Università degli Studi di Trieste

Corso di ingegneria industriale

Esercitazioni di Fondamenti di Informatica

Giacomo Strangolino

mailto/chat: delleceste@gmail.com

Materiale didattico anche su:

http://www.giacomos.it

(http://www.giacomos.it/didattica/units/2012/)

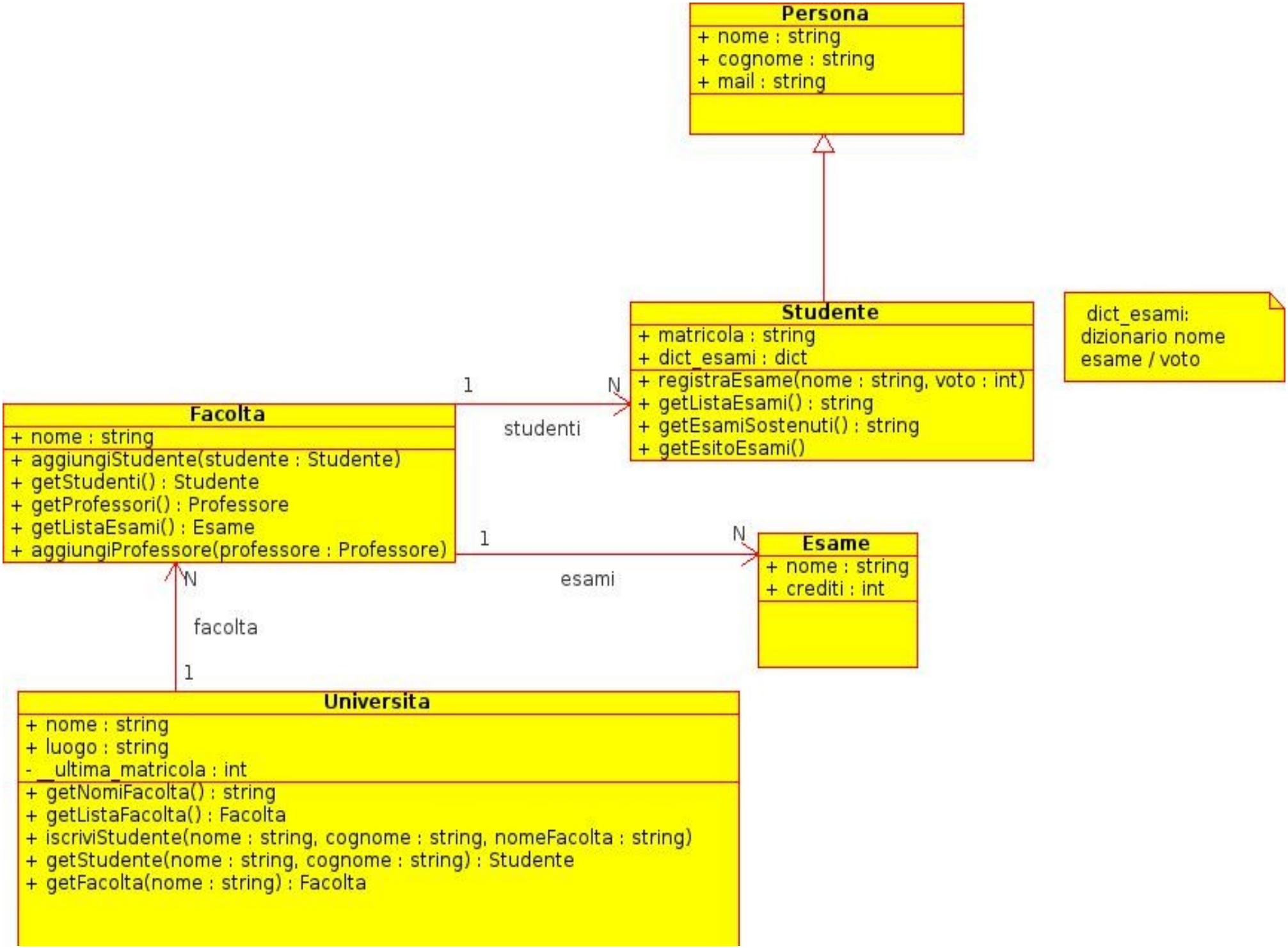
Lezione 9 (25/11/2013)

Classi e Programmazione ad Oggetti – Progetto Università

Si scrivano delle classi atte a modellare un'*università* con i propri *studenti*, *professori*, *esami* e *facoltà*. Deve essere possibile iscrivere studenti a una particolare facoltà, registrare esami, e chiedere alla classe università la lista delle facoltà e a ciascuna facoltà la lista degli studenti e di esami e quindi agli studenti i propri esami con il voto...

