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 15/04/2004

Roma, 22 Aprile 2004

Diagramma delle classi concettuale UML

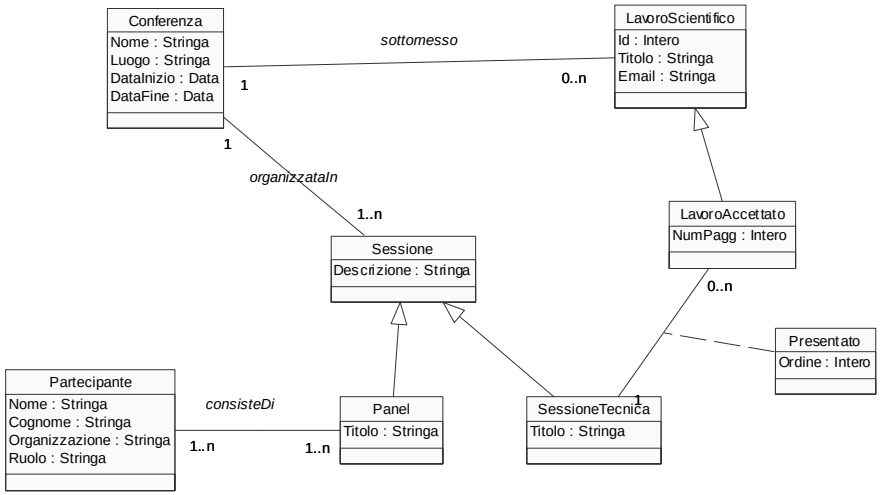
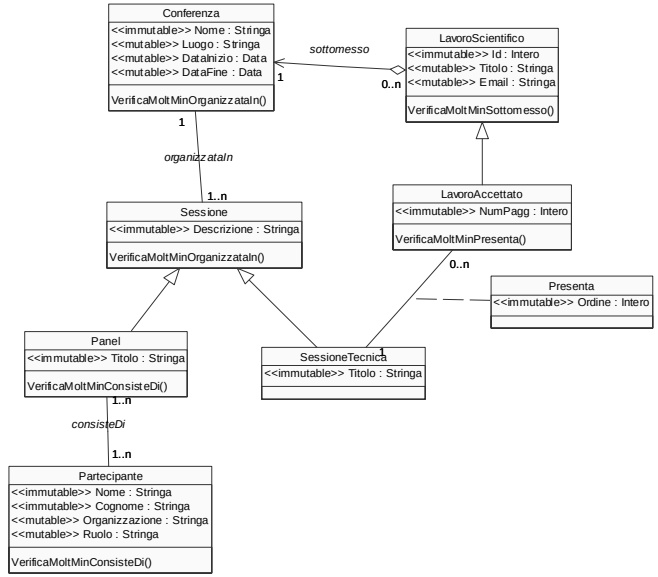


Diagramma delle classi realizzativo UML



Specifica degli use case

```
InizioSpecificaUseCase VerificheConferenze

  CalcolaPanel(c: Conferenza):int
     pre:  nessuna
     post: result è il numero di panel della conferenza c.

  CalcolaElencoLavori (s: SessioneTecnica): stringa
     pre:  nessuna
     post: result è l'elenco dei titoli dei lavori di s separati da uno spazio.

FineSpecifica
```


Responsabilità sulle associazioni

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

- *Conferenza* ha responsabilità su *organizzataIn*
- *LavoroScientifico* ha responsabilità su *sottomesso*
- *LavoroScientificoAccettato* ha responsabilità su *presenta*
- *sessione* ha responsabilità su *organizzataIn*
- *SessioneTecnica* ha responsabilità su *presenta*

- *Panel* ha responsabilità su *consisteDi*
- *Partecipante* ha responsabilità su *consisteDi*

La classe Java Conferenza

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

public class Conferenza implements SupportoAssociazioneOrganizzataIn{
private final String nome;
private String luogo;
private Data  dataInizio;
private Data dataFine;
private Set  insieme_link; /**rappresenta associazione organizzataIn*/

public static final int MINLINKORGANIZZATAIN=1;

/**costruttore*/
public Conferenza (String n, String l, Data i, Data f)
{nome=n;
luogo=l;
dataInizio=i;
dataFine=f;
insieme_link=new InsiemeArrayOmogeneo(TipoLinkOrganizzataIn.class);
}
/**metodi gestione campi dati*/
public String getNome(){return nome;
}
public String getLuogo(){return luogo;
}
public void setLuogo(String l){
luogo=l;
}
```

