Università di Roma "La Sapienza"
Facoltà di Ingegneria

Corso di
"PROGETTAZIONE DEL SOFTWARE I"
(Corso di Laurea in Ingegneria Informatica)
Proff. Marco Cadoli e Giuseppe De Giacomo
Canali A-L & M-Z
A.A. 2005-06

Compito d'esame del 13 settembre 2006

SOLUZIONE

Requisiti

L'applicazione da progettare riguarda la gestione di contratti per trasmissione dati e voce da parte degli utenti di una ditta di telecomunicazioni. Per un utente sono di interesse il codice fiscale, il nome e il cognome (stringhe), e le trasmissioni che effettua. Una trasmissione è caratterizzata dalla durata, in secondi. Alcuni utenti sono speciali, e per loro interessa il codice convenzione (una stringa). Esistono solo due tipi di trasmissioni, disgiunte fra loro: trasmissione dati e voce. Le prime sono caratterizzate dal volume dati, espresso in MegaByte, le seconde dal canale utilizzato (una stringa). Per le trasmissioni dati effettuate da un utente speciale è di interesse rappresentare la percentuale di compressione.

Un utente possiede almeno un contratto, che è caratterizzato dalla data di scadenza. I contratti sono di due tipi, disgiunti fra loro: contratto standard (caratterizzato dal codice, che è una stringa) e contratto fedeltà (caratterizzato dal numero di interventi di assistenza gratuiti previsti). Ogni utente può possedere al più due contratti fedeltà.

Requisiti (cont.)

Un contratto inizialmente viene catalogato come nuovo. Quando si avvicina la scadenza, viene catalogato come in scadenza. Se viene rinnovato, viene catalogato come rinnovato, se c'è ritardo viene catalogato come sollecitato. Da sollecitato può venire rinnovato, oppure divenire scaduto a seguito di ulteriore ritardo. Anche un contratto rinnovato viene catalogato come in scadenza quando si avvicina la data di scadenza,

L' Autorità per le Telecomunicazioni è interessata ad effettuare un'analisi di qualità dei servizi erogati, che si concretizza nei seguenti controlli:

- dato un contratto c, qual è, fra le trasmissioni effettuate dall'utente che possiede c, la percentuale di trasmissioni dati;

- dato un insieme I di utenti, qual è, fra le trasmissioni effettuate dagli utenti di I, la più alta percentuale di compressione.

