SAPIENZA Università di Roma

A.A. 2008-2009

Facoltà di Ingegneria

Corso di Laurea in Ingegneria Informatica

ESERCITAZIONI DI PROGETTAZIONE DEL SOFTWARE

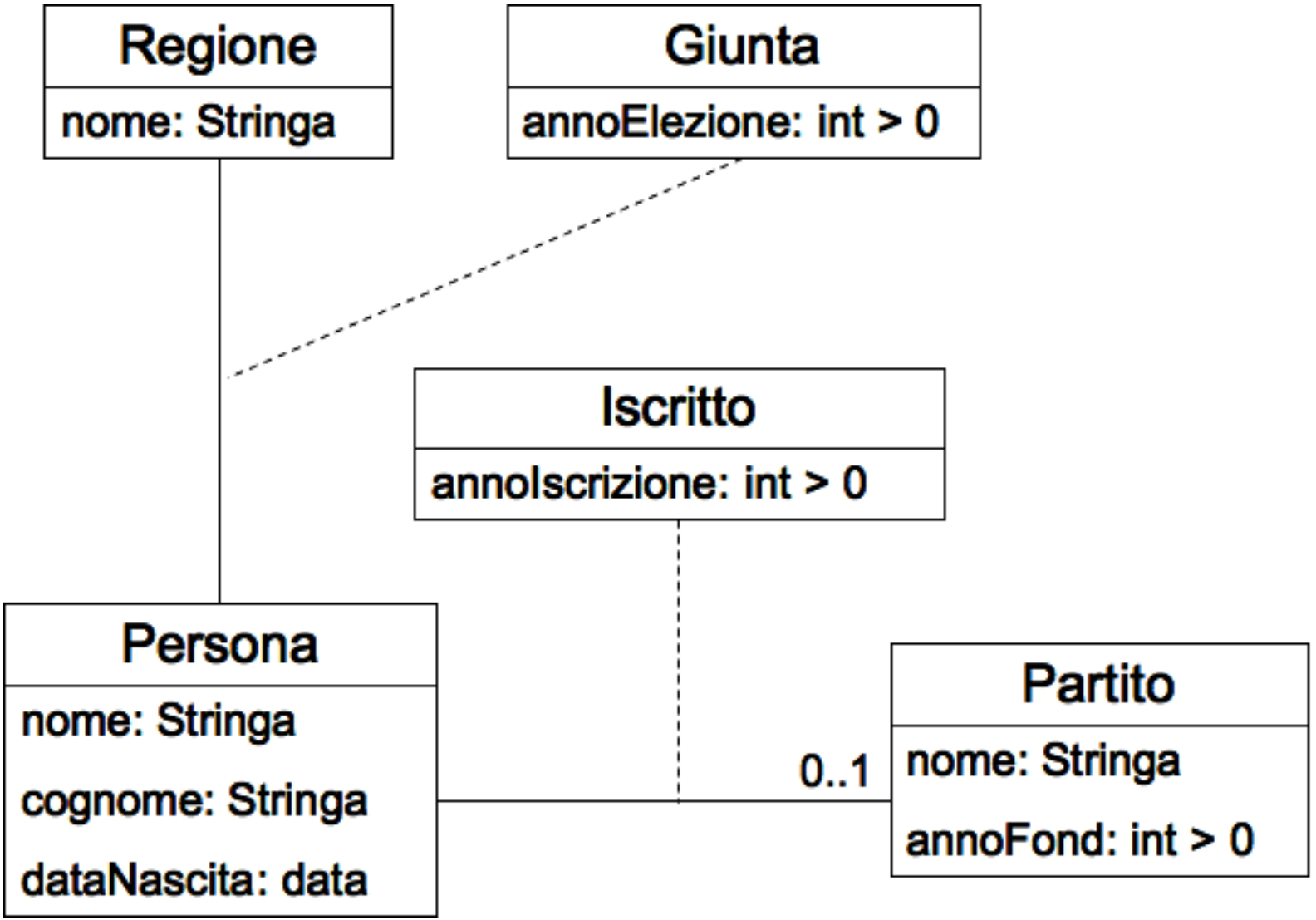
LA FASE DI REALIZZAZIONE

Realizzazione di classi e associazioni con responsabilità singola

(SOLUZIONE)

Fase di analisi

Diagramma delle classi



Fase di progetto

Responsabilità sulle associazioni

Riportiamo la tabella delle responsabilità.

