Università di Roma "La Sapienza", Facoltà di Ingegneria

Corso di
"PROGETTAZIONE DEL SOFTWARE" (Ing. Gestionale)
Proff. Giuseppe De Giacomo e Monica Scannapieco A.A. 2003-04

SOLUZIONE ESAME DEL 30/03/2004

Roma, 2 Aprile 2004

Diagramma delle classi concettuale UML

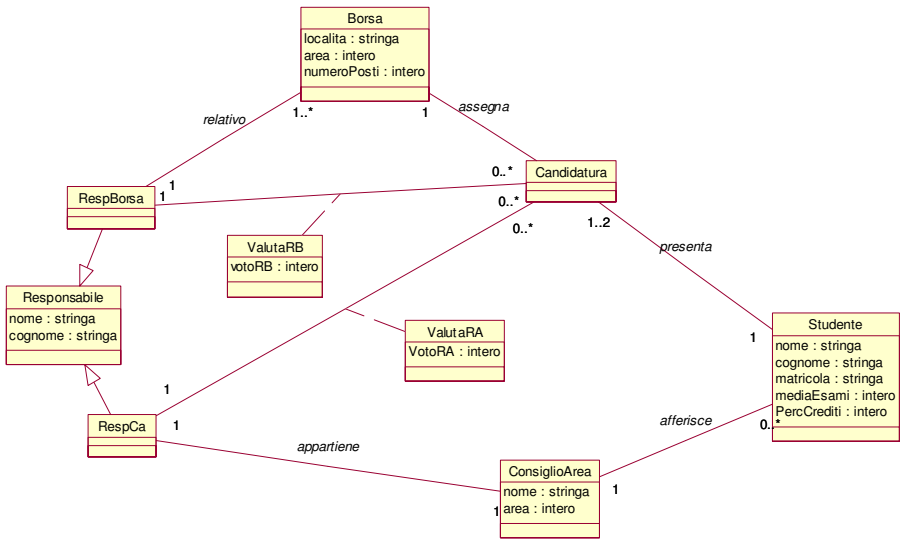
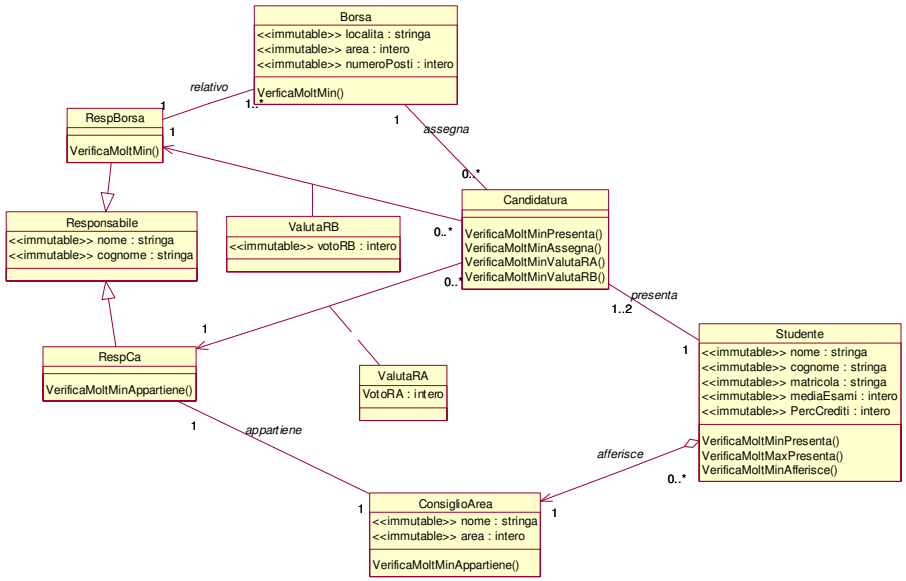


