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

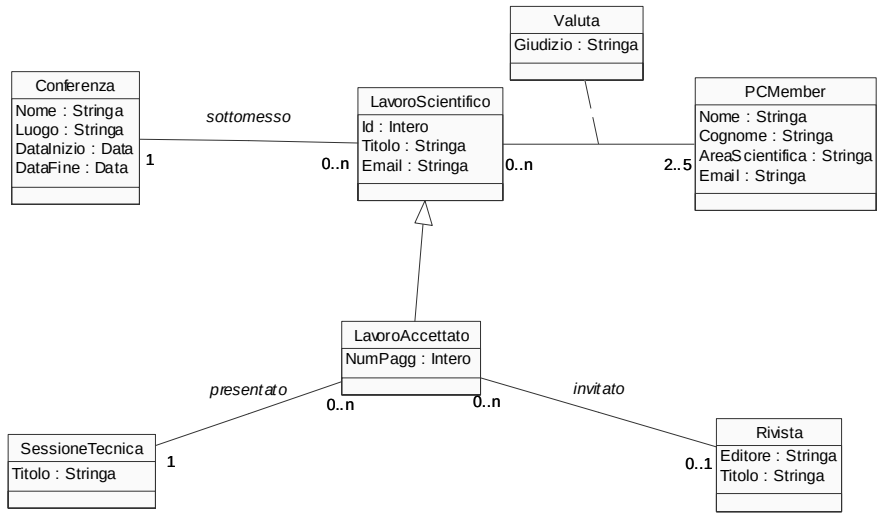
**SOLUZIONE ESAME DEL 15/04/2004**

Roma, 22 Aprile 2004

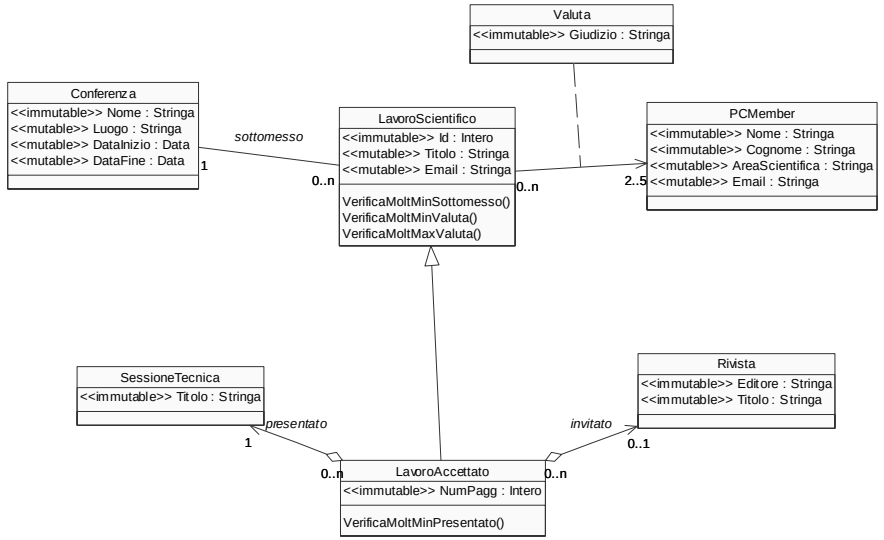


## Diagramma delle classi concettuale UML






## Diagramma delle classi realizzativo UML




# Specifica degli use case

```
InizioSpecificaUseCase VerificheConferenze

  percentualeLavoriAccettati(c: Conferenza):real
     pre:  nessuna
     post: result è la percentuale dei lavori sottomessi a c che sono stati accettati.

  verificaAreaScientifica(l: LavoroScientifico): booleano
     pre:  nessuna
     post: result e' true se tutti i PC Member che hanno valutato l appartengono alla stessa
     area scientifica, false altrimenti.


FineSpecifica
```



# Responsabilità sulle associazioni

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

- *Conferenza* ha responsabilità su *sottomesso*
- *LavoroScientifico* ha responsabilità su *sottomesso e valuta*
- *LavoroScientificoAccettato* ha responsabilità su *presentato*; si **assume** inoltre che abbia anche responsabilità su *invitato*
- le altre classi non hanno responsabilià.



