ESAME DI FONDAMENTI DI INFORMATICA T-2 del 10/09/2013
Proff. E. Denti – G. Zannoni
Tempo a disposizione: 4 ore MAX

NB: il candidato troverà nell'archivio ZIP scaricato da Esamix anche il software "Start Kit"

NOME PROGETTO ECLIPSE: CognomeNome-matricola (es. RossiMario-0000123456)

L'università di Dentinia, *TeethCollege,* non paga di aver gestito in passato le carriere dei propri studenti dal punto di vista del piano di studi, ha richiesto ora una nuova applicazione per gestire gli esami e i relativi voti.

DESCRIZIONE DEL DOMINIO DEL PROBLEMA

Ogni studente è caratterizzato da nome, cognome, una matricola univoca e una *carriera*, costituita da un insieme di *insegnamenti,* ognuno dei quali è caratterizzato a sua volta da codice univoco, denominazione e valore in crediti.

Per ogni insegnamento lo studente deve, prima o poi, sostenere il relativo *esame,* che dà luogo a un voto e si svolge in una certa *data*. Se l'esame ha esito positivo (voto≥18), non può essere ri-sostenuto, mentre in caso di esito negativo dovrà essere possibile ri-sostenerlo in una data successiva.

In ogni momento deve essere possibile sapere il totale dei *crediti acquisiti* dallo studente, intendendo con ciò i crediti corrispondenti agli esami già sostenuti con esito positivo, e la corrispondente *media pesata* in base ai crediti.

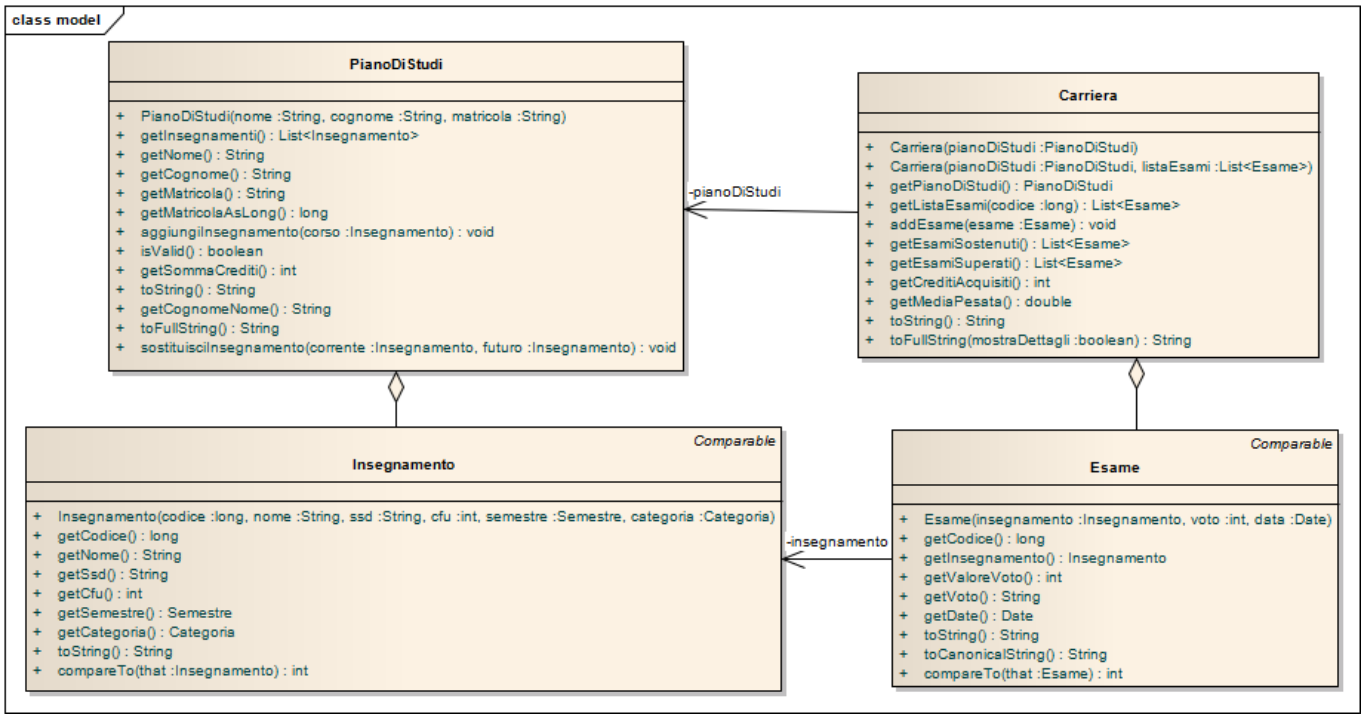
I file di testo Insegnamenti.txt e PianiDiStudi.txt contengono rispettivamente l'elenco degli insegnamenti attivi e i piani di studio di alcuni studenti: i relativi reader sono forniti già pronti nello Start Kit.