Diagramma delle classi realizzativo UML



Specifica degli use case

```
InizioSpecificaUseCase VerificheErasmus

  calcolaPunteggio(c: Candidatura):int
     pre:  nessuna
     post: se s è lo studente che ha presentato la candidatura c,
 result e' la somma della media degli esami di s,
 della percentuale dei crediti di s e dei voti assegnati a c dal responsabile della borsa
 e dal responsabile del consiglio d'area

  calcolaStudenti(b: borsa): Insieme
     pre:  nessuna
     post: result e' l'insieme degli studenti che hanno presentato una candidatura per b


FineSpecifica
```


Responsabilità sulle associazioni

Dalla specifica dello use case e delle molteplicità minime nel diagramma delle classi emerge che:

- *Candidatura* ha responsabilità su *presenta*,*assegna*,*ValutaRB*,*ValutaRA*
- *Studente* ha responsabilità su *presenta*, *afferisce*
- *ConsiglioArea* ha responsabilità su *appartiene*
- *Borsa* ha responsabilità su *assegna*,*relativo*
- *RespBorsa*ha responsabilità su *relativo*
- *RespCa* ha responsabilità su *appartiene*.

La classe Java Candidatura

```
public class Candidatura implements
SupportoAssociazionePresenta, SupportoAssociazioneAssegna
 { private TipoLinkPresenta link;
private TipoLinkValutaRB linkRB;
private TipoLinkValutaRA linkRA;
private TipoLinkAssegna linkAss;

/**realizzazione associazione presenta*/

public void inserisciLinkPresenta(AssociazionePresenta a){
if (a!=null && a.getLink().getCandidatura()==this && link==null)
link=a.getLink();
else throw new RuntimeException ("inserimento link impossibile");
}
public void eliminaLinkPresenta(AssociazionePresenta a){
if (a!=null && a.getLink().equals(link))
link=null;
else throw new RuntimeException ("eliminazione link impossibile");
}
public boolean VerificaMoltMin(){
return link!=null;
}
public TipoLinkPresenta getLinkPresenta(){
if (VerificaMoltMin()==false){
throw new RuntimeException("MoltMinViolata");
}
else return link;
}
```

```
/**realizzazione associazione ValutaRB*/
public void inserisciLinkValutaRB(TipoLinkValutaRB t){
if (t!=null && t.getCandidatura()==this)
linkRB=t;
}
public void eliminaLinkValutaRB(TipoLinkValutaRB t){
linkRB=null;
}
public boolean VerificaMoltMinValutaRB(){
return linkRB!=null;
}
public TipoLinkValutaRB getLinkValutaRB(){
if (VerificaMoltMinValutaRB()==false){
throw new RuntimeException("MoltMinViolata");
}
else return linkRB;
}
/**realizzazione associazione ValutaRA*/
public void inserisciLinkValutaRA(TipoLinkValutaRA t){
if (t!=null && t.getCandidatura()==this)
linkRA=t;
}
public void eliminaLinkValutaRA(TipoLinkValutaRA t){
linkRA=null;
}
public boolean VerificaMoltMinValutaRA(){
return linkRA!=null;
}
public TipoLinkValutaRA getLinkValutaRA(){
if (VerificaMoltMinValutaRA()==false){
throw new RuntimeException("MoltMinViolata");
```

```
}
else return linkRA;
}

/**realizzazione associazione Assegna*/
public void inserisciLinkAssegna(AssociazioneAssegna a){
if (a!=null && a.getLink().getCandidatura()==this && linkAss==null)
linkAss=a.getLink();
else throw new RuntimeException ("inserimento link impossibile");
}
public void eliminaLinkAssegna(AssociazioneAssegna a){
if (a!=null && a.getLink().equals(linkAss))
link=null;
else throw new RuntimeException ("eliminazione link impossibile");
}
public TipoLinkAssegna getLinkAssegna(){
return linkAss;
}

}
```

