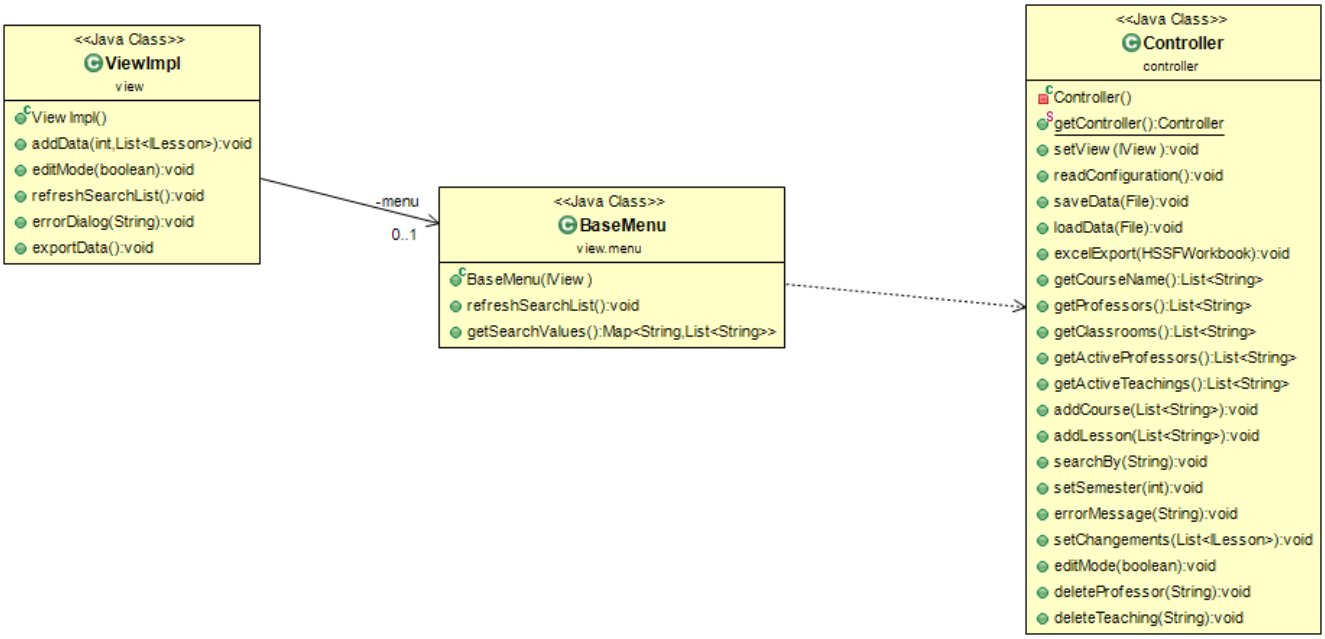


Per aggiungere un corso o una lezione o per eliminare un corso o un professore salvati, la ViewImpl usa le classi AddMenu e DeleteMenu che fanno apparire le dialog necessarie.



Questo UML fa vedere le dialog necessarie alle modifiche dei dati, tuttavia per poter funzionare correttamente esse necessitano di alcune informazioni del model che vengono passate attraverso la classe del Controller e che vengono organizzate in modo conveniente alle dialog tramite la classe DialogManager. Le addDialog e le deleteDialog fanno parte rispettivamente delle classi AddMenu e DeleteMenu viste sopra. La classe AddLessonDialog è responsabile per l'aggiunta delle lezioni all'orario e permette di scegliere quali professori sono presenti nelle lezioni che si stanno per aggiungere.

