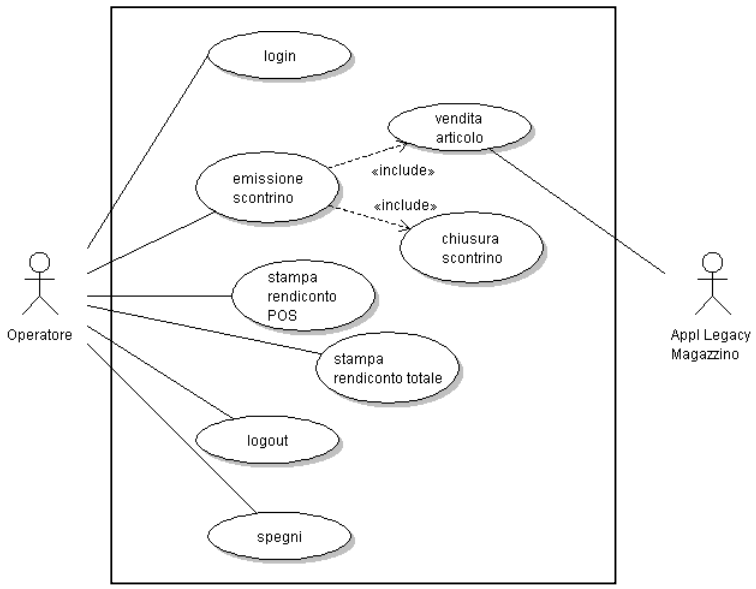
Ingegneria del Software a.a. 2011-12
Prova Scritta del 16 gennaio 2012
Soluzione

Esercizio 1

a) Operatore/utente del POS (chi usa la tastiera) – attore primario
Applicazione (legacy) per gestione magazzino – attore secondario

b) Di seguito il diagramma dei casi d'uso, i casi d'uso identificati sono quelli indicati nel diagramma.



Va bene anche con EmissioneScontrino senza esplicitare VenditaArticolo e ChiusuraScontrino, o inserendo un caso di'uso StampaRendiconto che poi viene specializzato in StampaRendicontoPOS e StampaRendicontoTotale. Per la stampa dei rendiconti si può prevedere o meno l'interazione con l'applicazione legacy.

È importante **identificare i confini** (boundary) del sistema ed **essere coerenti** tra le varie risposte. In particolare, nel testo si specificava che l'interfaccia del sistema era costituita da tastiera usata dall'operatore e applicazione legacy tutto il resto era parte del sistema. Sono state considerate valide anche soluzioni che avevano una visione più ristretta del sistema e indicavano come attori secondari la tastiera, il lettore di codici a barre e la stampante, purché il diagramma dei casi d'uso e la specifica del caso d'uso in c) fossero coerenti.

c) Specificare il caso d'uso "Emissione di uno scontrino", che prevede la registrazione della vendita di ogni articolo nello scontrino e la chiusura dello scontrino, con calcolo e stampa del totale. Le informazioni sulla descrizione e sul prezzo di vendita, dato il codice a barre del prodotto, vengono fornite dall'applicazione legacy, che viene anche utilizzata per aggiornare la disponibilità a magazzino del prodotto.

La bozza di specifica proposta è fin troppo dettagliata rispetto a quanto richiesto. In particolare non era richiesto di gestire l'annullamento. Era possibile consentire l'inserimento della quantità associata ad un certo codice a barre via tastiera. Era possibile prevedere altre extension.