La classe Java Borsa

```
import java.util.*;
import insiemearray.*;

public class Borsa implements SupportoAssociazioneAssegna,SupportoAssociazioneRelativo{
private final String localita;
private final int area;
private final int numPosti;
private Set insieme_link; /**rappresenta associazione assegna*/
private TipoLinkRelativo link;

/**costruttore*/
public Borsa (String loc,int a, int n)
{localita=loc;
area=a;
numPosti=n;
insieme_link=new InsiemeArrayOmogeneo(TipoLinkAssegna.class);
}
/**metodi gestione campi dati*/
public String getLocalita(){return localita;
}
public int getArea(){return area;
}
public int getNumPosti(){return numPosti;
}
/**realizzazione associazione Assegna*/

public void inserisciLinkAssegna(AssociazioneAssegna a){
if (a!=null && a.getLink().getBorsa()==this && a.getLink().getBorsa()!=null)
insieme_link.add(a.getLink());
```



```
else throw new RuntimeException ("inserimento link impossibile");
}
public void eliminaLinkAssegna(AssociazioneAssegna a){
if (a!=null && a.getLink().getBorsa()==this)
insieme_link.remove(a.getLink());
else throw new RuntimeException ("eliminazione link impossibile");
}

public Iterator getLinkAssegna(){
return insieme_link.iterator();
}
/**realizzazione associazione Relativo*/
public void inserisciLinkRelativo(AssociazioneRelativo a){
if (a!=null && a.getLink().getBorsa()==this && link==null)
link=a.getLink();
else throw new RuntimeException ("inserimento link impossibile");
}
public void eliminaLinkRelativo(AssociazioneRelativo a){
if (a!=null && a.getLink().equals(link))
link=null;
else throw new RuntimeException ("eliminazione link impossibile");
}
public boolean VerificaMoltMin(){
return link!=null;
}
public TipoLinkRelativo getLinkRelativo(){
if (VerificaMoltMin()==false){
throw new RuntimeException("MoltMinViolata");
}
else return link;
}
}
```

La classe Java Studente

```
import java.util.*;
import insiemearray.*;

public class Studente implements SupportoAssociazionePresenta{
private final String nome;
private final String cognome;
private final String matricola;
private final int mediaEsami;
private final int percCrediti;
private Set  insieme_link; /**rappresenta associazione presenta*/
private ConsiglioArea ca;/**rappresenta associazione afferisce*/

public static final int MINLINKPRESENTA=1;
public static final int MAXLINKPRESENTA=2;

/**costruttore*/
public Studente (String n, String c, String m, int media, int crediti, ConsiglioArea cons)
{nome=n;
cognome=c;
mediaEsami=media;
percCrediti=crediti;
ca=cons;
matricola=m;
insieme_link=new InsiemeArrayOmogeneo(TipoLinkPresenta.class);
}
/**metodi gestione campi dati*/
public String getNome(){return nome;
}
```



```
public String getCognome(){return cognome;
}
public String getMatricola(){return matricola;
}
public int getMediaEsami(){return mediaEsami;
}
public int getPercCrediti(){return percCrediti;
}
/**realizzazione associazione presenta*/

public void inserisciLinkPresenta(AssociazionePresenta a){
if (a!=null && a.getLink().getStudente()==this)
insieme_link.add(a.getLink());
else throw new RuntimeException ("inserimento link impossibile");
}
public void eliminaLinkPresenta(AssociazionePresenta a){
if (a!=null && a.getLink().getStudente()==this)
insieme_link.remove(a.getLink());
else throw new RuntimeException ("eliminazione link impossibile");
}

public Iterator getLinkPresenta(){
if (VerificaMoltMinPresenta()==false){
throw new RuntimeException("MoltMinViolata");
}
else return insieme_link.iterator();
}
 public boolean VerificaMoltMinPresenta(){
   return insieme_link.size()>MINLINKPRESENTA;
   }
public boolean VerificaMoltMax(){
```

```
   return insieme_link.size()<MAXLINKPRESENTA;
   }

/**realizzazione associazione afferisce*/

 public ConsiglioArea getConsiglioArea(){
if (VerificaMoltMinPresenta()==false)
throw new RuntimeException("MoltMinViolata");
else
   return ca;
    }

    public void setConsiglioArea(ConsiglioArea c)
    { ca=c;
    }
   public boolean VerificaMoltMinAfferisce(){
   return ca!=null;
   }
}
```

