Università di Roma "La Sapienza", Facoltà di Ingegneria

Corso di
"PROGETTAZIONE DEL SOFTWARE" (Ing. Gestionale)
Prof. Giuseppe De Giacomo, A.A. 2001-02

SOLUZIONE ESAME DEL 18/09/2002

Roma, 18 settembre 2002

Diagramma delle classi UML

```
//UML-class-diagram.txt
```

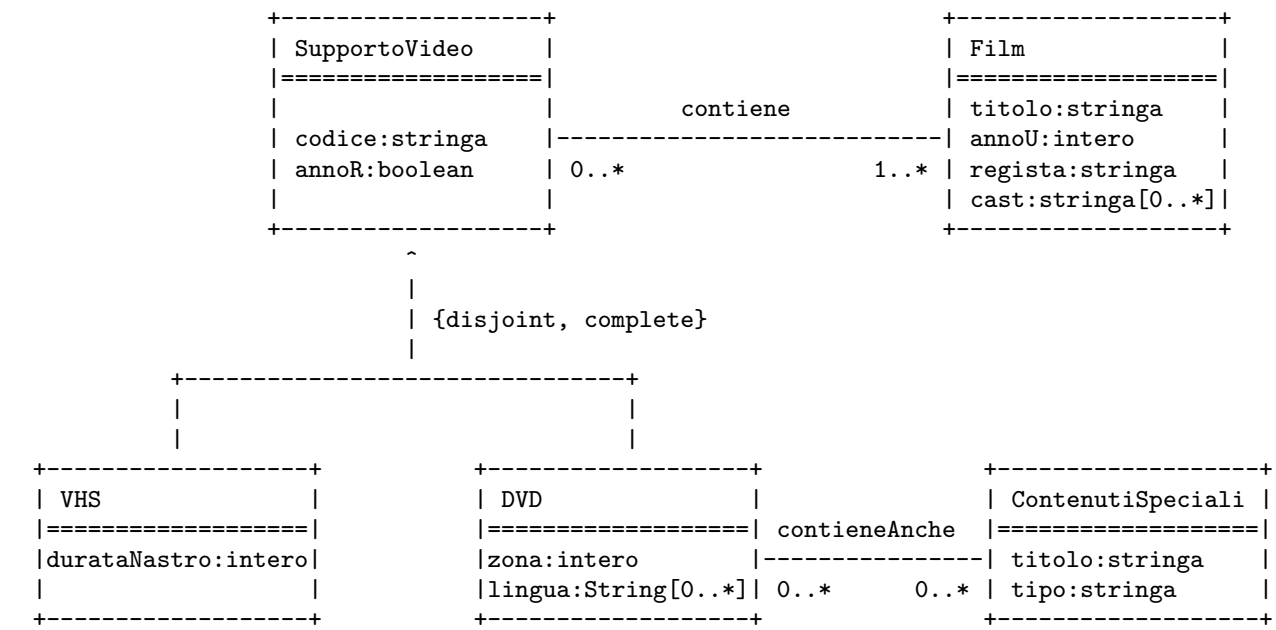

Diagramma e specifica degli use case

```
//UML-usecase-diagram.txt

         _
        |_|              /--------------\
        -|-  ------------| Verifiche    |
        / \              | Video        |
      Gestore            \--------------/
  servizio archiviazione


InizioSpecificaUseCase VerificheVideo

  estVHSeDVD(f: Film): booleano
     pre:  nessuna
     post: result e' true se il film f e' contenuto sia su una cassetta VHS
           che su un DVD

  getCodiciDVD(f: Film): Insieme(Stringa)
     pre:  nessuna
     post: result e' l'insieme dei codici dei DVD che contengono il film f

FineSpecifica
```


Responsabilità sulle associazioni

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

- entrambe *SupportoVideo* e *Film* hanno entrambe responsabilità su *contiene*;
- inoltre *DVD* ha responsabilità su *contieneAncora*;
- le altre classi non hanno responsabilità sulle associazioni.