Altri file di testo (uno per ogni studente, di nome *matricola*.txt dove matricola è la *matricola* dello specifico studente) mantengono l'elenco degli esami sostenuti da ogni studente: il loro formato è specificato più oltre.

Parte 1 (punti: 18)

Dati (namespace teethcollege.model) *(punti: 9)*

Il modello dei dati deve essere organizzato secondo il diagramma UML più sotto riportato.



SEMANTICA:

a) Gli enumerativi ***Categoria*** e ***Semestre*** (forniti) rappresentano rispettivamente le due categorie di insegnamenti (obbligatori/a scelta) e i due semestri (primo/secondo);

b) La classe ***Insegnamento*** (fornita) rappresenta un insegnamento con tutte le sue proprietà;

c) La classe ***PianoDiStudi*** (fornita) rappresenta il piano di studi di uno specifico studente, ossia l'insieme di insegnamenti di cui deve sostenere l'esame, coi relativi crediti;

d) La classe ***Esame*** (fornita) rappresenta un esame relativo a un certo insegnamento, sostenuto in una certa data; la rappresentazione interna del voto è un intero compreso fra 1 e 31, dove quest'ultimo rappresenta convenzionalmente il "30 e Lode": per questo motivo il metodo ***getValoreVoto*** restituisce il valore del voto (**fra 1 e 30**, quindi ripulito dalla lode) utile per il calcolo della media, mentre il metodo ***getVoto*** restituisce il voto inteso come stringa (incluso quindi "30L" nel caso del 30 e Lode). Si noti come un ***Esame*** si costruisca passando al costruttore i parametri ***Insegnamento*** (relativo all'***Esame***), voto e data e come, conseguentemente, dall'***Esame*** sia possibile risalire all' ***Insegnamento*** mediante il metodo ***getInsegnamento***. Si notino, infine, le diverse stringhe ottenibili tramite i due metodi ***toString*** e ***toCanonicalString***, quest'ultima utile per la persistenza (si veda oltre).

e) **La classe *Carriera* (da realizzare)** rappresenta la carriera di uno specifico studente intesa come coppia (piano di studi, esami sostenuti). La classe espone due costruttori:

- ***Carriera(PianoDiStudi)*** che costruisce una carriera con quel piano di studi e lista esami sostenuti vuota;
- ***Carriera(PianoDiStudi, List<Esame>)*** che costruisce una carriera con quel piano di studi e lista esami sostenuti inizializzata con gli esami forniti nella lista passata come secondo argomento;

La classe espone inoltre i seguenti metodi:

- ***getPianoDiStudi*** ritorna il piano di studi passato all'atto della costruzione;
- ***getListaEsami(long codiceInsegnamento)*** ritorna la lista dei soli esami sostenuti per un dato insegnamento;
- ***getEsamiSostenuti*** ritorna la lista di tutti gli esami sostenuti (compresi quelli con esito negativo);
- ***getEsamiSuperati*** che ritorna la lista di tutti, e soli, gli esami superati, ossia sostenuti con esito positivo;
- ***getCreditiAcquisiti*** ritorna il totale dei crediti acquisiti, ossia corrispondenti agli esami superati;
- ***getMediaPesata*** restituisce la media, pesata in base ai crediti, dei voti degli esami superati (sono esclusi gli esami con esito negativo): a questo fine, il "30 e Lode" dev'essere valutato come 30. Il risultato deve essere arrotondato alla seconda cifra decimale. Nel caso in cui la media non sia calcolabile (assenza di esami superati), restituisce 0.
- ***addEsame(Esame)*** aggiunge l'esame passato come argomento alla lista degli esami sostenuti per l'insegnamento corrispondente, lanciando ***IllegalArgumentException*** se l'esame non può essere aggiunto perché riferito a un insegnamento non in carriera o perché risulta già sostenuto con esito positivo: il messaggio associato all'eccezione deve descrivere il motivo del rifiuto.
- ***toString*** deve rimapparsi sulla omonima *toString* del ***PianoDiStudi*** interno;
- ***toFullString(boolean mostraDettagli)*** deve mostrare la situazione complessiva della carriera in termini di numero di esami sostenuti, numero di esami superati, crediti acquisiti e media pesata; se il flag *mostraDettagli* è vero, deve mostrare anche l'elenco dettagliato, esame per esame, sia di tutti gli esami sostenuti sia di quelli superati (si veda Fig. 3 in fondo per un esempio).

