

# ESAME DI FONDAMENTI DI INFORMATICA T-2 del 2/07/2013

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*, ha richiesto un'applicazione per gestire i piani di studio dei propri studenti.

## DESCRIZIONE DEL DOMINIO DEL PROBLEMA

All'atto dell'immatricolazione a un corso di studi, a ogni studente viene attribuita una **matricola univoca** e un **piano di studi**, costituita da un insieme prestabilito di **insegnamenti**.

### I piani di studio, gli insegnamenti, i semestri

Il piano di studi è costituito da un **insieme di insegnamenti** che lo studente deve seguire, collocati ciascuno in un **ben preciso semestre**. Ogni insegnamento è caratterizzato da codice univoco, denominazione, settore scientifico-disciplinare, valore in crediti, **categoria** (*obbligatorio* o *a scelta*) e **semestre**. Gli insegnamenti di categoria **a scelta** sono sostituibili con altri purché **a) l'insegnamento che si intende inserire non sia già presente nel piano di studi, b) l'insegnamento che si intende togliere sia effettivamente presente nel piano di studi, c) i due insegnamenti abbiano egual numero di crediti**.

I file di testo **Insegnamenti.txt** e **PianiDiStudi.txt** contengono rispettivamente **l'elenco degli insegnamenti attivi** e i **piani di studi di alcuni studenti** ( si veda più oltre per i dettagli).

## Parte 1

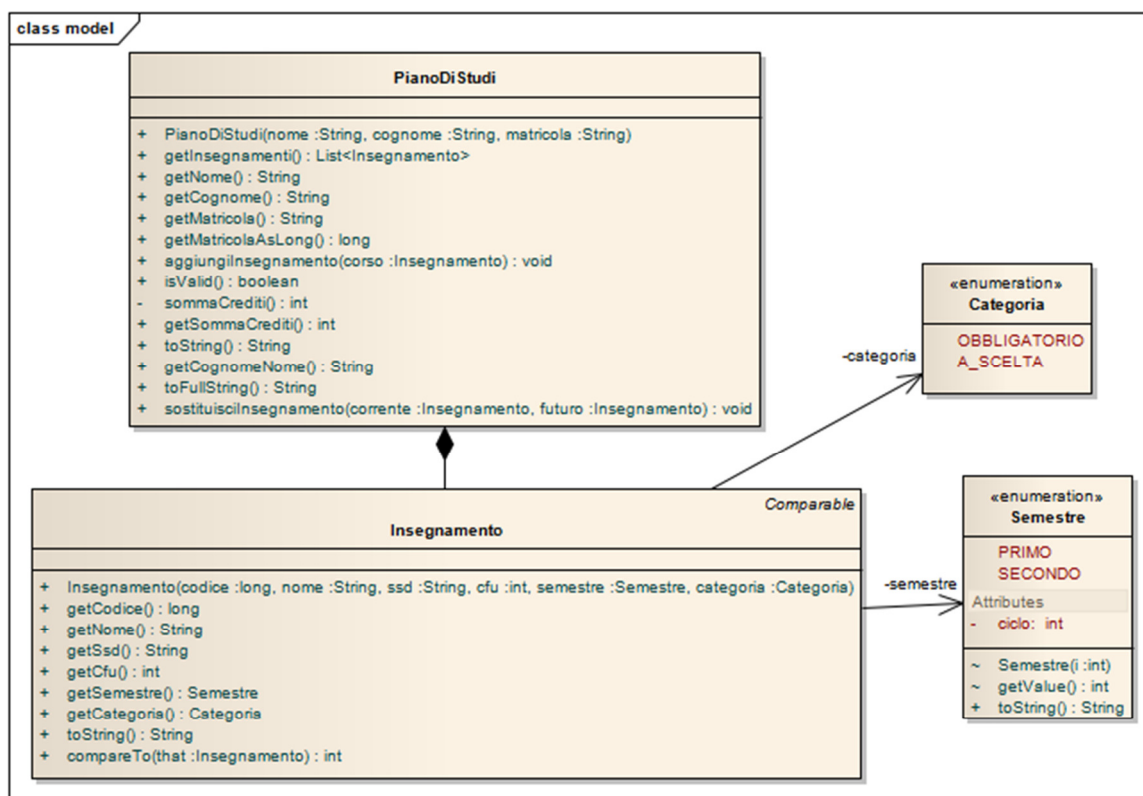
(punti: 18)

*Dati (namespace teethcollege.model)*

(punti: 6)