Fase di analisi

Diagramma delle classi

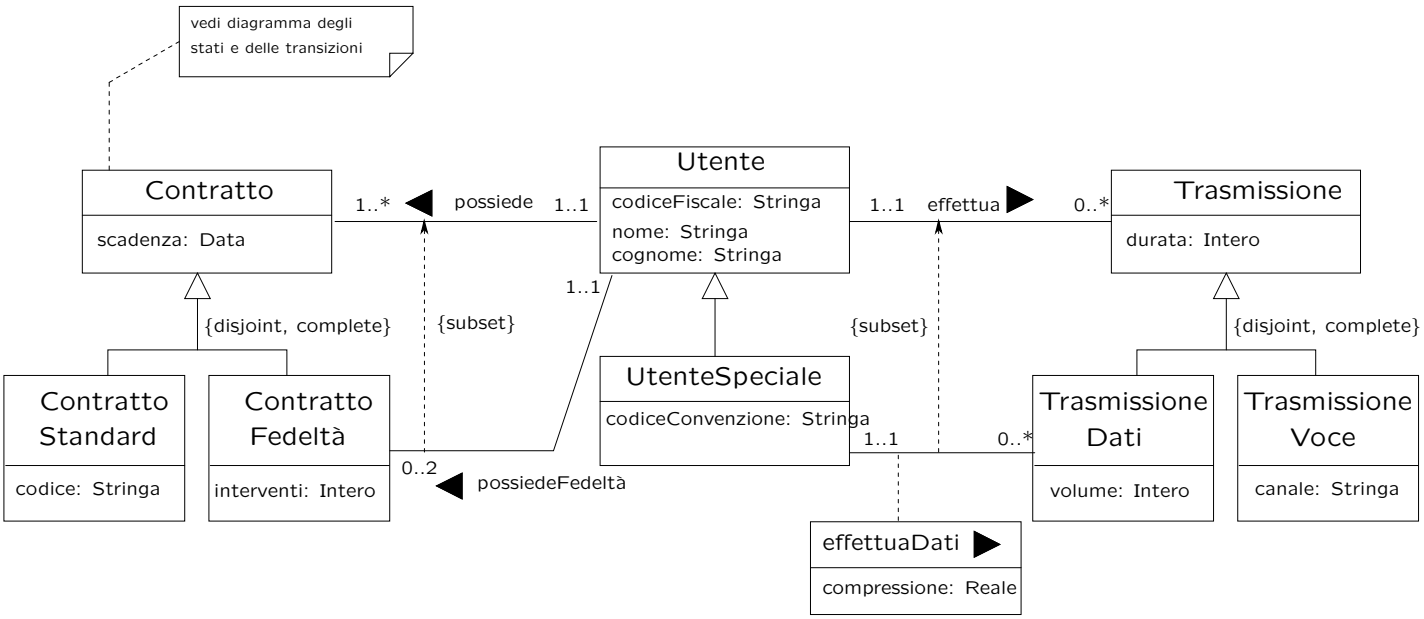


Diagramma degli stati e delle transizioni classe Contratto

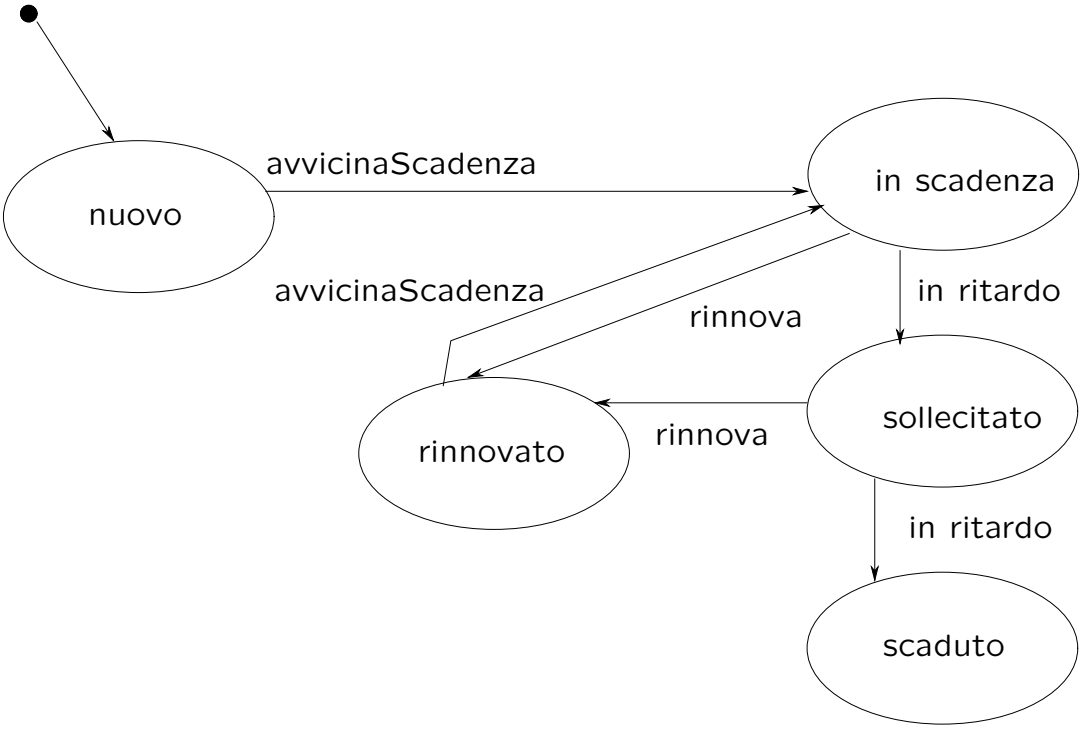
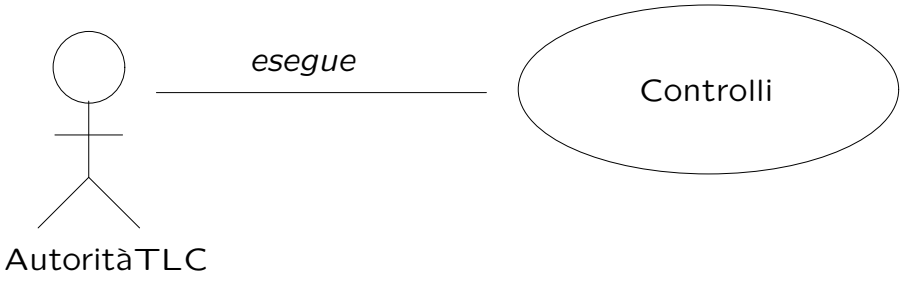


Diagramma degli use case



Specifica dello use case

InizioSpecificaUseCase Controlli

PercentualeTxDati *(c: Contratto): reale*

Definiamo gli insiemi:

$$Tx \doteq \{t \mid t \in Trasmissione \wedge \langle c.possiede, t\rangle \in effettua\},$$

$$TxD \doteq \{t \mid t \in TrasmissioneDati \wedge \langle c.possiede, t\rangle \in effettua\},$$

pre: $Tx \neq \emptyset$

post: *result* = 100 · card(TxD) / card(Tx)

...

Specifica dello use case (cont.)

...

