Università di Roma "La Sapienza", Facoltà di Ingegneria

Corso di
"PROGETTAZIONE DEL SOFTWARE I" (Ing. Informatica)
Proff. Marco Cadoli e Maurizio Lenzerini, Canali A-L & M-Z
A.A. 2003-04

Compito d'esame del 16 aprile 2004

SOLUZIONE

Requisiti

L'applicazione da progettare riguarda le informazioni su aziende e contenziosi tra aziende. Di ogni azienda interessa il nome e l'anno di fondazione (che non possono cambiare). Ogni contenzioso è caratterizzato dall'azienda accusatrice, l'azienda accusata, la persona eletta per operare come giudice, e l'anno di inizio. Si noti che una stessa persona non può fare da giudice in due contenziosi che coinvolgono la stessa azienda accusatrice e la stessa azienda accusata. Di ogni persona interessa il codice fiscale, il nome, il cognome (queste tre proprietà non possono ovviamente cambiare), ed i contenziosi di cui è giudice.

Esistono esattamente tre categorie di aziende, che sono fra loro disgiunte: piccole, medie e grandi. Di ogni azienda piccola interessa la città in cui ha sede. Di ogni azienda media interessa conoscere le persone che hanno stipulato un contratto di lavoro con esse, con l'anno di inizio del contratto e lo stipendio previsto. Si noti che una stessa persona può aver stipulato in anni diversi più contratti con la stessa azienda. Di ogni grande azienda interessa conoscere la persona che eventualmente funge da presidente.

Requisiti (cont.)

L'ufficio giudiziario del lavoro deve poter effettuare, come cliente della nostra applicazione, dei controlli sui contenziosi. A questo scopo, si faccia riferimento ad uno use case che prevede le seguenti operazioni:

- Dato un insieme S di aziende, restituire il sottoinsieme di S formato da tutte le aziende che sono accusatrici in almeno due contenziosi con la stessa azienda accusata.

- Data una grande azienda A ed una persona P, dire se A è accusata in almeno un contenzioso in cui P opera come giudice.