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

FIGURA



SEMANTICA:

- a) **L'enumerativo Semestre** (fornito nello start kit) definisce i **due valori possibili PRIMO e SECONDO**;

- b) L'enumerativo **Categoria** (fornito) definisce i due valori **OBBLIGATORIO** e **A\_SCELTA**;
- c) La classe **Insegnamento** (fornita) rappresenta un insegnamento con tutte le sue proprietà;
- d) **La classe PianoDiStudi (da realizzare)** rappresenta il piano di studi di uno specifico studente, ossia l'insieme di insegnamenti previsti nella sua carriera. Oltre agli opportuni accessor, la classe espone i metodi:
- **PianoDiStudi** (costruttore a tre argomenti **String**: nome, cognome, matricola): non deve essere possibile passare argomenti nulli e il parametro matricola deve essere convertibile a un **long** (lanciare **IllegalArgumentException**);
  - **aggiungiInsegnamento**, che aggiunge un insegnamento al piano di studi stesso: nel caso in cui un insegnamento sia già presente occorre lanciare una **IllegalArgumentException**
  - **sostituisciInsegnamento**, che sostituisce un insegnamento (primo argomento) con un altro (secondo argomento); l'operazione fallisce se l'insegnamento da sostituire non è di categoria **a scelta**, o l'insegnamento proposto in sostituzione non ha lo stesso numero di crediti di quello da eliminare, o è **obbligatorio** (si sostituiscono solo insegnamenti a scelta con altri insegnamenti a scelta), o è già presente nel piano di studi, o se l'insegnamento da togliere non è presente nel piano di studi; in tali casi, il metodo lancia una **IllegalArgumentException** con messaggio d'errore adeguato alla circostanza;
  - **isValid**, che verifica se il piano di studi è valido, ossia contiene insegnamenti per almeno 180 crediti;
  - **toString**, che produce una rappresentazione compatta (solo matricola, cognome e nome studente);
  - **toFullString**, che produce una rappresentazione estesa completa del piano di studi (matricola, cognome, nome ed elenco di tutti gli insegnamenti coi rispettivi dettagli, uno per riga): l'output è quello visibile in Fig. 4. Suggerimento: usare il metodo **toString** di **Insegnamento**. 

*Persistenza (namespace `teethcollege.persistence`)*

*(punti 12)*

Come anticipato, i file di testo **PianiDiStudi.txt** e **Insegnamenti.txt** contengono rispettivamente i piani di studi di alcuni studenti e l'elenco degli insegnamenti attivi. Più precisamente, il file **PianiDiStudi.txt** contiene una serie di record che descrivono il piano di studi di uno studente: la prima riga contiene la parola chiave **STUDENTE** seguita dal numero di matricola, dal cognome e dal nome dello studente, separati da tabulazioni dato che sia il nome sia il cognome possono contenere spazi; le righe successive elencano i singoli insegnamenti, riportandone i codici univoci (numeri long) separati da virgole e/o ritorni a capo. Il record è chiuso dalla riga contenente **"FINE STUDENTE"**.

*Formato del file **PianiDiStudi.txt***

```
STUDENTE 0000987654      De Rossi      Mario
27991, 28004, ..., 28072
...
FINE STUDENTE
...
```

I dettagli dei vari insegnamenti sono riportati nel file **Insegnamenti.txt** dove ogni riga contiene nell'ordine codice insegnamento, nome insegnamento, settore, crediti, categoria (una delle due stringhe "A\_SCELTA" o "OBBLIGATORIO") e semestre (quest'ultimo secondo la convenzione "S1" o "S2" rispettivamente per il primo e il secondo semestre), separati da virgole, come sotto riportato:

*Formato del file **Insegnamenti.txt***

```
27991, Analisi matematica 1, MAT/05, 9, OBBLIGATORIO, S1
28004, Fondamenti di Informatica 1, ING-INF/05, 12, OBBLIGATORIO, S1
28072, Amministrazione di sistemi, ING-INF/05, 9, A_SCELTA, S2
...
```