EMISSIONE SCONTRINO
Level: User-Goal
Primary Actor: Operatore POS

Main Success Scenario:

1. Per ogni articolo da includere nello scontrino
   Vendita articolo
2. Chiusura scontrino

Extension:

2a. L'emissione dello scontrino viene annullata

Nota: su questo è possibile seguire diverse strade, non è specificato nel testo – E' possibile annullare esplicitamente? Time-out dalla scansione dell'ultimo articolo senza chiudere lo scontrino? (non era richiesto di dettagliarlo)

2a.1 Per ogni articolo incluso
annulla vendita articolo (non la dettagliamo, non era richiesto)

VENDITA ARTICOLO

Level: User-Goal

Primary Actor: Operatore POS

Main Success Scenario:

1. L'operatore utilizza il lettore di codici a barre per rilevare il codice di ogni articolo acquistato.
2. Il POS riceve il codice dell'articolo dal lettore e lo invia all'applicazione legacy che restituisce il prezzo e la descrizione dell'articolo e decrementa di 1 la disponibilità a magazzino di tale prodotto.
3. Il POS stampa una riga dello scontrino con la descrizione dell'articolo ed il prezzo.
4. Lo use case termina con successo.

Extension:

1a. Il lettore non riconosce il codice
1a.1 L'operatore inserisce il codice dell'articolo attraverso la tastiera del POS

2a. L'applicazione restituisce un errore di codice non esistente
2a.1 Si torna a 1.

CHIUSURA SCONTRINO

Level: User-Goal

Primary Actor: Operatore POS

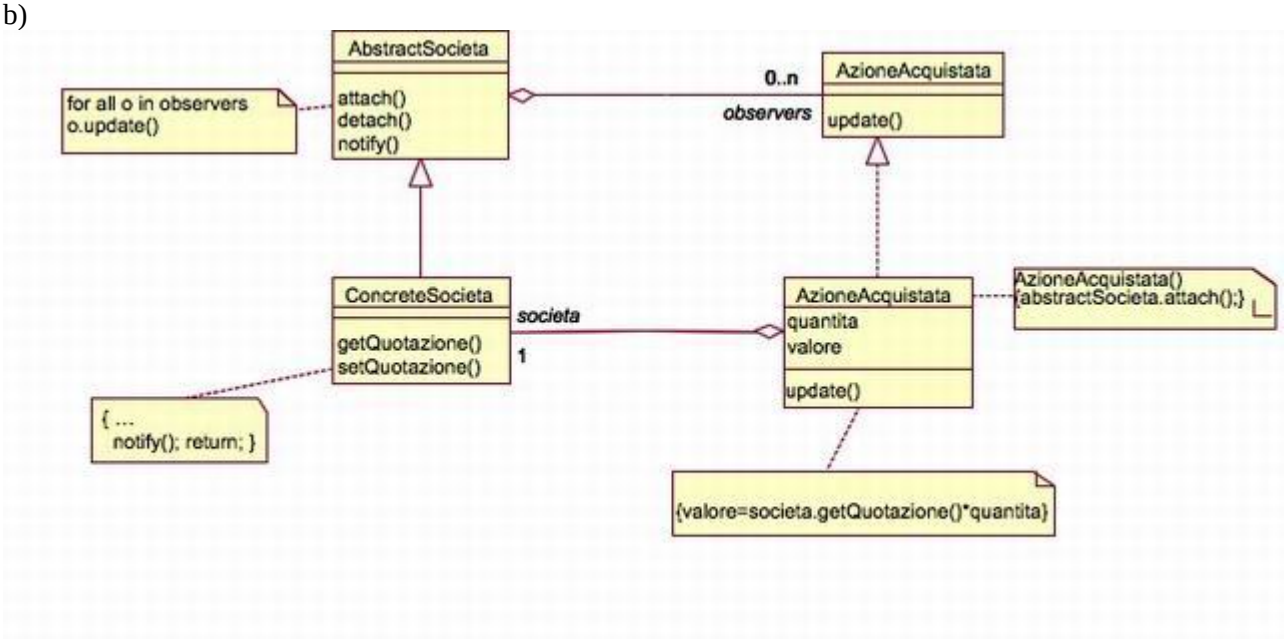
Main Success Scenario:

1. L'operatore preme il tasto di chiusura dello scontrino.
2. Il POS stampa nello scontrino una riga con il totale dello scontrino e la data.
3. Lo use case termina con successo.

