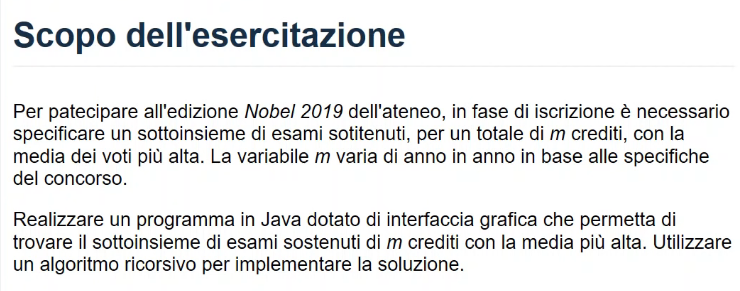
TECNICHE DI PROGRAMMAZIONE

09/04/19

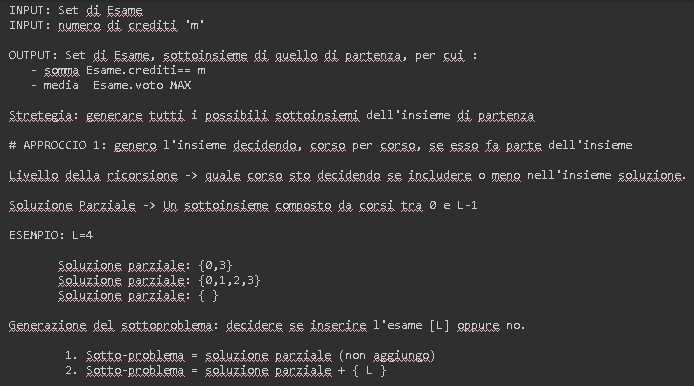
ESERCIZIO VotiNobel. Obbiettivo del programma è il seguente:

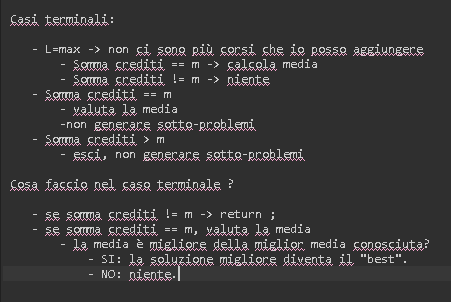


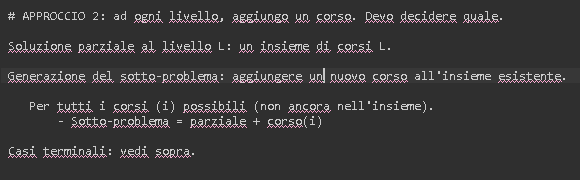
Aprendo il file .fxml possiamo vedere che l’interfaccia grafica è già stata preparata.

Come prima cosa da fare, importiamo il database su HeidiSQL. Successivamente ci chiediamo qual è l’impostazione ricorsiva che possiamo dare alla nostra soluzione.   
Creiamo un file di testo su Eclipse chiamato ‘ricorsione’ nel quale ci appuntiamo tutti i passaggi da seguire per impostare la ricorsione appunto.

Nella prima parte della lezione il professore ha spiegato come muoversi in questo problema compilando il file di testo con tutti i passaggi.







La complessità del primo approccio è 2^N mentre nel secondo è N!

ACCORTEZZA: permettere solo sequenze CRESCENTI di esami nella soluzione parziale

Complessità diventa: 2^N

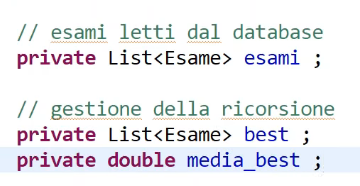
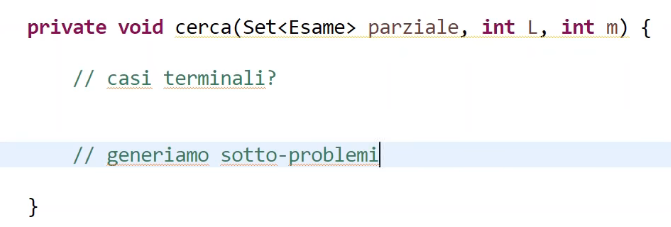
Passiamo al codice in sé per sé.