L'interfaccia Java SupportoAssociazionePresenta

```
public interface SupportoAssociazionePresenta
{void inserisciLinkPresenta(AssociazionePresenta a);
void eliminaLinkPresenta(AssociazionePresenta a);
}
```

La classe Java AssociazionePresenta

```
public class AssociazionePresenta
{private TipoLinkPresenta link;
private AssociazionePresenta(TipoLinkPresenta x)
{
link = x;
}

public TipoLinkPresenta getLink()
{return link;
}

public static void inserisci(TipoLinkPresenta y)
{if(y.getStudente()!=null && y.getCandidatura()!=null)
{AssociazionePresenta k= new AssociazionePresenta(y);
y.getStudente().inserisciLinkPresenta(k);
y.getCandidatura().inserisciLinkPresenta(k);
}
}
public static void elimina(TipoLinkPresenta y)
{if(y.getCandidatura() !=null && y.getStudente()!=null)
{AssociazionePresenta k= new AssociazionePresenta(y);
y.getStudente().eliminaLinkPresenta(k);
y.getCandidatura().eliminaLinkPresenta(k);
}
}
}
```


La classe Java TipoLinkPresenta

```
public class TipoLinkPresenta
{

private final Studente loStudente;
private final Candidatura laCandidatura;

public TipoLinkPresenta(Studente x, Candidatura y)
{ loStudente=x;
laCandidatura=y;
}

public Studente getStudente()
{
return loStudente;
}
public Candidatura getCandidatura()
{
return laCandidatura;
}
public boolean equals (Object o)
{
if (o!=null && getClass().equals(o.getClass()))
{TipoLinkPresenta b= (TipoLinkPresenta)o;
return b.loStudente!=null && b.laCandidatura!= null
&& b.loStudente==loStudente &&b.laCandidatura==laCandidatura;
}
else return false;
}

}
```


La classe Java ConsiglioArea

```
public class ConsiglioArea implements SupportoAssociazioneAppartiene
{
private final String nome;
private final int area;
    private TipoLinkAppartiene link_appartiene;

/**costruttore*/
public ConsiglioArea(String n, int a)
{ nome = n;
area=a;
link_appartiene=null;
}
/**gestione campi dati*/

public String getNome(){
return nome;
}
public int getArea(){
return area;
}
/**realizzazione associazione appartiene*/
public boolean VerificaMoltMin(){
return link_appartiene!=null;
}
public void inserisciLinkAppartiene(AssociazioneAppartiene a){
if (a!=null && a.getLink().getConsiglioArea()==this && link_appartiene==null)
link_appartiene=a.getLink();
else throw new RuntimeException ("inserimento link impossibile");
```



```
}
public void eliminaLinkAppartiene(AssociazioneAppartiene a){
if (a!=null && a.getLink().equals(link_appartiene))
link_appartiene=null;
else throw new RuntimeException ("eliminazione link impossibile");
}
public TipoLinkAppartiene getLinkAppartiene(){
if (VerificaMoltMin()==false){
throw new RuntimeException("MoltMinViolata");
}
else return link_appartiene;
}

public String toString(){
        return nome+" "+area;
    }
}
```

La classe Java Responsabile

```
public abstract class Responsabile
{
protected final String nome;
protected final String cognome;
public Responsabile(String n, String c)
{nome=n;
cognome=c;
}
public String getNome()
{return nome;
}
public String getCognome()
{return cognome;
}
}
```