Fase di analisi

Diagramma delle classi

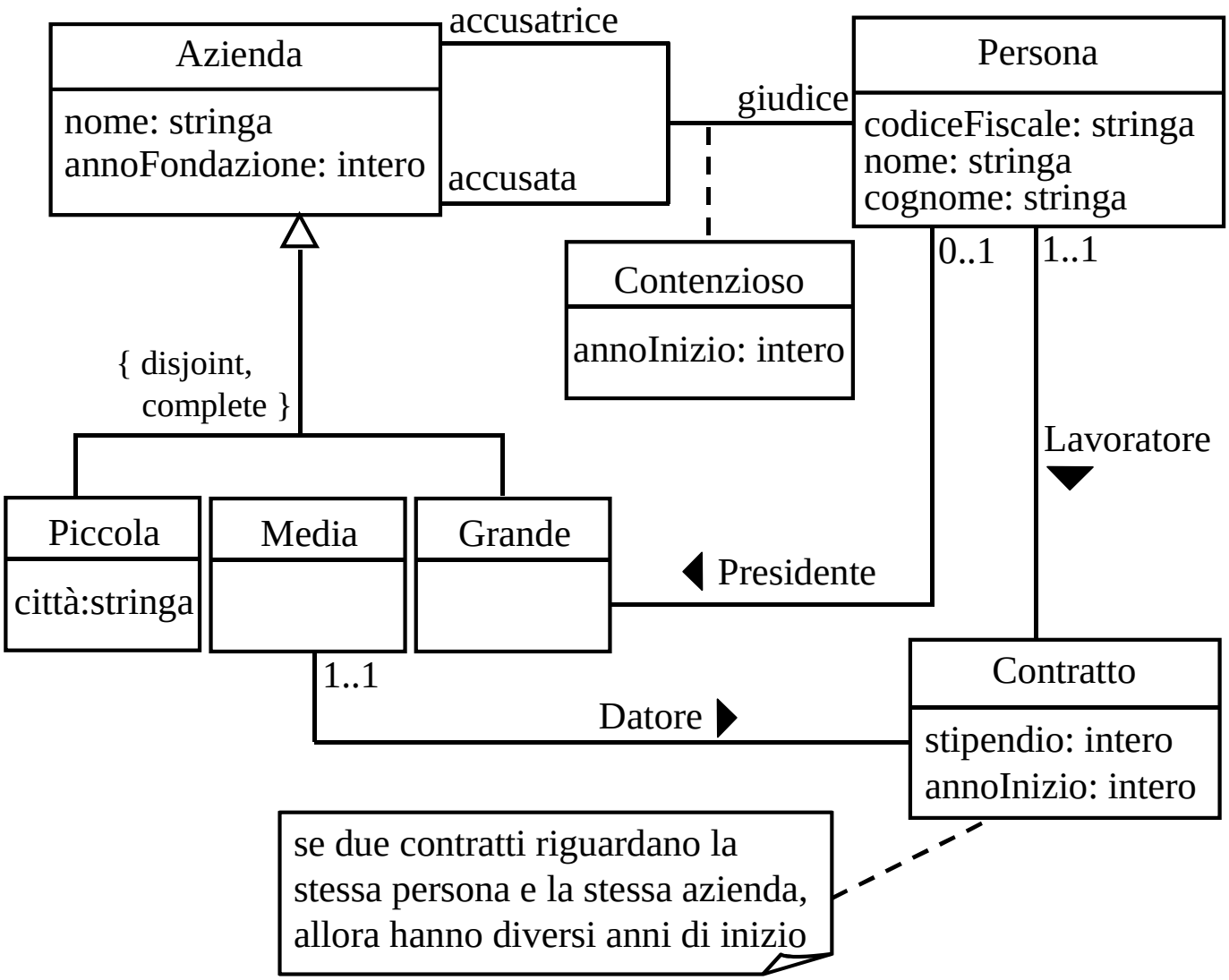
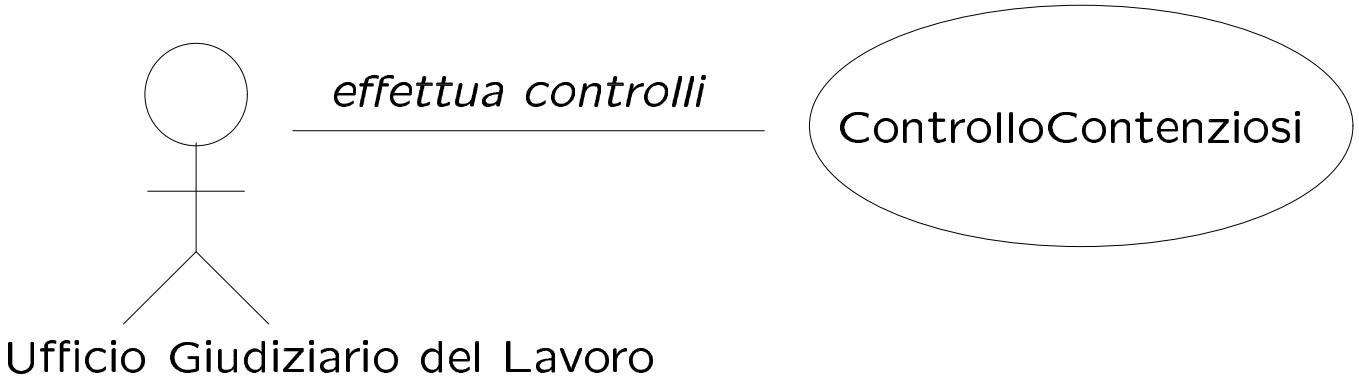


Diagramma degli use case



Specifica dello use case

InizioSpecificaUseCase ControlloContenziosi

DueContenziosi *(S: Insieme(Azienda)): Insieme(Azienda)*

pre: nessuna

post: *result* è il sottoinsieme di *S* formato da tutte le aziende che sono accusatrici in almeno due contenziosi con la stessa azienda accusata