Analizzando la classe VotiNobeloController notiamo che il bottone doCalcolaCombinazione è stato già scritto, infatti quando l’utente inserisce un valore, chiama un metodo del Model che si chiama calcolaSottoinsiemeEsami a cui viene passato il numero di crediti.

Noi dobbiamo andare a gestire questo metodo nel modello. **DA ADESSO IN POI SIAMO NEL MODEL!!!**

Decidiamo di usare l’approccio 1!!!

Incominciamo definendo la procedura ricorsiva “cerca” che riceve la lista degli esami completa (che dovremmo acquisire, ma dato che non cambia possiamo pensare di averla sotto forma di variabile Lista), l’insieme corrente di scelte fatte ( Set di Esame che chiamiamo “parziale”), il livello e il numero di crediti “m”. Successivamente dovrà anche sapere qual è la soluzione migliore.

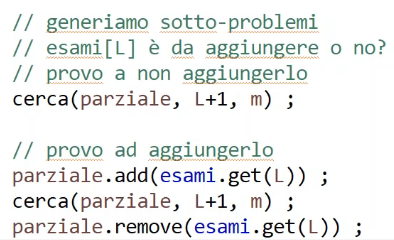
Variabili che ci servono. Metodo ricorsivo con impostazione di come affrontarlo.

In questa procedura immaginiamo di essere nel livello L e ci chiediamo se l’esame della lista “Esame” in posizione [L] è da aggiungere o no. Gestisco i 2 casi decidendo prima di non aggiungerlo e poi invece di aggiungerlo.

Nel primo caso (non aggiungo l’esame) richiamo ricorsivamente sempre la procedura “cerca” però con parametri (parziale, **L+1**, m) e non faccio nessun backtracking perché non ho modificato niente.

Nel secondo caso, aggiungo alla lista parziale l’esame in posizione L e poi richiamo di nuovo la procedura sempre con livello L+1. Infine, faccio backtracking mettendo così le cose a posto.

La situazione è la seguente (ci troviamo sempre all’interno della procedura ricorsiva “cerca”):



Ora gestiamo i casi terminali.

Per prima cosa controlliamo subito se la soluzione parziale ha più di “m” crediti. Se così fosse “scappiamo via più veloce che possiamo” cit.

Quindi calcoliamo i crediti della soluzione parziale e li inseriamo in una variabile di tipo integer chiamata “crediti”. Il metodo usato l’abbiamo chiamato sommaCrediti() ma dobbiamo ancora gestirlo.

Se il numero di crediti è maggiore di “m”, return.

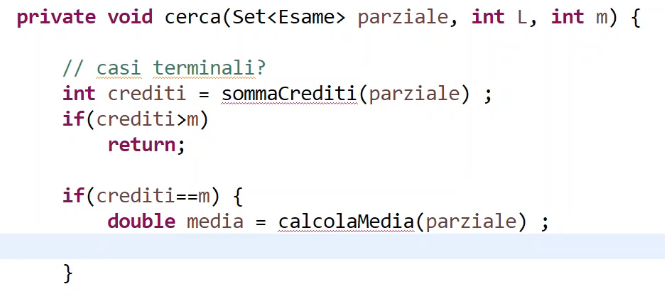
Nel caso in cui il numero di crediti sia invece =m, calcoliamo la media mediante un metodo “calcolaMedia()” che andremo ad implementare sempre in seguito.

Ora se la media trovata è maggiore della “media migliore” allora sostituisco alla “best” (la lista di esami che dobbiamo cercare noi), questa nostra nuova lista “parziale”. Infine, sostituisco alla “media best”, “media”.