MaxPercCompr *(I: Insieme(Utente)): reale*

Definiamo gli insiemi:

$$\begin{aligned} EffDat &\doteq \{\langle us, td\rangle \mid \\ &\quad us \in I \wedge td \in TrasmissioneDati \wedge \langle us, td\rangle \in effettuaDati\}, \\ Perc &\doteq \{p \mid \exists \langle us, td\rangle \in EffDat \ \wedge \ p = compressione(us, td)\} \end{aligned}$$

pre: $EffDat \neq \emptyset$

post: *result* = max($Perc$)

FineSpecifica

Fase di progetto

Algoritmi per le operazioni dello use-case

Adottiamo i seguenti algoritmi:

- Per l'operazione **PercentualeTxDati**:

```
Utente u = c.possiede;
int trasmissioni = 0;
int trasmissioniDati = 0;
per ogni link l di tipo effettua in cui u è coinvolto
        trasmissioni++;
        se (l.Trasmissione.class = TrasmissioneDati)
            trasmissioniDati++;
return (trasmissioniDati * 100)/trasmissioni;
```

- Per l'operazione **MaxPercCompr**:

```
float max = 0;
per ogni Utente u di I
        se u.class = UtenteSpeciale
                per ogni link l di tipo effettuaDati in cui u è coinvolto
                        se l.compressione > max
                                max = l.compressione
return max;
```


Responsabilità sulle associazioni

La seguente tabella delle responsabilità si evince da:

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

Associazione	Classe	ha resp.
possiede	*Utente*	SÌ[3]
	Contratto	SÌ[2,3]
possiedeFedeltà	*Utente*	SÌ[3]
	ContrattoFedeltà	SÌ[3]
effettua	*Utente*	SÌ[2]
	Trasmissione	SÌ[1,3]
effettuaDati	*UtenteSpeciale*	SÌ[2]
	TrasmissioneDati	SÌ[1,3]

Strutture di dati

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

- dei vincoli di molteplicità 0..* delle associazioni,
- delle variabili locali necessarie per vari algoritmi.

Per fare ciò, utilizzeremo le classi del collection framework di Java 1.5: `Set`, `HashSet`.

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`
data	`Data`
Insieme	`HashSet`

Utilizzeremo la classe Java `Data` presentata durante le lezioni del corso.

Tabelle di gestione delle proprietà di classi UML

Riassumiamo le nostre scelte differenti da quelle di default mediante la *tabella delle proprietà immutabili* e la *tabella delle assunzioni sulla nascita*.

Classe UML	**Proprietà immutabile**
Utente	*nome*
	cognome
	codiceFiscale
Trasmissione	*durata*
TrasmissioneDati	*volume*
TrasmissioneVoce	*canale*

	Proprietà	
Classe UML	**nota alla nascita**	**non nota alla nascita**

Altre considerazioni

Sequenza di nascita degli oggetti: Non dobbiamo assumere una particolare sequenza di nascita degli oggetti.

Valori alla nascita: Non sembra ragionevole assumere che per qualche proprietà esistano valori di default validi per tutti gli oggetti.

Rappresentazione degli stati in Java

Per la classe UML *Contratto*, ci dobbiamo occupare della rappresentazione in Java del diagramma degli stati e delle transizioni.

Scegliamo di rappresentare gli stati mediante una variabile `int`, secondo la seguente tabella.

Stato	Rappresentazione in Java	
	tipo var.	`int`
	nome var.	`stato`
nuovo	valore	1
inScadenza	valore	2
rinnovato	valore	3
sollecitato	valore	4
scaduto	valore	5

API delle classi Java progettate

Omesse per brevità (si faccia riferimento al codice Java).

Fase di realizzazione

Considerazioni iniziali

La traccia ci richiede di realizzare:

1. la classe UML *Contratto*;
2. la classe UML *Utente*
3. l'associazione UML *possiede* con responsabilità doppia e con vincoli di molteplicità 1..1 (molteplicità massima finita e molteplicità minima diversa da zero) e 1..* (molteplicità minima diversa da zero);
4. uno use case.

Nel seguito verranno realizzate tutte le classi e gli use case individuati in fase di analisi.

Struttura dei file e dei package

```
+---AppContratti
|   |   AssociazioneEffettua.java
|   |   AssociazioneEffettuaDati.java
|   |   AssociazionePossiede.java
|   |   AssociazionePossiedeFedelta.java
|   |   Controlli.java
|   |   EccezioneCardMax.java
|   |   EccezioneCardMin.java
|   |   EccezionePrecondizioni.java
|   |   EccezioneSubset.java
|   |   TestContratti.java
|   |   TipoLinkEffettua.java
|   |   TipoLinkEffettuaDati.java
|   |   TipoLinkPossiede.java
|   |   TipoLinkPossiedeFedelta.java
|   |
|   +---Contratto
|   |       Contratto.java
|   |
|   +---ContrattoFedelta
|   |       ContrattoFedelta.java
|   |
|   +---ContrattoStandard
|   |       ContrattoStandard.java
|   |
|   +---Trasmissione
|   |       Trasmissione.java
|   |
|   +---TrasmissioneDati
```



```
|   |       TrasmissioneDati.java
|   |
|   +---TrasmissioneVoce
|   |       TrasmissioneVoce.java
|   |
|   +---Utente
|   |       Utente.java
|   |
|   \---UtenteSpeciale
|           UtenteSpeciale.java
|
\---data
        Data.java
```