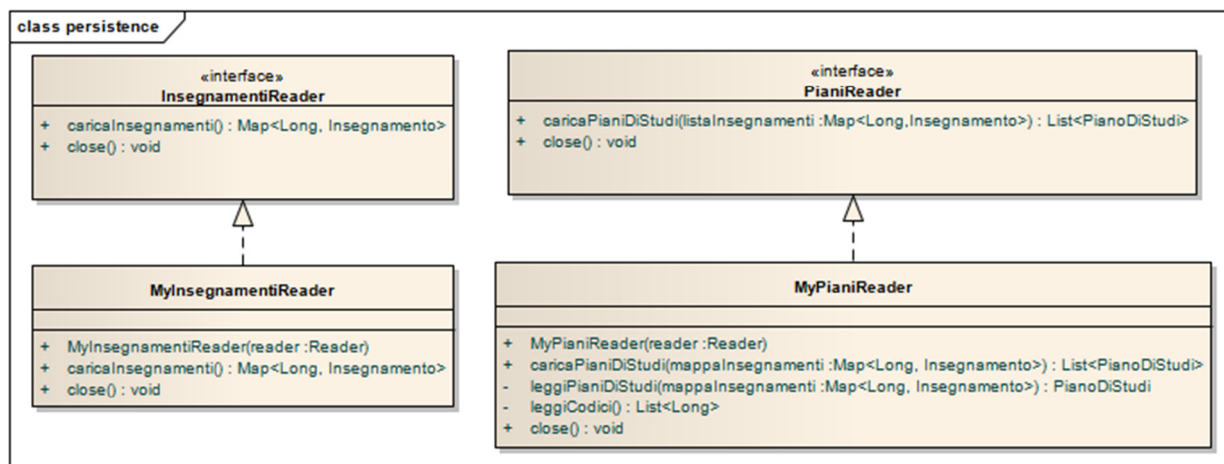
Le interfacce **PianiReader** e **InsegnamentiReader** (fornite) dichiarano rispettivamente i metodi **caricaPianiDiStudi** e **caricaInsegnamenti** che leggono da un **Reader** rispettivamente una lista di **PianoDiStudi** nel primo caso e una mappa di **Insegnamenti**, indicizzata per codice insegnamento, nel secondo caso.

**Le classi MyInsegnamentiReader e MyPianiReader (da realizzare)** implementano tali interfacce, effettuando i necessari controlli di formato ove opportuni e lanciando **MalformedFileException** (fornita) in caso di errore di formato,

con opportuno messaggio d'errore specializzato in base all'accaduto. I costruttori di entrambe le classi prendono un **Reader** come parametro e lanciano **IllegalArgumentException** in caso in cui sia null.

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

FIGURA



## Parte 2

(punti: 12)

L'obiettivo dell'applicazione è la gestione dei piani di studio degli studenti, permettendo in particolare la sostituzione di un insegnamento a scelta con un altro, nel rispetto dei vincoli sopra riportati.

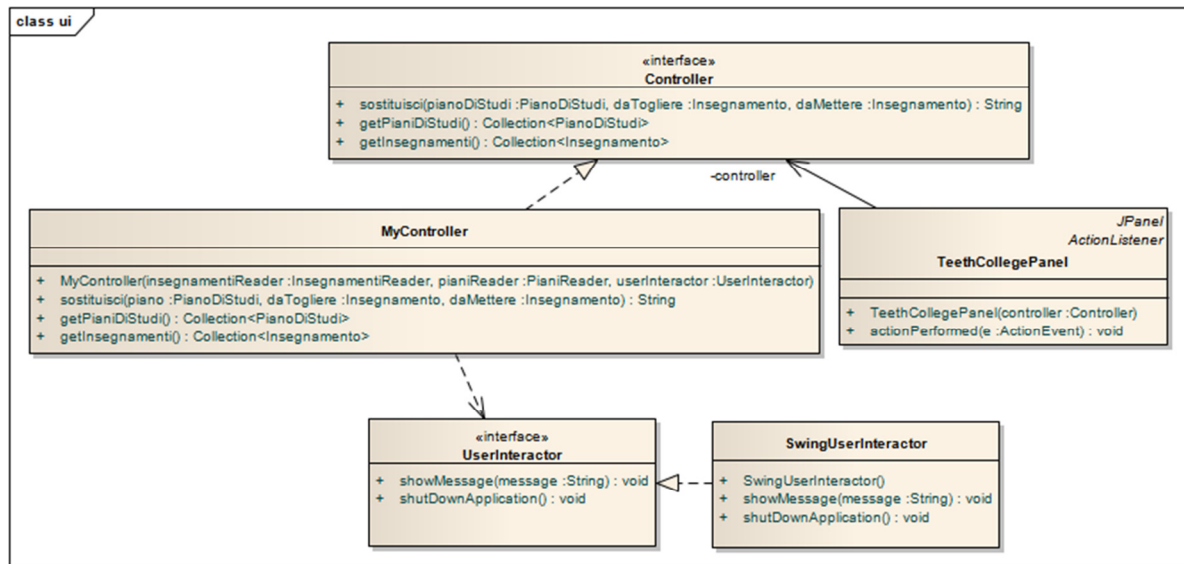
Controller (namespace *teethcollege.pianidistudi.ui*)

