Università di Roma "La Sapienza", Facoltà di Ingegneria

Corso di
"PROGETTAZIONE DEL SOFTWARE" (Ing. Gestionale)
Prof. Giuseppe De Giacomo, A.A. 2002-03

SOLUZIONE ESAME DEL 18/09/2003

Roma, 18 settembre 2003

Diagramma delle classi UML

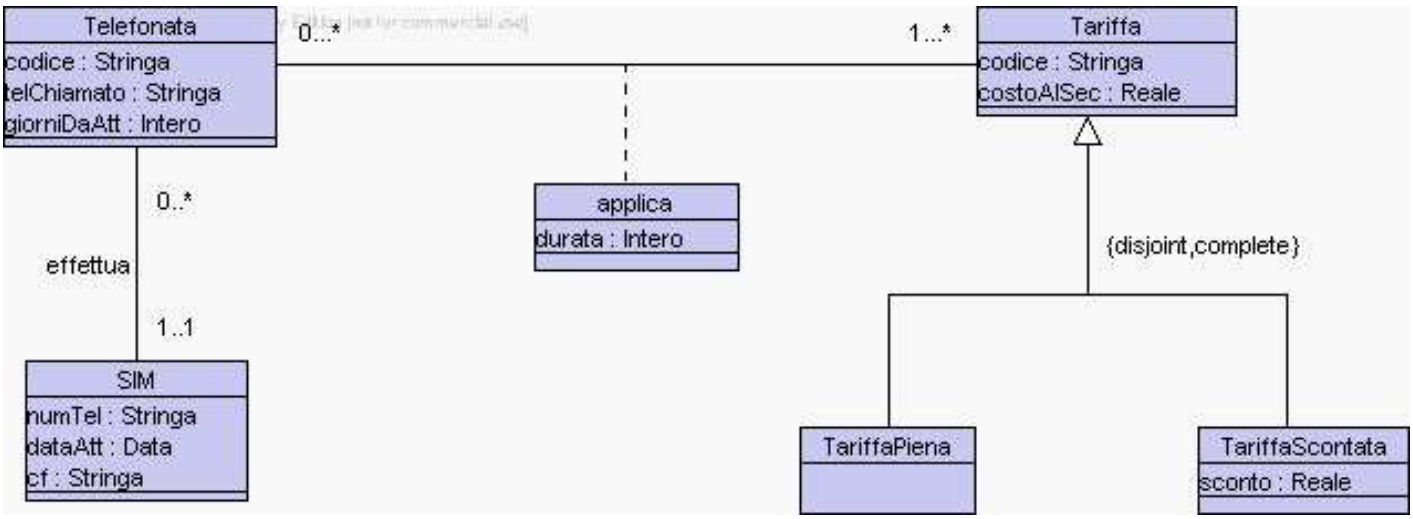

Diagramma e specifica degli use case


```
InizioSpecificaUseCase VerificheTariffe

  tariffaUtilizzata(t: TariffaScontata): Booleano
     pre:  nessuna
     post: result e' true se la tariffa scontata t e' stata applicata
           ad almeno una telefonata; false altrimenti

  costoComplessivo(s: SIM, n: Intero): Reale
     pre:  nessuna
     post: result e' il costo complessivo delle telefonate effettuate da s
           l'n-esimo giorno dalla attivazione, non considerando eventuali
           sconti relativi a tariffe scontate

FineSpecifica
```


Responsabilità sulle associazioni

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

- *SIM* ha responsabilità su *effettua*
- *Telefonata* ha responsabilità su *effettua* e su *applica*
- *Tariffa* ha responsabilità su *applica*
- *TariffaPiena* e *TariffaScontata* non hanno responsabilità specifiche

La classe Java TipoLinkEffetua

```
// File TipoLinkEffettua.java

public class TipoLinkEffettua {

  //Rappr. componenti della tupla
  private final SIM laSIM;
  private final Telefonata laTelefonata;

  //Costruttore
  public TipoLinkEffettua(SIM x, Telefonata y) {
    laSIM = x;
    laTelefonata = y;
  }

  //Accesso alle componenti
  public SIM getSIM() { return laSIM; }
  public Telefonata getTelefonata() { return laTelefonata; }

  //Overriding di funzioni speciali ereditate da Object
  //Nota TipoLinkEffettua e' un tipo non una classe:
  //va ridefinito equals per effettuare test di uguaglianza profonda
  public boolean equals(Object o) {
    if (o != null && getClass().equals(o.getClass())) {
      TipoLinkEffettua b = (TipoLinkEffettua)o;
      return b.laSIM != null && b.laTelefonata != null &&
        b.laSIM == laSIM &&
        b.laTelefonata == laTelefonata;
    }
    else return false;
```



```
  }
}
```