Associazione	Classe	ha resp.
Giunta	*Regione*	SÌ[1,2]
	Persona	NO
Iscritto	*Persona*	SÌ[1,2,3]
	Partito	NO

1. dai requisiti
2. dalle operazioni
3. dai vincoli di molteplicità

Strutture di dati

Abbiamo la necessità di rappresentare collezioni omogenee di oggetti, poiché la classe *Regione* ha responsabilità sull'associazione *Giunta*, la cui molteplicità è 0..*, per la realizzazione di quest'ultima avremo bisogno di rappresentare *insiemi di link*. Per fare ciò, utilizzeremo la classe Java `HashSet`.

Corrispondenza fra tipi UML e Java

La tabella di corrispondenza dei tipi UML è la seguente.

Tipo UML	Rappresentazione in Java
integer	`int`
posinteger	`int`
string	`String`
date	`int, int, int`
Insieme	`HashSet`

- Per tenere conto del fatto che `int` è semanticamente più esteso del tipo UML "posinteger", 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**
Regione	*nome*
Persona	*nome* *cognome* *dataNascita*

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

Sequenza di nascita degli oggetti e valori alla nascita

Gli oggetti nascono indipendentemente gli uni dagli altri.

Non possiamo fare alcuna assunzione sulla sequenza di nascita degli oggetti.

Per quanto riguarda le proprietà note alla nascita non è ragionevole assumere che esista un valore di default (valido per tutti gli oggetti).

API di ogni classe Java progettata

A titolo di esempio, riportiamo la API della classe Java `Partito`.

```
public class Partito {
// COSTRUTTORI
  public Partito(String n, int anno);
// GESTIONE ATTRIBUTI
  public String getNome();
  public int getAnnoFondazione();
}
```

Fase di realizzazione

Struttura dei file e dei package

```
\---AppGiunte
    |   Regione.java
    |   Partito.java
    |   Persona.java
    |   TipoLinkIscritto.java
    |   TipoLinkGiunta.java
    |   Controlli.java
    |   EccezionePrecondizioni.java
```

Classe Regione

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

public class Regione {
    private final String nome;
    private HashSet<TipoLinkGiunta> giunta;
    public Regione(String x) {
        nome = x;
        giunta = new HashSet<TipoLinkGiunta>();
    }
    public String getNome() { return nome; }
    public void inserisciInGiunta(TipoLinkGiunta g) {
        if (g != null && g.getRegione() == this)
            giunta.add(g);
    }
    public void eliminaInGiunta(TipoLinkGiunta g) {
        if (g != null && g.getRegione() == this)
            giunta.remove(g);
    }
    public Set<TipoLinkGiunta> getGiunta() {
        return (Set<TipoLinkGiunta>)giunta.clone(); }
}
```

Classe Persona

```
// File Persona.java
public class Persona {
    private final String nome, cognome;
    private final int giorno_nascita, mese_nascita,
        anno_nascita;
    private TipoLinkIscritto iscritto;
    public Persona(String n, String c, int g, int m, int a) {
        nome = n;
        cognome = c;
        giorno_nascita = g;
        mese_nascita = m;
        anno_nascita = a;
    }
    public String getNome() { return nome; }
    public String getCognome() { return cognome; }
    public int getGiornoNascita() { return giorno_nascita; }
    public int getMeseNascita() { return mese_nascita; }
    public int getAnnoNascita() { return anno_nascita; }
    public String toString() {
        return nome + ' ' + cognome + ", "  + giorno_nascita
            + "/" + mese_nascita + "/" + anno_nascita;
    }
    public TipoLinkIscritto getLinkIscritto() { return iscritto; }
    public void inserisciLinkIscritto(TipoLinkIscritto t) {
```