Accusata *(A: Grande, P: Persona): booleano*

pre: nessuna

post: *result* è true se *A* è accusata in almeno un contenzioso in cui *P* opera come giudice

FineSpecifica

Fase di progetto

Algoritmi per le operazioni dello use-case

Per l'operazione *DueContenziosi(S)* adottiamo il seguente algoritmo:

```
Insieme(Azienda) ins = insieme vuoto;
per ogni azienda a di S
    Insieme(Link di tipo Contenzioso) s1 = i link di tipo Contenzioso che
                                           coinvolgono a come accusatrice
    Insieme(Azienda) s2 = insieme vuoto;
    per ogni t di s1
         se l'azienda accusata di t è contenuta in s2
         allora inserisci a in ins
         altrimenti inserisci l'azienda accusata di t in s2
return ins;
```

Per l'operazione *Accusata(A,P)* adottiamo il seguente algoritmo:

```
Insieme(Link di tipo Contenzioso) c = i link di tipo Contenzioso che
                                      coinvolgono A come accusatrice
per ogni elemento e di c
    se (la persona che giudica in e è P)
    allora return true;
return false;
```

Responsabilità sulle associazioni

La seguente tabella delle responsabilità si evince da:

1. i requisiti,
2. i vincoli di molteplicità nel diagramma delle classi,
3. la specifica degli algoritmi per le operazioni dello use-case.

Associazione	Ruolo/Classe	ha resposabilitá
Contenzioso	*Accusatrice/Azienda*	SI[3]
Contenzioso	*Accusata/Azienda*	SI[3]
Contenzioso	*Giudice/Persona*	SI[1]
Presidente	*Persona*	NO
Presidente	*Grande*	SI[1]
Datore	*Media*	SI[1]
Datore	*Contratto*	SI[2]
Lavoratore	*Persona*	NO
Lavoratore	*Contratto*	SI[2]

Corrispondenza fra tipi UML e Java

Riassumiamo le nostre scelte nella seguente tabella di corrispondenza dei tipi UML.

Tipo UML	Rappresentazione in Java
intero	`int`
stringa	`String`
booleano	`boolean`
Insieme	`InsiemeListaOmogeneo`

Strutture di dati

Abbiamo la necessità di rappresentare collezioni omogenee di oggetti, a causa:

- dei vincoli di molteplicità 0..* di alcune associazioni,
- dei parametri di una operazione dello use-case,
- delle variabili locali necessarie per un algoritmo.

Per fare ciò, utilizzeremo la classe Java `InsiemeListaOmogeneo`.

API per le strutture di dati

```
// File insiemelista/InsiemeListaOmogeneo.java

package insiemelista;

public class InsiemeListaOmogeneo extends InsiemeLista {
  public InsiemeListaOmogeneo(Class cl)
  public InsiemeListaOmogeneo()
  public int size()
  public boolean isEmpty()
  public boolean contains(Object e)
  public boolean add(Object e)
  public boolean remove(Object e)
  public Iterator iterator()
  public boolean containsAll(Collection c)
  public Object[] toArray()
  public Object[] toArray(Object[] a)
  public boolean equals(Object o)
  public Object clone()
  public String toString()
}
```

Proprietà immutabili di classi UML

Dai requisiti evinciamo che le proprietà immutabili sono quelle riassunte nella seguente tabella. Dai requisiti si deduce anche tali proprietà sono tutte note al momento della nascita degli oggetti.

Classe UML	Proprietà immutabile
Persona	*nome*
	cognome
	codiceFiscale
Azienda	*nome*
	annoFondazione
Contratto	*annoInizio*
	stipendio
	Datore
	Lavoratore

Sequenza di nascita degli oggetti

Quando viene definito un contratto sono noti entrambi i contraenti, e non possono cambiare (vedi tabella proprietà immutabili). Ne segue che:

- quando nasce un oggetto Java corrispondente ad un contratto è nota l'azienda in associazione *Datore*;
- quando nasce un oggetto Java corrispondente ad un contratto è nota la persona in associazione *Lavoratore*;
- poiché la responsabilità su *Lavoratore* è della sola classe *Contratto*, per rispettare i vincoli di cardinalità massima e minima è sufficiente non offrire alcuna funzione di inserimento e cancellazione dei link di questo tipo nella classe Java `Contratto`;

Sequenza di nascita degli oggetti (cont.)