# La classe Java Conferenza

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

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

/**costruttore*/
public Conferenza (String n, String l, Data i, Data f)
{nome=n;
luogo=l;
dataInizio=i;
dataFine=f;
insieme_link=new InsiemeArrayOmogeneo(TipoLinkSottomesso.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 sottomesso*/

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

public Iterator getLinkSottomesso(){
return insieme_link.iterator();
}
}
```

# La classe Java LavoroScientifico

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

public class LavoroScientifico implements SupportoAssociazioneSottomesso{
private final int id;
private String titolo;
private String email;
private Set  insieme_link; /**rappresenta associazione valuta*/
private TipoLinkSottomesso link;/**rappresenta associazione sottomesso*/


public static final int MINLINKVALUTA=2;
public static final int MAXLINKVALUTA=5;

/**costruttore*/
public LavoroScientifico (int i, String t, String e)
{id=i;
titolo=t;
email=e;
link=null;
insieme_link=new InsiemeArrayOmogeneo(TipoLinkValuta.class);
}
/**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 void inserisciLinkSottomesso(AssociazioneSottomesso a){
if (a!=null && a.getLink().getLavoroScientifico()==this && link==null)
link=a.getLink();
else throw new RuntimeException ("inserimento link impossibile");
}
public void eliminaLinkSottomesso(AssociazioneSottomesso a){
if (a!=null && a.getLink().equals(link))
link=null;
else throw new RuntimeException ("eliminazione link impossibile");
}

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


/**realizzazione associazione valuta*/
```

```
public void inserisciLinkValuta(TipoLinkValuta v){
if (v!=null && v.getLavoroScientifico()==this)
insieme_link.add(v);
else throw new RuntimeException ("inserimento link impossibile");
}
public void eliminaLinkValuta(TipoLinkValuta v){
if (v!=null)
insieme_link.remove(v);
else throw new RuntimeException ("eliminazione link impossibile");
}

public Iterator getLinkValuta(){
if (VerificaMoltMinValuta()==false||VerificaMoltMaxValuta()==false){
throw new RuntimeException("MoltViolata");
}
else
if (VerificaMoltMaxValuta()==false){
throw new RuntimeException("MoltMaxViolata");
}
else
return insieme_link.iterator();
}
 public boolean VerificaMoltMinValuta(){
   return insieme_link.size()>=MINLINKVALUTA;
   }
public boolean VerificaMoltMaxValuta(){
   return insieme_link.size()<=MAXLINKVALUTA;
   }

}
```

# 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 AssociazioneSottomesso

```
public class AssociazioneSottomesso
{private TipoLinkSottomesso link;
private AssociazioneSottomesso(TipoLinkSottomesso x)
{
link = x;
}

public TipoLinkSottomesso getLink()
{return link;
}

public static void inserisci(TipoLinkSottomesso y)
{if(y.getConferenza()!=null && y.getLavoroScientifico()!=null)
{AssociazioneSottomesso k= new AssociazioneSottomesso(y);
y.getConferenza().inserisciLinkSottomesso(k);
y.getLavoroScientifico().inserisciLinkSottomesso(k);
}
}
public static void elimina(TipoLinkSottomesso y)
{if(y.getLavoroScientifico() !=null && y.getConferenza()!=null)
{AssociazioneSottomesso k= new AssociazioneSottomesso(y);
y.getConferenza().eliminaLinkSottomesso(k);
y.getLavoroScientifico().eliminaLinkSottomesso(k);
}
}
}
```



## L'interfaccia Java SupportoAssociazioneSottomesso

```
public interface SupportoAssociazioneSottomesso
{void inserisciLinkSottomesso(AssociazioneSottomesso a);
void eliminaLinkSottomesso(AssociazioneSottomesso a);
}
```