La classe Java Contratto

```
// File AppContratti/Contratto/Contratto.java
package AppContratti.Contratto;

import AppContratti.Contratto.*;
import AppContratti.*;
import AppContratti.ContrattoStandard.*;
import java.util.*;
import data.*;

public abstract class Contratto  {
    private final static int nuovo = 1;
    private final static int in_scadenza = 2;
    private final static int rinnovato = 3;
    private final static int sollecitato = 4;
    private final static int scaduto = 4;
    private int corrente = nuovo;
    protected Data scadenza;
    protected TipoLinkPossiede link;
    public Contratto(Data d) {
        scadenza = d;
        link = null;
    }
    public Data getScadenza() { return scadenza; }
    public void setScadenza(Data d) { scadenza = d; }
```



```
    public void avvicinaScadenza() {
        if (corrente == nuovo || corrente == rinnovato)
            corrente = in_scadenza;
    }
    public void rinnova() {
        if (corrente == in_scadenza)
            corrente = rinnovato;
    }
    public void in_ritardo() {
        if (corrente == in_scadenza)
            corrente = sollecitato;
        if (corrente == sollecitato)
            corrente = scaduto;
    }
    public int quantiPossiede() {
        if (link == null)
            return 0;
        else return 1;
    }
    public void inserisciLinkPossiede(AssociazionePossiede a) {
        if (a != null) link = a.getLink();
    }
    public void eliminaLinkPossiede(AssociazionePossiede a) {
        if (a != null) link = null;
    }
    public TipoLinkPossiede getLinkPossiede()
```

```
        throws EccezioneCardMin {
        if (link == null)
            throw new EccezioneCardMin("Cardinalita' minima violata");
        else
            return link;
    }
}
```

La classe Java ContrattoStandard

```
// File AppContratti/ContrattoStandard/ContrattoStandard.java
package AppContratti.ContrattoStandard;
import AppContratti.Contratto.*;
import data.*;
import AppContratti.*;
import java.util.*;

public class ContrattoStandard extends Contratto {
    protected String codice;
    public ContrattoStandard(Data scadenza, String cod) {
        super(scadenza);
        codice = cod;
    }
    public String getCodice() { return codice; }
    public void setCodice(String cod) { codice = cod; }
}
```

La classe Java ContrattoFedelta

```
// File AppContratti/ContrattoFedelta/ContrattoFedelta.java
package AppContratti.ContrattoFedelta;
import AppContratti.Contratto.*;
import AppContratti.*;
import data.*;
import java.util.*;

public class ContrattoFedelta extends Contratto {
    protected int interventi;
    protected TipoLinkPossiedeFedelta link_f;
    public ContrattoFedelta(Data scadenza, int inter) {
        super(scadenza);
        interventi = inter;
        link_f = null;
    }
    public int getInterventi() { return interventi; }
    public void setInterventi(int inter) { interventi = inter; }
    public int quantiPossiedeFedelta() {
        if (link_f == null)
            return 0;
        else return 1;
    }
    public void inserisciLinkPossiedeFedelta(AssociazionePossiedeFedelta a) {
        if (a != null) link_f = a.getLink();
```



```
    }
    public void eliminaLinkPossiedeFedelta(AssociazionePossiedeFedelta a) {
        if (a != null) link_f = null;
    }
    public TipoLinkPossiedeFedelta getLinkPossiedeFedelta()
        throws EccezioneCardMin, EccezioneSubset {
        if (link_f == null)
            throw new EccezioneCardMin("Cardinalita' minima violata");
        else if (link == null)
            throw new EccezioneSubset();
        else if (link.getContratto() != this |
                 link.getUtente() != link_f.getUtente())
            throw new EccezioneSubset();
        else
            return link_f;
    }
}
```

