Università di Roma "La Sapienza"

A.A. 2006-2007

Facoltà di Ingegneria

Corso di Laurea in Ingegneria Informatica

Corso di "PROGETTAZIONE DEL SOFTWARE I"
(Canale A-L & M-Z)

Esercitazione numero 9

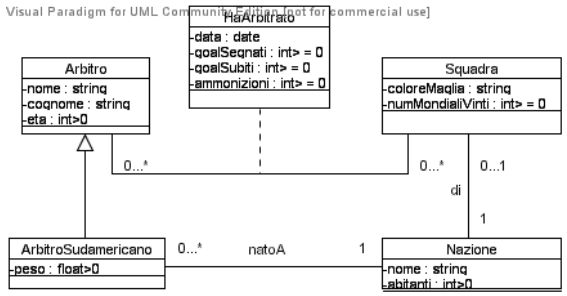
LA FASE DI REALIZZAZIONE

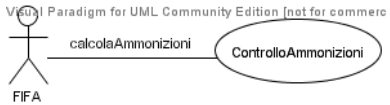
Realizzazione di associazioni a responsabilità multipla e ISA

(SOLUZIONE)

Fase di analisi

Diagramma delle classi e degli use-case





Specifica degli use case

InizioSpecificaUseCase Calcola Ammonizioni

TotaleAmmonizioni *(a: Arbitro): intero*
pre: true // nessuna precondizione
post: $result = \sum_{<a,s>\in HaArbitrato} HaArbitrato.ammonizioni(< a, s >)$.

FineSpecifica

Fase di progetto

Algoritmi per le operazioni dello use-case

Per l'operazione *totaleAmmonizioni(a: Arbitro) : int >= 0* adottiamo il seguente algoritmo:

```
result = 0;
per ogni link l di tipo HaArbitrato che coinvolge a {
  result += l.ammonizioni;
}
return result;
```


Responsabilità sulle associazioni

Riportiamo la tabella delle responsabilità.

Associazione	Classe	ha resp.
HaArbitrato	*Arbitro*	SÌ[1,2]
	Squadra	SÌ[1]
NatoA	*ArbitroSudamericano*	SÌ[3]
	Nazione	NO
Di	*Squadra*	SÌ[3]
	Nazione	SÌ[3]

1. dai requisiti
2. dagli algoritmi
3. dai vincoli di molteplicità

Strutture di dati

Abbiamo la necessità di rappresentare collezioni omogenee di oggetti, a causa dei vincoli di molteplicità 0..* delle associazioni e dei parametri e delle variabili locali dell'operazione dello use-case necessarie per l'algoritmo. Per fare ciò, utilizzeremo la classe Java `HashSet`.

Abbiamo anche la necessità di rappresentare date. A tale scopo utilizzeremo la classe Java `Data`.

API per le strutture di dati

```
package Data;

public class Data implements Cloneable {
  public Data();
  public Data(int a, int me, int g);
  public int giorno();
  public int mese();
  public int anno();
  public boolean prima(Data d);
  public void avanzaUnGiorno();
  public String toString();
  public Object clone();
  public boolean equals(Object o);
}
```


Corrispondenza fra tipi UML e Java

Possiamo riassumere il risultato delle nostre scelte nella seguente tabella di corrispondenza dei tipi UML.

Tipo UML	Rappresentazione in Java
int	`int`
int>0	`int`
int>=0	`int`
string	`String`
data	`Data`
float>0	`float`

Si noti come in tre casi (int>0, int>=0, e float>0) il tipo Java è semanticamente più esteso del corrispondente tipo UML. Prevediamo una verifica delle condizioni di ammissibilità sul lato server, perché è una soluzione di migliore qualità.

Tabelle di gestione delle proprietà di classi UML

Riassumiamo tutte 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**
Arbitro	*nome*
	cognome
Squadra	*di*
Nazione	*nome*
	di
ArbitroSudamericano	*natoA*

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

Sequenza di nascita degli oggetti

- Al momento della creazione di un oggetto di classe Squadra, possiamo assumere che sia nota la nazione di appartenenza;
- Al momento della creazione di un oggetto di classe ArbitroSudamericano, possiamo assumere che sia nota la nazione di nascita.

