SAPIENZA Università di Roma
Facoltà di Ingegneria

Corso di
"PROGETTAZIONE DEL SOFTWARE I"
(Laurea in Ingegneria Informatica)
Proff. Giuseppe De Giacomo e Domenico Lembo
Canali A-L & M-Z
A.A. 2007-08

Compito d'esame del 16 aprile 2008

SOLUZIONE

Requisiti

L'applicazione da progettare riguarda una parte del sistema di gestione di elezioni in un collegio elettorale del comune di Noncè. Alle elezioni vengono presentate delle liste, ciascuna con un nome ed un simbolo (una stringa che identifica il file con l'immagine). Le liste sono suddivise in liste per l'elezione del sindaco e liste per l'elezione del consiglio comunale. Ogni lista per il sindaco contiene un solo candidato. Mentre ogni lista per il consiglio contiene un insieme non vuoto e ordinato di candidati. Una lista per il consiglio può essere collegata ad una (e non più di una) lista per il sindaco. Ogni candidato è caratterizzato dal nome e dall'indirizzo web del suo blog (una stringa). Un candidato può presentarsi in al più una lista per il consiglio ed al più una lista per il sindaco. Nel comune di Noncè ci sono diversi seggi elettorali ciascuno contraddistinto da un codice e dal numero di cittadini aventi diritto al voto iscritti al seggio. Data una lista ed un seggio è di interesse memorizzare quanti voti ha preso la lista in quel seggio, ma solo se il numero di voti è maggiore di 0.

Requisiti (cont.)

L'ufficio elezioni è interessato ad effettuare diversi controlli sui seggi e le liste, in particolare:

- data una lista ℓ, restituire l'insieme dei candidati a sindaco contenuti in ℓ, cioè se ℓ è una lista per il sindaco restituire l'insieme costituito dal solo candidato presente in essa, se ℓ è una lista per il consiglio restituire l'insieme formato dai candidati presenti in essa che sono anche candidati in una lista per il sindaco;

- dato un seggio s, restituire la percentuale di votanti, cioè il rapporto tra voti ottenuti dalle varie liste nel seggio s e numero di iscritti ad s.

Requisiti (cont.)

Domanda 1. Basandosi sui requisiti riportati sopra, svolgere la fase di analisi producendo lo schema concettuale in UML per l'applicazione e motivando, qualora ce ne fosse bisogno, le scelte effettuate.

Domanda 2. Svolgere la fase di progetto, illustrando i prodotti rilevanti di tale fase e motivando, qualora ce ne fosse bisogno, le scelte effettuate. È obbligatorio solo progettare gli algoritmi e definire le responsabilità sulle associazioni.

Domanda 3. Svolgere la fase di realizzazione, producendo un programma Java e motivando, qualora ce ne fosse bisogno, le scelte effettuate. È obbligatorio realizzare in Java solo i seguenti aspetti dello schema concettuale:

- le classe `ListaConsiglio` e `Candidato`, le eventuali classi da cui sono derivate, e tutte le associazioni che le legano;
- il primo use case.

Fase di analisi

Diagramma delle classi

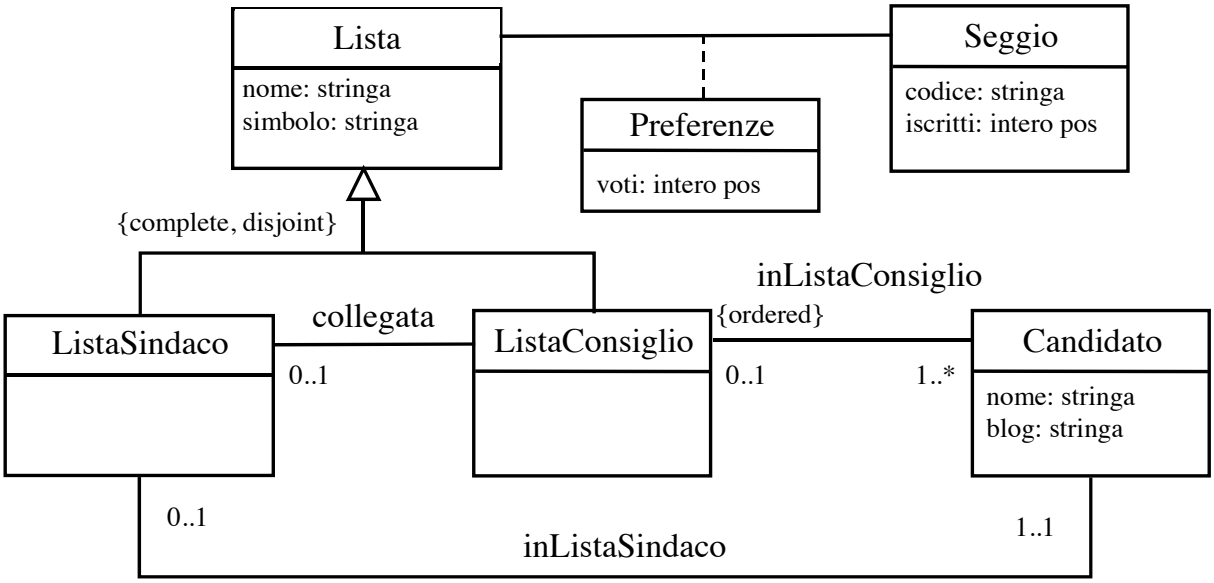

Diagramma degli use case

Specifica dello use case

InizioSpecificaUseCase Controlli

CandidatiSindaco *(l: Lista): Insieme(Candidato)*

pre: *true*

post: Definiamo l'insieme C tale che:

se $l \in ListaSindaco$

$$C \doteq \{c \in Candidato \mid (c, l) \in inListaSindaco\}$$

altrimenti, se $l \in ListaConsiglio$:

$$C \doteq \{c \in Candidato \mid (c, l) \in inListaConsiglio \wedge \exists l' \in ListaSindaco.(c, l') \in ListaSindaco\}$$

result $= C$

...