Esercizio 2

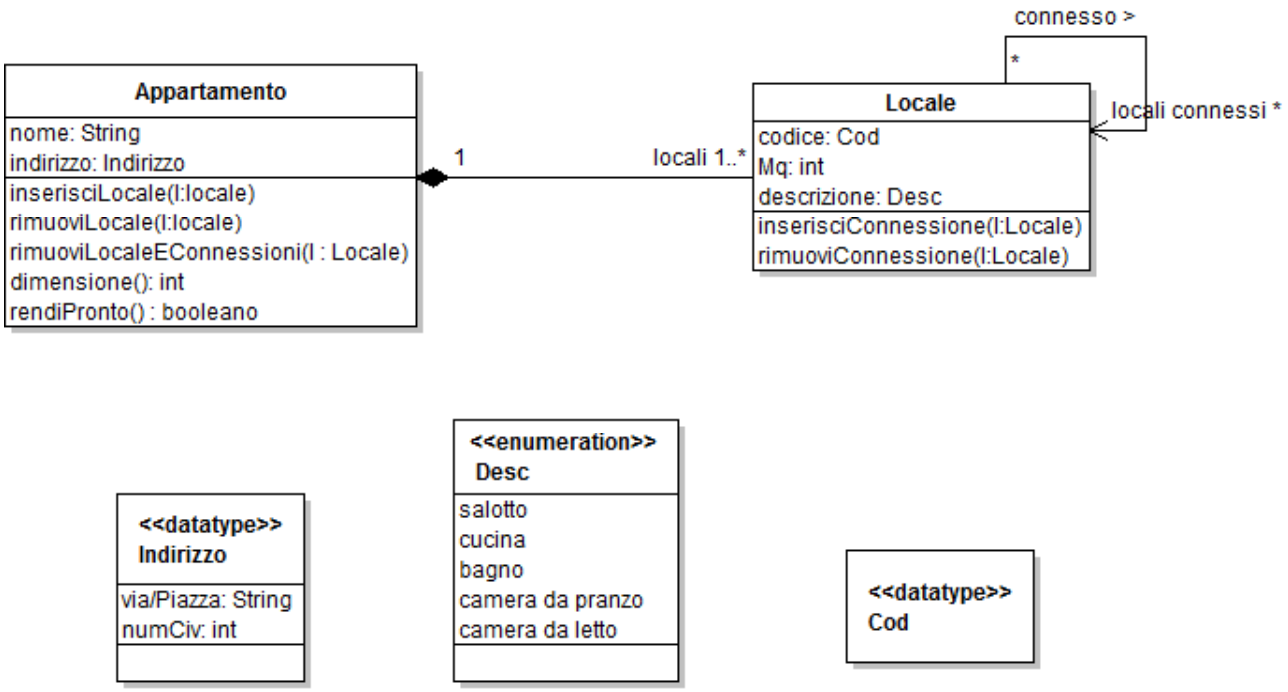
a) Observer

b)



