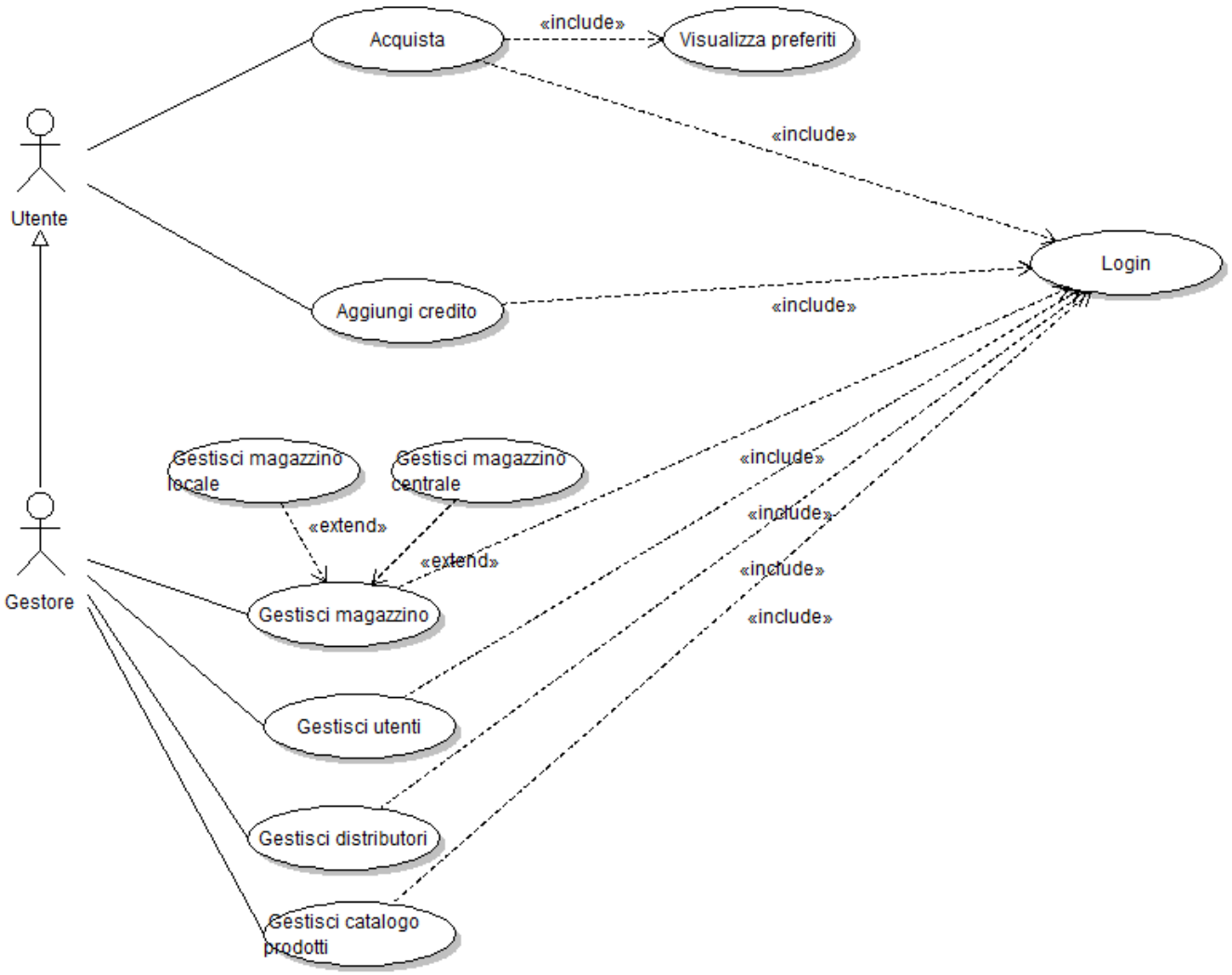
Ingegneria del Software (6 crediti) a.a. 2011-12
Prova Scritta del 13 luglio 2012 – Bozza di soluzione

