Ingegneria del Software (6 crediti) a.a. 2011-12
Prova Scritta del 16 gennaio 2012

Tempo a disposizione: 2 ore
Svolgere gli esercizi 1+2 e 3+4 su fogli protocollo separati

Esercizio 1

Un negozio, dotato di un'applicazione "legacy" per la gestione del magazzino, decide di dotarsi di un sistema di registratori di cassa "intelligenti" (POS) collegati ad un calcolatore dislocato nel negozio. Ogni POS è collegato con un dispositivo per la stampa degli scontrini, con un lettore di codici a barre e con una tastiera. Il dispositivo per la lettura dei codici a barre invia il codice letto (stringa di caratteri) al POS ogni volta che viene utilizzato. La tastiera del POS ha, oltre ai tasti numerici, sei tasti con cui si possono attivare diverse funzioni:

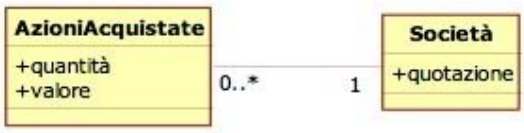
- *login* / *logout* / spegnimento del POS (tre tasti)
- chiusura di uno scontrino
- due tasti "programmabili" che si intendono usare per:
  - stampa del totale cumulativo di tutti gli scontrini emessi dal POS
  - stampa del totale cumulativo di tutti gli scontrini emessi da tutti i POS del negozio.

L'applicazione legacy esistente fornisce funzionalità di gestione della quantità disponibile e del prezzo di vendita per ogni prodotto venduto nel negozio.
Assumendo che i confini del sistema siano costituiti dall'applicazione legacy per la gestione del magazzino e dall'interfaccia utente del POS (tastiera):

a) Identificare gli attori, distinguendo in attori primari e secondari.
b) Identificare i casi d'uso del sistema e produrre un diagramma dei casi d'uso.
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.

Esercizio 2

Nell'ambito della progettazione di un sistema software che includa informazioni relative ad azioni di società quotate in borsa, una cui porzione di diagramma delle classi nella prospettiva concettuale è mostrata di fianco, si vuole gestire anche un attributo **`valore`** nella classe **`AzioniAcquistate`** che misuri il valore istantaneo di un insieme di azioni acquistate.



L'attributo è calcolato come:
`AzioniAcquistate.valore = Società.quotazione * AzioniAcquistate.quantità.`

Allorché il sistema aggiorna il valore della quotazione delle azioni di una società (ovvero l'attributo **`Società.quotazione`**), si vuole che gli attributi **`valore`** delle istanze delle classi **`AzioniAcquistate`** relativi a tale società siano automaticamente aggiornati.

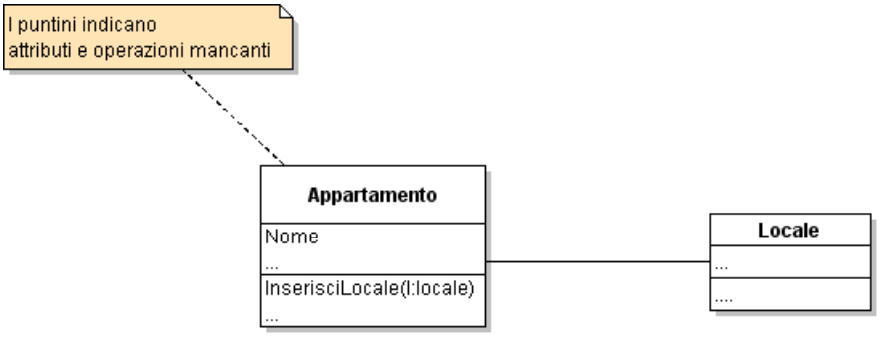
a) Identificare un Design Pattern che potrebbe essere istanziato per proporre una soluzione al problema precedente.
b) Produrre un diagramma delle classi che istanzi il Design Pattern individuato sul problema in esame, mettendo in corrispondenza le classi dell'applicazione con quelle del Design Pattern originario.