- poiché la responsabilità su *Datore* è doppia, per rispettare i vincoli di cardinalità massima e minima occorre:
  - evitare di definire la funzione di cancellazione di link di questo tipo nella classe Java `AssociazioneDatore` (e quindi anche nella classe `Media` e nella classe `Contratto`);
  - fare in modo che la funzione di inserimento della classe Java `AssociazioneDatore` esegua l'inserimento solo se il contratto non ha già un datore, in particolare al momento della creazione di oggetti di tipo `Contratto`, con una chiamata della funzione di inserimento da parte del costruttore della classe Java `Contratto`). Al contrario, fare in modo che la funzione di inserimento della classe Java `AssociazioneDatore` lanci una eccezione (di tipo `EccezionCardMax`) se il contratto che si vuole inserire ha già un contratto di lavoro.

Rappresentazione degli stati in Java

Passo non significativo.

API delle classi Java progettate

Per semplicità, le omettiamo.

Ulteriori considerazioni

Il vincolo di integrità specificato nel diagramma delle classi mediante il commento impone che se due contratti riguardano la stessa persona e la stessa azienda, allora hanno diversi anni di inizio.

Per fare in modo che questo vincolo sia sempre rispettato, al momento della creazione di un contratto (tramite il costruttore) con azienda *az*, lavoratore *lav* e anno *an*, si analizzano i contratti che coinvolgono *az*, e si lancia una eccezione (di tipo `EccezioneContratto`) se tra tali contratti ce n'è già uno che coinvolge lo stesso lavoratore *lav* e lo stesso anno *an*.

Struttura dei file e dei package

```
+---AppContenziosi
|    |   AssociazioneContenzioso.java
|    |   AssociazioneDatore.java
|    |   Contratto.java
|    |   ControlloContenziosi.java
|    |   EccezioneCardMax.java
|    |   EccezioneContratto.java
|    |   MainContenziosi.java
|    |   Persona.java
|    |   TipoLinkContenzioso.java
|    |   TipoLinkDatore.java
|    |
|    +---Azienda
|    |       Azienda.java
|    |
|    +---Grande
|    |       Grande.java
|    |
|    +---Media
|    |       Media.java
|    |
|    \---Piccola
|            Piccola.java
|
\---insiemelista
        InsiemeLista.java
        InsiemeListaOmogeneo.java
        IteratorInsiemeLista.java
```

Fase di realizzazione

Considerazioni iniziali

Dalle fasi precedenti traiamo come conseguenza che dobbiamo realizzare:

1. sei classi UML;
2. una associazione a responsabilità singola e con vincoli di molteplicità 0..* e 0..1:

   una associazione a responsabilità singola e con vincoli di molteplicità 0..* e 1..1;

   una associazione a responsabilità doppia e con vincoli di molteplicità 0..* e 1..1;

   una associazione con attributi e a responsabilità tripla;
3. uno use case.

Per facilitare la fase di test e debugging, prevediamo l'overriding della funzione `toString()` per ognuna delle classi Java di cui al punto 1.

La classe Java Persona

```
// File AppContenziosi/Persona.java
package AppContenziosi;
import insiemelista.*;

public class Persona {
    private final String nome, cognome, codiceFiscale;
    private InsiemeListaOmogeneo linkContenzioso;
    public Persona(String no, String co, String cf) {
        nome = no;
        cognome = co;
        codiceFiscale = cf;
        linkContenzioso = new InsiemeListaOmogeneo(TipoLinkContenzioso.class);
    }
    public String getNome() { return nome; }
    public String getCognome() { return cognome; }
    public String getCodiceFiscale() { return codiceFiscale; }
    public InsiemeListaOmogeneo getContenzioso() {
        return (InsiemeListaOmogeneo)linkContenzioso.clone();
    }
    public void inserisciLinkContenzioso(AssociazioneContenzioso a) {
        if (a != null) { linkContenzioso.add(a.getLink()); }
    }
    public void eliminaLinkContenzioso(AssociazioneContenzioso a) {
        if (a != null) { linkContenzioso.remove(a.getLink()); }
```