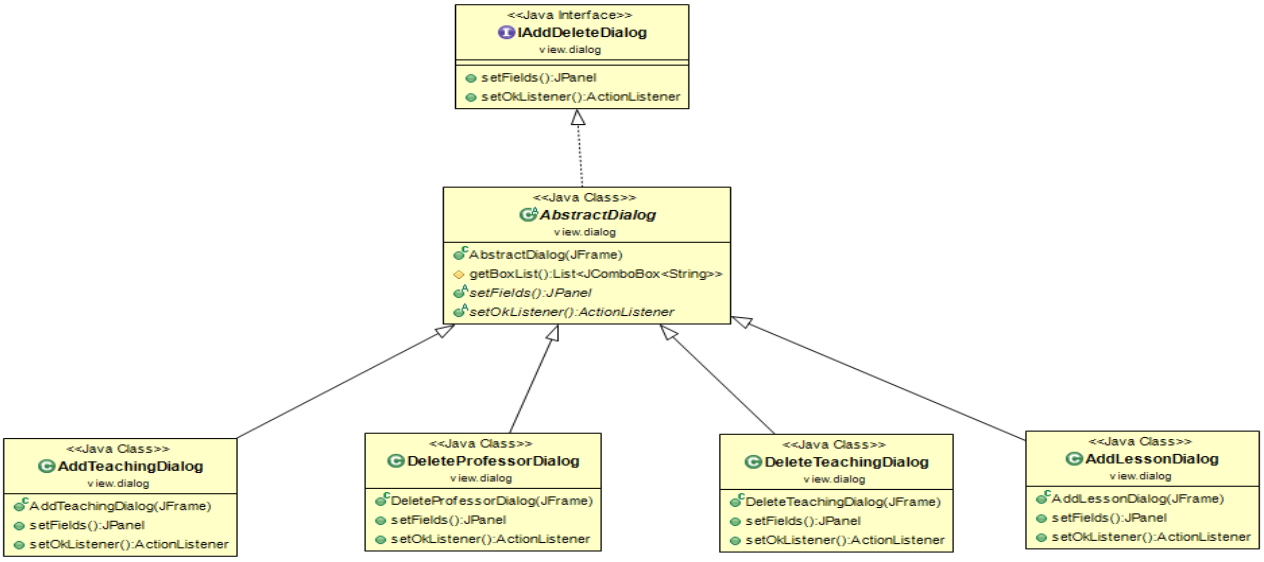
Per il salvataggio, caricamento dei file e per l'esportazione in excel:



La classe ViewImpl tramite la classe BaseMenu indica al controller quali file aprire, salvare o esportare in formato excel.

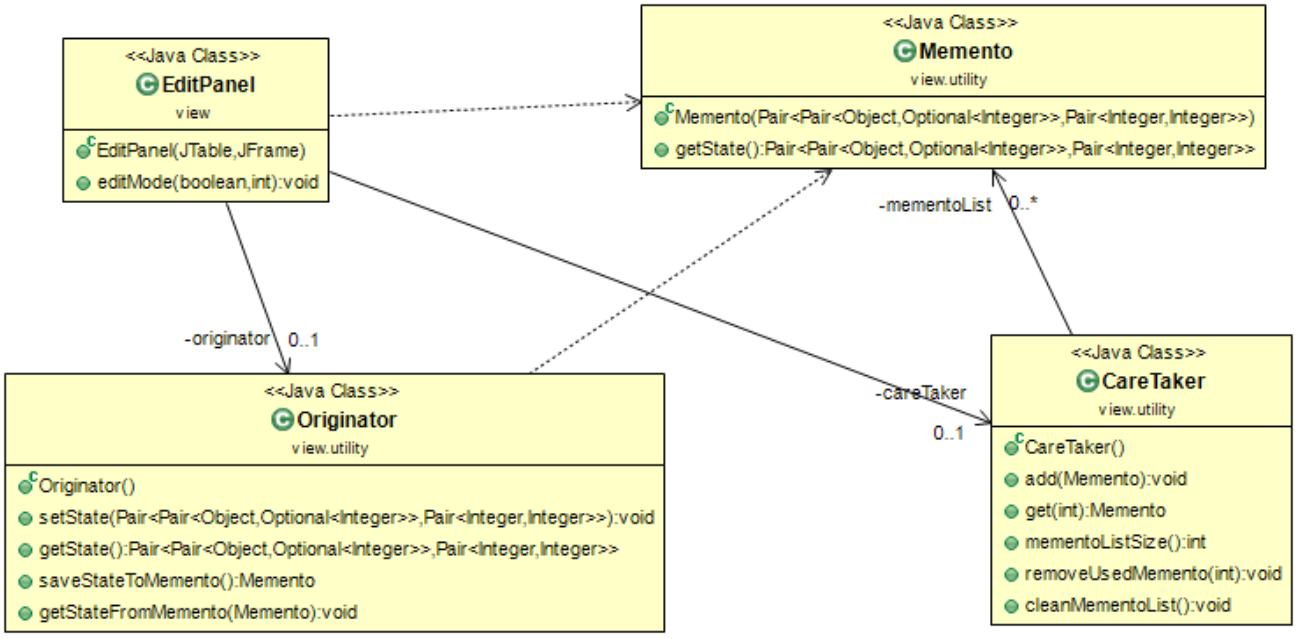
Pattern presenti:

Template method



Siccome le dialog sono praticamente uguali di aspetto, invece che stare a riscrivere le stesse parti ho preferito sfruttare il pattern "template method" per far si di dover scrivere solo 2 metodi che descrivevano l'aspetto e il comportamento finale della singola dialog.

Memento



Pattern "memento" usato dalla classe EditPanel per usare le funzioni di undo e redo. Gli stati delle modifiche vengono salvati sotto forma di oggetti della classe Memento, tramite la classe Originator e in seguito gestiti dalla classe CareTaker.

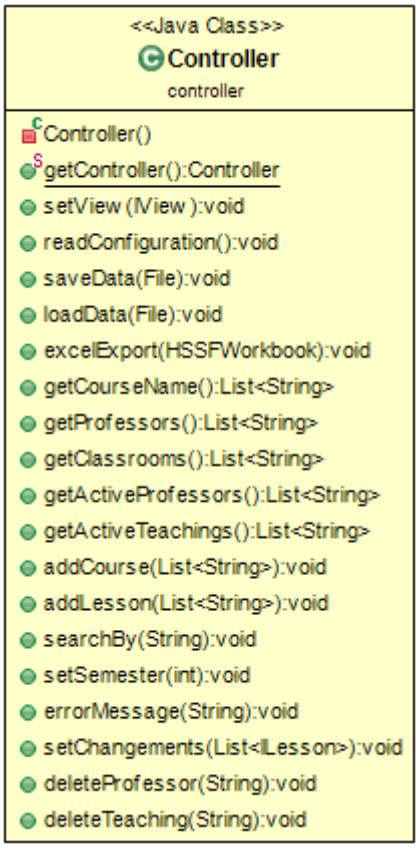
Elementi volti ad aiutare possibili modifiche future:

Le classi di utility sono comode per il fatto che contengono funzioni che determinano diversi comportamenti della view (tra cui gli algoritmi di organizzazione degli oggetti nella tabella) e che si possano fare grandi cambiamenti solo modificando quelle. Inoltre il pattern template method usato per le dialog si può continuare per creare nuove dialog che potrebbero servire per aggiungere nuove funzionalità. Per finire l'intera parte di view è frammentata il più possibile in modo tale da poter fare delle modifiche mirate a determinati componenti o funzioni più facilmente.