```
public Data getDataInizio(){return dataInizio;
}
public void setDataInizio(Data i){
dataInizio=i;
}
public Data getDataFine(){return dataFine;
}
public void setDataFine(Data f){dataFine=f;
}
/**realizzazione associazione OrganizzataIn*/

public void inserisciLinkOrganizzataIn(AssociazioneOrganizzataIn a){
if (a!=null && a.getLink().getConferenza()==this)
insieme_link.add(a.getLink());
else throw new RuntimeException ("inserimento link impossibile");
}
public void eliminaLinkOrganizzataIn(AssociazioneOrganizzataIn a){
if (a!=null && a.getLink().getConferenza()==this)
insieme_link.remove(a.getLink());
else throw new RuntimeException ("eliminazione link impossibile");
}

public Iterator getLinkOrganizzataIn(){
if (VerificaMoltMinOrganizzataIn()==false)
throw new RuntimeException ("cardinalità minima violata");
else
return insieme_link.iterator();
}

 public boolean VerificaMoltMinOrganizzataIn(){
```

```
   return insieme_link.size()>=MINLINKORGANIZZATAIN;
   }

}
```

La classe Java LavoroScientifico

```
public class LavoroScientifico {
private final int id;
private String titolo;
private String email;
private Conferenza conf;/**rappresenta associazione sottomesso*/


/**costruttore*/
public LavoroScientifico (int i, String t, String e)
{id=i;
titolo=t;
email=e;
conf=null;

}
/**metodi gestione campi dati*/
public int getId(){return id;
}
public String getTitolo(){return titolo;
}
public String getEmail(){return email;
}
public void setTitolo(String t){titolo=t;
}
public void setEmail(String e){email=e;
}
/**realizzazione associazione sottomesso*/
public Conferenza getConferenza(){
```



```
 if (VerificaMoltMinSottomesso()==false)
 throw new RuntimeException ("MoltMinViolata");
 else return conf;
 }
 public void setConferenza(Conferenza c){
 conf=c;
 }
public boolean VerificaMoltMinSottomesso(){
return conf!=null;
}

}
```

La classe Java Data

```
public class Data implements Cloneable
{   private int giorno;
private int mese;
private int anno;

public Data(int g, int m, int a)
{giorno=g;
mese=m;
anno=a;
}

public int getGiorno()
{return giorno;
}
    public void setGiorno(int g)
{giorno=g;
}

    public int getMese()
{return mese;
}
    public void setMese(int m)
{mese=m;
}
    public int getAnno()
{return anno;
}
public void setAnno(int a)
```



```
{anno=a;
}
    /**funzioni ereditate da Object*/
public String toString() {
    return giorno + "/" + mese + "/" + anno;
  }
  public Object clone() {
    try {
      Data d = (Data)super.clone();
      return d;
    } catch (CloneNotSupportedException e) {
     throw new InternalError(e.toString());
    }
  }
  public boolean equals(Object o) {
    if (o != null && getClass().equals(o.getClass())) {
      Data d = (Data)o;
      return d.giorno == giorno && d.mese == mese && d.anno == anno;
    }
    else return false;
  }


}
```

La classe Java LavoroScientificoAccettato

```
public class LavoroScientificoAccettato extends LavoroScientifico
implements SupportoAssociazionePresenta
{ private final int numpagg;
private TipoLinkPresenta link; //*rappresentazione associazione presenta*/
    public LavoroScientificoAccettato(int i, String t, String e, int num)
{super(i,t,e);
numpagg = num;
link=null;
}

public int getNumPagg()
{ return numpagg;
}
/**realizzazione associazione presenta*/

public void inserisciLinkPresenta(AssociazionePresenta a){
if (a!=null && a.getLink().getLavoroScientificoAccettato()==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 TipoLinkPresenta getLinkPresenta(){
```



```
if (VerificaMoltMinPresenta()==false){
throw new RuntimeException("MoltMinViolata");
}
else return link;
}
public boolean VerificaMoltMinPresenta(){
   return link!=null;
   }

}
```

La classe Java Sessione

```
public class Sessione implements SupportoAssociazioneOrganizzataIn
{
protected String descrizione;
TipoLinkOrganizzataIn link; /**rappresenta associazione organizzataIn*/


public Sessione(String d)
{ descrizione=d;
link=null;
}

/**realizzazione associazione organizzataIn*/

public void inserisciLinkOrganizzataIn(AssociazioneOrganizzataIn a){
if (a!=null && a.getLink().getSessione()==this && link==null)
link=a.getLink();
else throw new RuntimeException ("inserimento link impossibile");
}
public void eliminaLinkOrganizzataIn(AssociazioneOrganizzataIn a){
if (a!=null && a.getLink().equals(link))
link=null;
else throw new RuntimeException ("eliminazione link impossibile");
}

public TipoLinkOrganizzataIn getLinkOrganizzataIn(){
if (VerificaMoltMinOrganizzataIn()==false){
throw new RuntimeException("MoltMinViolata");
}
```