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

Esercizio 1



Caso d'uso Acquista
Breve
Descrizione
L'utente si autentica e seleziona uno o piu prodotti disponibili
in locale. Il costo è scalato dal credito.
Attori Primari Utente.
Precondizioni Nessuna.
Sequenza principale degli eventi
1. Login.
2. Visualizza preferiti

3. Utente effettua scelta prodotto
4. Aggiorna credito disponibile, disponibilità prodotto, preferenze utente
5. Consegna prodotto

Postcondizioni Utente servito. Disponibilità prodotto, credito disponibile e lista preferenze aggiornati

Extension

1a. Autenticazione fallita.
    1a.1 Torna a 1.
3a. Credito insufficiente.
    3a.1 Aggiungi credito
3b. Prodotto non disponibile localmente
    3b.1 Mostra altri distributori dove è disponibile
    3b.2 Torna a 3
3c. Non viene effettuata scelta entro timeout
    3c.1 Torna a 1.

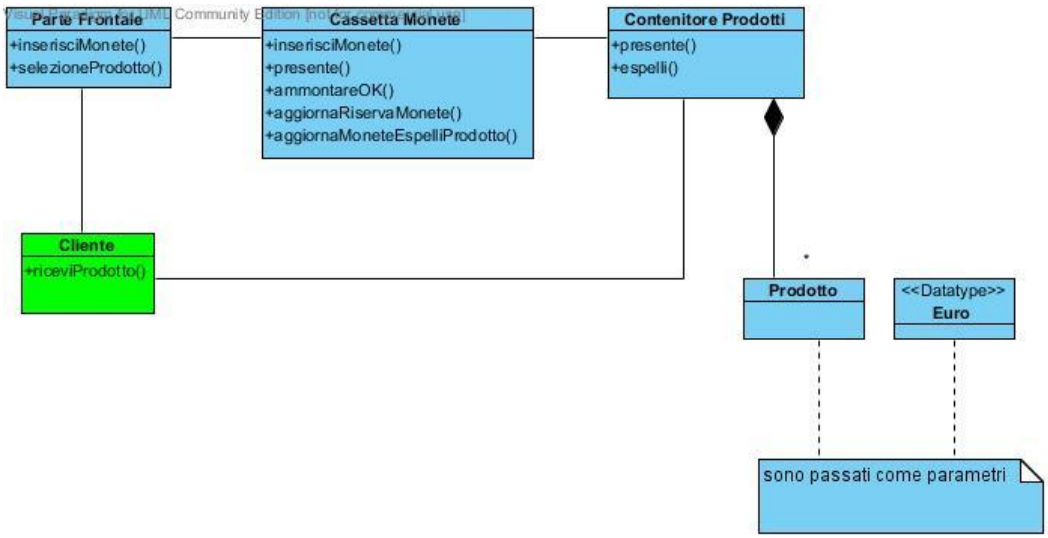
Esercizio 2

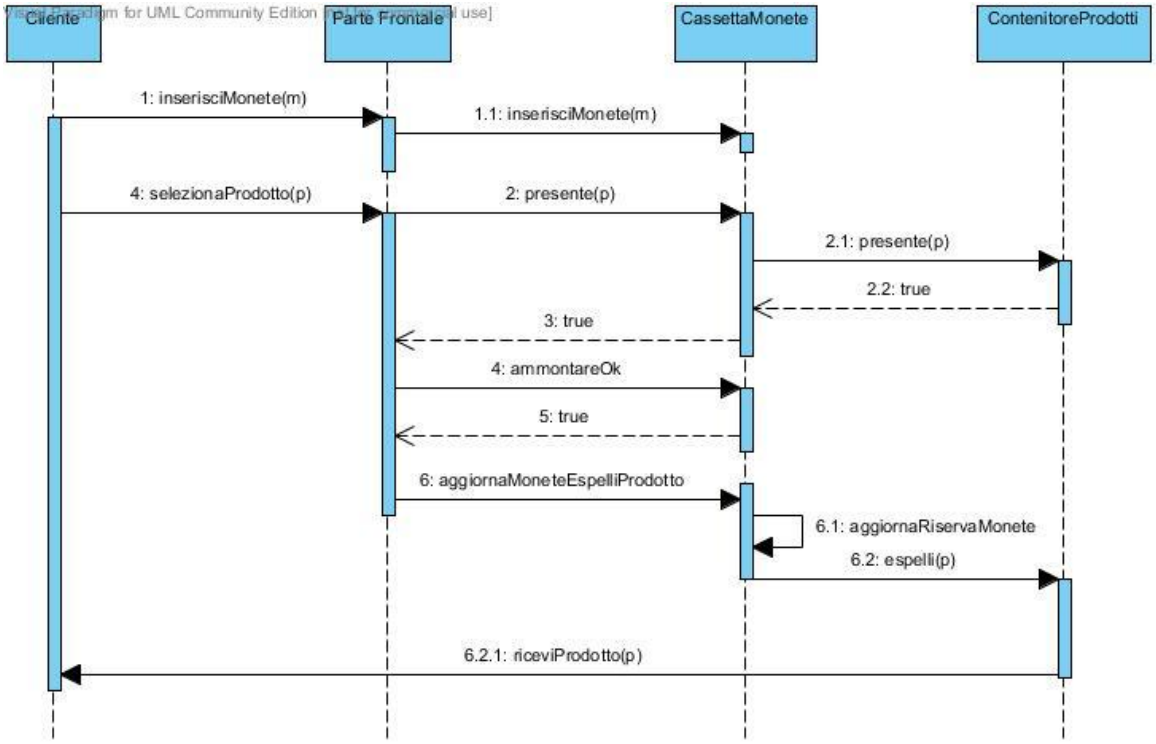
1. Composite
    a. Shape - Component
    b. Circle, Trinagle, Square - Leaf
    c. Assemnly – Composite
2. Visitor
   Con VisitGetArea, VisitGetColor e VisitGetDominantColor

3. Aumenta coupling (più scambi di messaggi tra oggetti per realizzare operazioni), Aumenta coesione (le parti di calcolo dell'area, del colore dominante sono racchiuse in un'unica classe e non "sparse" nelle varie forme)

Esercizio 3

a)





b)

```
public Class ParteFrontale {
        private CassettaMonete cassetta; // si suppone già creata

        public void inserisciMonete (Euro m){ // tutte le monete in un colpo solo
                cassetta.inserisciMonete(m);
        }

        public void selezionaProdotto (Prodotto p){
                if (cassetta.presente(p)) {
                        if (cassetta.ammontareOK()){
                                cassetta.aggiornaMoneteEspelliProdotto();
                        }
                        else  {
                                print("ammontare non sufficiente");
                        }
                }
                else  {
                        print("prodotto non presente");
                }
        }
}

public Class CassettaMonete {
        private Euro totaleMoneteCassetta; // contiene il totale delle monete presenti nella cassetta
        private Euro moneteInserite;
        private Prodotto prodottoSelezionato;
        private ContenitoreProdotti contenitore; // si suppone già creato
        private HashMap[Prodotto, Euro] listaPrezzi; // si suppone già riempita; ad ogni prodotto è collegato
                                                     // il prezzo

        public void inserisciMonete (Euro m) {
                moneteInserite=m;
```