(punti 6)

La classe **MyController** (da realizzare) implementa l'interfaccia **Controller** (fornita). Il costruttore riceve come argomenti un **InsegnamentoReader**, un **PianiReader** e uno **UserInteractor**, e provvede al popolamento delle strutture dati necessarie; una volta caricati i dati, verifica che i piani siano validi e scarta quelli eventualmente non validi notificandolo all'utente mediante un messaggio; in caso di errore nella lettura del file, notifica l'utente e termina l'applicazione (catturare le opportune eccezioni), mentre in caso di piani scartati visualizza l'elenco dei piani scartati (per comporre il messaggio è sufficiente usare il metodo **toString** della collezione). A tal fine usare i metodi di **UserInteractor**: **showMessage** per mostrare gli errori (Fig. 1 e Fig. 2 sotto) e **shutdownApplication** per terminare l'applicazione.

I metodi **getInsegnamenti** e **getPianiDiStudi** restituiscono rispettivamente una **Collection** di **Insegnamento** e di **PianoDiStudi** (entrambe ottenute nel costruttore mediante i reader). Il metodo **sostituisci** effettua la sostituzione, nel **PianoDiStudi** passato come primo argomento, dell'**Insegnamento** passato come secondo argomento con quello passato come terzo argomento; tale metodo restituisce una string che in caso di successo è null, mentre in caso d'errore contiene il messaggio d'errore relativo ottenuto catturando le opportune eccezioni (si veda l'implementazione del metodo **sostituisciInsegnamento** nel **PianoDiStudi**).

FIGURA



Interfaccia utente (namespace *teethcollege.pianidistudi.ui*)

(punti 6)

La classe ***teethcollege.pianidistudi.Program*** (non mostrata nel diagramma UML, ma fornita nello start kit) contiene il **main di partenza dell'intera applicazione**. L'interfaccia utente deve essere simile (non necessariamente identica) agli esempi mostrati di seguito.

**Appena l'applicazione parte**, vengono **caricati dal file Insegnamenti.txt tutti gli insegnamenti** e dal file **PianiDiStudi.txt tutti i piani di studi** (classi ***Program*** + ***MyController***).

La finestra principale dell'applicazione è mostrata in Fig. 3: la combo in alto contiene i soli piano di studi validi. La GUI, espressa dalla **classe *TeethCollegePanel* (da realizzare)**, consente:

- di **scegliere uno studente fra quelli il cui piano di studi è disponibile** (nella combo box in alto, Fig. 3);
- di **visualizzarne gli insegnamenti in dettaglio** (nella text area centrale, Fig. 4);
- di **sostituire un insegnamento a scelta con un altro**, scegliendo sia l'insegnamento da rimuovere che quello da inserire da due combo (in basso), **popolate entrambe con l'elenco di tutti gli insegnamenti attivi**: l'esito dell'operazione è mostrato nel campo di testo in basso, in verde in caso positivo (Fig. 5), in rosso in caso negativo (Fig. 6).

Tutte le operazioni sui dati che la classe ***TeethCollegePanel*** devono essere **effettuate mediante il *Controller*** che è preso in ingresso come argomento del costruttore.

**Nota Bene:** ***TeethCollegePanel*** è un pannello pertanto deve essere derivato da *JPanel*; tale pannello viene integrato in un opportuno frame dalla classe *Program* (fornita nello start kit).