SUGGERIMENTO: si suggerisce di mantenere internamente gli esami sostenuti sotto forma di mappa indicizzata per codice insegnamento, in modo da poter reperire velocemente i soli esami relativi a un dato insegnamento.

Persistenza (namespace teethcollege.persistence) *(punti 9)*

Come già anticipato, i file di testo Insegnamenti.txt e PianiDiStudi.txt contengono l'elenco degli insegnamenti attivi e i piani di studio di alcuni studenti: i relativi reader sono forniti già pronti nello Start Kit, quindi il formato di questi file non viene dettagliato. Gli esami di ciascuno studente, invece, sono mantenuti in un file di testo distinto per ogni studente, di nome *matricola*.txt dove *matricola* è la matricola dello studente: ogni riga descrive un esame, nel formato codice insegnamento, voto (31 per il "30 e Lode"), data (nel formato GG/MM/AA o GG/MM/AAAA):

```
Formato di ciascun file matricola.txt
27991,  22, 18/06/2013
```

```
29228,  31, 13/07/13
...
```

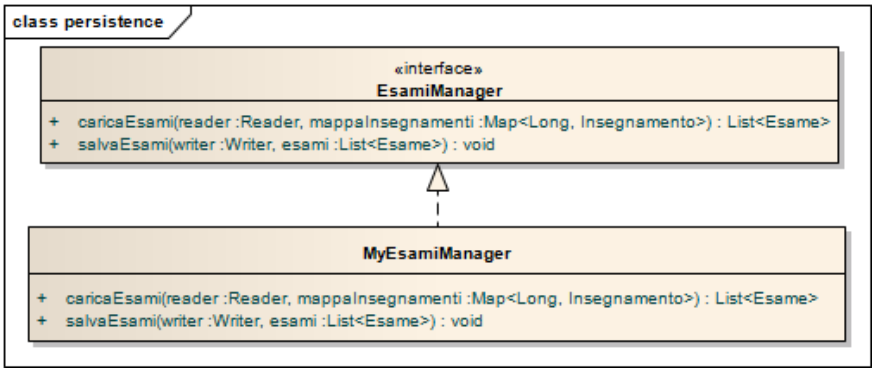
Uno studente che non abbia ancora sostenuto esami non ha inizialmente alcun file *matricola*.txt a sé associato.

L'interfaccia ***EsamiManager*** (fornita) dichiara i due metodi ***caricaEsami*** per caricare una lista di esami partendo da un ***Reader*** e da una mappa di insegnamenti (la mappa serve per recuperare l'insegnamento da associare all'esame, dato il codice) e ***salvaEsami*** per salvare una lista di esami su un ***Writer***.

La classe *MyEsamiManager* (da realizzare) implementa tale interfaccia, lanciando ***MalformedFileException*** (fornita) in caso di errore nel file, con messaggio specializzato in base all'accaduto. Per evitare perdite di dati e accavallamenti fra letture e scritture sui file, è indispensabile che ***caricaEsami*** e ***salvaEsami*** aprano e chiudano i file ogni volta che vengono invocati, evitando di lasciare file aperti fra una invocazione e l'altra.

SUGGERIMENTO: per la realizzazione della scrittura, si suggerisce di sfruttare i metodi disponibili nella classe ***Esame*** (in particolare *toCanonicalString*).

Lo Start Kit contiene anche i test (da includere nel progetto) per verificare il funzionamento di questa classe.



Parte 2 (punti: 12)

L'applicazione consente sia di vedere il piano di studi degli studenti (senza poterlo però modificare), sia di operare sulla carriera: in questo senso consente sia di vedere gli esami sostenuti con i relativi dati (crediti acquisiti, media, ecc.), sia di inserire nuovi esami con i relativi voti. Caratteristica cruciale dell'applicazione è il meccanismo di salvataggio automatico dei dati inseriti, che garantisce che la situazione degli esami di ogni studente sia sempre salvata su disco a ogni modifica.