La classe Java Film

```
// File Film.java

public class Film {

    //Rappr. attributi
    private final String titolo;
    private final int annoUscita;
    private final String regista;
    private InsiemeSS cast; //questo e' un attributo multivalore

    //Rappr. associazione "contiene"
    private InsiemeSS insieme_link;

    //Costruttore
    public Film(String t, int a, String r) {
        titolo = t;
        annoUscita = a;
regista = r;
cast = new InsiemeSS(String.class);
insieme_link = new InsiemeSS(TipoLinkContiene.class);
    }

    //Accesso agli attributi
    public String getTitolo() { return titolo; }
    public int getAnnoUscita() { return annoUscita; }
    public String getRegista() { return regista; }

    public InsiemeSS getCast() { return (InsiemeSS)cast.clone(); }
    public void inserisciInCast(String a) { cast.inserisci(a); }
```



```
    public void eliminaDaCast(String a) { cast.elimina(a); }


    //Accesso alla associazione "contiene"
    public InsiemeSS getLinksContiene() {
        return (InsiemeSS)insieme_link.clone();
    }

    //funzioni ausiliarie per l'inserimento e la cancellazione
    //che devono essere richamate dalle funzioni d'inserimento ed eliminazione
    //della classe AssociazioneContiene
    public void inserisciLinkContiene(AssociazioneContiene a) {
        if (a != null  && a.getLink().getFilm() == this)
            insieme_link.inserisci(a.getLink());
    }
    public void eliminaLinkContiene(AssociazioneContiene a) {
        if (a != null  && a.getLink().getFilm() == this)
            insieme_link.elimina(a.getLink());
    }

    //Overriding delle funzione speciali (cioe' solo toString)
    public String toString() {
        return "Film " + titolo;
    }
}
```

La classe Java SupportoVideo

```
// File SupportoVideo.java

abstract public class SupportoVideo {

    //Rappr. attributi
    private final String codice;
    private final int annoRealizzazione;

    //Rappr. associazione "contiene"
    private InsiemeSS insieme_link;

    //Costruttore
    public SupportoVideo(String c, int a) {
        codice = c;
        annoRealizzazione = a;
insieme_link = new InsiemeSS(TipoLinkContiene.class);
    }

    //Accesso agli attributi
    public String getCodice() { return codice; }
    public int getAnnoRealizzazione() { return annoRealizzazione; }

    //Accesso alla associazione "contiene"
    public InsiemeSS getLinksContiene() {
        return (InsiemeSS)insieme_link.clone();
    }

    //funzioni ausiliarie per l'inserimento e la cancellazione
    //che devono essere richamate dalle funzioni d'inserimento ed eliminazione
```



```
    //della classe AssociazioneContiene
    public void inserisciLinkContiene(AssociazioneContiene a) {
        if (a != null  && a.getLink().getSupportoVideo() == this)
            insieme_link.inserisci(a.getLink());
    }
    public void eliminaLinkContiene(AssociazioneContiene a) {
        if (a != null  && a.getLink().getSupportoVideo() == this)
            insieme_link.elimina(a.getLink());
    }

    //Overriding delle funzione speciali (cioe' solo toString)
    public String toString() {
        return "SupportoVideo " + codice;
    }
}
```

La classe Java TipoLinkContiene

```
// File TipoLinkContiene.java

public class TipoLinkContiene {

    //Rappr. componenti della tupla
    private final SupportoVideo ilSupportoVideo;
    private final Film ilFilm;

    //Rappr. attributi della tupla
    //non ce ne sono

    //Costruttore
    public TipoLinkContiene(SupportoVideo x, Film y) {
        ilSupportoVideo = x;
        ilFilm = y;
    }

    //Accesso alle componenti
    public SupportoVideo getSupportoVideo() { return ilSupportoVideo; }
    public Film getFilm() { return ilFilm; }

    //Overriding di funzioni speciali ereditate da Object
    //Nota TipoLinkContiene e' un tipo non una classe:
    //va rdefinito equals per effettuare test di uguaglianza profonda
    public boolean equals(Object o) {
        if (o != null && getClass().equals(o.getClass())) {
            TipoLinkContiene b = (TipoLinkContiene)o;
            return b.ilSupportoVideo != null && b.ilFilm != null &&
                b.ilSupportoVideo == ilSupportoVideo &&
```



```
                b.ilFilm == ilFilm;
        }
        else return false;
    }
}
```