Parte controller

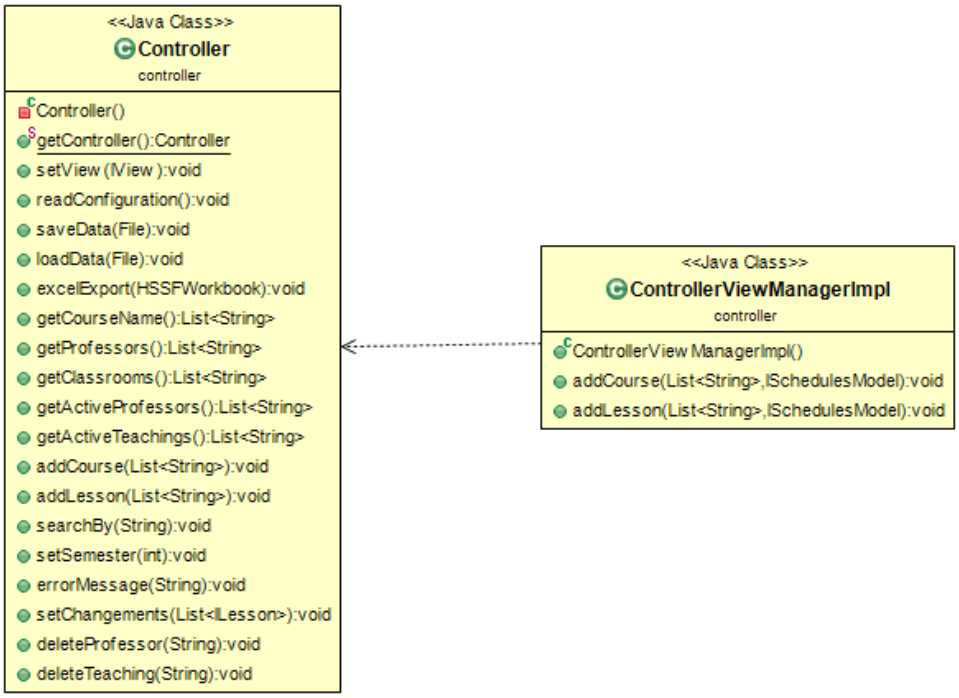
Il Controller presenta solamente un singolo package che è per l'appunto il package "controller" contenente tutte le sue classi principali.

Elementi volti a risolvere i problemi riscontrati nell'analisi:



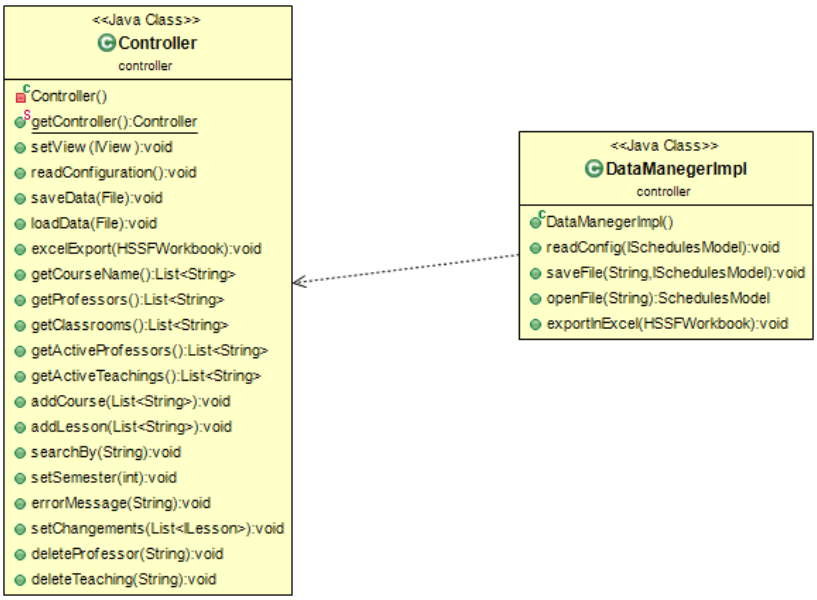
L'intera classe controller è responsabile della comunicazione principale tra view e model riorganizzando i dati presi sia dall'utente che dal model.

Per le modifiche da parte dell'utente:



Come già detto la classe Controller è responsabile delle interazioni fra model e view. Di conseguenza gestisce anche le modifche effettuate dall'utente. Menzione particolare va alla classe ControllerViewManagerImpl che gestisce le informazioni prese dall'utente per aggiungere corsi e lezioni al model.

