Ingegneria del Software (6 crediti) a.a. 2011-12
Prova Scritta del 11 aprile 2012 --- Bozza di soluzione

Esercizio 1

a) L'accoppiamento tra le componenti è elevato. In particolare

i. Uso di alfa come variabile nel server utilizzata esplicitamente da client, senza esplicitare che alfa è il valore cercato nella funzione Cerca.

Implicazioni: impone alle due componenti di usare alfa in maniera coerente, difficoltà di comprensione, difficoltà di manutenzione (se cambio qualcosa in una delle componenti, devo stare attento di non avere problemi nell'altra)

Altro problema:

ii. Sarebbe meglio effettuare il controllo di lista vuota dentro a Cerca, piuttosto che in Client prima di invocare la funzione

Implicazioni: se voglio modificare il comportamento in caso di lista vuota non è facile individuare dove devono essere effettuate le modifiche

b) Riprogettazione delle due componenti che risolva problemi rilevati

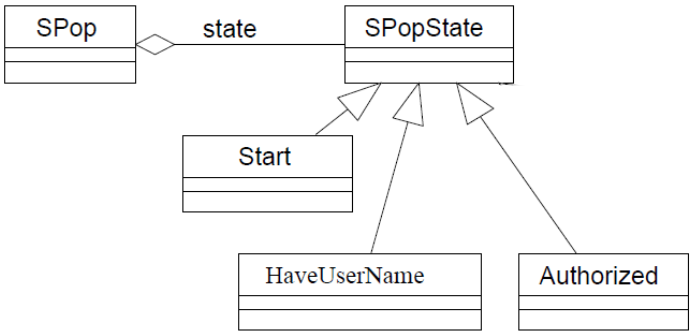
```
public class Server {
    public static bool Cerca(Lista x, int alfa){
   //  è esplicito che alfa è il valore cercato, minor livello di accoppiamento: non si usa più variabile di altra classe
   // Cerca effettua anche il controllo di lista vuota
       while (x != null)
      { if (x.info == alfa) return true
        else x = x.next;
       }
     return false;
    }
}

public class Client {
   static Lista MiaLista() { … }
   public static void main(String[] args) {
        Lista s = MiaLista();
        System.out.print("Inserisci un intero: ");
        int alfa = InOut.readInt();
        if (Server.Cerca(s, alfa))
        // BASSO ACCOPPIAMENTO: il cliente si concentra sulla richiesta del servizio
            System.out.print(Server.alfa+" presente");
       else
           System.out.print(Server.alfa+" non presente");}
}
```

Esercizio 2

a) Identificare un Design Pattern che potrebbe essere istanziato per proporre una soluzione al problema precedente.

State pattern

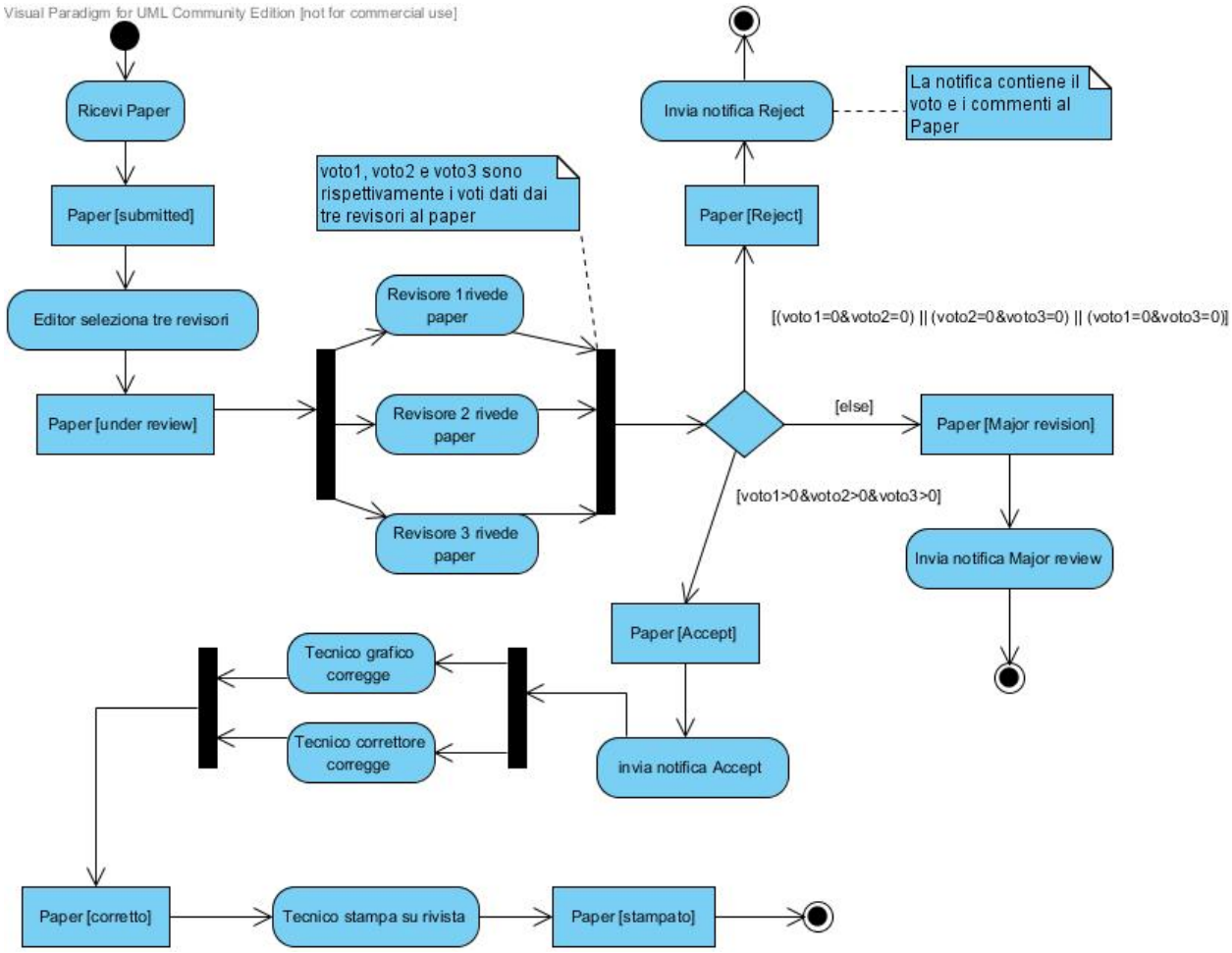
b) Produrre un diagramma delle classi che istanzi il Design Pattern individuato sul problema in esame, mettendo in corrispondenza le classi dell'applicazione con quelle del Design Pattern originario.




```
public class SPopState {
public SPopState user(String userName) {default action here}
public SPopState pass(String password) {default action here}
public SPopState list(int messageNumber) {default action here}
public SPopState retr(int messageNumber) {default action here}
public SPopState quit() {default action here}
}

In Start ridefinisco user

In HaveUserName ridefinisco pass

In Authorized ridefinisco list, retr, quit
```