Controller (namespace teethcollege.exams.ui) *(punti 6)*

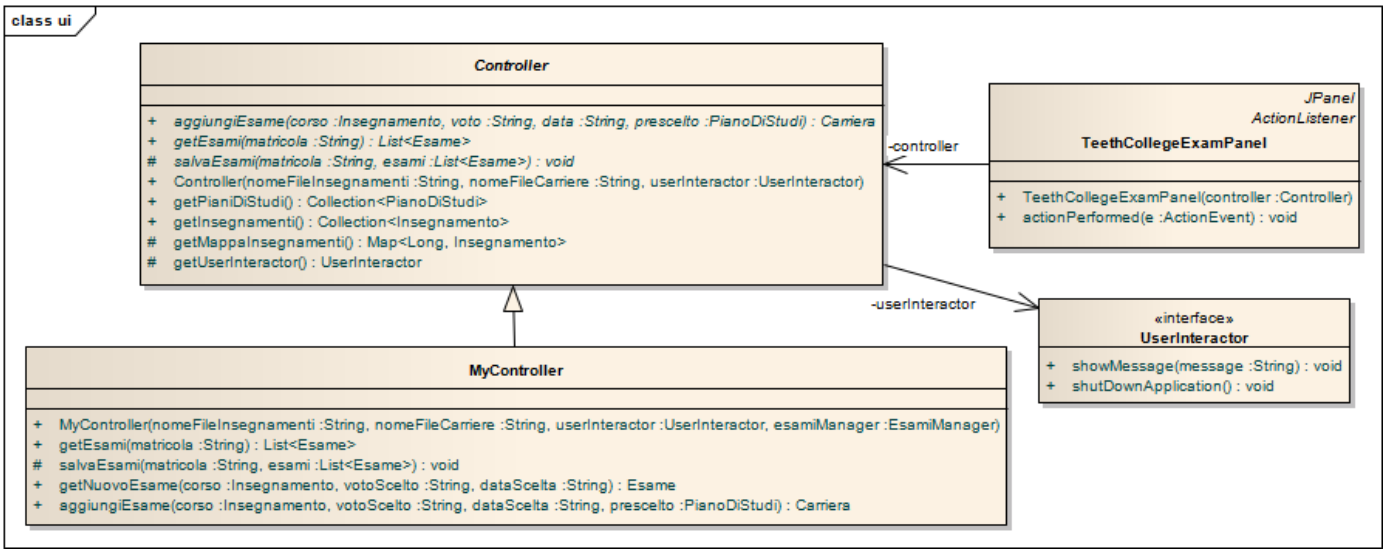
La classe astratta ***Controller*** (fornita) contiene già l'implementazione di tutta la parte di gestione di insegnamenti e piani di studio: rimangono astratti soltanto i metodi specifici della gestione di esami e carriere.

La classe *MyController* (da realizzare) estende ***Controller*** implementando i metodi astratti. In particolare:

- il costruttore prende in ingresso il nome del file contenente gli insegnamenti, il nome del file contenente i piani di studi, uno ***UserInteractor*** utile per mostrare un messaggio all'utente (***showMessage***) e chiudere l'applicazione (***shutDownApplication***), e un ***EsamiManager***: questo costruttore invoca il costruttore della classe base (il quale effettua la lettura di insegnamenti e piani di studi) e assegna l'***EsamiManager*** ad un opportuno campo privato della classe;
- ***getPianiDiStudi*** (ereditato da ***Controller***) restituisce l'elenco dei piani di studi caricati
- ***getInsegnamenti*** (ereditato da ***Controller***) restituisce l'elenco degli insegnamenti disponibili;
- ***getMappaInsegnamenti*** (ereditato da ***Controller*** – è un metodo *protected*) restituisce la mappa degli insegnamenti (la chiave è il codice);
- ***getUserInteractor*** (ereditato da ***Controller***) restituisce uno ***UserInteractor***;
- ***getEsami(String matricola)*** recupera dall'opportuno file di testo gli esami sostenuti dallo studente specificato dalla matricola passata come argomento, appoggiandosi al metodo ***caricaEsami*** di