La classe Java RespCa

```
public class RespCa extends Responsabile implements SupportoAssociazioneAppartiene
{
private TipoLinkAppartiene link_appartiene;

/**Costruttore*/

public RespCa(String n, String c){
super(n,c);
link_appartiene=null;
}
/**Associazione appartiene*/
public boolean VerificaMoltMin(){
return link_appartiene!=null;
}

public void inserisciLinkAppartiene(AssociazioneAppartiene a){
if (a!=null && a.getLink().getRespCa()==this && link_appartiene==null)
link_appartiene=a.getLink();
else throw new RuntimeException ("inserimento link impossibile");

}
public void eliminaLinkAppartiene (AssociazioneAppartiene a)
{
if (a!=null && a.getLink().equals(link_appartiene))
link_appartiene=null;
else throw new RuntimeException ("eliminazione link impossibile");
}
```



```
public TipoLinkAppartiene getLinkAppartiene(){
if (VerificaMoltMin()==false){
throw new RuntimeException("MoltMinViolata");
}
else return link_appartiene;
}
}
```

La classe Java RespBorsa

```
import java.util.*;
import insiemearray.*;
public class RespBorsa extends Responsabile {
/**Costruttore*/
Set insieme_link;
public static final int MINLINKRELATIVO=1;
public RespBorsa(String n, String c){
super(n,c);
insieme_link=new InsiemeArrayOmogeneo(TipoLinkRelativo.class);
}
public void inserisciLinkRelativo(AssociazioneRelativo a){
if (a!=null && a.getLink().getRespBorsa()==this)
insieme_link.add(a.getLink());
else throw new RuntimeException ("inserimento link impossibile");
}
public void eliminaLinkRelativo(AssociazioneRelativo a){
if (a!=null && a.getLink().getRespBorsa()==this)
insieme_link.remove(a.getLink());
else throw new RuntimeException ("eliminazione link impossibile");
}

public Iterator getLinkRelativo(){
if (VerificaMoltMinRelativo()==false){
throw new RuntimeException("MoltMinViolata");
}
else return insieme_link.iterator();
}
 public boolean VerificaMoltMinRelativo(){
   return insieme_link.size()<MINLINKRELATIVO;
```

```
   }
}
```

L'interfaccia SupportoAssociazioneAppartiene

```
public interface SupportoAssociazioneAppartiene
{void inserisciLinkAppartiene(AssociazioneAppartiene a);
void eliminaLinkAppartiene(AssociazioneAppartiene a);
}
```


La classe Java AssociazioneAppartiene

```
public class AssociazioneAppartiene
{private TipoLinkAppartiene link;
private AssociazioneAppartiene(TipoLinkAppartiene x)
{link = x;
}
public TipoLinkAppartiene getLink()
{return link;
}

public static void inserisci(TipoLinkAppartiene y)
{if(y.getConsiglioArea()!=null && y.getRespCa()!=null)
{AssociazioneAppartiene k= new AssociazioneAppartiene(y);
y.getConsiglioArea().inserisciLinkAppartiene(k);
y.getRespCa().inserisciLinkAppartiene(k);
}
}
public static void elimina(TipoLinkAppartiene y)
{if(y.getConsiglioArea() !=null && y.getRespCa()!=null)
{AssociazioneAppartiene k= new AssociazioneAppartiene(y);
y.getConsiglioArea().eliminaLinkAppartiene(k);
y.getRespCa().eliminaLinkAppartiene(k);
}
}
}
```


La classe Java TipoLinkAppartiene

```
public class TipoLinkAppartiene
{

private final ConsiglioArea ilConsiglioArea;
private final RespCa ilResp;

public TipoLinkAppartiene(ConsiglioArea x, RespCa y)
{ ilConsiglioArea=x;
ilResp=y;
}

public ConsiglioArea getConsiglioArea()
{
return ilConsiglioArea;
}
public RespCa getRespCa()
{
return ilResp;
}
public boolean equals (Object o)
{
if (o!=null && getClass().equals(o.getClass()))
{TipoLinkAppartiene b= (TipoLinkAppartiene)o;
return b.ilConsiglioArea!=null && b.ilResp!= null && b.ilConsiglioArea==ilConsiglioArea &&b.ilResp==ilResp;
}
else return false;
}

}
```