```
else return link;
}
public boolean VerificaMoltMinOrganizzataIn(){
   return link!=null;
   }
}
```

La classe Java AssociazioneOrganizzataIn

```
public class AssociazioneOrganizzataIn
{private TipoLinkOrganizzataIn link;
private AssociazioneOrganizzataIn(TipoLinkOrganizzataIn x)
{
link = x;
}

public TipoLinkOrganizzataIn getLink()
{return link;
}

public static void inserisci(TipoLinkOrganizzataIn y)
{if(y.getConferenza()!=null && y.getSessione()!=null)
{AssociazioneOrganizzataIn k= new AssociazioneOrganizzataIn(y);
y.getConferenza().inserisciLinkOrganizzataIn(k);
y.getSessione().inserisciLinkOrganizzataIn(k);
}
}
public static void elimina(TipoLinkOrganizzataIn y)
{if(y.getSessione() !=null && y.getConferenza()!=null)
{AssociazioneOrganizzataIn k= new AssociazioneOrganizzataIn(y);
y.getConferenza().eliminaLinkOrganizzataIn(k);
y.getSessione().eliminaLinkOrganizzataIn(k);
}
}
}
```


La classe Java TipoLinkOrganizzataIn

```
public class TipoLinkOrganizzataIn
{

private final Conferenza laConferenza;
private final Sessione laSessione;

public TipoLinkOrganizzataIn(Conferenza x, Sessione y)
{ laConferenza=x;
laSessione=y;
}

public Conferenza getConferenza()
{
return laConferenza;
}
public Sessione getSessione()
{
return laSessione;
}

public boolean equals (Object o)
{
if (o!=null && getClass().equals(o.getClass()))
{TipoLinkOrganizzataIn b= (TipoLinkOrganizzataIn)o;
return b.laConferenza!=null && b.laSessione!= null && b.laConferenza==laConferenza &&b.laSessione==laSes
}
else return false;
}

}
```

L'interfaccia Java SupportoAssociazioneOrganizzataIn

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


La classe Java SessioneTecnica

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

public class SessioneTecnica extends Sessione
implements SupportoAssociazionePresenta
{ private final String titolo;
private Set insieme_link;

public SessioneTecnica(String d, String t)
{super(d);
titolo =t;
}

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

public Iterator getLinkPresenta(){
```



```
return insieme_link.iterator();
}

}
```

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

La classe Java TipoLinkPresenta

```
public class TipoLinkPresenta
{

private final SessioneTecnica laSessioneTecnica;
private final LavoroScientificoAccettato ilLavoroScientificoAccettato;
private final String ordine;

public TipoLinkPresenta(SessioneTecnica x, LavoroScientificoAccettato y, String o)
{ laSessioneTecnica=x;
ilLavoroScientificoAccettato=y;
ordine=o;
}

public SessioneTecnica getSessioneTecnica()
{
return laSessioneTecnica;
}
    public LavoroScientificoAccettato getLavoroScientificoAccettato()
{
return ilLavoroScientificoAccettato;
}
public String getOrdine()
{
return ordine;
}
public boolean equals (Object o)
{
if (o!=null && getClass().equals(o.getClass()))
{TipoLinkPresenta b= (TipoLinkPresenta)o;
```



```
return b.laSessioneTecnica!=null && b.ilLavoroScientificoAccettato!= null
&& b.laSessioneTecnica==laSessioneTecnica &&
b.ilLavoroScientificoAccettato==ilLavoroScientificoAccettato;
}
else return false;
}

}
```

L'interfaccia Java SupportoAssociazionePresenta

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


La classe Java Panel

```
import java.util.*;
import insiemearray.*;
public class Panel extends Sessione implements SupportoAssociazioneConsisteDi{
private final String titolo;
    private Set insieme_link; /**rappresenta associazione consisteDi*/
    public static final int MINLINKCONSISTEDI=1;

    public Panel(String d, String t)
{super(d);
titolo=t;
insieme_link=new InsiemeArrayOmogeneo(TipoLinkConsisteDi.class);}

public String getTitolo()
{ return titolo;
}

public void inserisciLinkConsisteDi(AssociazioneConsisteDi a){
if (a!=null && a.getLink().getPanel()==this)
insieme_link.add(a.getLink());
else throw new RuntimeException ("inserimento link impossibile");
}
public void eliminaLinkConsisteDi(AssociazioneConsisteDi a){
if (a!=null && a.getLink().getPanel()==this)
insieme_link.remove(a.getLink());
else throw new RuntimeException ("eliminazione link impossibile");
}

public Iterator getLinkConsisteDi(){
if (VerificaMoltMinConsisteDi()==false)
```

```
throw new RuntimeException ("cardinalità minima violata");
else
return insieme_link.iterator();
}

 public boolean VerificaMoltMinConsisteDi(){
   return insieme_link.size()>=MINLINKCONSISTEDI;
   }

}
```