La classe Java AssociazioneContiene

```
// File AssociazioneContiene.java

public class AssociazioneContiene {
    private AssociazioneContiene(TipoLinkContiene x) { link = x; }
    private TipoLinkContiene link;
    public TipoLinkContiene getLink() { return link; }
    public static void inserisci(TipoLinkContiene y) {
        if (y.getSupportoVideo() != null && y.getFilm() != null) {
            AssociazioneContiene k = new AssociazioneContiene(y);
            y.getSupportoVideo().inserisciLinkContiene(k);
            y.getFilm().inserisciLinkContiene(k);
        }
    }
    public static void elimina(TipoLinkContiene y) {
        if (y.getSupportoVideo() != null && y.getFilm() != null) {
            AssociazioneContiene k = new AssociazioneContiene(y);
            y.getSupportoVideo().eliminaLinkContiene(k);
            y.getFilm().eliminaLinkContiene(k);
        }
    }
}
```


La classe Java VHS

```
// File VHS.java
public class VHS extends SupportoVideo {
    private final int durata;
    public VHS(String c, int a, int d) {
        super(c,a);
        durata = d;
    }
    public int getDurataNastro() { return durata; }
}
```

La classe Java DVD

```
// File DVD.java
public class DVD extends SupportoVideo {

    //Rappr. attributi
    private final int tipo;
    private InsiemeSS lingue;

    //Rappr. associazione "contieneAncora"
    private InsiemeSS contSpec;

    //Costruttore
    public DVD(String c, int a, int t) {
        super(c,a);
        tipo = t;
lingue = new InsiemeSS(String.class);
contSpec = new InsiemeSS(ContenutiSpeciali.class);
    }

    //Accesso agli attributi
    public int getTipo() { return tipo; }
    public InsiemeSS getLingue() { return (InsiemeSS)lingue.clone(); }
    public void inserisciInLingue(String a) { lingue.inserisci(a); }
    public void eliminaDaLingue(String a) { lingue.elimina(a); }

    //Accesso alla associazione "contieneAncora"
    public InsiemeSS getContenutiSpeciali() {
return (InsiemeSS)contSpec.clone();
    }
    public void inserisciInContSpec(String a) { contSpec.inserisci(a); }
    public void eliminaDaContSpec(String a) { contSpec.elimina(a); }
}
```


La classe Java ContenutiSpeciali

```
// File ContenutiSpeciali.java

public class ContenutiSpeciali {
    //Rappr. attributi
    private final String titolo;
    private final String tipo;

    //Costruttore
    public ContenutiSpeciali(String tt, String tp) {
        titolo = tt;
        tipo = tp;
    }

    //Accesso agli attributi
    public String getTitolo() { return titolo; }
    public String getTipo() { return tipo; }

    //Overriding delle funzione speciali (cioe' solo toString): come prima
    public String toString() {
        return "ContenutiSpeciali: " + titolo + " - " + tipo;
    }
}
```