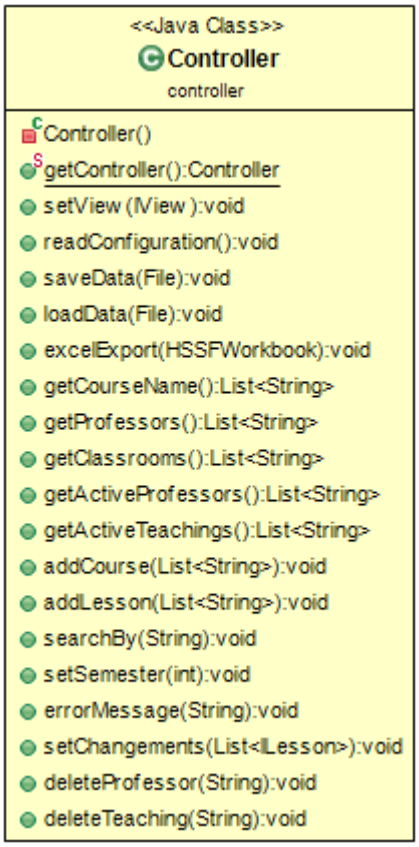
Per il salvataggio, caricamento dei file e per l'esportazione in excel:



Il controller, per le operazioni di caricamento, salvataggio ed esportazione in excel dei file, sfrutta la classe DataManagerImpl.

Pattern utilizzati:

Singleton



Il controller fa parte del pattern Singleton. In questo modo può essere richiamato da tutte le classi della view che ne necessitano senza il bisogno di dover passare ogni volta il riferimento di un oggetto Controller.

Elementi volti ad aiutare possibili modifiche future:

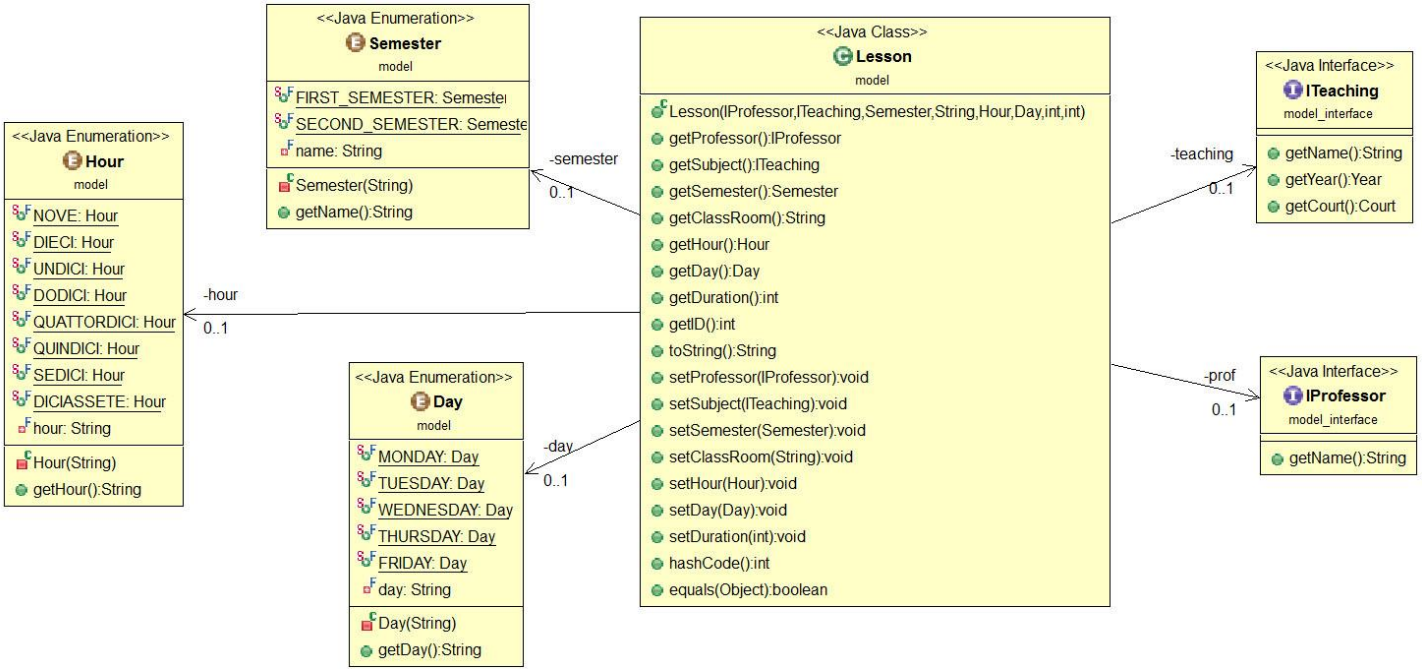
L'uso del pattern singleton permette di aggiungere metodi o cambiare quelli già presenti con discreta facilità, dato che i metodi presenti sono per la maggior parte semplici ed alcune diverse funzioni complesse sono rilegate in altre classi.