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

La classe Java Azienda

```
// File AppContenziosi/Azienda/Azienda.java
package AppContenziosi.Azienda;
import AppContenziosi.*;
import insiemelista.*;

public abstract class Azienda {
    protected final String nome;
    protected final int annoFondazione;
    protected InsiemeListaOmogeneo linkAccusatrice;
    protected InsiemeListaOmogeneo linkAccusata;
    public Azienda(String no, int an) {
        nome = no;
        annoFondazione = an;
        linkAccusatrice = new InsiemeListaOmogeneo(TipoLinkContenzioso.class);
        linkAccusata= new InsiemeListaOmogeneo(TipoLinkContenzioso.class);
    }
    public String getNome() { return nome; }
    public int getAnnoFondazione() { return annoFondazione; }
    public InsiemeListaOmogeneo getAccusatrice() {
      return (InsiemeListaOmogeneo)linkAccusatrice.clone();
    }
    public InsiemeListaOmogeneo getAccusata() {
      return (InsiemeListaOmogeneo)linkAccusata.clone();
    }
```

```
    public void inserisciLinkAccusatrice(AssociazioneContenzioso a) {
      if (a != null) { linkAccusatrice.add(a.getLink()); }
    }
    public void eliminaLinkAccusatrice(AssociazioneContenzioso a) {
      if (a != null) { linkAccusatrice.remove(a.getLink()); }
    }
    public void inserisciLinkAccusata(AssociazioneContenzioso a) {
      if (a != null) { linkAccusata.add(a.getLink()); }
    }
    public void eliminaLinkAccusata(AssociazioneContenzioso a) {
      if (a != null) { linkAccusata.remove(a.getLink()); }
    }
    public String toString() {
        return nome + " " + annoFondazione;
    }
}
```

La classe Java Grande

```
// File AppContenziosi/Grande/Grande.java
package AppContenziosi.Grande;
import AppContenziosi.*;
import AppContenziosi.Azienda.*;

public class Grande extends Azienda {
    protected Persona presidente;
    public Grande(String no, int an) {
        super(no,an);
    }
    public Persona getPresidente() {
        return presidente;
    }
    public void setPresidente(Persona p) {
        presidente = p;
    }
    public String toString() {
        return super.toString() + " " + presidente;
    };
}
```

La classe Java Media

```
// File AppContenziosi/Media/Media.java
package AppContenziosi.Media;
import AppContenziosi.*;
import AppContenziosi.Azienda.*;
import insiemelista.*;

public class Media extends Azienda {
    protected InsiemeListaOmogeneo linkDatore;
    public Media(String no, int an) {
        super(no,an);
        linkDatore = new InsiemeListaOmogeneo(TipoLinkDatore.class);
    }
    public InsiemeListaOmogeneo getLinkDatore() {
        return (InsiemeListaOmogeneo)linkDatore.clone();
    }
    public void inserisciLinkDatore(AssociazioneDatore a) {
        if (a != null) linkDatore.add(a.getLink());
    }
    public String toString() {
        return super.toString();
    };
}
```

La classe Java Piccola

```
// File AppContenziosi/Piccola/Piccola.java
package AppContenziosi.Piccola;
import AppContenziosi.*;
import AppContenziosi.Azienda.*;

public class Piccola extends Azienda {
    protected String citta;
    public Piccola(String no, int an, String c) {
        super(no,an);
        citta = c;
    }
    public String getCitta() {
        return citta;
    }
    public void setCitta(String c) {
            citta = c;
    }
    public String toString() {
        return super.toString() + " " + citta;
    };
}
```

