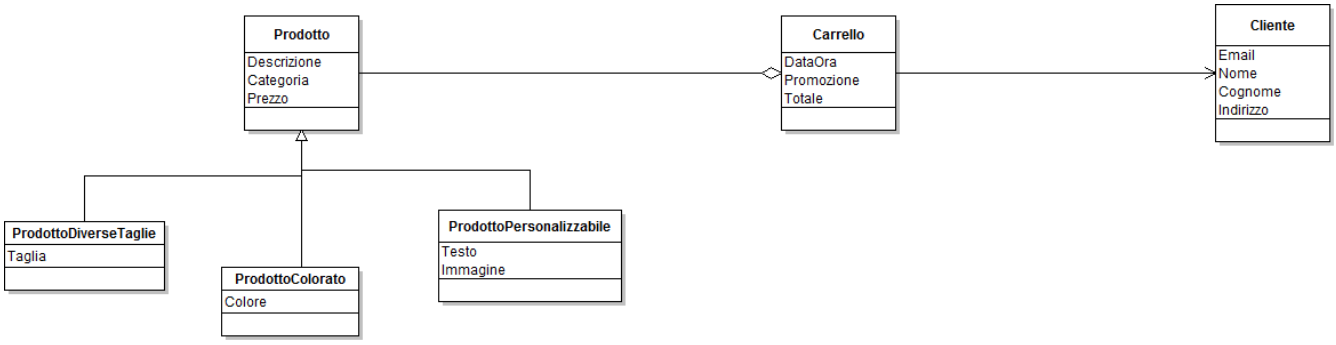
Ingegneria del Software a.a. 2013-14
Prova Scritta del 11 Settembre 2014

Esercizio 1

Si supponga di voler sviluppare un'applicazione Java per il supporto di transazioni di commercio elettronico. Si supponga che le classi di dominio siano quelle raffigurate nel seguente Class Diagram.



a) Si supponga che i dati corrispondenti vadano archiviati su un DBMS relazionale. Presentare gli schemi relazionali[1] che si utilizzerebbe per l'archiviazione di tali dati, in accordo alle tre alternative viste a lezione (vertical, horizontal, filtered mapping) discutendo quale si ritiene più adatto in questo contesto e perché.

Vertical:

PRODOTTO(IdP,Descrizione,Categoria,Prezzo)
PRODOTTODIVERSETAGLIE(IdP^{PRODOTTO}, Taglia)
PRODOTTOCOLORATO(IdP^{PRODOTTO}, Colore)
PRODOTTOPERSONALIZZABILE(IdP^{PRODOTTO}, Testo, Immagine)
CLIENTE(IdCl, *Email*, Nome, Cognome, Indirizzo)
CARRELLO(IdB, IdCl^{CLIENTE}, DataOra, Promozione, Totale)
CONTIENE(IdB^{CARRELLO},IdP^{PRODOTTO})

Horizontal:

PRODOTTODIVERSETAGLIE(IdP, Descrizione,Categoria,Prezzo, Taglia)
PRODOTTOCOLORATO(IdP, Descrizione,Categoria,Prezzo, Colore)
PRODOTTOPERSONALIZZABILE(IdP, Descrizione,Categoria,Prezzo,Testo, Immagine)
CLIENTE(IdCl, *Email*, Nome, Cognome, Indirizzo)
CARRELLO(IdB, IdCl^{CLIENTE}, DataOra, Promozione, Totale)
CONTIENET(IdB^{CARRELLO},IdP^{PRODOTTODIVERSETAGLIE})
CONTIENEC(IdB^{CARRELLO},IdP^{PRODOTTOCOLORATO})
CONTIENET(IdB^{CARRELLO},IdP^{PRODOTTOPERSONALIZZABILE})

Filtered:

PRODOTTO (IdP, Descrizione,Categoria,Prezzo, Taglia$_O$, Colore$_O$, Testo$_O$, Immagine$_O$)
CLIENTE(IdCl, *Email*, Nome, Cognome, Indirizzo)
CARRELLO(IdB, IdCl^{CLIENTE}, DataOra, Promozione, Totale)
CONTIENE(IdB^{CARRELLO},IdP^{PRODOTTO})

Horizontal è da escludersi perché la gerarchia non è totale e non sarebbe in grado di memorizzare prodotti senza taglia, non colorati, non personalizzabili. Il vantaggio di filtered è di avere una tabella sola (schema più compatto, meno join) a discapito di spreco di spazio per valori nulli.