EsamiManager, restituendo infine la relativa lista; se il file non può essere aperto, o in caso di errori di I/O, informa l'utente mediante lo ***UserInteractor*** (***showMessage***), restituendo *null*;

- ***salvaEsami(String matricola, List<Esame> listaesami)*** salva sull'opportuno file di testo gli esami sostenuti dallo studente specificato dalla matricola passata come primo argomento, appoggiandosi al metodo ***salvaEsami*** di ***EsamiManager***; se il file non può essere aperto, o in caso di errori di I/O, informa l'utente mediante lo ***UserInteractor*** (***showMessage***);
- ***getNuovoEsame*** costruisce e restituisce un nuovo ***Esame*** a partire dall'insegnamento, dal voto e dalla data specificate. È suo compito gestire correttamente il voto "30L" ed effettuare il parsing della data tramite un opportuno ***DateFormat***. Se l'operazione non può essere completata a causa di data errata o voto fuori range, informa l'utente mediante lo ***UserInteractor*** (***showMessage***), restituendo *null*;
- ***aggiungiEsame*** ottiene un nuovo esame mediante il metodo ***getNuovoEsame*** (passando i parametri opportuni) e lo aggiunge alla carriera relativa al ***PianoDiStudi*** specificato, restituendo un oggetto ***Carriera*** perfettamente configurato. A tal fine, ottiene la lista degli esami sostenuti mediante il metodo ***getEsami***, crea un'istanza di Carriera passando il PianoDiStudi specificato e gli esami appena ottenuti, tenta di aggiungere l'esame ricevuto come parametro con il metodo ***addEsame*** e, in caso di successo, effettua il salvataggio tramite il metodo ***salvaEsami*** di cui sopra, sia al metodo di ***Carriera***; se ***addEsame*** solleva eccezione, ***aggiungiEsame*** segnala che l'operazione non può andare a buon fine, informa l'utente mediante lo ***UserInteractor*** (***showMessage***) e non inserisce l'esame in carriera. In tutti i casi (anche nel caso in cui ***getNuovoEsame*** ritorni null), però, restituisce sempre un oggetto ***Carriera*** opportunamente configurato.



Interfaccia utente (namespace teethcollege.exams.ui) *(punti 6)*

L'interfaccia utente deve essere simile (non necessariamente identica) all'esempio mostrato di seguito.

> PREMESSA: appena l'applicazione parte, vengono caricati dal file Insegnamenti.txt tutti gli insegnamenti e dal file PianiDiStudi.txt tutti i piani di studi: un eventuale errore di formato nei file è mostrato tramite un'opportuna finestra di dialogo, nel qual caso l'applicazione esce immediatamente senza neanche mostrare la GUI. Se, invece, il formato è corretto ma uno o più piani di studio non sono validi (ossia non contengono insegnamenti per almeno 180 crediti), tali piani vengono scartati e non saranno considerati nell'applicazione: la GUI però partirà normalmente, riportando i piani scartati in un'opportuna finestra di dialogo. Nessuna di queste situazioni è mostrata nel seguito.

Se il caricamento preliminare ha esito positivo, compare la finestra principale dell'applicazione (Fig. 1): la combo in alto mostra gli studenti con un piano di studio valido. La GUI (**classe *TeethCollegeExamPanel*, da realizzare**) consente:

- di scegliere uno studente fra quelli la cui carriera è disponibile nella combobox in alto (Fig. 1 e 2);

- di vederne sia il piano di studi (default, Fig. 2) sia gli esami (Fig. 3): in questo caso vengono mostrati tutti i relativi dati (numero esami sostenuti, numero esami superati, crediti acquisiti, media pesata, seguiti dall'elenco di tutti gli esami in dettaglio); se lo studente non ha ancora dato alcun esame, verrà mostrata comunque una sintesi della situazione (Fig. 4), ovviamente senza dettagli relativi agli esami;
- di inserire un nuovo esame, specificando l'insegnamento, il voto e la data di sostenimento (Fig. 5): l'inserimento avviene premendo il pulsante INSERISCI, con conseguente aggiornamento della view (Fig. 6). Se l'inserimento non è possibile per qualche violazione (esame già dato con esito positivo, formato data errato), vengono mostrati opportuni messaggi in finestra di dialogo, e l'operazione non viene effettuata (Fig. 7 e 8).