Parte model

Il model si compone di due packages: model e model_interface
model: questo package contiene tutte le classi utilizzate dal model
model.interface: questo package contiene le interfacce delle rispettive classi
Elementi volti a risolvere i problemi riscontrati nell'analisi:

Il Model, per una questione di manutenzione e sviluppo futuro, è stato realizzato in modo tale che fosse il più semplice e intuitivo possibile. E' organizzato in maniera che riproduca la struttura di un database.

Organizzazione dati strutturali:

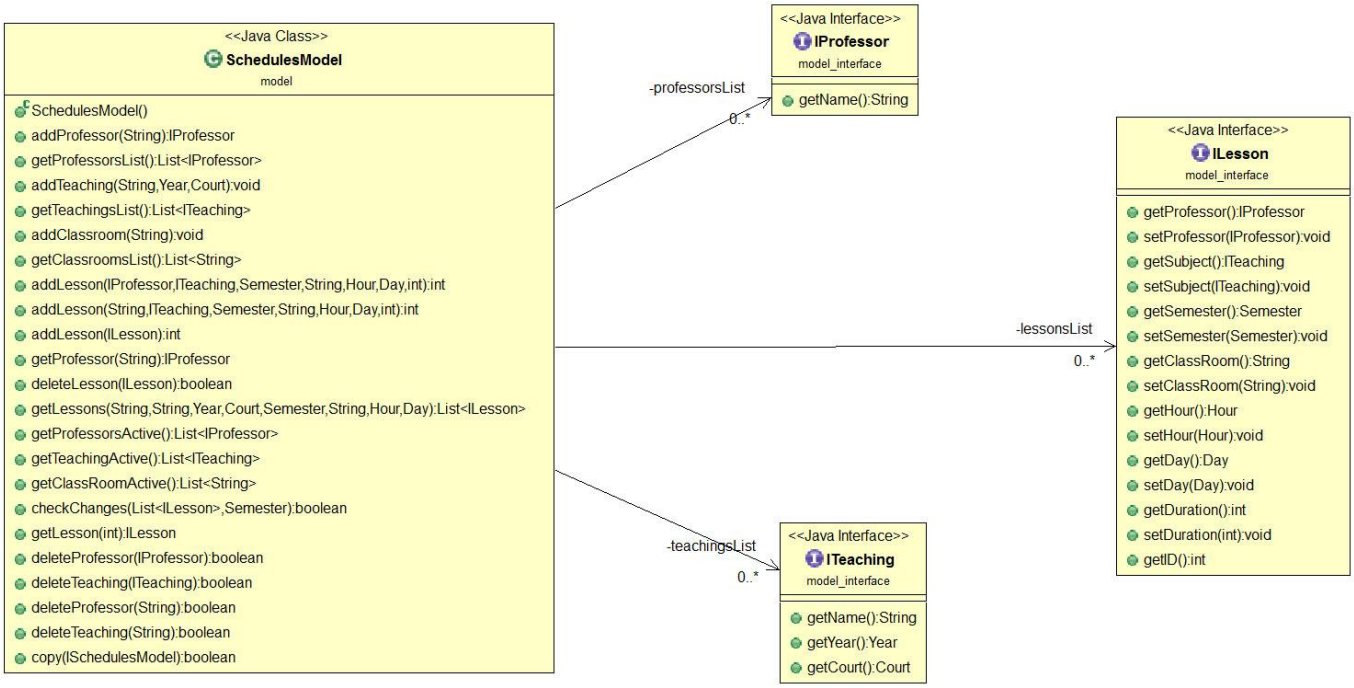


Per identificare i singoli oggetti utili all'organizzazione del Sistema, è stato scelto di organizzarli in classi; una per ogni entità (Professor, Teaching, Lesson). In questo modo sarà possibile ampliare la struttura di ogni oggetto se si presenterà la necessità.