La classe Java TipoLinkValutaRB

```
public class TipoLinkValutaRB
{ private final RespBorsa ilRespBorsa;
private final Candidatura laCandidatura;
private final int votoRB;
public TipoLinkValutaRB(RespBorsa  x, Candidatura y, int voto)
{ ilRespBorsa=x;
laCandidatura=y;
votoRB=voto;
}

public RespBorsa  getRespBorsa ()
{
return ilRespBorsa;
}
public Candidatura getCandidatura()
{
return laCandidatura;
}

public int getVotoRB(){
return votoRB;}

public boolean equals (Object o)
{
if (o!=null && getClass().equals(o.getClass()))
{TipoLinkValutaRB b= (TipoLinkValutaRB)o;
return b.ilRespBorsa!=null && b.laCandidatura!= null
&& b.ilRespBorsa==ilRespBorsa &&b.laCandidatura==laCandidatura;
```



```
}
else return false;
}

}
```

La classe Java TipoLinkValutaRA

```
public class TipoLinkValutaRA
{ private final RespCa ilRespCa;
private final Candidatura laCandidatura;
private final int votoRA;
public TipoLinkValutaRA(RespCa  x, Candidatura y, int voto)
{ ilRespCa=x;
laCandidatura=y;
votoRA=voto;
}

public RespCa getRespCa ()
{
return ilRespCa;
}
public Candidatura getCandidatura()
{
return laCandidatura;
}

public int getVotoRA(){
return votoRA;}

public boolean equals (Object o)
{
if (o!=null && getClass().equals(o.getClass()))
{TipoLinkValutaRA b= (TipoLinkValutaRA)o;
return b.ilRespCa!=null && b.laCandidatura!= null && b.ilRespCa==ilRespCa &&
b.laCandidatura==laCandidatura;
}
```



```
else return false;
}

}
```

L'interfaccia SupportoAssociazioneRelativo

```
public interface SupportoAssociazioneRelativo
{void inserisciLinkRelativo(AssociazioneRelativo a);
void eliminaLinkRelativo(AssociazioneRelativo a);
}
```


La classe Java AssociazioneRelativo

```
public class AssociazioneRelativo
{private TipoLinkRelativo link;
private AssociazioneRelativo(TipoLinkRelativo x)
{
link = x;
}

public TipoLinkRelativo getLink()
{return link;
}

public static void inserisci(TipoLinkRelativo y)
{if(y.getBorsa()!=null && y.getRespBorsa()!=null)
{AssociazioneRelativo k= new AssociazioneRelativo(y);
y.getBorsa().inserisciLinkRelativo(k);
y.getRespBorsa().inserisciLinkRelativo(k);
}
}
public static void elimina(TipoLinkRelativo y)
{if(y.getRespBorsa() !=null && y.getBorsa()!=null)
{AssociazioneRelativo k= new AssociazioneRelativo(y);
y.getBorsa().eliminaLinkRelativo(k);
y.getRespBorsa().eliminaLinkRelativo(k);
}
}
}
```


La classe Java TipoLinkRelativo

```
public class TipoLinkRelativo{
private final RespBorsa ilRespBorsa;
private final Borsa laBorsa;

public TipoLinkRelativo(RespBorsa  x, Borsa y)
{ ilRespBorsa=x;
laBorsa=y;
}

public RespBorsa  getRespBorsa ()
{
return ilRespBorsa;
}
public Borsa getBorsa()
{
return laBorsa;
}


public boolean equals (Object o)
{
if (o!=null && getClass().equals(o.getClass()))
{TipoLinkRelativo b= (TipoLinkRelativo)o;
return b.ilRespBorsa!=null && b.laBorsa!= null && b.ilRespBorsa==ilRespBorsa &&b.laBorsa==laBorsa;
}
else return false;
}

}
```


L'interfaccia SupportoAssociazioneAssegna

```
public interface SupportoAssociazioneAssegna
{void inserisciLinkAssegna(AssociazioneAssegna a);
void eliminaLinkAssegna(AssociazioneAssegna a);

}
```

