Ingegneria del Software (6 crediti) a.a. 2011-12
Prova Scritta del 31 agosto 2012: Soluzioni

Esercizio 1

a) Per avere 100% statement coverage del metodo prelievo occorrono 4 casi di test: due relativi alle due eccezioni e due che testano il prelievo con i due tipi di commissioni (famiglia e azienda)

Una possibilità con JUnit 3.8 è:

```
public void testPrelievoFamiglia() {
            Account a = new Account(0);
            assertTrue(a.getSaldo()==0);
            try {
                  a.versamento(100);
                  assertEquals(90, a.getSaldo(),  0);
                  a.prelievo(10);
                  assertEquals(72, a.getSaldo(), 0);
            } catch (Exception e) {
                  // TODO Auto-generated catch block
                  e.printStackTrace();
            }
      }

      public void testVersamentoAzienda() {
            Account a = new Account(1);
            assertTrue(a.getSaldo()==0);
            try {
                  a.versamento(100);
                  assertEquals(a.getSaldo(), 95, 0);
                  a.prelievo(10);
                  assertEquals(80.75, a.getSaldo(), 0);
            } catch (Exception e) {
                  // TODO Auto-generated catch block
                  e.printStackTrace();
            }
      }

      public void testSommaNegativa() {
            Account a = new Account(0);
            try {
                  a.prelievo(-10);
                  fail();
            } catch (Exception e) {
                  assertTrue(true);
            }
      }

      public void testSaldoMinoreSomma() {
            Account a = new Account(0);
            assertTrue(a.getSaldo()==0);
            try {
                  a.versamento(100);
                  assertEquals(a.getSaldo(), 95, 0);
                  a.prelievo(105);
                  fail();
            } catch (Exception e) {
                  assertTrue(true);
            }
      }
```

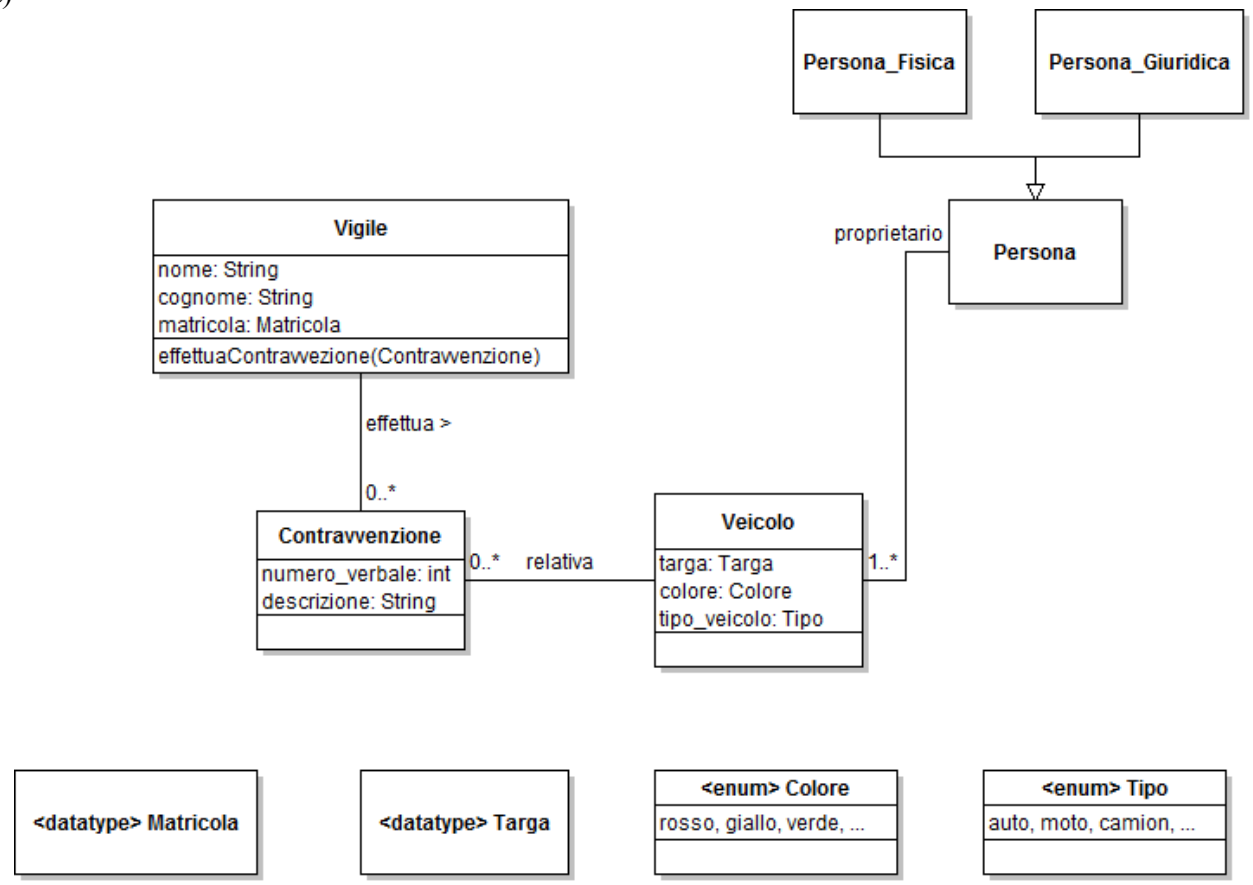
b+c) dopo i due refactoring il codice è:

```
public abstract class Account {

  private float saldo;

  public Account(){
    this.saldo=0;
    //this.tipoConto=tipoConto;
  }

  public float getSaldo(){
    return this.saldo;
  }

  public void versamento(float somma) throws Exception{
    if (somma <= 0){
      throw new Exception("Somma rifiutata: " + somma);
    }
    this.saldo = this.saldo + somma;

    // calcola gli interessi di commissione sull'operazione
    float commissioneOperazione = calcolaCommissione();

    // sottraggo la commisione calcolata al saldo
    this.saldo = this.saldo - ((this.saldo * commissioneOperazione)/100);
  }

  public void prelievo(float somma) throws Exception{
    if (somma < 0){
      throw new Exception("Somma negativa: " + somma);
    }
    if (this.saldo < somma){
      throw new Exception("Saldo insufficiente : " + somma);
    }
    this.saldo = this.saldo - somma;

    float commissioneOperazione = calcolaCommissione();

    // sottraggo la commiszione calcolata al saldo
    this.saldo = this.saldo - ((this.saldo * commissioneOperazione)/100);
  }

  public abstract float calcolaCommissione();
}

public class AccountFamiglia extends Account {
       public AccountFamiglia() {
            super();
            // TODO Auto-generated constructor stub
      }

      public float calcolaCommissione(){
                return 10;
             }
}

public class AccountAzienda extends Account {
      public AccountAzienda() {
            super();
            // TODO Auto-generated constructor stub
      }
```