La classe Java Utente

```
// File AppContratti/Utente/Utente.java
package AppContratti.Utente;
import AppContratti.*;
import AppContratti.Contratto.*;
import java.util.*;

public class Utente {
    protected String nome, cognome, codiceFiscale;
    protected HashSet<TipoLinkEffettua> effettua;
    protected HashSet<TipoLinkPossiede> possiede;
    protected HashSet<TipoLinkPossiedeFedelta> possiedeFedelta;
    public static final int MIN_LINK_POSSIEDE = 1;
    public static final int MAX_LINK_POSSIEDE_FEDELTA = 2;
    public Utente(String n, String c, String cf) {
        nome = n;
        cognome = c;
        codiceFiscale = cf;
        effettua = new HashSet<TipoLinkEffettua>();
        possiede = new HashSet<TipoLinkPossiede>();
        possiedeFedelta = new HashSet<TipoLinkPossiedeFedelta>();
    }
    public String getNome() { return nome; }
    public String getCognome() { return cognome; }
    public String getCodiceFiscale() { return codiceFiscale; }
```



```
    public void inserisciLinkEffettua(AssociazioneEffettua a)  {
        if (a != null) effettua.add(a.getLink());
    }
    public void eliminaLinkEffettua(AssociazioneEffettua a) {
        if (a != null) effettua.remove(a.getLink());
    }
    public Set<TipoLinkEffettua> getLinkEffettua() {
        return (HashSet<TipoLinkEffettua>)effettua.clone();
    }
    public int quantiPossiede() {
        return possiede.size();
    }
    public void inserisciLinkPossiede(AssociazionePossiede a)  {
        if (a != null) possiede.add(a.getLink());
    }
    public void eliminaLinkPossiede(AssociazionePossiede a) {
        if (a != null) possiede.remove(a.getLink());
    }
    public Set<TipoLinkPossiede> getLinkPossiede() throws EccezioneCardMax {
        if (possiede.size() < MIN_LINK_POSSIEDE)
            throw new EccezioneCardMax("Cardinalita' minima violata");
        else return (HashSet<TipoLinkPossiede>)possiede.clone();
    }
    public void inserisciLinkPossiedeFedelta(AssociazionePossiedeFedelta a)  {
        if (a != null) possiedeFedelta.add(a.getLink());
    }
```

```
    public void eliminaLinkPossiedeFedelta(AssociazionePossiedeFedelta a) {
        if (a != null) possiedeFedelta.remove(a.getLink());
    }
    public Set<TipoLinkPossiedeFedelta> getLinkPossiedeFedelta()
        throws EccezioneCardMax, EccezioneSubset {
        if (possiedeFedelta.size() > MAX_LINK_POSSIEDE_FEDELTA)
            throw new EccezioneCardMax("Cardinalita' massima violata");
        else {
            Set<TipoLinkPossiede> c = this.getLinkPossiede();
            Iterator<TipoLinkPossiede> it = possiede.iterator();
            while (it.hasNext()) {
                Contratto f = it.next().getContratto();
                if (!c.contains(new TipoLinkPossiede(this,f)))
                    throw new EccezioneSubset
                        ("possiede non è un subset di contiene");
            }
            return (HashSet<TipoLinkPossiedeFedelta>)possiedeFedelta.clone();
        }
    }
    public String toString() {
        return nome + "/" + cognome + "/" + codiceFiscale;
  }
}
```

La classe Java UtenteSpeciale

```
// File AppContratti/UtenteSpeciale/UtenteSpeciale.java
package AppContratti.UtenteSpeciale;
import AppContratti.Utente.*;
import AppContratti.*;
import data.*;
import java.util.*;

public class UtenteSpeciale extends Utente {
    protected String codiceConvenzione;
    protected HashSet<TipoLinkEffettuaDati> effettuaDati;
    public UtenteSpeciale(String n, String c, String cf, String cc) {
        super(n,c,cf);
        codiceConvenzione = cc;
        effettuaDati = new HashSet<TipoLinkEffettuaDati>();
    }
    public String getCodiceConvenzione() { return codiceConvenzione; }
    public void setCodiceConvenzione(String cc) { codiceConvenzione = cc; }
    public void inserisciLinkEffettuaDati(AssociazioneEffettuaDati a) {
        if (a != null) effettuaDati.add(a.getLink());
    }
    public void eliminaLinkEffettuaDati(AssociazioneEffettuaDati a) {
        if (a != null) effettuaDati.remove(a.getLink());
    }
    public Set<TipoLinkEffettuaDati> getLinkEffettuaDati() {
```

```
        return (HashSet<TipoLinkEffettuaDati>)effettuaDati.clone();
    }
}
```

La classe Java Trasmissione

```
// File AppContratti/Trasmissione/Trasmissione.java
package AppContratti.Trasmissione;

import AppContratti.*;
import java.util.*;

public abstract class Trasmissione  {
    protected int durata;
    protected TipoLinkEffettua link;
    public static final int MIN_LINK_EFFETTUA = 1;
    public Trasmissione(int dur) {
        durata = dur;
    }
    public int getDurata() { return durata; }
    public void setDurata(int n) { durata = n; }
    public int quantiEffettua() {
        if (link == null)
            return 0;
        else return 1;
    }
    public void inserisciLinkEffettua(AssociazioneEffettua a) {
        if (a != null) link = a.getLink();
    }
    public void eliminaLinkEffettua(AssociazioneEffettua a) {
```

```
        if (a != null) link = null;
    }
    public TipoLinkEffettua getLinkEffettua() throws EccezioneCardMin {
        if (link == null)
            throw new EccezioneCardMin("Cardinalita' minima violata");
        else
            return link;
    }
}
```