La classe Java AssociazioneAssegna

```
public class AssociazioneAssegna
{private TipoLinkAssegna link;
private AssociazioneAssegna(TipoLinkAssegna x)
{
link = x;
}

public TipoLinkAssegna getLink()
{return link;
}

public static void inserisci(TipoLinkAssegna y)
{if(y.getBorsa()!=null && y.getCandidatura()!=null)
{AssociazioneAssegna k= new AssociazioneAssegna(y);
y.getBorsa().inserisciLinkAssegna(k);
y.getCandidatura().inserisciLinkAssegna(k);
}
}
public static void elimina(TipoLinkAssegna y)
{if(y.getCandidatura()!=null && y.getBorsa()!=null)
{AssociazioneAssegna k= new AssociazioneAssegna(y);
y.getBorsa().eliminaLinkAssegna(k);
y.getCandidatura().eliminaLinkAssegna(k);
}
}
}
```


La classe Java TipoLinkAssegna

```
public class TipoLinkAssegna
{
private final Borsa laBorsa;
private final Candidatura laCandidatura;

public TipoLinkAssegna(Borsa x, Candidatura y)
{ laBorsa=x;
laCandidatura=y;
}

public Borsa getBorsa()
{
return laBorsa;
}
public Candidatura getCandidatura()
{
return laCandidatura;
}
public boolean equals (Object o)
{
if (o!=null && getClass().equals(o.getClass()))
{TipoLinkAssegna b= (TipoLinkAssegna)o;
return b.laBorsa!=null && b.laCandidatura!= null && b.laBorsa==laBorsa &&b.laCandidatura==laCandidatura;
}
else return false;
}

}
```


Realizzazione in Java dello use case

```
import java.util.*;
import insiemearray.*;
public class usecase
{
public static int calcolaPunteggio(Candidatura c)
{TipoLinkPresenta link=c.getLinkPresenta();
TipoLinkValutaRB linkValutaRB= c.getLinkValutaRB();
TipoLinkValutaRA linkValutaRA= c.getLinkValutaRA();
return (link.getStudente().getPercCrediti()+link.getStudente().getMediaEsami()+linkValutaRB.getVotoRB()+linkVa
}
public static Set calcolaStudenti(Borsa b){
Iterator it=b.getLinkAssegna();
Set result= new InsiemeArrayOmogeneo(Studente.class);
while (it.hasNext())
{TipoLinkAssegna link=(TipoLinkAssegna)it.next();
Candidatura c= link.getCandidatura();
result.add(c.getLinkPresenta().getStudente());
}
return result;
}
}
```