API delle classi Java

A titolo di esempio, viene fornita l'API della classe Arbitro:

```
public class Arbitro {
    public Arbitro(String n, String c, int e);
    public String getNome();
    public String getCognome();
    public int getEta();
    public void setEta(int e);
    public void inserisciLinkHaArbitrato(AssocHaArbitrato a);
    public void eliminaLinkHaArbitrato(AssocHaArbitrato a);
    public Set getLinkHaArbitrato();
    public String toString();
}
```


Fase di realizzazione

Considerazioni

Dalle fasi precedenti traiamo come conseguenza che dobbiamo realizzare:

- quattro classi UML,
- una associazione (*HaArbitrato*) con attributi e a responsabilità doppia, con vincoli di molteplicità 0..* e 0..*;
- una associazione (*natoA*) senza attributi e a responsabilità singola, con vincoli di molteplicità 0..* e 1..1;
- una associazione (*di*) senza attributi e a responsabilità doppia, con vincoli di molteplicità 0..1 e 1..1;
- uno use case.

Considerazioni (cont.)

Notiamo che le associazioni *natoA* e *di* hanno un vincolo di molteplicità minima diverso da zero. Sappiamo che, in generale, ciò ci obbliga a prendere dei provvedimenti, in particolare generando opportune eccezioni nel caso in cui un oggetto sia in uno stato tale da non rispettare tale vincolo.

Nel caso in esame, possiamo evitare tali provvedimenti, in quanto gli oggetti di interesse non possono mai essere in uno stato inconsistente. Infatti, quando nasce un oggetto di classe `Squadra`, ne è nota la `Nazione`, che è immutabile. Analogamente, quando nasce un oggetto di classe `ArbitroSudamericano`, ne è nota la `Nazione`, che è immutabile.

Considerazioni (cont.)

L'associazione *di*, a responsabilità doppia, richiede considerazioni particolari, in quanto deve essere gestita durante la nascita dell'oggetto di classe `Squadra`.

`AssociazioneDi`: solo `inserisci()`, ma non `elimina()`, in quanto si tratta di una proprietà immutabile per entrambe le classi.

`Nazione`: costruttore senza parametro di classe `Squadra`; solo `inserisci()`, ma non `elimina()`, in quanto si tratta di una proprietà immutabile per la classe.

`Squadra`: costruttore con parametro di classe `Nazione`, in quanto si tratta di una proprietà nota alla nascita; il costruttore invoca `AssociazioneDi.inserisci()`; solo `inserisci()`, ma non `elimina()`, in quanto si tratta di una proprietà immutabile per la classe.

Struttura dei file e dei package

```
+---AppArbitri
|   |   AssociazioneDi.java
|   |   AssociazioneHaArbitrato.java
|   |   CalcolaAmmonizioni.java
|   |   EccezionePrecondizioni.java
|   |   Nazione.java
|   |   Squadra.java
|   |   Test.java
|   |   TipoLinkDi.java
|   |   TipoLinkHaArbitrato.java
|   |
|   +---Arbitro
|   |       Arbitro.java
|   |
|   \---ArbitroSudamericano
|           ArbitroSudamericano.java
|
\---Data
        Data.java
```


La classe Java Arbitro

```
// File Arbitro/Arbitro.java
package AppArbitri.Arbitro;

import AppArbitri.*;
import java.util.*;

public class Arbitro {
    protected final String nome;
    protected final String cognome;
    protected int eta;
    protected HashSet <TipoLinkHaArbitrato> insiemeLinkHaArbitrato;
    public Arbitro(String n, String c, int e) throws EccezionePrecondizioni {
        if (eta < 0)  // CONTROLLO PRECONDIZIONI
            throw new EccezionePrecondizioni("L'eta' deve essere non nulla");
        nome = n;
        cognome = c;
        eta = e;
        insiemeLinkHaArbitrato = new
            HashSet<TipoLinkHaArbitrato>();
    }
    public String getNome() { return nome; }
    public String getCognome() { return cognome; }
    public int getEta() { return eta; }
    public void setEta(int e) throws EccezionePrecondizioni {
        if (eta < 0)  // CONTROLLO PRECONDIZIONI
            throw new EccezionePrecondizioni("L'eta' deve essere non nulla");
        eta = e;
    }
    public void inserisciLinkHaArbitrato(AssociazioneHaArbitrato a) {
        if (a != null  && a.getLink().getSquadra() != null &&
            a.getLink().getArbitro() == this)
            insiemeLinkHaArbitrato.add(a.getLink());
    }
    public void eliminaLinkHaArbitrato(AssociazioneHaArbitrato a) {
        if (a != null  && a.getLink().getArbitro() == this)
            insiemeLinkHaArbitrato.remove(a.getLink());
    }
    public Set getLinkHaArbitrato() {
        return new HashSet<TipoLinkHaArbitrato>(insiemeLinkHaArbitrato);
    }
    public String toString() { // per il test
        return nome + " " + cognome + " (" + eta + " anni)";
    }
}
```