[1] Utilizzando la notazione vista nel corso di Basi di Dati: gli attributi sottolineati sono chiavi primarie, quelli in italico chiavi alternative, gli apici indicano per le chiavi esterne la relazione riferita.

b) Discutere quali degli approcci di gestione della persistenza visti a lezione risultano applicabili in questo contesto e quale è secondo voi preferibile, motivando le affermazioni.

Approcci alla persistenza visti:
forza bruta, DAO, persistence framework (vedi slide per descrizioni e vantaggi/svantaggi)
In un contesto molto semplice come questo forse il preferibile è DAO: non serve "complessità" di persistence framework ma incapsula in strato separato la gestione della persistenza.

c) Supponendo di utilizzare un approccio alla gestione della persistenza basato su framework di persistenza (es. Hibernate) presentare un esempio di mapping, esplicitando dove e come viene utilizzato.

Es. di porzione di mapping Hibernate

```
<!DOCTYPE hibernate-mapping PUBLIC
    "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
    "http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd">
 <hibernate-mapping package="…">
    <class name="Prodotto" table="Prodotto">
   <property name="Categoria" type="string" column="Categoria"/>
   <property name="Descrizione"/>
   <property name="Prezzo"/>
   </class>
</hibernate-mapping>
```

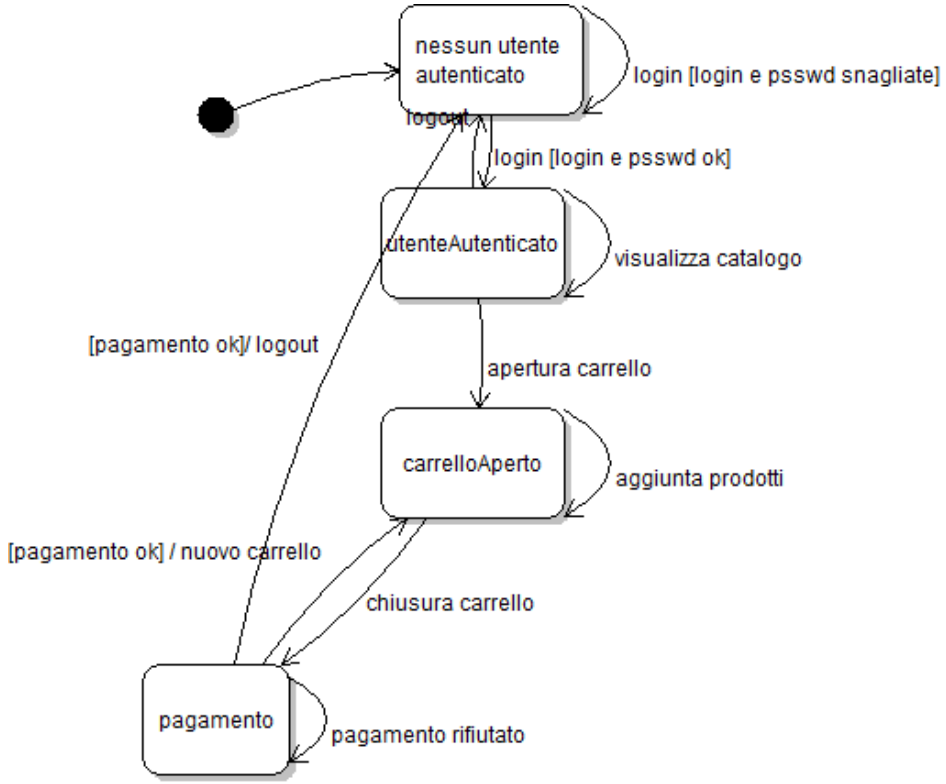
Viene usato come in schema sotto per mettere in corrispondenza tabelle e classi.

d) Considerare ora l'aspetto dinamico della nostra applicazione e in particolare il fatto che per ogni transazione si intende operare nei seguenti stadi:

a. L'applicazione è in attesa di login e password (se nessun utente è collegato si può solo esaminare il catalogo)
b. L'autenticazione ha successo e un utente è collegato. Tale utente ha associato un carrello a cui possono essere aggiunti acquisti fino a quando l'utente chiede di procedere con il pagamento e il carrello viene chiuso.
c. L'applicazione chiede i dati per il pagamento (estremi della carta di credito). Possono solo essere inseriti tali dati, fino a che i dati non vengono confermati.
d. Il pagamento è confermato e l'utente può aprire un nuovo carrello o scollegarsi.