La classe Java Contratto

```
// File AppContenziosi/Contratto.java
package AppContenziosi;
import AppContenziosi.Media.*;

public class Contratto {
    private final int annoInizio, stipendio;
    private Persona lavoratore;
    private TipoLinkDatore datore;
    public Contratto(int ai, int stip, Persona lav, Media dat) throws EccezioneContratto,
       EccezioneCardMax {
        annoInizio = ai;
        stipendio = stip;
        lavoratore = lav;
        datore = null;
        AssociazioneDatore.inserisci(new TipoLinkDatore(dat,this));
    }
    public int getAnnoInizio() { return annoInizio; }
    public int getStipendio() { return stipendio; }
    public Persona getLavoratore() { return lavoratore; }
    public TipoLinkDatore getDatore() { return datore; }
    public void inserisciLinkDatore(AssociazioneDatore a) {
            if (a != null) datore = a.getLink();
    }
    public String toString() {
```

```
        return lavoratore  + " " + datore.getAzienda() + " " +
            annoInizio + " " + stipendio;
    }
}
```

La classe Java TipoLinkContenzioso

```
// File AppContenziosi/TipoLinkContenzioso.java
package AppContenziosi;
import AppContenziosi.Azienda.*;

public class TipoLinkContenzioso {
    private final Azienda laAccusatrice;
    private final Azienda laAccusata;
    private final Persona ilGiudice;
    private final int annoInizio;
    public TipoLinkContenzioso(Azienda a, Azienda t, Persona g, int anno) {
        laAccusatrice = a; laAccusata = t; ilGiudice = g; annoInizio = anno;
    }
    public boolean equals(Object o) {
        if (o != null && getClass().equals(o.getClass())) {
            TipoLinkContenzioso b = (TipoLinkContenzioso)o;
            return b.laAccusatrice != null && b.laAccusata != null &&
                   b.ilGiudice != null && b.laAccusatrice == laAccusatrice &&
                   b.laAccusata == laAccusata && b.ilGiudice == ilGiudice;
        }
        else return false;
    }
    public Azienda getAccusatrice() { return laAccusatrice; }
    public Azienda getAccusata() { return laAccusata; }
    public Persona getGiudice() { return ilGiudice; }
```

```
    public int getAnnoInizio() { return annoInizio; }
    public String toString() {
        return laAccusatrice + " " + laAccusata + " " + ilGiudice + " " + annoInizio;
    }
}
```

La classe Java AssociazioneContenzioso

```
// File AppContenziosi/AssociazioneContenzioso.java
package AppContenziosi;

public class AssociazioneContenzioso {
    private AssociazioneContenzioso(TipoLinkContenzioso x) { link = x; }
    private TipoLinkContenzioso link;
    public TipoLinkContenzioso getLink() { return link; }
    public static void inserisci(TipoLinkContenzioso y) {
        if (y != null && y.getAccusatrice() != null && y.getAccusata() != null &&
            y.getGiudice() != null) {
            AssociazioneContenzioso k = new AssociazioneContenzioso(y);
            y.getAccusatrice().inserisciLinkAccusatrice(k);
            y.getAccusata().inserisciLinkAccusata(k);
            y.getGiudice().inserisciLinkContenzioso(k);
        }
    }
    public static void elimina(TipoLinkContenzioso y) {
        if (y != null && y.getAccusatrice() != null && y.getAccusata() != null &&
            y.getGiudice() != null) {
            AssociazioneContenzioso k = new AssociazioneContenzioso(y);
            y.getAccusatrice().eliminaLinkAccusatrice(k);
            y.getAccusata().eliminaLinkAccusata(k);
            y.getGiudice().eliminaLinkContenzioso(k);
        }
```

```
    }
}
```

Le classi Java per le eccezioni

```
// File AppContenziosi/EccezioneContratto.java
package AppContenziosi;

public class EccezioneContratto extends Exception {
  private String messaggio;
  public EccezioneContratto(String m) {
    messaggio = m;
  }
  public String toString() {
    return messaggio;
  }
}
```

```
// File AppContenziosi/EccezioneCardMax.java
package AppContenziosi;

public class EccezioneCardMax extends Exception {
  private String messaggio;
  public EccezioneCardMax(String m) {
    messaggio = m;
  }
  public String toString() {
    return messaggio;
  }
}
```