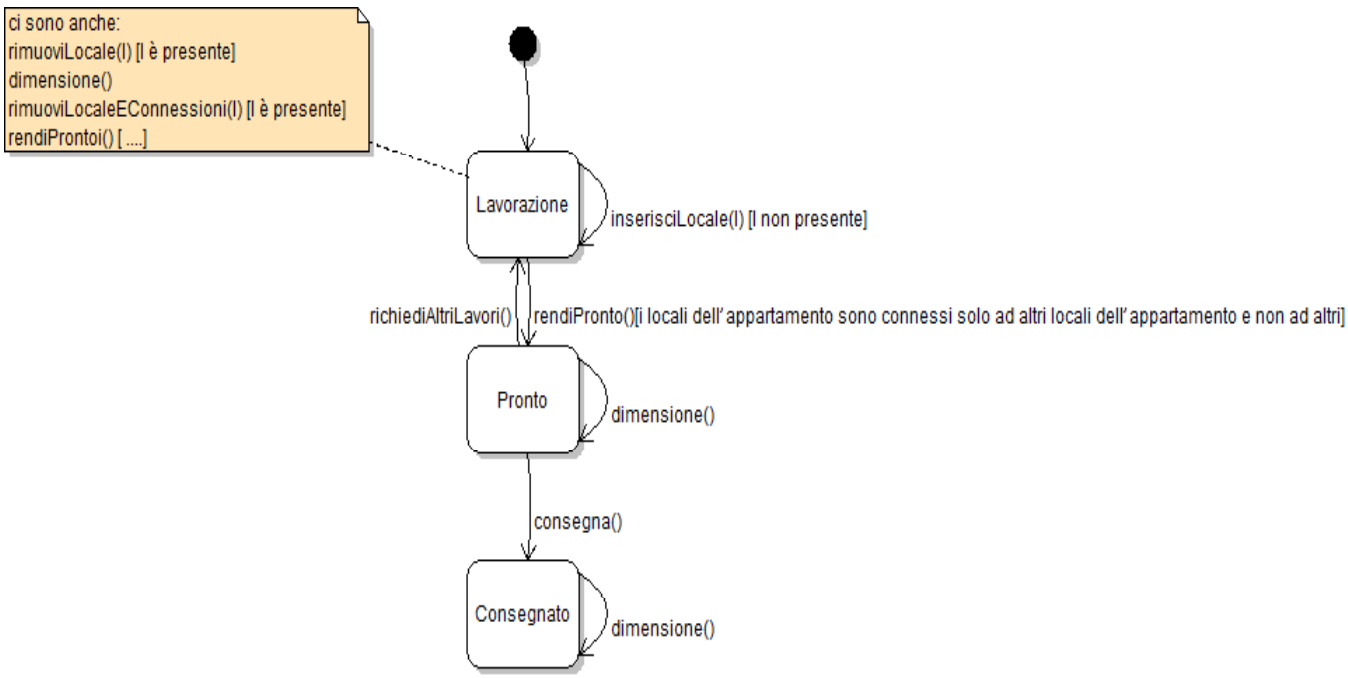
Esercizio 3

a)



b)

Data la povertà dello strumento Violet viene presentata una soluzione molto semplificata. La nota sostituisce il fatto che esistono altri "autoanelli" nello stato Lavorazione (non si riescono ad esprimere con Violet)



c)

Una possibile soluzione usando lo pseudocodice.
vediLocale(i) restituisce il locale i-esimo.
numeroLocali() restituisce il numero di locali dell'appartamneto.

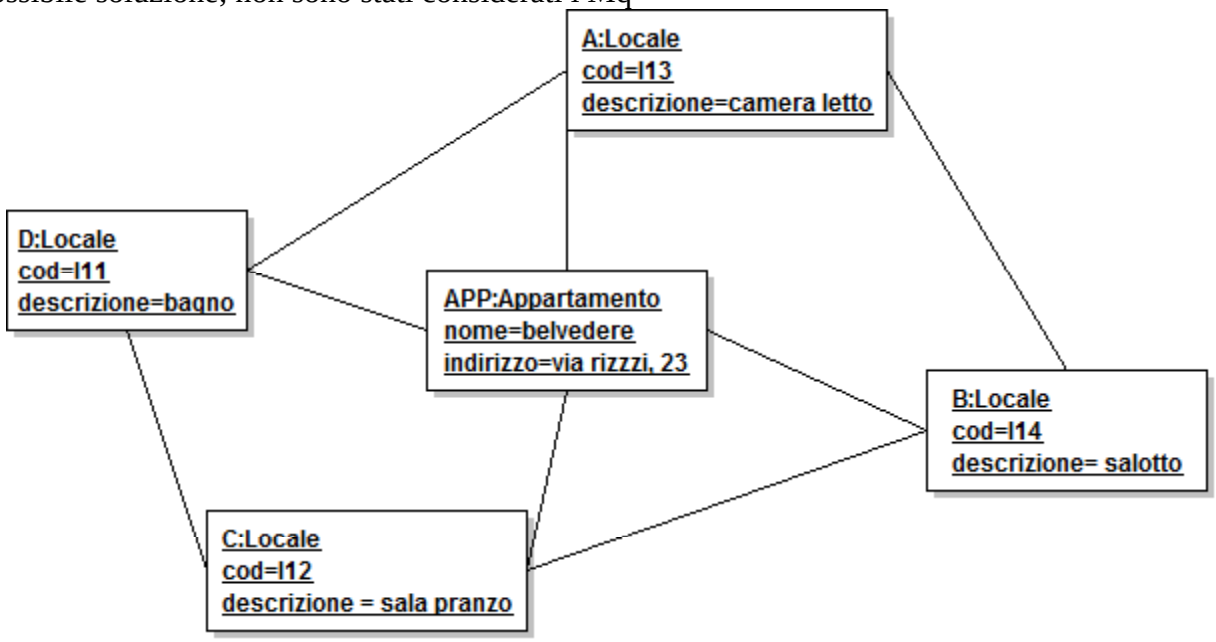
```
boolean VerificaAbitabilita() {
        // verifica che sia presente esattamente una cucina e almeno un bagno
        bagni = 0, cucine = 0;
        for (i = 0; i < this.numeroLocali(); i++) {
                if (this.vediLocale(i).descrizione == "bagno")
                        bagni++;
                else if (this.vediLocale(i).descrizione == "cucina")
                        cucine++;
        }
        return (cucine == 1 && bagni >= 1);
}
```