Modellare il comportamento dell'applicazione con un diagramma UML che ritenete più opportuno.

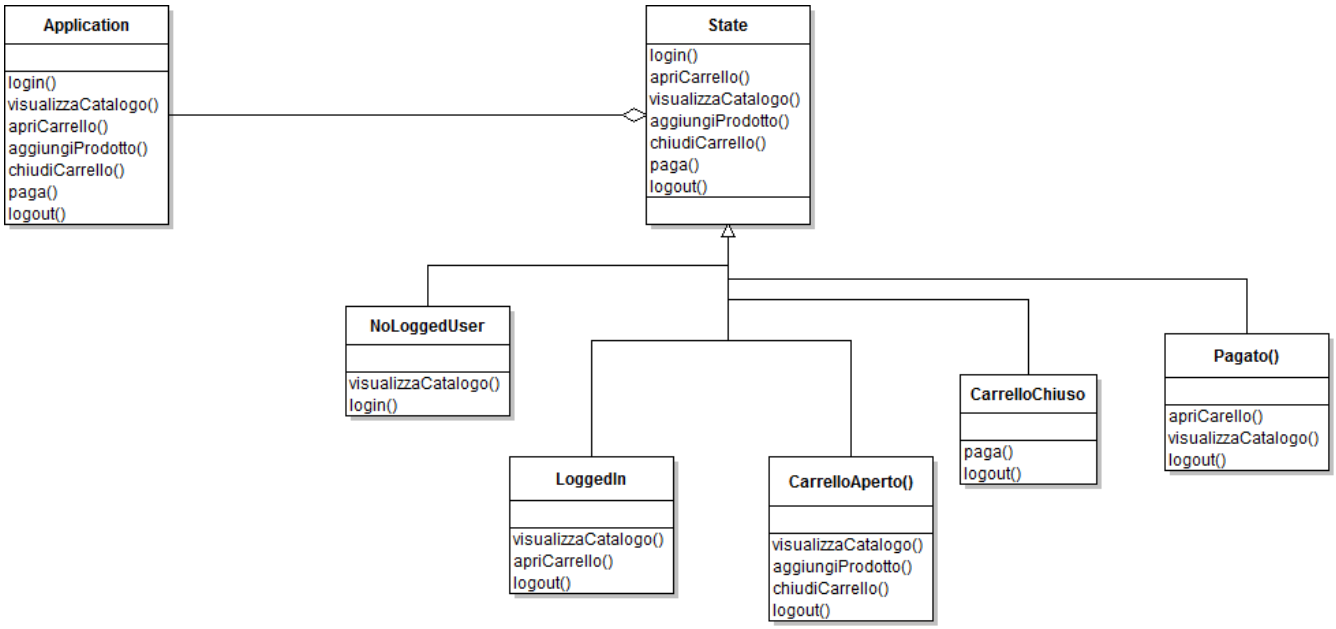
State diagram, vedi bozza seguente (raffinabile, ad esempio inserendo la possibilità di richiesta esplicita dal sistema in vari stati) [mancano dei visualizza carrello che non modificano lo stato e dei logout che "riportano" a nessun utente autenticato].



e) Quale dei design pattern visti a lezione ritenete sia utile per implementare tale comportamento e perché?

State, visto che il comportamento è ben modellato in termini di transizioni tra stati.

f) Instanziare il design pattern al punto d) abbozzando così un class diagram che sia un utile punto di partenza per l'implementazione dell'applicazione.