La classe Java Squadra

```
// File Squadra.java
package AppArbitri;

import java.util.*;

public class Squadra {
    private String coloreMaglia;
    private int mondialiVinti;
    private HashSet<TipoLinkHaArbitrato> insiemeLinkHaArbitrato;
    private TipoLinkDi linkDi;
    public Squadra(Nazione n, String c, int m)
        throws EccezionePrecondizioni {
        if (m < 0)  // CONTROLLO PRECONDIZIONI
            throw new EccezionePrecondizioni
                ("Il numero di mondiali vinti deve essere positivo");
        TipoLinkDi td = null;
        try {
            td = new TipoLinkDi(this,n);
        }
        catch (EccezionePrecondizioni e) {
            System.out.println(e);
        }
        AssociazioneDi.inserisci(td);
        coloreMaglia = c;
        mondialiVinti = m;
        insiemeLinkHaArbitrato =
            new HashSet<TipoLinkHaArbitrato>();
    }
    public TipoLinkDi getLinkDi() {
        return linkDi;
    }
    public void inserisciLinkDi(AssociazioneDi a) {
        if (a != null) linkDi = a.getLink();
    }
    public String getColoreMaglia() { return coloreMaglia; }
    public void setColoreMaglia(String c) { coloreMaglia = c; }
    public int getNumeroMondialiVinti() { return mondialiVinti; }
    public void setNumeroMondialiVinti(int n)
        throws EccezionePrecondizioni {
        if (n < 0)  // CONTROLLO PRECONDIZIONI
            throw new EccezionePrecondizioni
                ("Il numero di mondiali vinti deve essere positivo");
        mondialiVinti = n;
    }
    // In alternativa:
    public void incrNumeroMondialiVinti() { mondialiVinti++; }
    public void inserisciLinkHaArbitrato(AssociazioneHaArbitrato a) {
        if (a != null  && a.getLink().getArbitro() != null &&
            a.getLink().getSquadra() == this)
            insiemeLinkHaArbitrato.add(a.getLink());
    }
    public void eliminaLinkHaArbitrato(AssociazioneHaArbitrato a) {
        if (a != null  && a.getLink().getSquadra() == this)
            insiemeLinkHaArbitrato.remove(a.getLink());
    }
    public Set getLinkHaArbitrato() {
        return new HashSet<TipoLinkHaArbitrato>(insiemeLinkHaArbitrato);
    }
    public String toString() { // per il test
        return linkDi.getNazione().getNome(); }
}
```

La classe Java ArbitroSudamericano

```
// File ArbitroSudamericano/ArbitroSudamericano.java
package AppArbitri.ArbitroSudamericano;

import AppArbitri.Arbitro.*;
import AppArbitri.*;

public class ArbitroSudamericano extends Arbitro {
    protected float peso;
    protected final Nazione nazioneNascita;
    public ArbitroSudamericano(String nome, String cognome, int eta,
                               float pe, Nazione naz)
        throws EccezionePrecondizioni {
        super(nome, cognome, eta);
        if (pe <= 0)  // CONTROLLO PRECONDIZIONI
            throw new EccezionePrecondizioni("Il peso deve essere positivo");
        peso = pe;
        nazioneNascita = naz;
    }
    public float getPeso() { return peso; }
    public Nazione getNazioneNascita() { return nazioneNascita; }
    public void setPeso(float pe)
        throws EccezionePrecondizioni {
        if (peso <= 0)  // CONTROLLO PRECONDIZIONI
            throw new EccezionePrecondizioni("Il peso deve essere positivo");
```

```
        peso = pe;
    }
    public String toString() { // per il test
        return super.toString() + " nato in " + nazioneNascita;
    }
}
```

La classe Java Nazione

```
// File Nazione.java
package AppArbitri;

public class Nazione {
    private Squadra squadra;
    private final String nome;
    private int abitanti;
    private TipoLinkDi linkDi;
    public Nazione(String n, int a) {
        nome = n;
        abitanti = a;
        squadra = null;
    }
    public void inserisciLinkDi(AssociazioneDi a) {
        if (a != null) linkDi = a.getLink();
    }
    public TipoLinkDi getLinkDi() {
        return linkDi;
    }
    public String getNome() { return nome; }
    public float getAbitanti() { return abitanti; }
    public void setAbitanti(int a) { abitanti = a; }
    public Squadra getSquadra() { return squadra; }
    public String toString() { // per il test
```

```
        return nome + " - " + abitanti + " mln ab.";
    }
}
```

