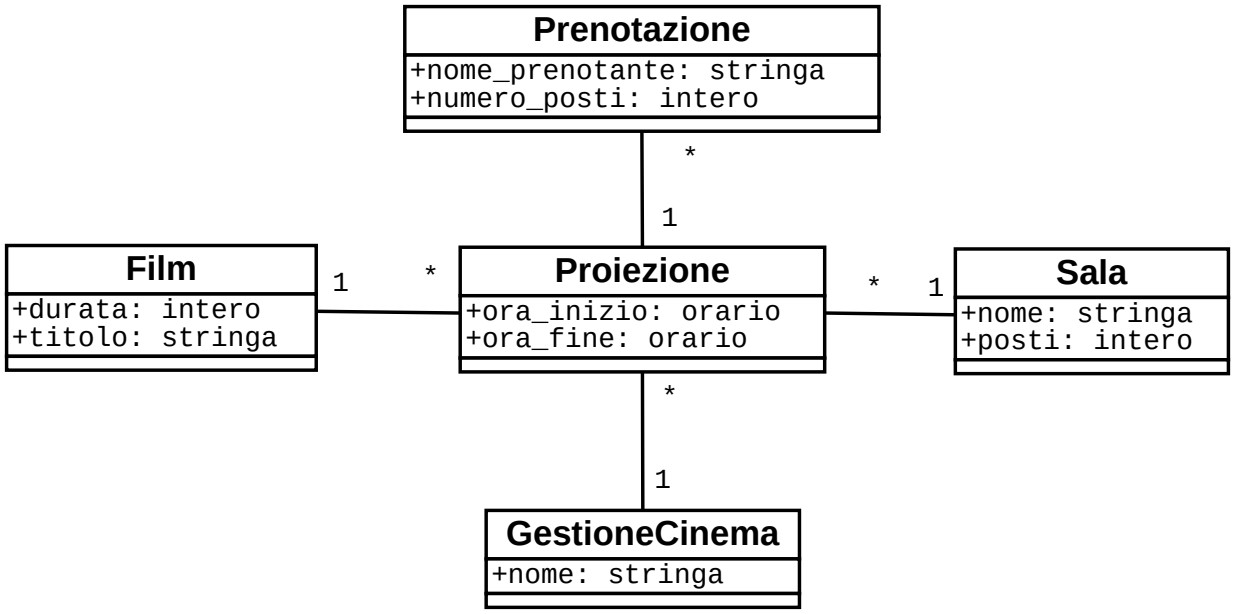
Si vuole progettare una classe per la gestione del sistema di prenotazione per le proiezioni giornaliere di un cinema multisala. Una proiezione è caratterizzata dal film proiettato, la sala in cui è proiettato ed un insieme di prenotazioni ad essa associate.

L'applicazione è descritta dal seguente schema in UML, senza operazioni. Le operazioni della classe GestioneCinema sono specificate di seguito; le classi Sala, Cinema e Prenotazione non hanno operazioni; le semplici operazioni della classe Proiezione sono state omesse.



Le funzioni della classe GestioneCinema sono specificate come segue:

$InserisciProiezione(p : Proiezione)$: p viene aggiunta alla programmazione.
Precondizioni: la proiezione p non è già in programmazione; la sala di p è libera da altre proiezioni per tutto l'orario di p; la proiezione p non ha prenotazioni.

$RimuoviProiezione(p : Proiezione)$: p viene rimossa della programmazione.
Precondizioni: la proiezione p è in programmazione.

$InserisciPrenotazione(pn : Prenotazione, p : Proiezione) : booleano$ se il numero di posti già prenotati per p sommati ai posti di pn non supera la capienza delle sala di p, allora pn viene aggiunta alle prenotazioni di p e viene restituito $true$, altrimenti la prenotazione non viene inserita e viene restituito $false$.
Precondizioni: p è una proiezione già presente in programmazione.

$CambiaOrario(p : Proiezione, i : Orario, f : Orario)$: l'orario di p viene cambiato in $i \div f$ (ora_inizio: i, ora_fine: f)
Precondizioni: p è una proiezione già presente in programmazione; la sala di p è libera per tutto l'orario $i \div f$.

$CambiaSala(p : Proiezione, s : Sala)$: la sala di p viene cambiata in s
Precondizioni: p è una proiezione già presente in programmazione; la sala s ha un numero di posti sufficiente per le prenotazioni correnti di p; la sala s è libera per l'orario di p.

Esercizio 1 (punti 10) Si scrivano le definizioni delle classi C++ `Film`, `Sala`, `Prenotazione`, `Proiezione` e `GestioneCinema`. Si assuma già disponibile la classe `Orario`, in cui alcune definizioni di funzione sono omesse per brevità (ma possono essere assunte già realizzate).

```
class Orario
{ friend bool operator<(const Orario& o1, const Orario& o2);
  friend bool operator<=(const Orario& o1, const Orario& o2);
  friend bool operator==(const Orario& o1, const Orario& o2);
  friend int operator-(const Orario& o1, const Orario& o2);  // differenza in minuti
public:
  Orario(unsigned o, unsigned m);
  unsigned Ore() const { return ore; }
  unsigned Minuti() const { return minuti; }
private:
  unsigned ore, minuti;
};
```

Esercizio 2 (punti 10) Si scrivano le definizioni delle funzioni delle classi gestendo le precondizioni attraverso la funzione `assert()`.

Esercizio 3 (punti 4) Si scriva una funzione esterna che prenda come parametro un oggetto di tipo `GestioneCinema` e restituisca *true* se esiste un film che abbia durata maggiore della lunghezza della sua proiezione, $false$ altrimenti. A questo scopo si utilizzi l'operatore "`-`" della classe `Orario` che restituisce la differenza in minuti tra due orari.

Esercizio 4 (punti 6) Si scriva una funzione esterna che prenda come parametri un oggetto di tipo `GestioneCinema`, un vettore di nomi di persone, ed un intero k e restituisca il numero di persone del vettore che hanno fatto delle prenotazioni nel cinema per un numero complessivo di posti pari o superiore a k.