d)

Context Locale
Inv: self.localiconnessi --> forAll(l:locale | l.appartamento = self.appartamento)

e)

Una possibile soluzione; non sono stati considerati i Mq



Esercizio 4

a)

Si è scelta la strategia "equivalence partitioning".

Primo step: si dividono gli input in legali e non legali.

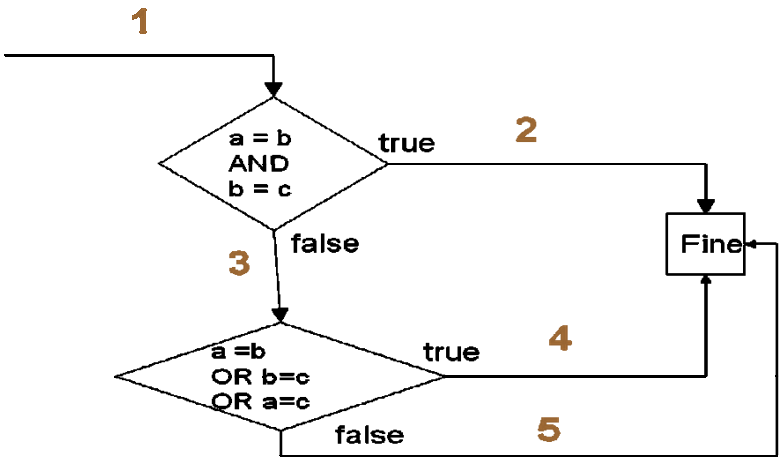
Legali: (a sinistra di --> c'e' l'input mentre a destra c'e' l'output)
a=1, b=1, c=1 --> 1
a=1, b=1, c=3 -->2
a=1, b=2, c=3-->3

non legali: (negativi e somma di due lati < del terzo)
a=-1, b=1, c=1 -->0
a=1, b=-1, c=1 -->0
a=1, b=1, c=-1 -->0
a=100, b=1, c=1 -->0
a=1, b=100, c=1 -->0
a=1, b=1, c=100 -->0

b)

L'implementazione non rispetta la specifica. Questo è chiaro quando si esegue l'implementazione sugli input non legali (si ottiene 1 al posto do 0)

c)



CC is the **number of decision points** in the method's control flow graph **incremented by one. Quindi in questo caso CC = 3**

d)

a=1, b=1, c=1 --> 1 cammino 1,2
a=1, b=1, c=3 -->2 cammino 1,3,4
a=1, b=2, c=3-->3 cammino 1,3,5