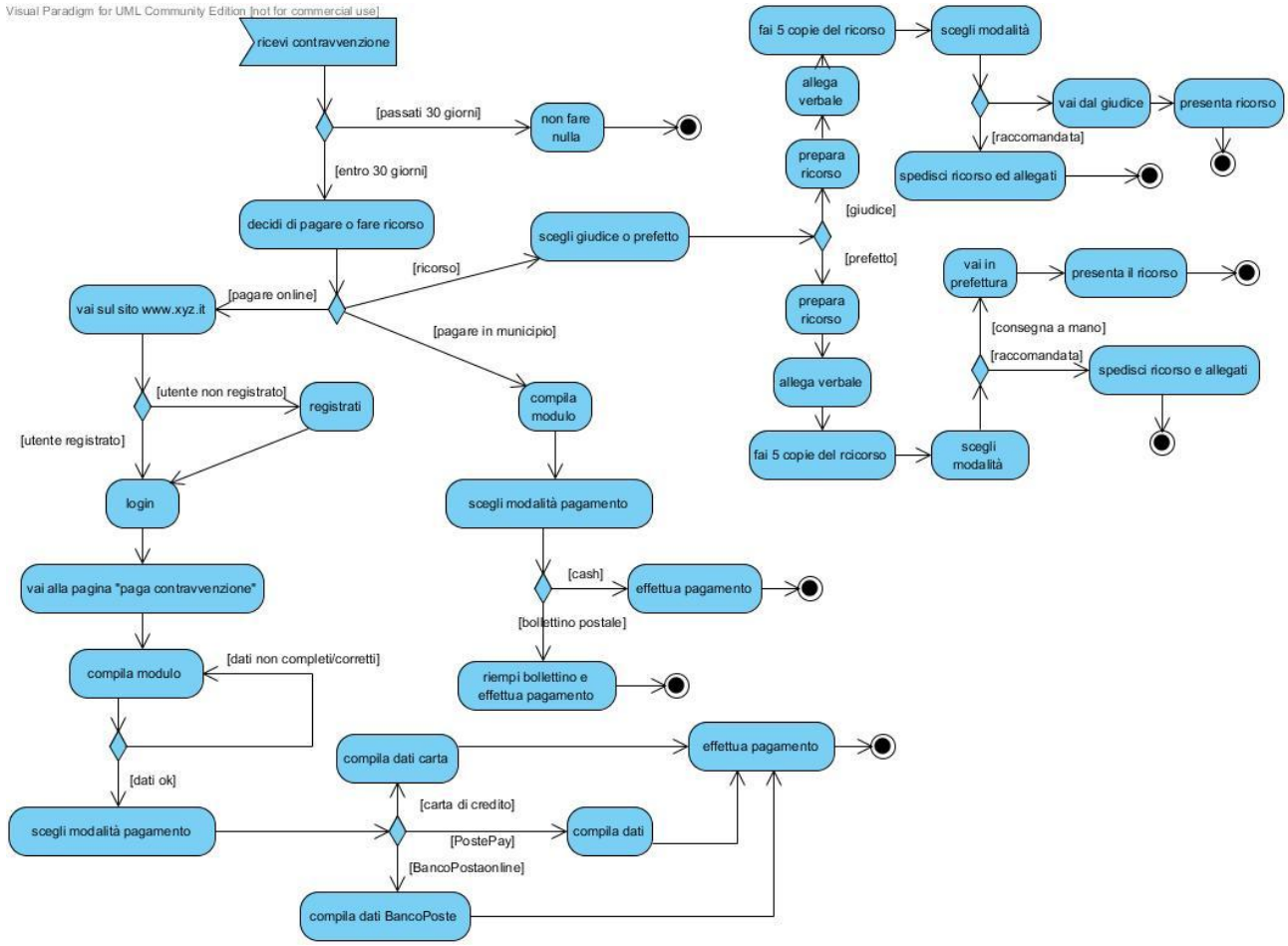
```
        public float calcolaCommissione(){
            return 5;
          }
}
```

Esercizio 2

a)



b) diagramma più opportuno: activity diagram



Esercizio 3

Esercizio 4