La classe Java TrasmissioneDati

```
// File AppContratti/TrasmissioneDati/TrasmissioneDati.java
package AppContratti.TrasmissioneDati;
import AppContratti.Trasmissione.*;
import AppContratti.*;
import java.util.*;

public class TrasmissioneDati extends Trasmissione {
    protected int volume;
    public static final int MIN_LINK_EFFETTUA_DATI = 1;
    protected TipoLinkEffettuaDati link_e;
    public TrasmissioneDati(int d, int v) {
        super(d);
        volume = v;
        link_e = null;
    }
    public int getVolume() { return volume; }
    public int quantiEffettuaDati() {
        if (link_e == null)
            return 0;
        else return 1;
    }
    public void inserisciLinkEffettuaDati(AssociazioneEffettuaDati a) {
        if (a != null) link_e = a.getLink();
    }
```

```
    public void eliminaLinkEffettuaDati(AssociazioneEffettuaDati a) {
        if (a != null) link_e = null;
    }
    public TipoLinkEffettuaDati getLinkEffettuaDati()
        throws EccezioneCardMin, EccezioneSubset {
        if (link_e == null)
            throw new EccezioneCardMin("Cardinalita' minima violata");
        else if (link == null)
            throw new EccezioneSubset();
        else if (link.getTrasmissione() != this |
                 link.getUtente() != link_e.getUtenteSpeciale())
            throw new EccezioneSubset();
        else
            return link_e;
    }
}
```

La classe Java TrasmissioneVoce

```
// File AppContratti/TrasmissioneVoce/TrasmissioneVoce.java
package AppContratti.TrasmissioneVoce;
import AppContratti.Trasmissione.*;
import AppContratti.*;
import java.util.*;

public class TrasmissioneVoce extends Trasmissione {
    protected String canale;
    public TrasmissioneVoce(int d, String c) {
        super(d);
        canale = c;
    }
    public String getCanalee() { return canale; }
}
```

La classe Java TipoLinkPossiede

```
// File AppContratti/TipoLinkPossiede.java
package AppContratti;
import data.*;
import AppContratti.Contratto.*;
import AppContratti.Utente.*;

public class TipoLinkPossiede {
    private final Utente lUtente;
    private final Contratto ilContratto;
    public TipoLinkPossiede(Utente u, Contratto c)
        throws EccezionePrecondizioni {
        if (u == null || c == null)  // CONTROLLO PRECONDIZIONI
            throw new EccezionePrecondizioni
                ("Gli oggetti devono essere inizializzati");
        lUtente = u;
        ilContratto = c;
    }
    public boolean equals(Object o) {
        if (o != null && getClass().equals(o.getClass())) {
            TipoLinkPossiede b = (TipoLinkPossiede)o;
            return b.ilContratto == ilContratto &&
                b.lUtente == lUtente;
        }
        else return false;
```



```
    }
    public int hashCode() {
        return lUtente.hashCode() + ilContratto.hashCode() +
            lUtente.hashCode();
    }
    public Utente getUtente() { return lUtente; }
    public Contratto getContratto() { return ilContratto; }
    public String toString() {
        return "<" + lUtente + ", " + ilContratto + ">";
    }
}
```

La classe Java AssociazionePossiede

```
// File AppContratti/AssociazionePossiede.java
package AppContratti;

public final class AssociazionePossiede {
    private AssociazionePossiede(TipoLinkPossiede x) { link = x; }
    private TipoLinkPossiede link;
    public TipoLinkPossiede getLink() { return link; }
    public static void inserisci(TipoLinkPossiede y) {
        if (y != null && y.getContratto().quantiPossiede() == 0) {
            AssociazionePossiede k = new AssociazionePossiede(y);
            y.getUtente().inserisciLinkPossiede(k);
            y.getContratto().inserisciLinkPossiede(k);
        }
    }
    public static void elimina(TipoLinkPossiede y)
    throws EccezioneCardMin, EccezioneCardMax {
        if (y != null && y.getContratto().getLinkPossiede().equals(y)) {
            AssociazionePossiede k = new AssociazionePossiede(y);
            y.getUtente().eliminaLinkPossiede(k);
            y.getContratto().eliminaLinkPossiede(k);
        }
    }
}
```


La classe Java TipoLinkPossiedeFedelta

```
// File AppContratti/TipoLinkPossiedeFedelta.java
package AppContratti;
import data.*;
import AppContratti.ContrattoFedelta.*;
import AppContratti.Utente.*;

public class TipoLinkPossiedeFedelta {
    private final Utente lUtente;
    private final ContrattoFedelta ilContrattoFedelta;
    public TipoLinkPossiedeFedelta(Utente u, ContrattoFedelta c)
        throws EccezionePrecondizioni {
        if (u == null || c == null)  // CONTROLLO PRECONDIZIONI
            throw new EccezionePrecondizioni
                ("Gli oggetti devono essere inizializzati");
        lUtente = u;
        ilContrattoFedelta = c;
    }
    public boolean equals(Object o) {
        if (o != null && getClass().equals(o.getClass())) {
            TipoLinkPossiedeFedelta b = (TipoLinkPossiedeFedelta)o;
            return b.ilContrattoFedelta == ilContrattoFedelta &&
                b.lUtente == lUtente;
        }
        else return false;
```

```
    }
    public int hashCode() {
        return lUtente.hashCode() + ilContrattoFedelta.hashCode() +
            lUtente.hashCode();
    }
    public Utente getUtente() { return lUtente; }
    public ContrattoFedelta getContrattoFedelta() { return ilContrattoFedelta; }
    public String toString() {
        return "<" + lUtente + ", " + ilContrattoFedelta + ">";
    }
}
```