L'oggetto principale è quello identificato dalla classe Lesson. Questo infatti è costituito da un professore (classe Professor) e da una materia (classe Teaching). In oltre per identificare (in modo univoco) una lezione sono necessari campi che riguardano il giorno, l'ora e il semestre. Per la loro realizzazione, cercando una maggiore semplicità di gestione e efficenza, è stato scelto l'utilizzo di enumerazioni (Day, Hour, Semester).

Le aule, anch'esse identificative delle lezioni, sono gestite in modo differente; infatti vengono inserite attraverso un file di configurazione.

Operazioni sui dati:



La classe SchedulesModel gestisce tutte le operazioni sui dati. Essa infatti contiene tre liste: una per tutti i professori (IProfessor), una per tutti gli insegnamenti (ITeaching) e una per tutte le lezioni (ILesson), che rappresentano la base dati da gestire.

Per ognuna di queste liste sono previsti i metodi per l'aggiunta e la cancellazione degli elementi.

Per la lista delle aule, invece, questi metodi non sono presenti in quanto è stato deciso di caricare gli elementi all'apertura del programma, prendendoli dal file di configurazione.

Per consentire alla vista di visualizzare solo una parte dell'archivio di lezioni (ricerca) è stato deciso di implementare il metodo getLessons in maniera che definendo più o meno parametri si possano filtrare le lezioni che li soddisfano. I parametri lasciati null non influenzano la maschera di ricerca.

Questo costente di apportare modifiche o estensioni alla vista senza dover agire sul modello in quanto già prevede ogni combinazione di ricerca.

Capitolo 3

Sviluppo

3.1 Testing automatizzato

Nel model sono stati effettuati test automatizzati per verificare il corretto funzionamento delle operazioni di aggiunta, cancellazione e prelievo (get) di ogni elemento dell'archivio dati. Altri test automatizzati sono stati eseguiti per le operazioni di ricerca, più o meno selettiva, delle lezioni.

Per il testing automatico è stata utilizzata la suite Junit.

3.2 Metodologia di lavoro

Ognuno dei membri si è concentrato sul proprio ruolo basandosi sulla suddivisione del pattern MVC, tuttavia con la mancanza inaspettata di un membro del gruppo non si è potuta rispettare questa suddivisione e di conseguenza alcune parti di lavoro sono state sviluppate insieme dai due membri rimanenti. In particolare lo studente Filippo Ghinelli si è occupato completamente della parte di view e controller, tranne la classe responsabile del caricamento e salvataggio di file (DataManagerImpl) di cui si è occupato solamente per l'esportazione in excel della tabella. La studentessa Martina Magnani si è occupata della parte di model e della classe citata sopra.

Durante l'integrazione delle tre parti principali è venuto fuori il problema di come poter unire efficacemente il controller con la view, in modo tale che non si dovesse portare un riferimento del controller in tutta la view. Di conseguenza si è pensato di usare il Controller come singleton in modo tale da poterlo richiamare senza problemi in ogni punto della view in cui è richiesto. Per poter collegare il controller al model, si è deciso di inserire dentro il controller un riferimento alla classe entry point del model.

Come DVCS è stato utilizzato Bitbucket (dove si è creato un repository vuoto) per poi essere collegato al progetto in Eclipse tramite l'apposito plugin di Mercurial. Il plugin citato è stato utilizzato dall'inizio fino alla fine del progetto, per fare le operazioni di commit, push, pull e merge operando principalemente sul branch di deafult e creando solo un branch a parte chiamato "stable" per la distribuzione finale del software.