Lezione 9 (25/11/2013)
Classi e Programmazione ad Oggetti – Progetto Università

- La classe **Persona** avrà un nome, un cognome e un indirizzo email
- La classe **Studente** *è* una Persona, in più ha una matricola, appartiene a una Facolta e avrà sostenuto una serie di Esami
- La classe **Facolta** è caratterizzata da un nome, da un certo numero di Esame e di Studente. Avrà dei metodi per aggiungere uno studente, ottenere la lista degli studenti, ottenere la lista degli esami.
- La classe **Esame** ha un nome e un numero di crediti.
- La classe **Universita** avrà un nome, un luogo e diverse Facolta.
- Deve essere possibile ottenere la lista delle Facolta, iscrivere uno studente, ottenere uno studente sapendo il suo nome e cognome e ottenere un riferimento ad una Facolta attraverso il suo nome.


Persona
+ nome : string
+ cognome : string
+ mail : string
Studente
+ matricola : string
+ dict_esami : dict
+ registraEsame(nome : string, voto : int)
+ getListaEsami() : string
+ getEsamiSostenuti() : string
+ getEsitoEsami()
dict_esami:
dizionario nome
esame / voto
1
N
studenti
Facolta
+ nome : string
+ aggiungiStudente(studente : Studente)
+ getStudenti() : Studente
+ getProfessori() : Professore
+ getListaEsami() : Esame
+ aggiungiProfessore(professore : Professore)
1
N
esami
Esame
+ nome : string
+ crediti : int
N
facolta
1
Universita
+ nome : string
+ luogo : string
- __ultima_matricola : int
+ getNomiFacolta() : string
+ getListaFacolta() : Facolta
+ iscriviStudente(nome : string, cognome : string, nomeFacolta : string)
+ getStudente(nome : string, cognome : string) : Studente
+ getFacolta(nome : string) : Facolta

Lezione 9 (25/11/2013)
Homework

- Si completi il progetto *università* gestendo anche i professori.

  Ogni professore insegnerà uno o più esami, apparterrà ad una ed una sola facoltà ed avrà uno o al più due assistenti, che saranno rappresentati da una nuova classe derivata da persona.

- Si modifichi il progetto università in modo che carichi gli studenti e gli esami da 2 file: il primo contiene righe del tipo:

  Bruno, Resistenza, Ingegneria

  Federico, Nice, Filosofia

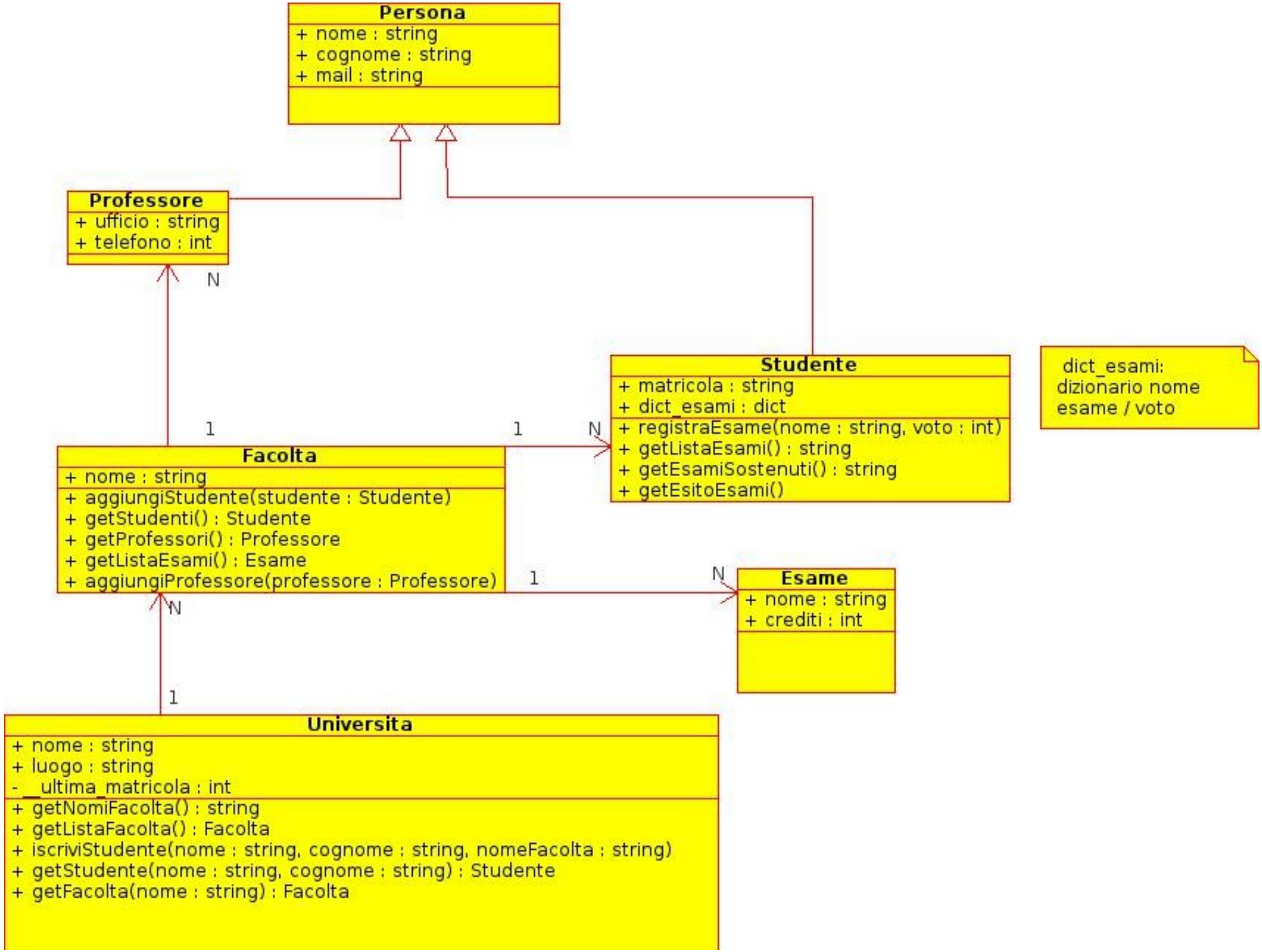
  …

  mentre il secondo:

  Federico, Nice, Storia della Filosofia, 30

  Federico, Nice, Filosofia Teoretica, 28

  Tommaso, D'Aquino, Teologia, 30


Persona
+ nome : string
+ cognome : string
+ mail : string
Professore
+ ufficio : string
+ telefono : int
N
Studente
+ matricola : string
+ dict_esami : dict
+ registraEsame(nome : string, voto : int)
+ getListaEsami() : string
+ getEsamiSostenuti() : string
+ getEsitoEsami()
dict_esami:
dizionario nome
esame / voto
1
1
N
Facolta
+ nome : string
+ aggiungiStudente(studente : Studente)
+ getStudenti() : Studente
+ getProfessori() : Professore
+ getListaEsami() : Esame
+ aggiungiProfessore(professore : Professore)
1
N
Esame
+ nome : string
+ crediti : int
N
1
Universita
+ nome : string
+ luogo : string
- __ultima_matricola : int
+ getNomiFacolta() : string
+ getListaFacolta() : Facolta
+ iscriviStudente(nome : string, cognome : string, nomeFacolta : string)
+ getStudente(nome : string, cognome : string) : Studente
+ getFacolta(nome : string) : Facolta

Homework

Si progetti un sistema per gestire contenitori e oggetti della spesa. ★

- La classe **Contenitore** è una *classe astratta*, con i seguenti *metodi astratti*: tara() e dimensioni(). dimensioni() restituisce una lista (o tupla) di tre float [larghezza, altezza, profondita]. La classe Contenitore implementa i metodi pesoNetto(), che restituisce in un float la somma dei pesi degli oggetti (in grammi o kg, a scelta), e quello analogo pesoLordo(). La classe contiene una lista di oggetti (attributo pubblico) che viene inizializzata nel costruttore, ed essendo pubblica, può essere modificata in seguito.
- **Oggetto** è un'interfaccia, con i metodi astrattti peso(), dimensioni(), nome(). Si restituisca None in ognuno di essi per convenzione.
- La classe **Pomodoro** *estende* la classe **Oggetto** implementandone l'interfaccia. Quando creo un Pomodoro, il costruttore deve accettare come parametri in ingresso il peso e una lista [l, h, p] per le dimensioni. La classe ha un metodo peso() che restituisce il valore dell'attributo peso.