La classe Java TipoLinkApplica

```
// File TipoLinkApplica.java

public class TipoLinkApplica {

  //Rappr. componenti della tupla
  private final Telefonata laTelefonata;
  private final Tariffa laTariffa;
  private final int durata;

  //Costruttore
  public TipoLinkApplica(Telefonata x, Tariffa y, int d) {
    laTelefonata = x;
    laTariffa = y;
    durata = d;
  }

  //Accesso alle componenti
  public Telefonata getTelefonata() { return laTelefonata; }
  public Tariffa getTariffa() { return laTariffa; }
  public int getDurata() { return durata; }

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



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

La classe Java AssociazioneEffettua

```
// File AssociazioneEffettua.java

public 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.getSIM() != null && y.getTelefonata() != null) {
            AssociazioneEffettua k = new AssociazioneEffettua(y);
            y.getSIM().inserisciLinkEffettua(k);
            y.getTelefonata().inserisciLinkEffettua(k);
        }
    }
    public static void elimina(TipoLinkEffettua y) {
        if (y != null && y.getSIM() != null && y.getTelefonata() != null) {
            AssociazioneEffettua k = new AssociazioneEffettua(y);
            y.getSIM().eliminaLinkEffettua(k);
            y.getTelefonata().eliminaLinkEffettua(k);
        }
    }
}
```


La classe Java AssociazioneApplica

```
// File AssociazioneApplica.java

public class AssociazioneApplica {
    private AssociazioneApplica(TipoLinkApplica x) { link = x; }
    private TipoLinkApplica link;
    public TipoLinkApplica getLink() { return link; }
    public static void inserisci(TipoLinkApplica y) {
        if (y != null && y.getTelefonata() != null && y.getTariffa() != null) {
            AssociazioneApplica k = new AssociazioneApplica(y);
            y.getTelefonata().inserisciLinkApplica(k);
            y.getTariffa().inserisciLinkApplica(k);
        }
    }
    public static void elimina(TipoLinkApplica y) {
        if (y != null && y.getTelefonata() != null && y.getTariffa() != null) {
            AssociazioneApplica k = new AssociazioneApplica(y);
            y.getTelefonata().eliminaLinkApplica(k);
            y.getTariffa().eliminaLinkApplica(k);
        }
    }
}
```

La classe Java SIM

```
// File SIM.java
import java.util.*;

public class SIM {
  //Rappr. attributi
  private final String numTel;
  private final String dataAtt;
  private final String cf;

  //Rappr. associazione "effettua"
  private InsiemeArray links_effettua;

  //Costruttore
  public SIM(String n, String d, String cf) {
    numTel = n;
    dataAtt = d;
    this.cf = cf;
    links_effettua = new InsiemeArray(TipoLinkEffettua.class);
  }

  //Accesso agli attributi
  public String getNumTel() { return numTel; }
  public String getDataAtt() { return dataAtt; }
  public String getCF() { return cf; }

  //Accesso alla associazione "effettua"

  public Set getLinkEffettua() {
    return (InsiemeArray)links_effettua.clone();
```



```
  }

  public void inserisciLinkEffettua(AssociazioneEffettua a) {
    if (a != null  && a.getLink().getSIM() == this &&
        a.getLink().getTelefonata() != null)
      links_effettua.add(a.getLink());
  }
  public void eliminaLinkEffettua(AssociazioneEffettua a) {
    if (a != null  && a.getLink().getSIM() == this)
      links_effettua.remove(a.getLink());
  }

  //Overriding delle funzione speciali (cioe' solo toString)
  public String toString() {
    return "SIM " + numTel + ", " + dataAtt.toString() + ", " + cf;
  }
}
```

La classe Java Telefonata

```
// File Telefonata.java
import java.util.*;

public class Telefonata {
  //Rappr. attributi
  private final String codice;
  private final String telChiamato;
  private final int giorniDaAtt;

  //Rappr. associazione "esegue"
  private TipoLinkEffettua link_effettua;

  //Rappr. associazione "applica"
  private InsiemeArray links_applica;

  //Costruttore
  public Telefonata(String c, String t, int g) {
    codice = c;
    telChiamato = t;
    giorniDaAtt = g;
    link_effettua = null;
    links_applica = new InsiemeArray(TipoLinkApplica.class);
  }

  //Accesso agli attributi
  public String getCodice() { return codice; }
  public String getTelChiamato() { return telChiamato; }
  public int getGiorniDaAtt() { return giorniDaAtt; }
```