La classe Java Partecipante

```
import java.util.*;
import insiemearray.*;
public class Partecipante implements SupportoAssociazioneConsisteDi
{
private final String nome;
private final String cognome;
private String  organizzazione;
private String ruolo;

public static final int MINLINKCONSISTEDI=1;
private Set  insieme_link; /**rappresenta associazione consisteDi*/
public Partecipante(String n,String c,String o,String r)
{
nome=n;
cognome=c;
organizzazione=o;
ruolo=r;
insieme_link=new InsiemeArrayOmogeneo(TipoLinkConsisteDi.class);
}

public void inserisciLinkConsisteDi(AssociazioneConsisteDi a){
if (a!=null && a.getLink().getPartecipante()==this)
insieme_link.add(a.getLink());
else throw new RuntimeException ("inserimento link impossibile");
}
public void eliminaLinkConsisteDi(AssociazioneConsisteDi a){
if (a!=null && a.getLink().getPartecipante()==this)
insieme_link.remove(a.getLink());
else throw new RuntimeException ("eliminazione link impossibile");
```



```
}

public Iterator getLinkConsisteDi(){
if (VerificaMoltMinConsisteDi()==false)
throw new RuntimeException ("cardinalità minima violata");
else
return insieme_link.iterator();
}

 public boolean VerificaMoltMinConsisteDi(){
   return insieme_link.size()>=MINLINKCONSISTEDI;
   }
}
```

La classe Java AssociazioneConsisteDi

```
public class AssociazioneConsisteDi
{private TipoLinkConsisteDi link;
private AssociazioneConsisteDi(TipoLinkConsisteDi x)
{
link = x;
}

public TipoLinkConsisteDi getLink()
{return link;
}

public static void inserisci(TipoLinkConsisteDi y)
{if(y.getPanel()!=null && y.getPartecipante()!=null)
{AssociazioneConsisteDi k= new AssociazioneConsisteDi(y);
y.getPanel().inserisciLinkConsisteDi(k);
y.getPartecipante().inserisciLinkConsisteDi(k);
}
}
public static void elimina(TipoLinkConsisteDi y)
{if(y.getPartecipante() !=null && y.getPanel()!=null)
{AssociazioneConsisteDi k= new AssociazioneConsisteDi(y);
y.getPanel().eliminaLinkConsisteDi(k);
y.getPartecipante().eliminaLinkConsisteDi(k);
}
}
}
```

La classe Java TipoLinkConsisteDi

```
public class TipoLinkConsisteDi
{

private final Panel ilPanel;
private final Partecipante ilPartecipante;

public TipoLinkConsisteDi(Panel x, Partecipante y)
{ ilPanel=x;
ilPartecipante=y;
}

public Panel getPanel()
{
return ilPanel;
}
public Partecipante getPartecipante()
{
return ilPartecipante;
}

public boolean equals (Object o)
{
if (o!=null && getClass().equals(o.getClass()))
{TipoLinkConsisteDi b= (TipoLinkConsisteDi)o;
return b.ilPanel!=null && b.ilPartecipante!= null && b.ilPanel==ilPanel &&b.ilPartecipante==ilPartecipar
}
else return false;
}

}
```


L'interfaccia Java SupportoAssociazioneConsisteDi

```
public interface SupportoAssociazioneConsisteDi
{void inserisciLinkConsisteDi(AssociazioneConsisteDi a);
void eliminaLinkConsisteDi(AssociazioneConsisteDi a);
}
```


Realizzazione in Java dello use case

```
import java.util.*;
import insiemearray.*;
public class UseCase
{
public static int CalcolaPanel(Conferenza c)
{
Iterator insiemeSessioni=c.getLinkOrganizzataIn();
int somma= 0;

while (insiemeSessioni.hasNext())
{TipoLinkOrganizzataIn link= (TipoLinkOrganizzataIn)insiemeSessioni.next();
Sessione sess=link.getSessione();
if (sess.getClass().equals(Panel.class))
somma++;
}
return somma;
}


public static String CreaElenco(SessioneTecnica s)
{
Iterator insiemeLavori=s.getLinkPresenta();
String result="";

while (insiemeLavori.hasNext())
{TipoLinkPresenta link= (TipoLinkPresenta)insiemeLavori.next();
LavoroScientificoAccettato la=link.getLavoroScientificoAccettato();
result=result+" "+la.getTitolo();
```



```
}
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");
  }
}
```