Esercizio 2

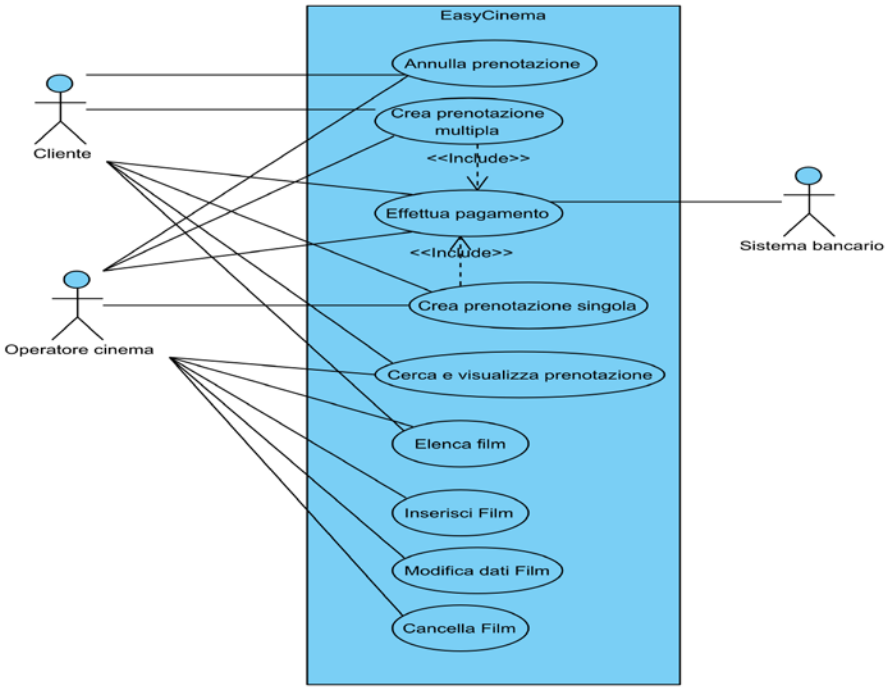
Si supponga di voler progettare un'applicazione per la gestione delle prenotazioni per un Cinema chiamata **EasyCinema**. I film hanno informazioni riguardanti titolo e data dell'evento.

Ogni prenotazione ha un nome di riferimento, riguarda un numero di posti ed è identificata da un codice intero. I posti sono numerati, e ciascun posto è identificato da due interi rappresentanti la fila e la colonna (la sala è rettangolare e i valori partono da 0,0 che rappresenta l'angolo della sala in alto a sinistra).
Il sistema deve consentire di fare le seguenti operazioni:

- **Nuova prenotazione multipla senza preferenze**: Inserisce nel sistema una prenotazione per un numero di posti pari a n. I posti assegnati sono scelti scandendo la sala per file partendo dal posto (0,0) e prendendo i primi n posti liberi. L'operazione restituisce il codice *c* della prenotazione, il quale viene generato automaticamente dal sistema in modo progressivo. Nel caso in cui non vi siano posti liberi a sufficienza, l'operazione restituisce -1 e nessuna prenotazione viene inserita.
- **Nuova prenotazione singola con preferenza**: Inserisce nel sistema una prenotazione per un singolo posto esattamente nella posizione *p* (fila, colonna). L'operazione restituisce il codice *c* della prenotazione, che viene generato come per la prenotazione multipla senza preferenze. Nel caso in cui il posto *p* sia già prenotato, l'operazione restituisce -1 e nessuna prenotazione viene generata.
- **Annulla prenotazione**. Rimuove la prenotazione corrispondente al codice *c*, libera tutti i posti associati e restituisce true. Se invece il codice *c* non esiste, l'operazione non ha effetto e restituisce false.
- **Vedi prenotazione**. Restituisce il vettore di posti associato alla prenotazione di codice *c*. Se il codice *c* non esiste, restituisce un vettore vuoto.

Un uso tipico del sistema prevede che l'operatore del cinema venga contattato telefonicamente[2] per una nuova prenotazione, chieda se il cliente ha preferenze oppure no, chieda il nome al cliente, gli estremi della carta di credito[3] e se l'operazione va a buon fine, comunichi il numero di prenotazione con l'elenco dei posti assegnati e il costo totale. Il sistema può anche essere usato autonomamente dal cliente tramite un interfaccia Web.

a) Rappresentare con uno use case diagram gli attori e gli use case (cioè le funzionalità offerte) di **EasyCinema**