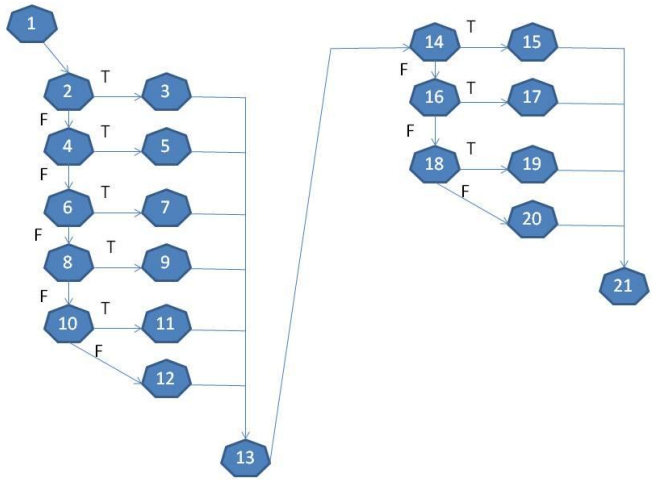
```
        }

        public bool presente (Prodotto p) {
                prodottoSelezionato=p;
                return contenitore.presente(p);
        }

        //  semplificazione: l'ammontare è OK se le monete inserite sono uguali o di più di quelle richieste
        // cioè non si considera il resto ....
        public bool ammontareOK (){
                if (moneteInserite>=listaPrezzi[prodottoSelezionato]) // si suppone di avere ">=" tra EURO
                        return true;
                else
                        return false;
        }

        public void aggiornaMoneteEspelliProdotti() {
                aggiornaRiservaMonete();
                contenitore.espelli(p);
        }

        public void aggiornaRiservaMonete() {
                totaleMoneteCassetta= totaleMoneteCassetta+ moneteInserite;
        }
}
public Class ContenitoreProdotti {
        private Cliente cliente // per semplicità si simula "la consegna" del prodotto con un operazione
                                // riceviProdotto() sulla classe fittizia Cliente
        private List[Prodotto] listaProdotti; // per semplicità si usa una lista ed ogni prodotto di un tipo è
                                              //  ripetuto più volte se presente nel contenitore
                                              // es. listaProdotti=[a,b,b,a,a,c] - 3 prodotti di tipo a, 2 b, 1 c
                                              // la lista si suppone già riempita

        public bool presente (Prodotto p) {
                return listaProdotti.IsIn(p) // operazione usuale IsIn che ritorna true/false
        }

        public espelli (Prodotto p) {
                //  bisognerebbe aggiornare la listaProdotti togliendo p; non si dettaglia ...
                cliente.riceviProdotto(p)
        }

}
```

Esercizio 4

a) CFG solo di stampaCarta. Numeriamo gli statement dell'operazione stampaCarta()

```
public static void stampaCarta(int valore, int seme) {
        1 System.out.println();
        1 System.out.println("\t **********");
        1 System.out.println("\t*          *");
        2 if (valore == 1)
           3 System.out.println("\t*   asso   *");
        4 else if (valore == 11)
           5 System.out.println("\t*   fante  *");
        6 else if (valore == 12)
```

```
      7 System.out.println("\t*  regina  *");
   8 else if (valore == 13)
      9 System.out.println("\t*    re    *");
   10 else if (valore == 10)
      11 System.out.println("\t*    10    *");
   else
      12 System.out.println("\t*    " + valore + "     *");
   13 System.out.println("\t*          *");
   13 System.out.println("\t*    di    *");
   13 System.out.println("\t*          *");
   13 System.out.println("\t*          *");
   14 if (seme == 0)
      15 System.out.println("\t*  cuori   *");
   16 else if (seme == 1)
      17 System.out.println("\t*  quadri  *");
   18 else if (seme == 2)
      19 System.out.println("\t*  fiori   *");
     else
      20 System.out.println("\t*  picche  *");
   21 System.out.println("\t*          *");
   21 System.out.println("\t ********** ");
   21 System.out.println();
}
```



b) Occorre modificare il codice dell'operazione stampaCarta(), ad esempio facendo in modo che l'operazione ritorni una stringa che corrisponde al valore + seme della carta. Es. "asso di picche". Sono sufficienti 6 casi di test per avere statement coverage. Ad esempio:
stampaCarta(1,0)
stampaCarta(11,1)
stampaCarta(12,2)
stampaCarta(13,3)
stampaCarta(10,0)
stampaCarta(5,0)

In Junit ogni asserzione sarà così del tipo
assertEquals("asso di cuori", s.stampaCarta(1,0)) dove s è un oggetto di tipo ScegliCarta