La classe Java AssociazionePossiedeFedelta

```
// File AppContratti/AssociazionePossiedeFedelta.java
package AppContratti;

public final class AssociazionePossiedeFedelta {
    private AssociazionePossiedeFedelta(TipoLinkPossiedeFedelta x) { link = x; }
    private TipoLinkPossiedeFedelta link;
    public TipoLinkPossiedeFedelta getLink() { return link; }
    public static void inserisci(TipoLinkPossiedeFedelta y) {
        if (y != null &&
            y.getContrattoFedelta().quantiPossiedeFedelta() == 0) {
            AssociazionePossiedeFedelta k = new AssociazionePossiedeFedelta(y);
            y.getUtente().inserisciLinkPossiedeFedelta(k);
            y.getContrattoFedelta().inserisciLinkPossiedeFedelta(k);
        }
    }
    public static void elimina(TipoLinkPossiedeFedelta y)
        throws EccezioneCardMin, EccezioneCardMax {
        if (y != null &&
            y.getContrattoFedelta().getLinkPossiedeFedelta().equals(y)) {
            AssociazionePossiedeFedelta k = new AssociazionePossiedeFedelta(y);
            y.getUtente().eliminaLinkPossiedeFedelta(k);
            y.getContrattoFedelta().eliminaLinkPossiedeFedelta(k);
        }
    }
}
```

La classe Java TipoLinkEffettua

```
// File AppContratti/TipoLinkEffettua.java
package AppContratti;
import data.*;
import AppContratti.Trasmissione.*;
import AppContratti.Utente.*;
import AppContratti.*;

public class TipoLinkEffettua {
    private final Utente lUtente;
    private final Trasmissione laTrasmissione;
    public TipoLinkEffettua(Utente u, Trasmissione t)
        throws EccezionePrecondizioni {
        if (u == null || t == null)  // CONTROLLO PRECONDIZIONI
            throw new EccezionePrecondizioni
                ("Gli oggetti devono essere inizializzati");
        lUtente = u;
        laTrasmissione = t;
    }
    public boolean equals(Object o) {
        if (o != null && getClass().equals(o.getClass())) {
            TipoLinkEffettua b = (TipoLinkEffettua)o;
            return b.laTrasmissione == laTrasmissione &&
                b.lUtente == lUtente;
        }
```



```
        else return false;
    }
    public int hashCode() {
        return lUtente.hashCode() + laTrasmissione.hashCode() +
            lUtente.hashCode();
    }
    public Utente getUtente() { return lUtente; }
    public Trasmissione getTrasmissione() { return laTrasmissione; }
    public String toString() {
        return "<" + lUtente + ", " + laTrasmissione + ">";
    }
}
```

La classe Java AssociazioneEffettua

```
// File AppContratti/AssociazioneEffettua.java
package AppContratti;

public final class AssociazioneEffettua {
    private AssociazioneEffettua(TipoLinkEffettua x) { link = x; }
    private TipoLinkEffettua link;
    public TipoLinkEffettua getLink() { return link; }
    public static void inserisci(TipoLinkEffettua y) {
        if (y != null &&
            y.getTrasmissione().quantiEffettua() == 0) {
            AssociazioneEffettua k = new AssociazioneEffettua(y);
            y.getUtente().inserisciLinkEffettua(k);
            y.getTrasmissione().inserisciLinkEffettua(k);
        }
    }
    public static void elimina(TipoLinkEffettua y)
        throws EccezioneCardMin {
        if (y != null && y.getTrasmissione().getLinkEffettua().equals(y)) {
            AssociazioneEffettua k = new AssociazioneEffettua(y);
            y.getUtente().eliminaLinkEffettua(k);
            y.getTrasmissione().eliminaLinkEffettua(k);
        }
    }
}
```