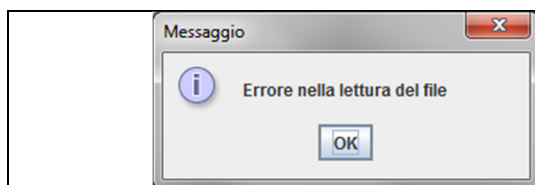


Fig. 1

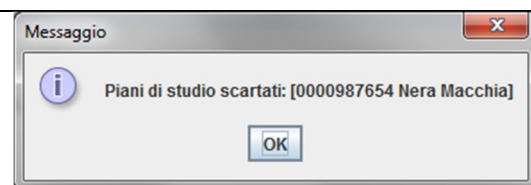


Fig. 2

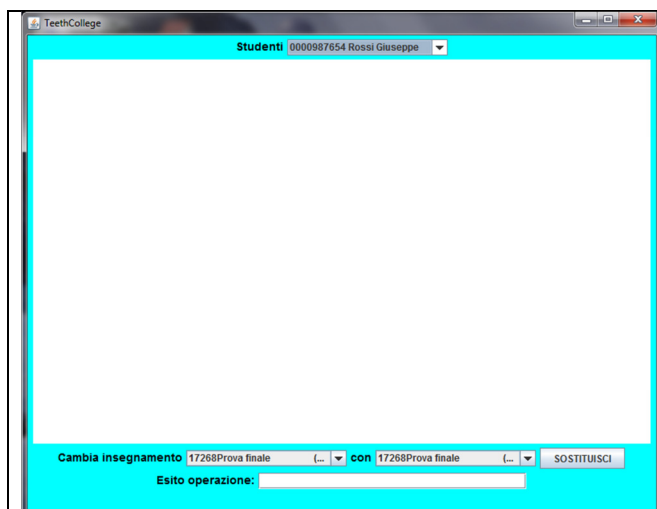


Fig. 3

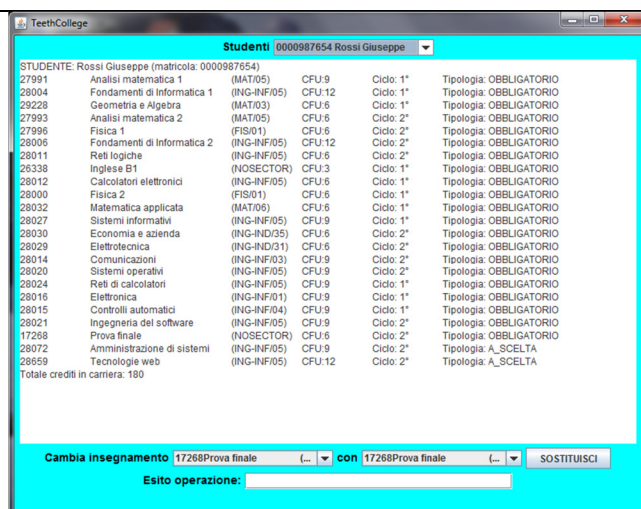


Fig. 4

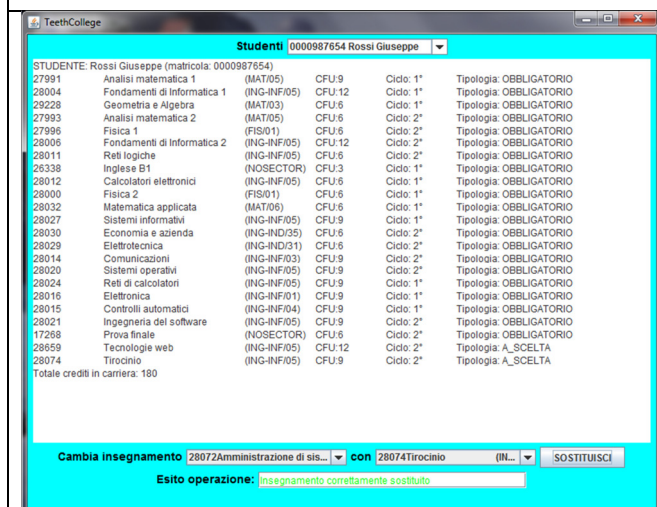


Fig. 5

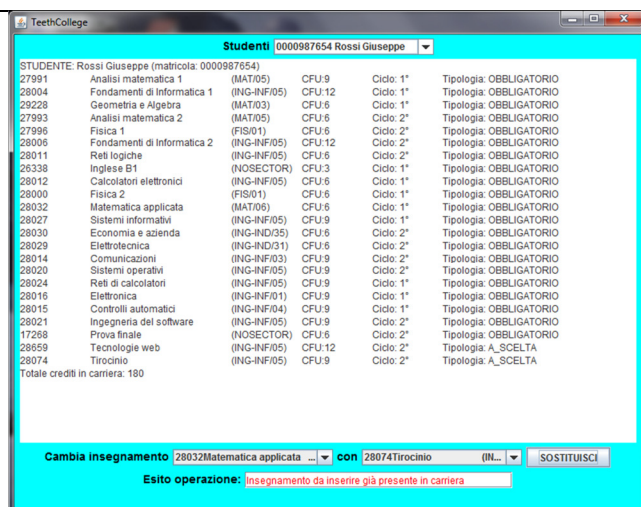


Fig. 6

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