Realizzazione in Java dello use case

```
// File VerificheFishSitting.java

public class VerificheVideo {

    public static boolean estVHSeDVD(Film f) {
        boolean ceVHS = false;
boolean ceDVD = true;
InsiemeSS tuple = f.getLinksContiene();
while(!tuple.estVuoto()) {
    TipoLinkContiene t = (TipoLinkContiene)tuple.scegli();
    if (t.getSupportoVideo().getClass().equals(VHS.class))
ceVHS = true;
    else ceDVD = true; //e' un DVD
    if (ceVHS && ceDVD) return true;
    tuple.elimina(t);
}
return false;

    }

    public static InsiemeSS getCodiciDVD(Film f) {
InsiemeSS codici = new InsiemeSS(String.class);
        InsiemeSS tuple = f.getLinksContiene();
while(!tuple.estVuoto()) {
    TipoLinkContiene t = (TipoLinkContiene)tuple.scegli();
    if (t.getSupportoVideo().getClass().equals(DVD.class)) {
DVD d = (DVD)t.getSupportoVideo();
codici.inserisci(d.getCodice());
    }
```



```
    tuple.elimina(t);
}
return codici;
    }
}
```

InsiemeSS

```
// File InsiemeSS.java

public class InsiemeSS implements Cloneable {
    // funzioni proprie del tipo astratto
    public InsiemeSS(Class cl) {
        // costruttore, realizza la funzione InsVuoto del tipo astratto Insieme
        inizio = null;
        cardinalita = 0;
        elemClass = cl;
    }
    public boolean estVuoto() {
        return inizio == null;
    }
    public boolean membro(Object e) {
        if (!elemClass.isInstance(e)) return false;
        else return appartiene(e,inizio);
    }
    public void inserisci(Object e) {
        if (!elemClass.isInstance(e)) return;
        else if (appartiene(e,inizio)) return;
        else {
            Lista l = new Lista();
            l.info = e;
            l.next = inizio;
            inizio = l;
            cardinalita = cardinalita + 1;
        }
    }
    public void elimina(Object e) {
```



```
        if (!appartiene(e,inizio)) return;
        else {
            inizio = cancella(e,inizio);
            cardinalita = cardinalita - 1;
        }
    }
    public Object scegli() {
        if (inizio == null) return null;
        else return inizio.info;
    }
    public int cardinalita() {
        return cardinalita;
    }
    // funzioni speciali ereditate da Object
    public boolean equals(Object o) {
        if (o != null && getClass().equals(o.getClass())) {
            InsiemeSS ins = (InsiemeSS)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
                Lista l = inizio;
                while (l != null) {
                    if (!appartiene(l.info,ins.inizio))
                        return false;
                    l = l.next;
                }
                return true;
            }
```

```
        }
        else return false;
    }
    public Object clone() {
        try {
            InsiemeSS ins = (InsiemeSS) super.clone();
            // chiamata a clone() di Object che esegue la copia campo a campo;
            // questa copia e' sufficiente per i campi cardinalita e elemClass
            // ma non per il campo inizio del quale va fatta una copia profonda
            ins.inizio = copia(inizio);
            return ins;
        } catch(CloneNotSupportedException e) {
            // non puo' accadere perche' implementiamo l'interfaccia cloneable,
            // ma va comunque gestita
            throw new InternalError(e.toString());
        }
    }
    public String toString() {
        String s = "{";
        Lista l = inizio;
        while (l != null) {
            s = s + l.info + " ";
            l = l.next;
        }
        s = s + "}";
        return s;
    }

    // campi dati
    protected static class Lista {
        Object info;









        Lista next;
    }
    protected Lista inizio;
    protected int cardinalita;
    protected Class elemClass;

    // funzioni ausiliarie

    protected static boolean appartiene(Object e, Lista l){
        return (l != null) && (l.info.equals(e) || appartiene(e,l.next));
    }
    protected static Lista copia (Lista l) {
        if (l == null) return null;
        else {
            Lista ll = new Lista();
            ll.info = l.info;
            ll.next = copia(l.next);
            return ll;
        }
    }
    protected static Lista cancella(Object e, Lista l) {
        if (l == null) return null;
        else  if (l.info.equals(e)) return l.next;
        else {
            l.next = cancella(e,l.next);
            return l;
        }
    }
}
```