★ Esercizio di un esame orale anno 2010-2011

Homework

- La classe **BorsaDellaSpesa** è un **Contenitore**, il costruttore accetta una lista di **Oggetto** (eventualmente vuota, modificabile a posteriori, pubblica), ha una tara di 10 grammi e dimensioni 50x40x25 (cm).
- Si disegni il diagramma *UML* delle classi e si scriva l'implementazione nel linguaggio *Python*.
- Si decidano liberamente le unità di misura per le dimensioni e i pesi, ricordando che devono essere consistenti per tutte le classi.

- **USO**

- Si deve creare una borsa della spesa e inserire alcuni pomodori.
- Si devono chiedere alla borsa della spesa la tara, il peso netto e il peso lordo.

Lezione 9
Homework – Progetto biblioteca

Si progetti un software (diagramma UML e codice python) per gestire una biblioteca, avente le seguenti classi

- **Articolo**, con gli attributi *collocazione:String, titolo:String, autore:String, tipo:String* (può avere i valori 'libro' e 'cd') e il metodo *durataPrestito(),* che restituisce un intero pari al numero dei giorni (30) del prestito;
- **Cd**, è un **Articolo**, in più ha l'attributo *genere* e la *durataPrestito* restituisce 10 invece di 30. L'attributo *tipo* vale 'CD';
- **Libro**, è un **Articolo**, con l'attributo *genere* in più. L'attributo *tipo* vale 'libro';

Lezione 9 - **Progetto biblioteca**

- **Biblioteca**, con gli attributi *nome* e *luogo* (stringhe) e i seguenti *metodi*:
- getListaPrestiti: restituisce una lista di oggetti Prestito (v. seguito);
- aggiungiStudente(nome, cognome, universita: String);
- aggiungiStudente(nome, cognome);
- getListaClienti(): restituisce la lista degli oggetti Cliente
- getListaArticoli(): restituisce la lista degli Articoli;
- cercaArticolo(titolo, autore): restituisce l'Articolo con quel titolo e autore;
- registraPrestito(titolo, autore, nomeCliente, cognomeCliente, dataPrestito:String): aggiunge un nuovo Prestito associato a un cliente con quel nome e cognome. Si supponga che il cliente debba essere gia` iscritto mediante aggiungiCliente
- aggiungiArticolo(collocazione:String, titolo, autore, genere, tipo ('CD' o 'libro'): aggiunge un nuovo libro o cd a seconda del tipo;
- cercaCliente(nome, cognome): restituisce il cliente con quel nome e cognome, se iscritto.

Lezione 9 - **Progetto biblioteca**

- Classe **Cliente**, con gli attributi nome, cognome e i metodi *bonusGiorniPrestito()* i e *isStudente()*, che restituisce False per un cliente generico. Anche il bonus giorni prestito vale 0 per un cliente generico.
- Classe **Studente***,* è un *Cliente*, ma ha un bonusGiorniPrestito che vale 7 (giorni in più in cui è valido il prestito) e il suo metodo *isStudente()* dovrà restituire il valore booleano opportuno.

  La classe **Studente** ha in più l'attributo *universita:String,* che memorizza il nome dell'universita` a cui lo studente è iscritto.
- Classe **Prestito**, con attributi *cliente*, di tipo *Cliente* e a*rticolo*¸di tipo *Articolo*.

  Infatti, la classe *Prestito associa* un cliente a un articolo. Ha inoltre l'attributo *dataInizioPrestito:String* e il metodo *durataPrestito()*, che restituirà un valore pari alla durata del prestito di ogni articolo generico più un eventuale *bonusGiorniPrestito* che dipende dal tipo di cliente ed è specifico del tipo di cliente (ad esempio uno studente ha un bonus di 7 giorni in più).

Lezione 9 - **Progetto biblioteca**

Popolare quindi la biblioteca con alcuni Libri e CD, e iscrivere qualche cliente normale e qualche studente. Sempre utilizzando i metodi della classe Biblioteca, interrogare la biblioteca circa la lista degli articoli, dei clienti, contare gli studenti, contare i cd e cercare clienti e articoli.