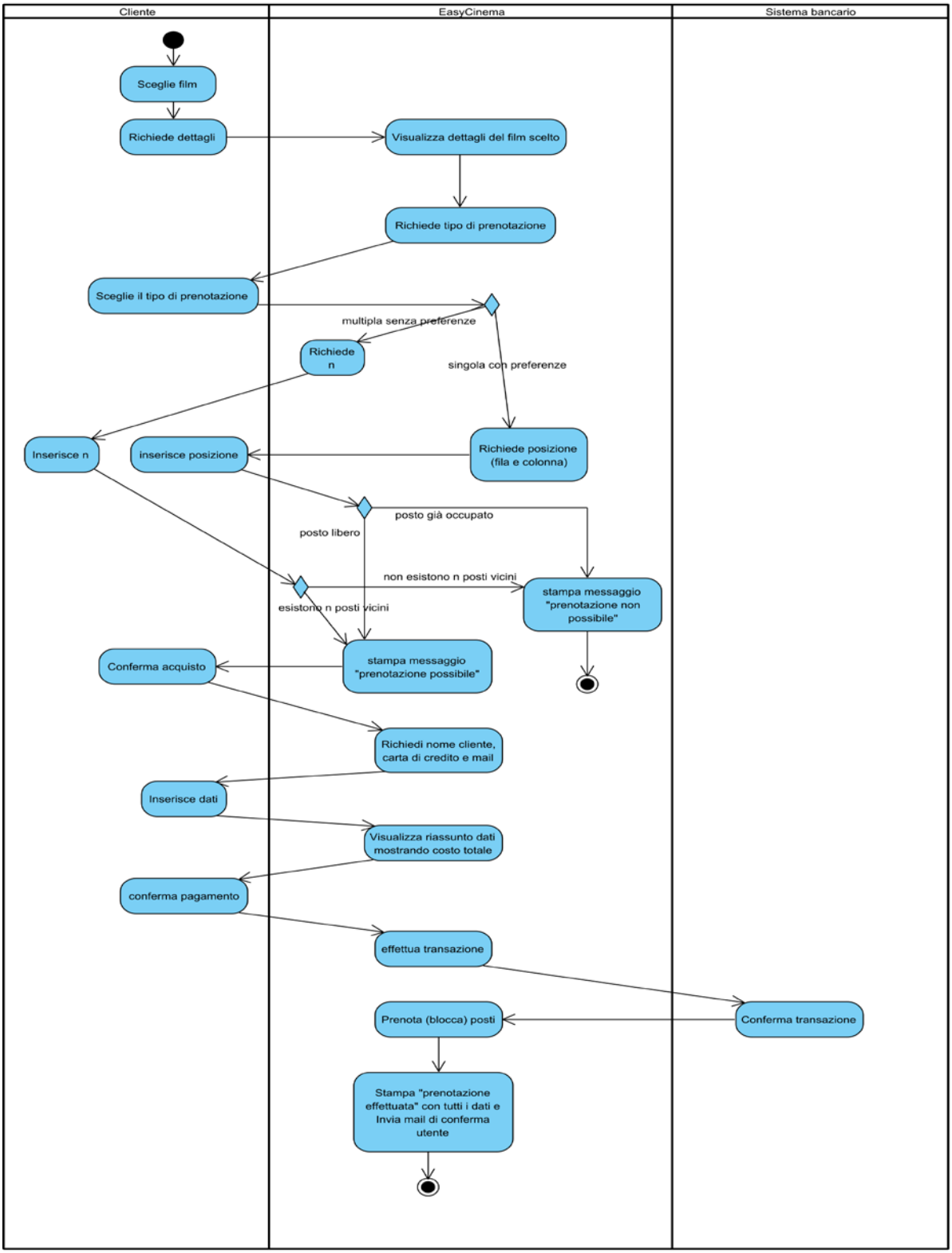


[2] Per semplicità si suppone che il cliente possa effettuare la prenotazione solo in due modi: chiamando l'operatore o attraverso un interfaccia Web (vedi dopo).

[3] Per semplicità si suppone che l'unico modo per effettuare un pagamento è utilizzare la carta di credito.