InsiemeArray

```
//Realizzazione dell'interfaccia Set con un array invece che con una lista

import java.util.*;

public class InsiemeArray implements Set, Cloneable {


  // campi dati
  protected Object[] array;
  protected static final int dimInit = 10; //dim. iniz. array

  protected int cardinalita;
  protected Class elemClass;

  // costruttori
  public InsiemeArray(Class cl) {
    array = new Object[dimInit];
    cardinalita = 0;
    elemClass = cl;
  }

  // funzioni proprie della classe
  // (realizzazione delle funzioni di Set)

  // basic operations
  public int size() {
    return cardinalita;
  }
```

```
public boolean isEmpty() {
  return cardinalita == 0;
}

public boolean contains(Object e) {
  if (!elemClass.isInstance(e)) return false;
  else return appartiene(e);
}

public boolean add(Object e) {
  if (!elemClass.isInstance(e)) return false;
  else if (appartiene(e)) return false;
  else {
    if (cardinalita == array.length) {// raddoppia array
      Object[] aux = new Object[array.length*2];
      for(int i = 0; i < array.length; i++)
        aux[i] = array[i];
      array = aux;
    }
    array[cardinalita] = e;
    cardinalita++;
    return true;
  }
}

public boolean remove(Object e) {
  if (!elemClass.isInstance(e)) return false;
  if (!appartiene(e)) return false;
  else {
    int k = 0;                  // trova l'elemento
    while (!array[k].equals(e))
      k++;
    for(int i = k; i < cardinalita-1; i++) // sposta di una poss
       array[i] = array[i+1];              // verso il basso gli
                                           // elementi dell'array
    cardinalita--;

    // rimpicciolisci l'array se e' il caso
    if (cardinalita > dimInit && cardinalita < array.length/3) {
      Object[] aux = new Object[array.length/2];
      for(int i = 0; i < cardinalita; i++)
        aux[i]=array[i];
      array = aux;
    }
    return true;
  }
}

public Iterator iterator() {
  return new IteratorInsiemeArray(this);
}


// bulk operations
public boolean containsAll(Collection c) {
  Iterator it = c.iterator();
  while (it.hasNext()) {
    Object e = it.next();
    if (!contains(e)) return false;
  }
  return true;
}
```

```
public boolean addAll(Collection c){
  throw new UnsupportedOperationException("addlAll() non e' supportata");
}
public boolean removeAll(Collection c) {
  throw new UnsupportedOperationException("removeAll() non e' supportata");
}
public boolean retainAll(Collection c) {
  throw new UnsupportedOperationException("retainAll() non e' supportata");
}
public void clear() {
  throw new UnsupportedOperationException("clear() non e' supportata");
}

// array operations
public Object[] toArray() {
  Object[] a = new Object[size()];
  int i = 0;
  Iterator it = iterator();
  while (it.hasNext()) {
    a[i] = it.next();
    i++;
  }
  return a;
}

public Object[] toArray(Object[] a) {
  if (a.length < size())
    a = new Object[size()];
  int i = 0;
  Iterator it = iterator();
  while (it.hasNext()) {
    a[i] = it.next();
    i++;
  }
  for (; i < a.length; i++)
    a[i] = null;
  return a;
}


// funzioni speciali ereditate da Object
public boolean equals(Object o) {
  if (o != null && getClass().equals(o.getClass())) {
    InsiemeArray ins = (InsiemeArray)o;
    if (!elemClass.equals(ins.elemClass)) return false;
    // ins non e' un insieme del tipo voluto
    else if (cardinalita != ins.cardinalita) return false;
    // ins non ha la cardinalita' giusta
    else {
      // verifica che gli elementi nella lista siano gli stessi
      for(int i = 0; i < ins.cardinalita; i++)
        if (!appartiene(ins.array[i])) return false;
      return true;
    }
  }
  return false;
}

public Object clone() {
  try {
    InsiemeArray ins = (InsiemeArray) super.clone();
```

```
      ins.array = new Object[array.length];
      for(int i = 0; i < cardinalita; i++)
        ins.array[i]=array[i];
      return ins;
    } catch(CloneNotSupportedException e) {
      throw new InternalError(e.toString());
    }
  }
  public String toString() {
    String s = "{ ";
    for(int i = 0; i < cardinalita; i++)
      s = s + array[i] + " ";
    s = s + "}";
    return s;
  }

  // funzioni ausiliarie

  protected boolean appartiene(Object e) {
    for(int i = 0; i < cardinalita; i++)
      if (array[i].equals(e)) return true;
    return false;
  }
}
```

IteratorInsiemeArray

```
// Quanto segue deve stare nello stesso package di InsiemeArray

import java.util.*;

public class IteratorInsiemeArray implements Iterator {

  private InsiemeArray insiemeArray;
  private int indice;

  public IteratorInsiemeArray(InsiemeArray ia) {
    insiemeArray = ia;
    indice = 0;
  }

  // Realizzazione funzioni di Itarator
  public boolean hasNext() {
    return indice < insiemeArray.cardinalita;
  }

  public Object next() {
    Object e = insiemeArray.array[indice];
    indice++;
    return e;
  }

  public void remove() {
    throw new UnsupportedOperationException("remove() non e' supportata");
  }
}
```