Esercizio 3

L'applicazione di interesse riguarda la progettazione di appartamenti. Di ogni appartamento interessano l'indirizzo, il nome, i locali che lo compongono e come questi sono connessi tra loro (si noti che se il locale A e connesso con il locale B, allora anche B e connesso con A). Di ciascun locale interessano il codice identificativo (di 3 caratteri), i metri quadri ed una descrizione testuale, tipo salotto, bagno, cucina, etc.
Un appartamento è inizialmente in lavorazione, poi alla fine dei lavori diviene pronto per la consegna; a questo punto possono essere richiesti ulteriori lavori (e quindi torna ad essere in preparazione) o può essere consegnato. Solo quando un appartamento è in lavorazione si possono aggiungere ed eliminare locali da esso.
Le operazioni della classe Appartamento sono specificate come segue:

- **inserisciLocale**. Il locale l viene inserito nell'appartamento. Precondizione: Il locale l non è presente nell'appartamento (cioè non è presente un locale con lo stesso codice). L'appartamento è in lavorazione.
- **rimuoviLocale.** Il locale l viene eliminato nell'appartamento. Precondizione: Il locale l è presente nell'appartamento. L'appartamento è in lavorazione.
- **rimuoviLocaleEConnessioni**. Il locale l viene eliminato nell'appartamento e vengono rimosse tutte le sue connessioni (ma non i locali connessi) a locali presenti nell'appartamento. Precondizione: Il locale l è presente nell'appartamento. L'appartamento è in lavorazione.
- **dimensione**. Restituisce la dimensione totale dell'appartamento ottenuta come somma dei metri quadri dei suoi locali.
- **rendiPronto**. Se i locali dell'appartamento sono connessi solo ad altri locali dell'appartamento e non ad altri, l'appartamento passa nello stato di pronto e viene restituito true; altrimenti lo stato non cambia e viene restituito false. Precondizione: L'appartamento è in lavorazione.

a) Completare il seguente class diagram (aggiungere attributi, operazioni, associazioni, molteplicità, …) considerando la descrizione data.



b) Produrre il diagramma degli stati UML per la classe Appartamento. Può essere necessario aggiungere delle operazioni.

c) Si aggiunga alla classe Appartamento l'operazione VerificaAbitabilita() che restituisce true se e solo se nell'appartamento è presente esattamente una cucina e almeno un bagno e false altrimenti. Descrivere il behaviour di VerificaAbitabilita() mediante un activity diagram o pseudocodice (tipo Java).

d) **Facoltativo.** Si esprima in OCL il seguente vincolo: "un locale può essere connesso solo a locali dello stesso appartamento".

Esercizio 4

Si consideri la specifica del seguente metodo:

```
static int triangolo (int a, int b, int c) {
 // REQUIRE: a>0 and b>0 and c>0
 // EFFECT: restituisce:
 // 1 se i tre lati definiscono un triangolo equilatero,
 // 2 se definiscono un triangolo isoscele,
 // 3 se definiscono un triangolo scaleno,
 // 0 se a, b, e c non possono essere lati di un triangolo oppure se i tre
 // lati non formano un triangolo
      }
```

a) Si definisca un insieme di casi di test in base a una strategia di tipo "black box" (cioè funzionale), motivando sinteticamente ciascuna scelta.

b) Applicare i casi di test prodotti per **a)** alla seguente implementazione del metodo triangolo. L'implementazione soddisfa la specifica data? Commentare brevemente i risultati ottenuti.

```
static int triangolo (int a, int b, int c) {
      if ((a==b) && (b==c)) return 1;
      if ((a==b) || (b==c) || (a==c)) return 2;
      else return 3;
}
```

c) Disegnare il CFG (Control Flow Graph) e calcolare la complessità ciclomatica del metodo triangolo così come implementato in **b)**. Commentare come è stato effettuato il calcolo.

d) Si definisca un insieme di casi di test per il metodo triangolo così come implementato in **b)** affinché sia soddisfatto il criterio di branch coverage. Far vedere sul CFG i cammini intrapresi dall'esecuzione di triangolo dai casi di test scelti.