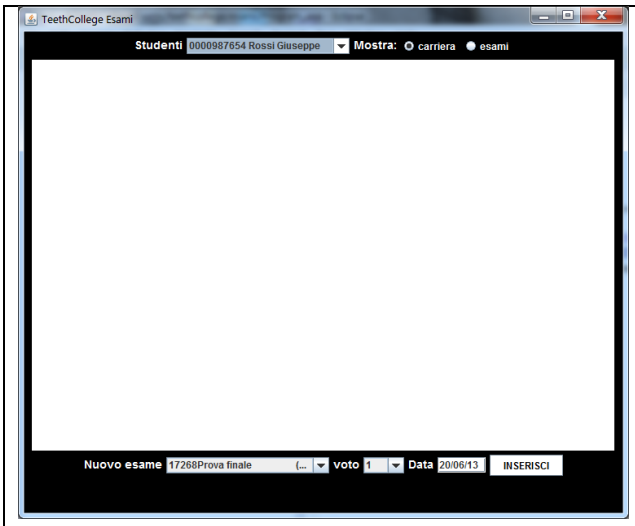


Fig. 1

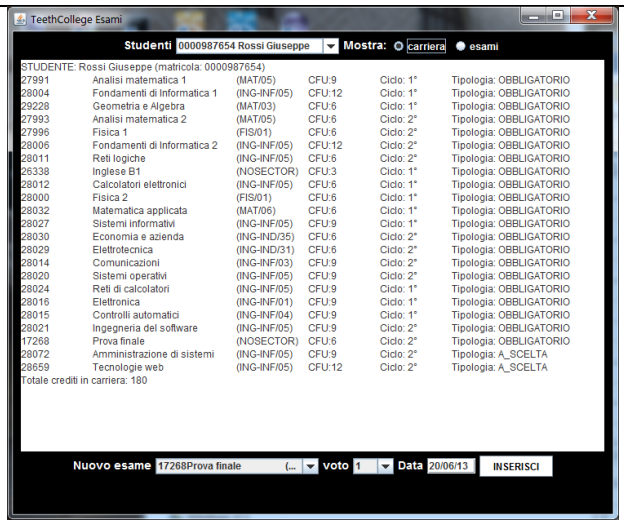


Fig. 2

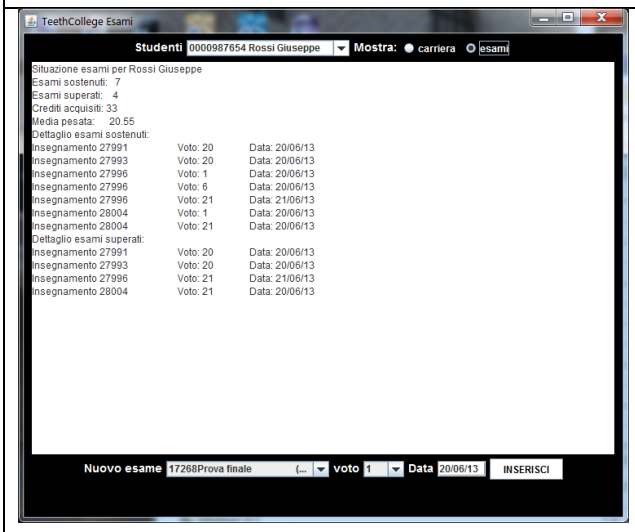


Fig. 3

Fig. 4

Fig. 5

Fig. 6

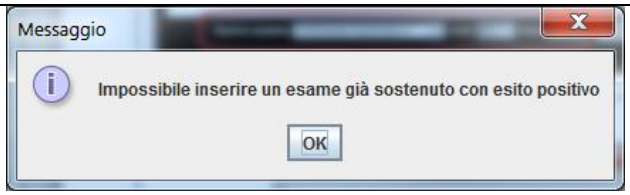


Fig. 7

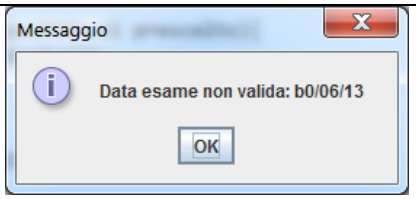


Fig. 8

Come sempre, la classe che realizza la GUI riceve nel proprio costruttore il riferimento al ***Controller*** associato.

La classe ***teethcollege.exams.Program*** (non mostrata nel diagramma UML, ma fornita nello start kit) contiene il main di partenza dell'intera applicazione.