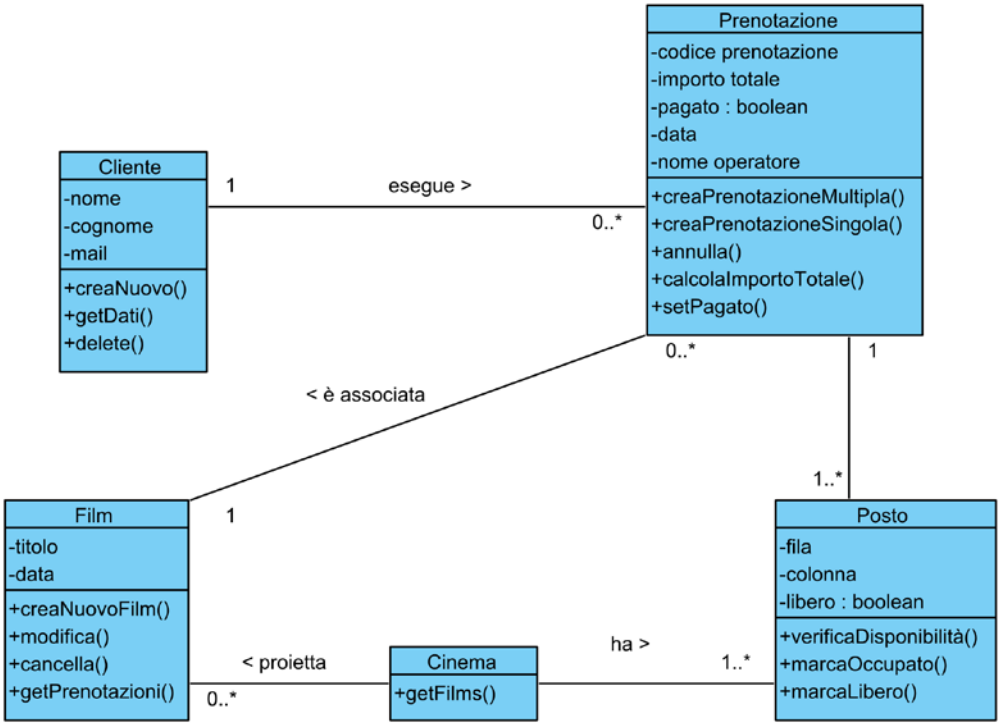
b) Rappresentare con un activity diagram la procedura di prenotazione (sia multipla senza preferenze che singola con preferenze) di un cliente che utilizza il sistema in modo autonomo attraverso l'interfaccia Web (suddividere l'activity diagram in tre swimlanes: cliente, EasyCinema e sistema bancario)

Sono state fatte diverse semplificazioni. Ad esempio le azioni conferma acquisto, pagamento e transazione vanno sempre a buon fine.



c) Rappresentare con un class diagram le relazioni che intercorrono tra le seguenti entità: Cliente, Prenotazione, Film, Posto. Includere attributi e operazioni nel class diagram.

E' stata inserita la classe Cinema per esplicitare la relazione di proietta e il fatto che un cinema ha diversi posti a sedere. Sono state inserite le operazioni e gli attributi più importanti.



d) Implementare utilizzando lo pseudocodice (oppure se preferite direttamente il linguaggio Java) l'operazione "**Nuova prenotazione multipla senza preferenze"** della classe Prenotazione

Implementiamo usando lo pseudocodice.

La sala è rappresentata da una matrice:

```
int sala[][] = new int[numero_file][numero_colonne];
```

con la convenzione che un valore 0 contenuto nella matrice in una posizione (i, j) vuol dire posto libero mentre 1 occupato (già prenotato).

Si suppone che gli n posti liberi per effettuare una prenotazione multipla debbano essere tutti sulla stessa fila e consecutivi (sono accettate soluzioni anche diverse). Cioè non è possibile effettuare una prenotazione multipla che si riferisce a due o più file diverse "andando a capo".

```
Public Int nuovaPrenotazioneMultiplaNopreferenze(n: int, int sala[][]) {

    int Tot;
    boolean fuoriFila;

     for (int i=0; i<numero_file; i++) { \\ scorri le file
      for (int j=0; j<numero_colonne; j++) { \\ scorri le colonne
       Tot=0; fuoriFila=false;
       for (int k=0; k<n; k++) { \\ scorri gli n posti liberi da cercare
         if ((j+k) > numero_colonne) { \\ non ci stanno sulla stessa fila ...
           fuoriFila=true;
```

```
            exit
          }
          Tot = Tot + sala[i][j+k];
        }
        If (Tot == 0 and fuoriFila = false) { \\ n posti liberi sulla fila
         Print (“prenotazione posti dal (”+i+”,”+j+”) al (”+i+”,”+(j+n)+”)”);
         prenota(n, sala[][], i, j) \\ metti gli ‘1’ nei posti prenotati
         return ClasseGeneraCodicePrenotazione.generaCodice()
        }
      } \\ for colonne
    }  \\ for file
   return -1; \\ no posti liberi a sufficienza
}
```

e) Mostrare una possibile architettura software per l’applicazione **EasyCinema** utilizzando un diagramma UML che ritenete opportuno

Si utilizza un component diagram. La segnatura delle interfacce fornite dalle componenti software non è esplicitata ma è facilmente desumibile a partire dal class diagram dato nella risposta c). I dati dei clienti si è deciso di non memorizzarli in modo persistente.