```
  //Accesso alla associazione "effettua"

  public boolean estSigLinkEffettua() {
    return link_effettua != null;
  }

  public TipoLinkEffettua getLinkEffettua() {
    if (!estSigLinkEffettua())
      throw new RuntimeException(
        "Formato: partecipazione obbligatoria in effettua violata");
    return link_effettua;
  }

  public void inserisciLinkEffettua(AssociazioneEffettua a) {
    if (a != null  && a.getLink().getTelefonata() == this &&
        a.getLink().getSIM() != null)
      link_effettua = a.getLink();
  }
  public void eliminaLinkEffettua(AssociazioneEffettua a) {
    if (a != null && a.getLink().getTelefonata() == this)
      link_effettua = null;
  }

  //Accesso alla associazione "applica"

  public int quantiApplica() {
    return links_applica.size();
  }

  public Set getLinkApplica() {
    if (quantiApplica() < 1)
```

```
      throw new RuntimeException(
        "Formato: partecipazione obbligatoria in applica violata");
    return (InsiemeArray)links_applica.clone();
  }

  public void inserisciLinkApplica(AssociazioneApplica a) {
    if (a != null  && a.getLink().getTelefonata() == this &&
        a.getLink().getTariffa() != null)
      links_applica.add(a.getLink());
  }
  public void eliminaLinkApplica(AssociazioneApplica a) {
    if (a != null && a.getLink().getTelefonata() == this)
      links_applica.remove(a.getLink());
  }


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

La classe Java Tariffa

```
// File Tariffa.java
import java.util.*;

public abstract class Tariffa {
  //Rappr. attributi
  protected final String codice;
  protected final double costoAlSec;

  //Rappr. associazione "applica"
  private InsiemeArray links_applica;

  //Costruttore
  public Tariffa(String c, double cs) {
    codice = c;
    costoAlSec = cs;
    links_applica = new InsiemeArray(TipoLinkApplica.class);
  }

  //Accesso agli attributi
  public String getCodice() { return codice; }
  public double getCostoAlSec() { return costoAlSec; }

  //Accesso alla associazione "applica"

  public Set getLinkApplica() {
    return (InsiemeArray)links_applica.clone();
  }

  public void inserisciLinkApplica(AssociazioneApplica a) {
```

```
    if (a != null  && a.getLink().getTariffa() == this &&
        a.getLink().getTelefonata() != null)
      links_applica.add(a.getLink());
  }
  public void eliminaLinkApplica(AssociazioneApplica a) {
    if (a != null && a.getLink().getTariffa() == this)
      links_applica.remove(a.getLink());
  }

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

La classe Java TariffaPiena

```
// file TariffaPiena.java

public class TariffaPiena extends Tariffa {

  // costruttore
  public TariffaPiena(String c, double cs) {
    super(c,cs);
  }

  //Overriding delle funzione speciali (cioe' solo toString)
  public String toString() {
    return "Tariffa Piena" + codice + ", " + costoAlSec;
  }

}


// Nota questa classe non ha campi dati propri, ma solo quelli
// ereditati da Tariffa
```

La classe Java TariffaScontata

```
// file TariffaScontata.java

public class TariffaScontata extends Tariffa {
  // rappresentazione degli attributi propri della classe
  private double sconto;

  // costruttore
  public TariffaScontata(String c, double cs, double s) {
    super(c,cs);
    sconto = s;
  }

  //Overriding delle funzione speciali (cioe' solo toString)
  public String toString() {
    return "Tariffa Piena" + codice + ", " + costoAlSec + ", " + sconto;
  }

}
```


Realizzazione in Java dello use case

```
// File VerificheTariffe.java
import java.util.*;

public class VerificheTariffe {


  public static boolean tariffaUtilizzata(TariffaScontata t) {
    return !t.getLinkApplica().isEmpty();
  }

  public static double costoComplessivo(SIM s, int n) {
    double ris = 0;
    Set tuple = s.getLinkEffettua();
    Iterator it = tuple.iterator();
    while(it.hasNext()) {
      TipoLinkEffettua t = (TipoLinkEffettua)it.next();
      Telefonata tl = t.getTelefonata();
      if (tl.getGiorniDaAtt() == n)
        ris = ris + calcolaCostoTelefonata(tl); //chiamata a metodo ausiliario
    }
    return ris;
  }


  // metodo ausiliario
  private static double calcolaCostoTelefonata(Telefonata tel) {
    double costo = 0;
    Set tuple = tel.getLinkApplica();
    Iterator it = tuple.iterator();
```

```
    while(it.hasNext()) {
      TipoLinkApplica t = (TipoLinkApplica)it.next();
      double d = t.getDurata();
      double cs = t.getTariffa().getCostoAlSec();
      costo = costo + d*cs;
    }
    return costo;
  }

}
```

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");
  }
}
```