La classe Java AssociazioneDatore

```
// File AppContenziosi/AssociazioneDatore.java
package AppContenziosi;
import java.util.*;
import insiemelista.*;

public class AssociazioneDatore {
    private AssociazioneDatore(TipoLinkDatore x) { link = x; }
    private TipoLinkDatore link;
    public TipoLinkDatore getLink() { return link; }
    public static void inserisci(TipoLinkDatore y) throws EccezioneContratto,
                                                          EccezioneCardMax {
        if (y != null && y.getAzienda() != null && y.getContratto() != null)
            if (y.getContratto().getDatore() == null) {
            AssociazioneDatore k = new AssociazioneDatore(y);
            InsiemeListaOmogeneo s = y.getAzienda().getLinkDatore();
            Iterator j = s.iterator();
            while (j.hasNext()) {
                TipoLinkDatore t = (TipoLinkDatore)j.next();
                if (t.getContratto().getLavoratore() == y.getContratto().getLavoratore() &&
                    t.getContratto().getAnnoInizio() == y.getContratto().getAnnoInizio())
                    throw new EccezioneContratto
                        ("Esiste gia' contratto con stesso datore, lavoratore e anno");
            }
            y.getAzienda().inserisciLinkDatore(k);
```

```
            y.getContratto().inserisciLinkDatore(k);
            }
            else throw new EccezioneCardMax("Il contratto ha gia' un datore");
    }
}
```

La classe Java TipoLinkDatore

```
// File AppContenziosi/TipoLinkDatore.java
package AppContenziosi;
import AppContenziosi.Media.*;

public class TipoLinkDatore {
    private final Media laAzienda;
    private final Contratto ilContratto;
    public TipoLinkDatore(Media m, Contratto c) {
        laAzienda = m; ilContratto = c;
    }
    public boolean equals(Object o) {
        if (o != null && getClass().equals(o.getClass())) {
            TipoLinkDatore b = (TipoLinkDatore)o;
            return b.laAzienda != null && b.ilContratto != null &&
                   b.laAzienda == laAzienda && b.ilContratto == ilContratto;
        }
        else return false;
    }
    public Media getAzienda() { return laAzienda; }
    public Contratto getContratto() { return ilContratto; }
    public String toString() {
        return laAzienda + " " + ilContratto;
    }
}
```

Realizzazione in Java dello use case

```
// File AppContenziosi/ControlloContenziosi.java
package AppContenziosi;
import java.util.*;
import insiemelista.*;
import AppContenziosi.Azienda.*;
import AppContenziosi.Grande.*;

public final class ControlloContenziosi {
    private ControlloContenziosi() { }
    public static InsiemeListaOmogeneo DueContenziosi(InsiemeListaOmogeneo i) {
        InsiemeListaOmogeneo ins = new InsiemeListaOmogeneo(Azienda.class);
        Iterator it = i.iterator();
        while(it.hasNext()) {
            Azienda a = (Azienda)it.next();
            InsiemeListaOmogeneo s1 = (InsiemeListaOmogeneo)a.getAccusatrice();
            InsiemeListaOmogeneo s2 = new InsiemeListaOmogeneo(Azienda.class);
            Iterator j = s1.iterator();
            while (j.hasNext()) {
                 TipoLinkContenzioso t = (TipoLinkContenzioso)j.next();
                 if (s2.contains(t.getAccusata()))
                     ins.add(a);
                 else s2.add(t.getAccusata());
            }
        }
```

```
        return ins;
    }
    public static boolean Accusata(Grande a, Persona p) {
        InsiemeListaOmogeneo s = a.getAccusata();
        Iterator it = s.iterator();
        while(it.hasNext()) {
            TipoLinkContenzioso c = (TipoLinkContenzioso)it.next();
            if (c.getGiudice() == p) return true;
        }
        return false;
    }
}
```