# La classe Java TipoLinkSottomesso

```
public class TipoLinkSottomesso
{

private final Conferenza laConferenza;
private final LavoroScientifico ilLavoroScientifico;

public TipoLinkSottomesso(Conferenza x, LavoroScientifico y)
{ laConferenza=x;
ilLavoroScientifico=y;
}

public Conferenza getConferenza()
{
return laConferenza;
}
public LavoroScientifico getLavoroScientifico()
{
return ilLavoroScientifico;
}

public boolean equals (Object o)
{
if (o!=null && getClass().equals(o.getClass()))
{TipoLinkSottomesso b= (TipoLinkSottomesso)o;
return b.laConferenza!=null && b.ilLavoroScientifico!= null && b.laConferenza==laConferenza
&& b.ilLavoroScientifico==ilLavoroScientifico;
}
else return false;
```



```
}

}
```

# La classe Java TipoLinkValuta

```
public class TipoLinkValuta
{

private final PCMember IlPCMember;
private final LavoroScientifico ilLavoroScientifico;
private final String giudizio;

public TipoLinkValuta(PCMember x, LavoroScientifico y, String g)
{ IlPCMember=x;
ilLavoroScientifico=y;
giudizio=g;
}

public PCMember getPCMember()
{
return IlPCMember;
}
    public LavoroScientifico getLavoroScientifico()
{
return ilLavoroScientifico;
}
public String getGiudizio()
{
return giudizio;
}
public boolean equals (Object o)
{
if (o!=null && getClass().equals(o.getClass()))
{TipoLinkValuta b= (TipoLinkValuta)o;
```



```
return b.IlPCMember!=null && b.ilLavoroScientifico!= null && b.IlPCMember==IlPCMember
&& b.ilLavoroScientifico==ilLavoroScientifico;
}
else return false;
}

}
```

# La classe Java LavoroScientificoAccettato

```
public class LavoroScientificoAccettato extends LavoroScientifico
{

private final int numpagg;
private SessioneTecnica sess; //*rappresentazione associazione presenta*/
private Rivista riv; //*rappresentazione associazione invitato*/

public LavoroScientificoAccettato(int i, String t, String e, int num)
{super(i,t,e);
numpagg = num;
}

public int getNumPagg()
{ return numpagg;
}

 public Rivista getRivista(){
  return riv;
 }
 public void setRivista(Rivista r){
 riv=r;
 }

 public SessioneTecnica getSessioneTecnica(){
 if (VerificaMoltMinPresentato()==false)
 throw new RuntimeException ("MoltMinViolata");
 else return sess;
 }
 public void setSessioneTecnica(SessioneTecnica s){
```



```
 sess=s;
 }
public boolean VerificaMoltMinPresentato(){
return sess!=null;
}

}
```

# La classe Java SessioneTecnica

```
public class SessioneTecnica
{

private final String titolo;

public SessioneTecnica(String t)
{
titolo =t;
}

public String getTitolo()
{
return titolo;
}
}
```



# La classe Java Rivista

```
public class Rivista
{
private final String titolo;
private final String editore;

public Rivista(String t, String e)
{
titolo =t;
editore=e;
}

public String getTitolo()
{
return titolo;
}

public String getEditore()
{
return editore;
}

}
```

# Realizzazione in Java dello use case

```
import java.util.*;
import insiemearray.*;
public class UseCase
{public static double PercentualeLavoriAccettati(Conferenza c)
{

Iterator insiemeLavoriSottomessi=c.getLinkSottomesso();
int somma= 0;
double cont=0;
while (insiemeLavoriSottomessi.hasNext())
{TipoLinkSottomesso link= (TipoLinkSottomesso)insiemeLavoriSottomessi.next();
LavoroScientifico ls=link.getLavoroScientifico();
if (ls.getClass().equals(LavoroScientificoAccettato.class))
somma++;
cont++;
}
return (somma/cont)*100;
}
public static boolean VerificaAreaAppartenenza(LavoroScientifico l)
{
Iterator insiemeValutatori=l.getLinkValuta();
TipoLinkValuta link= (TipoLinkValuta)insiemeValutatori.next();
String area1=link.getPCMember().getAreaInteresse();
while (insiemeValutatori.hasNext())
{TipoLinkValuta link2= (TipoLinkValuta)insiemeValutatori.next();
String area2=link2.getPCMember().getAreaInteresse();
if (!area1.equals(area2))
return false;
}
```



```
return true;
}
}
```

# 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");
  }
}
```