```
        if (iscritto == null && t != null && t.getPersona() == this)
            iscritto = t;
    }
    public void eliminaLinkIscritto() {
        iscritto = null;
    }
}
```

Classe Partito

```
// File Partito.java
public class Partito {
    private final String nome;
    private final int anno_fondazione;
    public Partito(String n, int anno) throws EccezionePrecondizioni {
        if (anno <= 0)  // CONTROLLO PRECONDIZIONI
            throw new EccezionePrecondizioni
                ("L'anno di fondazione deve essere positivo");
        nome = n;
        anno_fondazione = anno;
    }
    public String getNome() { return nome; }
    public int getAnnoFondazione() { return anno_fondazione; }
}
```

Classe TipoLinkIscritto

```
// File TipoLinkIscritto.java
public class TipoLinkIscritto {
    private final Partito ilPartito;
    private final Persona laPersona;
    private final int annoIscrizione;
    public TipoLinkIscritto(Partito x, Persona y, int a)
        throws EccezionePrecondizioni {
        if (x == null || y == null|| a <= 0)  // CONTROLLO PRECONDIZIONI
            throw new EccezionePrecondizioni
                ("Gli oggetti devono essere inizializzati\n" +
                 "e l'anno di iscrizione deve essere positivo");
        ilPartito = x; laPersona = y; annoIscrizione = a;
    }
    public Partito getPartito() { return ilPartito; }
    public Persona getPersona() { return laPersona; }
    public int getAnnoIscrizione() { return annoIscrizione; }
    public boolean equals(Object o) {
        if (o != null && getClass().equals(o.getClass())) {
            TipoLinkIscritto b = (TipoLinkIscritto)o;
            return b.laPersona == laPersona && b.ilPartito == ilPartito;
        }
        else return false;
    }
    public int hashCode() {
```

```
        return laPersona.hashCode() + ilPartito.hashCode();
    }
}
```

Classe TipoLinkGiunta

```
// File TipoLinkGiunta.java
public class TipoLinkGiunta {
    private final Regione laRegione;
    private final Persona laPersona;
    private final int annoElezione;
    public TipoLinkGiunta(Regione x, Persona y, int a)
        throws EccezionePrecondizioni {
        if (x == null || y == null || a <= 0)  // CONTROLLO PRECONDIZIONI
            throw new EccezionePrecondizioni
                ("Gli oggetti devono essere inizializzati\n" +
                 "e l'anno di elezione deve essere positivo");
        laRegione = x;     laPersona = y;
        annoElezione = a;
    }
    public Regione getRegione() { return laRegione; }
    public Persona getPersona() { return laPersona; }
    public int getAnnoElezione() { return annoElezione; }
    public boolean equals(Object o) {
        if (o != null && getClass().equals(o.getClass())) {
            TipoLinkGiunta t = (TipoLinkGiunta)o;
            return t.laPersona == laPersona && t.laRegione == laRegione;
        }
        else return false;
    }
```

```
    public int hashCode() {
        return laPersona.hashCode() + laRegione.hashCode();
    }
}
```

Realizzazione Operazione

```
// File Controlli.java
import java.util.*;
public final class Controlli {
    public static Partito monocolore(Regione r) {
        Set<TipoLinkGiunta> s = r.getGiunta();
        if (s == null) return null;
        Iterator<TipoLinkGiunta> iter = s.iterator();
        Persona per;
        Partito par;
        if (!iter.hasNext()) return null;

        TipoLinkGiunta link = iter.next();
        per = link.getPersona();
        TipoLinkIscritto li = per.getLinkIscritto();
        if (li == null) return null;
        par = li.getPartito();
        while (iter.hasNext()) {
            link = iter.next();
            Persona per1 = link.getPersona();
            TipoLinkIscritto li1 = per1.getLinkIscritto();
            if (li1 == null) return null;
            Partito par1 = per1.getLinkIscritto().getPartito();
            if (par != par1) return null;
        }
```

```
        return par;
    }
    private Controlli() { }
}
```