3.3 Note di sviluppo

Nella parte di view, sviluppata dallo studente Filippo Ghinelli, sono state prese diverse parti già sviluppate da altre persone e riadattate, in particolare:

Per le classi responsabili della JTable e del suo funzionamento (MyRenderer, MyTableModel), ho preso parti di codice già fatte da uno studente dell'università (Lorenzo Cottignoli) che fece un progetto simile al nostro, modificate e sviluppate adeguatamente, in parte grazie ai Java tutorial ufficiali presenti su google.

Per il pattern Memento mi sono dovuto informare su google per vedere com'è strutturato e dopo averne preso un esempio (sul sito tutorialspoint) l'ho modificato in base all'uso di cui ne ebbi bisogno.

Infine ho preso da stackoverflow parti del codice e consigli sull'implementazione degli eventi da mouse e tastiera presenti nel programma, modificate e riadattate adeguatamente.

Inoltre per l'esportazione della tabella in excel ho sfruttato le Apache API trovate sempre su google, imparandole ad usare in parte grazie ad un video tutorial su YouTube e in parte grazie a miei esperimenti.

Capitolo 4

Commenti finali

4.1 Autovalutazione e lavori futuri

Il progetto è venuto bene, presenta tutte le funzionalità richieste e discretamente complesso, tuttavia non si è certi del completo rispetto del pattern MVC e di conseguenza non si è certi che il progetto sia strutturato come si dovrebbe. In particolare la view è stata frammentata in modo tale da separare le varie funzioni per aiutare sviluppi o modifiche future, è abbastanza facile capire come interagirci e inoltre esprime le informazioni necessarie in modo efficace. Il controller d'altro canto non è stato gestito nel migliore dei modi strutturalmente parlando, tuttavia è semplice e facile da modificare e/o ampliare.

Il model è stato trutturato da assolvere I compiti richiesti e organizzato per consentire una facile manutenzione e estensione.

Il lavoro è completamente funzionante e risponde a tutti i requisiti richiesti dal committente.
Tuttavia, in futuro, si potrebbe estendere l'uso a più università in modo che si possa scegliere quale file di configurazione aprire all'avvio del programma (invece di modificare lo stesso file). Si poterbbero, inoltre, sviluppare nuove funzioni di registro per ogni singola lezione, oppure evidenziare eventi speciali nell'orario al di fuori delle normali lezioni.
Altri possibili sviluppi sono: l'estensione dell'undo e redo anche per l'aggiunta dei corsi e delle lezioni, e la selezione della cartella di destinazione per l'esportazione in excel.

Appedice A

Guida utente

Class Schedules Manager è un programma semplice e immediato nell'uso. Tuttavia ci sono alcuni dettagli che vanno specificati per chiarire il suo funzionamento.

Interazione con file esterni al programma:
Il file di configurazione deve rispettare una specifica sintassi. Siccome il file serve per il caricamento delle aule scolastiche, l'utente deve scrivere:
aula: Nome Aula

Il file di configurazione chiamato "config.yml" è nella cartella chiamata “Class Schedules Manager” all’interno della user home del proprio computer.
Il caricamento del file è possibile solo per file salvati precedentemente dal programma.
L'esportazione del file in excel avviene in modo rapido e automatico e il file si troverà all'interno della cartella dell'utente nominato "Lessons.xls".

Specifiche dell'editing:
Quanto si vuole editare una tabella degli orari, si ha la possibilità di eliminare o spostare delle lezioni, quest'ultima funzione si esegue prendendo momentaneamente fino al massimo di due lezioni con il tasto "Keep" per poi riposizionarle nella tabella premendo il tasto della lezione desiderata che comparirà in seguito alla loro rimozione, tutto ciò si può realizzare cliccando nella tabella le celle desiderate, sia per prenderle, sia per posizionarle. Durante l'editing si hanno a disposizione le funzioni di undo e redo, che si possono eseguire rispettivamente premendo "ctrl z" e "ctrl y".