La classe Java TipoLinkEffettuaDati

```
// File AppContratti/TipoLinkEffettuaDati.java
package AppContratti;
import AppContratti.UtenteSpeciale.*;
import AppContratti.TrasmissioneDati.*;
import data.*;

public class TipoLinkEffettuaDati {
    private final UtenteSpeciale lUtenteSpeciale;
    private final TrasmissioneDati laTrasmissioneDati;
    private final float compressione;
    public TipoLinkEffettuaDati(UtenteSpeciale a, TrasmissioneDati d,
                                float c)
        throws EccezionePrecondizioni {
        if (a == null || d == null)
            throw new EccezionePrecondizioni
                ("Gli oggetti devono essere inizializzati");
        lUtenteSpeciale = a; laTrasmissioneDati = d;
        compressione = c;
    }
    public boolean equals(Object o) {
        if (o != null && getClass().equals(o.getClass())) {
            TipoLinkEffettuaDati b = (TipoLinkEffettuaDati)o;
            return b.lUtenteSpeciale == lUtenteSpeciale &&
                b.laTrasmissioneDati == laTrasmissioneDati;
```

```
        }
        else return false;
    }
    public int hashCode() {
        return laTrasmissioneDati.hashCode() +
            lUtenteSpeciale.hashCode();
    }
    public UtenteSpeciale getUtenteSpeciale() {
        return lUtenteSpeciale; }
    public TrasmissioneDati getTrasmissioneDati() {
        return laTrasmissioneDati; }
    public float getCompressione() {
        return compressione; }
    public String toString() {
        return "(<" + lUtenteSpeciale + ", " +
            laTrasmissioneDati + ">)";
    }
}
```

La classe Java AssociazioneEffettuaDati

```
// File AppContratti/AssociazioneEffettuaDati.java
package AppContratti;

public final class AssociazioneEffettuaDati {
    private AssociazioneEffettuaDati(TipoLinkEffettuaDati x) { link = x; }
    private TipoLinkEffettuaDati link;
    public TipoLinkEffettuaDati getLink() { return link; }
    public static void inserisci(TipoLinkEffettuaDati y) {
        if (y != null &&
            y.getTrasmissioneDati().quantiEffettuaDati() == 0) {
            AssociazioneEffettuaDati k = new AssociazioneEffettuaDati(y);
            y.getUtenteSpeciale().inserisciLinkEffettuaDati(k);
            y.getTrasmissioneDati().inserisciLinkEffettuaDati(k);
        }
    }
    public static void elimina(TipoLinkEffettuaDati y)
        throws EccezioneCardMin, EccezioneCardMax {
        if (y != null &&
            y.getTrasmissioneDati().getLinkEffettuaDati().equals(y)) {
            AssociazioneEffettuaDati k = new AssociazioneEffettuaDati(y);
            y.getUtenteSpeciale().eliminaLinkEffettuaDati(k);
            y.getTrasmissioneDati().eliminaLinkEffettuaDati(k);
        }
    }
}
```

Realizzazione in Java dello use case

```
// File AppContratti/Controlli.java
package AppContratti;
import java.util.*;
import AppContratti.Contratto.*;
import AppContratti.Utente.*;
import AppContratti.UtenteSpeciale.*;
import AppContratti.Trasmissione.*;
import AppContratti.TrasmissioneDati.*;

public final class Controlli {
    private Controlli() { }
    public static float percentualeTxDati(Contratto c)
        throws EccezioneCardMin, EccezioneCardMax {
        int trasmissioni = 0, trasmissioniDati = 0;
        Utente u = c.getLinkPossiede().getUtente();
        Set<TipoLinkEffettua> ins = u.getLinkEffettua();
        Iterator<TipoLinkEffettua> it = ins.iterator();
        while(it.hasNext()) {
            TipoLinkEffettua t = it.next();
            trasmissioni++;
            if (t.getTrasmissione().getClass().equals(TrasmissioneDati.class))
                trasmissioniDati++;
        }
        return (float)(trasmissioniDati * 100)/trasmissioni;
```



```
    }
    public static float maxPercCompr(Set<Utente> i) {
        return (float)0.0;
    }
}
```

Realizzazione in Java delle classi per eccezioni

```
// File AppContratti/EccezioneCardMin.java
package AppContratti;

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

```
// File AppContratti/EccezioneCardMax.java
package AppContratti;
public class EccezioneCardMax extends Exception {
    private String messaggio;
    public EccezioneCardMax(String m) {
        messaggio = m;
    }
    public String toString() {
        return messaggio;
    }
}
```



```
// File AppContratti/EccezionePrecondizioni.java
package AppContratti;

public class EccezionePrecondizioni extends RuntimeException {
  private String messaggio;
  public EccezionePrecondizioni(String m) {
    messaggio = m;
  }
  public EccezionePrecondizioni() {
    messaggio = "Si e' verificata una violazione delle precondizioni";
  }
  public String toString() {
    return messaggio;
  }
}
```

```
// File AppContratti/EccezioneSubset.java
package AppContratti;

public class EccezioneSubset extends RuntimeException {
  private String messaggio;
  public EccezioneSubset(String m) {
    messaggio = m;
  }
  public EccezioneSubset() {
    messaggio = "Si e' verificata una violazione del vincolo di subset";
```

```
  }
  public String toString() {
    return messaggio;
  }
}
```