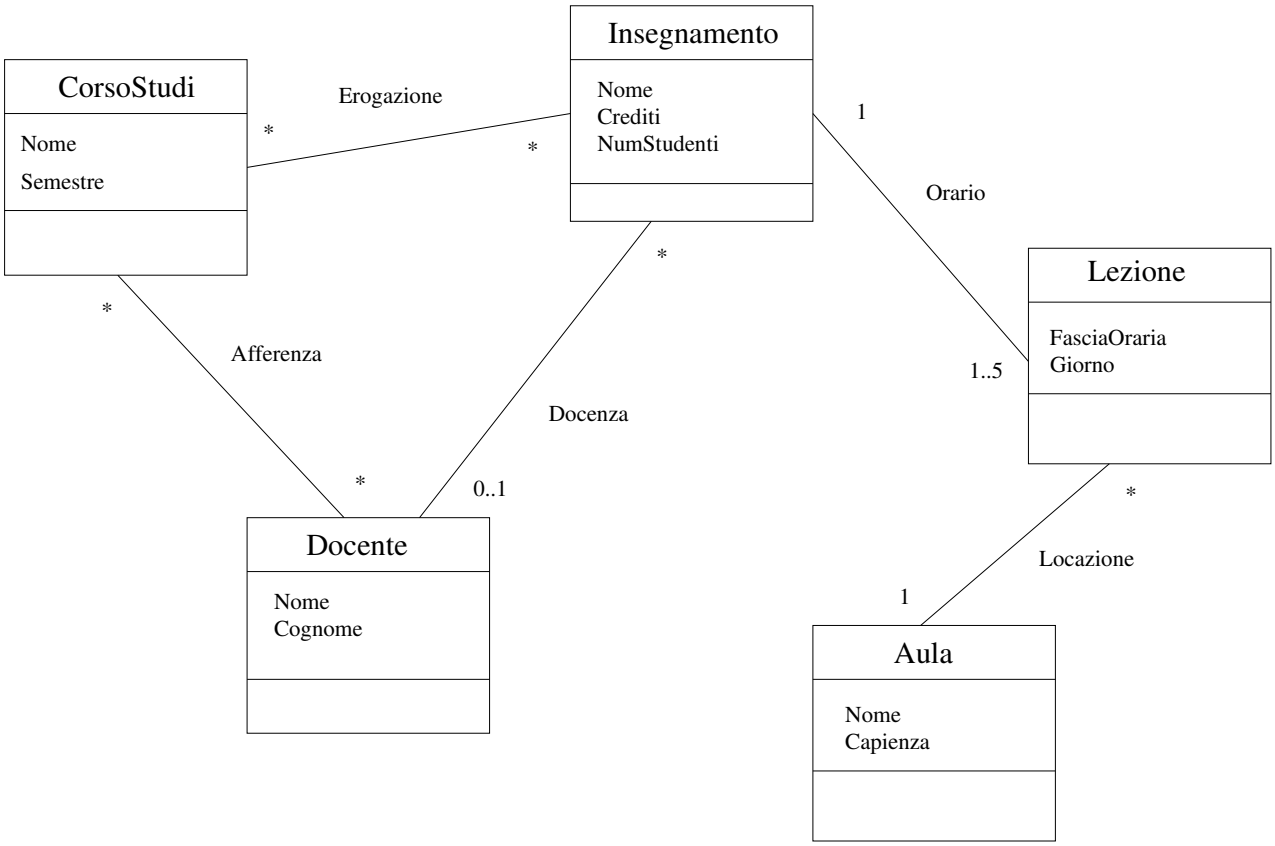
Analisi e Progettazione del Software
Prova scritta del 10 luglio 2017

Si vuole progettare un'applicazione per la gestione del calendario settimanale delle lezioni degli insegnamenti offerti da un corso di studi universitario in uno specifico periodo didattico. L'applicazione è descritta dal seguente diagramma UML (in cui non sono specificate le operazioni).



Nella classe Lezione l'attributo giorno rappresenta il giorno della settimana (Lunedì, Martedì, ...), mentre l'attributo FasciaOraria rappresenta la fascia della giornata codificata come segue: 1 = 8:30–10:30, 2 = 10:30–12:30, ... 5 = 16:30–18:30.

Le operazioni della classe CorsoStudi sono specificate come segue:

InserisciInsegnamento(i : *Insegnamento*)**:** L'insegnamento i viene inserito nel corso di studi.
Precondizioni: non è già presente un insegnamento con lo stesso nome di i.

InserisciDocente(d : *Docente*)**:** Il docente d viene inserito nel corso di studi. Se però è già presente un docente con lo stesso nome e cognome, allora l'operazione non ha effetto.

AttribuisciInsegnamento(i : *Insegnamento*, d : *Docente*)**:** L'insegnamento i viene attribuito al docente d.
Precondizioni: L'insegnamento i e il docente d sono presenti nell'offerta del corso di studi; il corso i non è assegnato ad alcun docente oppure è assegnato ad un docente non del corso di studi.

InserisciLezione(i : *Insegnamento*, l : *Lezione*)**:** La lezione l viene aggiunta alle lezioni dell'insegnamento i.

Precondizioni: L'insegnamento i è presente nell'offerta e non ha già una lezione nello stesso giorno e nella stessa fascia oraria; l'insegnamento i non ha già raggiunto il massimo numero di lezioni.

CreaLezione($i : Insegnamento, a : Aula, f : intero, g : stringa$)**:** Una lezione in aula a nel giorno g e fascia oraria f viene aggiunta alle lezioni dell'insegnamento i.
Precondizioni: L'insegnamento i non ha già una lezione nel giorno g e fascia oraria f; l'insegnamento i non ha già raggiunto il massimo numero di lezioni.

Si considerino già disponibili le classi `Aula` e `Docente` definite come segue.

```
class Docente
{
 public:
  Docente(string n, string c) : nome(n), cognome(c) {}
  string Nome() const { return nome; }
  string Cognome() const { return cognome; }
 private:
  string nome, cognome;
};

class Aula
{
 public:
  Aula(string n, int c) : nome(n) { capienza = c; }
  string Nome() const { return nome; }
  unsigned Capienza() const { return capienza; }
 private:
  string nome;
  unsigned capienza;
};
```

Esercizio 1 (punti 2) Nel diagramma UML si specifichino le operazione della classe Insegnamento e si attribuiscano le responsabilità delle associazioni.

Esercizio 2 (punti 6) Si scrivano le definizioni delle classi C++ per l'applicazione.

Esercizio 3 (punti 12) Si scrivano le definizioni delle funzioni delle classi (fare eventualmente solo quelle che si ritengono più significative).

Esercizio 4 (punti 4) Si scriva l'operatore di output della classe `CorsoStudi`, in modo che stampi anche tutti i dati ad essa collegati.

Esercizio 5 (punti 6) Si scriva una funzione esterna che prenda come parametro un corso di studi e un vettore di puntatori a insegnamenti e calcoli il numero di lezioni in "sovrapposizione" (stesso giorno e stessa fascia) tra due corsi del vettore.