La classe Java TipoLinkHaArbitrato

```
// File TipoLinkHaArbitrato.java
package AppArbitri;

import AppArbitri.Arbitro.*;
import AppArbitri.*;
import Data.*;

public class TipoLinkHaArbitrato {
    private final Arbitro arbitro;
    private final Squadra squadra;
    private final Data data;
    private final int goalSegnati;
    private final int goalSubiti;
    private final int ammonizioni;
    public Arbitro getArbitro() { return arbitro; }
    public Squadra getSquadra() { return squadra; }
    public Data getData() { return data; }
    public int getGoalSegnati() { return goalSegnati; }
    public int getGoalSubiti() { return goalSubiti; }
    public int getAmmonizioni() { return ammonizioni; }
    public TipoLinkHaArbitrato(Arbitro a, Squadra s, Data d, int gseg,
                               int gsub, int amm)
        throws EccezionePrecondizioni {
        if (a == null || s == null || d == null ||
```

```
            gseg < 0 || gsub < 0  || amm < 0)  // CONTROLLO PRECONDIZIONI
            throw new EccezionePrecondizioni
                ("Gli oggetti devono essere inizializzati\n" +
                 "e i goal e le ammonizioni devono essere positivi");
        arbitro = a;
        squadra = s;
        data = d;
        goalSegnati = gseg;
        goalSubiti = gsub;
        ammonizioni = amm;
    }
    public boolean equals(Object o) {
        if (o != null && getClass().equals(o.getClass())) {
            TipoLinkHaArbitrato a = (TipoLinkHaArbitrato)o;
            return arbitro == a.arbitro && squadra == a.squadra;
        }
        else return false;
    }
}
```

La classe Java AssociazioneHaArbitrato

```
// File AssocHaArbitrato.java
package AppArbitri;

import AppArbitri.Arbitro.*;

public class AssociazioneHaArbitrato {
  private AssociazioneHaArbitrato(TipoLinkHaArbitrato x) { link = x; }
  private TipoLinkHaArbitrato link;
  public TipoLinkHaArbitrato getLink() { return link; }
  public static void inserisci(TipoLinkHaArbitrato y) {
    if (y != null && y.getArbitro() != null && y.getSquadra() != null) {
      AssociazioneHaArbitrato k = new AssociazioneHaArbitrato(y);
      k.link.getArbitro().inserisciLinkHaArbitrato(k);
      k.link.getSquadra().inserisciLinkHaArbitrato(k);
    }
  }
  public static void elimina(TipoLinkHaArbitrato y) {
    if (y != null && y.getArbitro() != null && y.getSquadra() != null) {
      AssociazioneHaArbitrato k = new AssociazioneHaArbitrato(y);
      k.link.getArbitro().eliminaLinkHaArbitrato(k);
      k.link.getSquadra().eliminaLinkHaArbitrato(k);
    }
  }
}
```


La classe Java TipoLinkDi

```
// File TipoLinkDi.java
package AppArbitri;

import AppArbitri.Arbitro.*;

public class TipoLinkDi {
    private final Squadra laSquadra;
    private final Nazione laNazione;
    public TipoLinkDi(Squadra x, Nazione y)
        throws EccezionePrecondizioni {
        if (x == null || y == null)  // CONTROLLO PRECONDIZIONI
            throw new EccezionePrecondizioni
                ("Gli oggetti devono essere inizializzati");
        laSquadra = x; laNazione = y;
    }
    public boolean equals(Object o) {
        if (o != null && getClass().equals(o.getClass())) {
            TipoLinkDi b = (TipoLinkDi)o;
            return b.laNazione == laNazione && b.laSquadra == laSquadra;
        }
        else return false;
    }
    public Squadra getSquadra() { return laSquadra; }
    public Nazione getNazione() { return laNazione; }
}
```

La classe Java AssociazioneDi

```
// File AssocDi.java
package AppArbitri;

import AppArbitri.Arbitro.*;

public class AssociazioneDi {
    private AssociazioneDi(TipoLinkDi x) { link = x; }
    private TipoLinkDi link;
    public TipoLinkDi getLink() { return link; }
    public static void inserisci(TipoLinkDi y) {
        if (y != null &&
            y.getSquadra().getLinkDi() == null &&
            y.getNazione().getLinkDi() == null) {
            AssociazioneDi k = new AssociazioneDi(y);
            y.getNazione().inserisciLinkDi(k);
            y.getSquadra().inserisciLinkDi(k);
        }
    }
}
```

Realizzazione in Java dello use case

```
// File CalcolaAmmonizioni.java
package AppArbitri;

import java.util.*;
import AppArbitri.Arbitro.*;

public final class CalcolaAmmonizioni {
    private CalcolaAmmonizioni() { };
    public static int totaleAmmonizioni(Arbitro a) {
        int result = 0;
        Set insiemeLink = a.getLinkHaArbitrato();
        Iterator it = insiemeLink.iterator();
        while(it.hasNext()) {
            TipoLinkHaArbitrato link =
                (TipoLinkHaArbitrato)it.next();
            result += link.getAmmonizioni();
        }
        return result;
    }
}
```