ATTENZIONE: dobbiamo sempre creare un clone della lista parziale, altrimenti facciamo riferimento sempre allo stesso oggetto!! Quindi **non fare mai**:

best = parziale;

Quindi il codice fino ad adesso sarà il seguente:

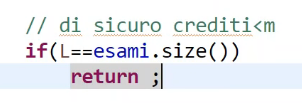




Ricordarsi di mettere il return alla fine in quanto noi, dopo la fine di questo ramo dell’if, non vogliamo più scendere. Se non l’avessi messo, avrei trovato una soluzione migliore, ma poi il programma avrebbe continuato comunque a cercare soluzione migliori. Nell’else finale bisogna anche indicare di voler uscire con il return.

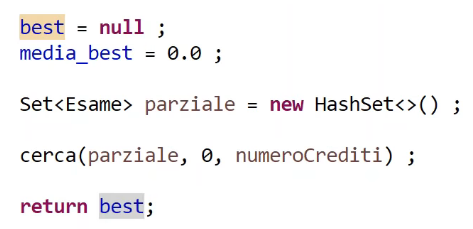
Abbiamo considerato tutti i casi termi nali? Crediti>m si, Crediti==m sì, ma Crediti<m no.

Gestiamo allora questo ultimo caso. L’unica cosa da fare è controllare che io abbia ancora esami da aggiungere,quindi:



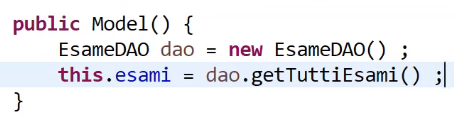
Abbiamo finito di gestire i casi terminali, rimane il caso iniziale, quindi l’avvio della ricorsione.

L’avvio di quest’ultima è compito della procedura chiamante che deve predisporre tutte le variabili e le collection in modo che la ricorsione possa poi lavorare correttamente. Quindi nella nostra procedura “CalcolaSottoinsiemeEsami” inizializziamo tutte le nostre variabili e chiamiamo la prima volta la procedura ricorsiva “cerca”.

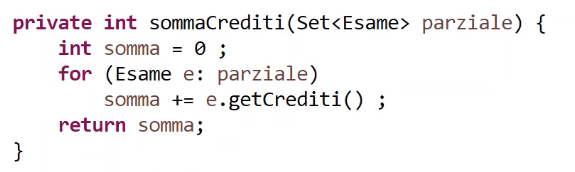


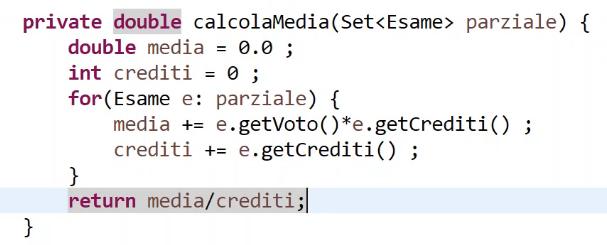
Codice all’interno della procedura “CalcolaSottoinsiemiEsami”. Variabile best impostata uguale a null così nel caso in cui “cerca” non trova niente, il return mi ritorna null. Importante chiamare la prima volta la procedura “cerca” passando al posto di “m” il numeroCrediti che vengono a loro volta passati da input dall’utente dall’interfaccia grafica.

Successivamente abbiamo constatato che era meglio leggere la lista di esami direttamente dal Model, quindi abbiamo creato un costruttore che chiede al Dao di leggere gli esami. Così facendo, quando costruisco il Model, lui si va a leggere tutti i dati dal database. **SIAMO ANCORA NEL MODEL!**



Ora implementiamo i due metodi “SommaCrediti” e “CalcolaMedia”.





Come ultima cosa da gestire, ritorniamo nel Controller perché dobbiamo dire ancora al bottone di stampare il risultato.



FINE.