Esercizio 3

a)



b) // non vengono considerati gli input illegali (es. voto1=7)

```
public Paper ValutazionePaper(voto1:int, voto2:int, voto3: int, p:Paper) {
        if ((voto1=0&voto2=0)||(voto2=0&voto3=0)||(Voto1=0&voto3=0))
                p.state=reject;
        if (voto1>0&voto2>0&voto3>0)
                p.state=accept;
        else
                p.state=major_revision;
        return p;
}
```

Casi di test:

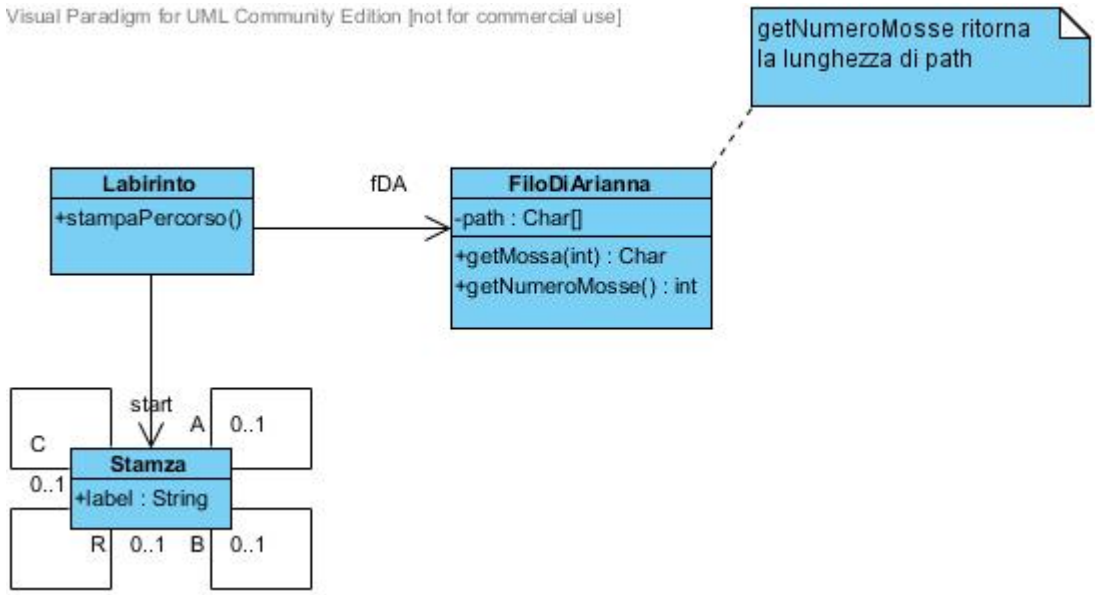
Legali)
0 0 0 --> reject
0 0 1 --> reject
1 1 1 --> accept
1 1 2 --> accept
2 2 2 --> accept

0 0 2 --> major revision
2 1 0 --> major revision

Non legali)
1 6 0
0 0 -2
3 3 3

c)
- se il paper viene sottomesso dopo una major review i revisori sono gli stessi oppure vengono ri-selezionati dall'editor (come per una nuova sottomissione)?
- cosa succede se il ricercatore non sottomette il paper entro 2 mesi che ha ricevuto una major review?
- serve una login oppure chiunque può sottomettere un paper?
- esiste un tempo per i revisori? Cosa succede se un revisore non li rispetta? L'editor può sostituire un revisore e se si come?

Esercizio 4



b)

```
public void stampaPercorso() {
        Stanza s =start;
        int n = FdA.getNumeroMosse();
        for (int i=0; i<n;i++) {
                System.out.println(s.label);
                Switch(FdA.getMossa(i)) {
                        Case 'A': s=s.A; break;
                        Case 'B': s=s.B; break;
                        Case 'C': s=s.C; break;
                        Case 'R': s=s.R; break;
                }
        }
}
```