Specifica dello use case (cont.)

...

PercentualeVotanti *(s: Seggio): reale*

pre: $s.iscritti \neq 0$

post: Definiamo l'insieme

$$C \doteq \{(l, v) \mid (s, l) \in preferenze \wedge preferenze.voti(s, l) = v \}$$

result $= (\sum_{(l,v) \in C} v \backslash\, s.iscritti) * 100$

FineSpecifica

Fase di progetto

Algoritmi per le operazioni dello use-case

Adottiamo i seguenti algoritmi:

- Per l'operazione **CandidatiSindaco**:

```
Insieme(Candidato) result = ∅
se l.class = ListaSindaco
   result=result ∪ {l.inListaSindaco};
altrimenti
   per ogni link ll di tipo inListaConsiglio in cui l è coinvolta
        se esiste un link di tipo inListaSindaco in cui ll.Candidato è coinvolto
           result = result ∪ {ll.Candidato};
return result;
```

- Per l'operazione **PercentualeVotanti**:

```
int sum = 0;
per ogni link l di tipo preferenze in cui s è coinvolto
    sum = sum + l.voti;
return (sum\s.iscritti)*100;
```


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.
preferenze	*Seggio*	SÌ[2]
	Lista	NO
collegata	*ListaSindaco*	NO
	ListaConsiglio	SI[3]
inListaConsiglio	*ListaConsiglio*	SÌ[2,3]
	Candidato	SÌ[3]
inListaSindaco	*ListaSindaco*	SÌ[2,3]
	Candidato	SÌ[2,3]

Strutture di dati

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

- dei vincoli di molteplicità 0..* ed 1..* delle associazioni,
- dei risultati delle operazioni e delle variabili locali necessarie per i vari algoritmi.

Per fare ciò, utilizzeremo l'interfaccia `Set` e la classe `HashSet` del collection framework di Java 1.5, per collezioni non ordinate. Inoltre, per rappresentare l'associazione ordinata `inListaConsiglio` useremo l'interfaccia `List` e la classe `LinkedList`.

Corrispondenza fra tipi UML e Java

Riassumiamo le nostre scelte nella seguente tabella di corrispondenza dei tipi UML.

Tipo UML	Rappresentazione in Java
intero pos	`int`
stringa	`String`
reale	`double`
Insieme	`HashSet`
Collezione ordinata	`LinkedList`

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**
Lista	*nome*
Seggio	*codice*
Candidato	*nome*

	Proprietà	
Classe UML	**nota alla nascita**	**non nota alla nascita**
ListaSindaco	–	*Candidato*

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.

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 *ListaConsiglio*;
2. la classe UML *Candidato*
3. l'associazione UML *inListaConsiglio* con responsabilità doppia e con vincoli di molteplicità 0..1 (molteplicità massima finita) e 1..* (molteplicità minima diversa da zero);
4. il primo use case.

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

Struttura dei file e dei package

```
AppElezioni
  |   AssociazioneInListaConsiglio.java
  |   AssociazioneInListaSindaco.java
  |   Candidato.java
  |   Controlli.java
  |   EccezioneMoltMax.java
  |   EccezioneMoltMin.java
  |   EccezionePrecondizioni.java
  |   Seggio.java
  |   TestElezioni.java
  |   TipoLinkInListaConsiglio.java
  |   TipoLinkInListaSindaco.java
  |   TipoLinkPreferenze.java
  |
  +---Lista
  |       Lista.java
  |
  +---ListaConsiglio
  |       ListaConsiglio.java
  |
  \---ListaSindaco
          ListaSindaco.java
```

La classe Java Lista

```
// File AppElezioni/Lista/Lista.java
package AppElezioni.Lista;

import AppElezioni.*;
import java.util.*;

public abstract class Lista  {
    protected String nome;
    protected String simbolo;

    public Lista(String n, String s) {
        nome = n;
        simbolo = s;
    }

    public String getNome() { return nome; }

    public String getSimbolo() { return simbolo; }

    public void setSimbolo(String s) { simbolo = s; }
}
```


La classe Java ListaConsiglio

```
// File AppElezioni/ListaConsiglio/ListaConsiglio.java
package AppElezioni.ListaConsiglio;

import AppElezioni.*;
import AppElezioni.Lista.*;
import AppElezioni.ListaSindaco.*;
import java.util.*;

public class ListaConsiglio extends Lista {
    protected LinkedList<TipoLinkInListaConsiglio> insieme_link;
    protected ListaSindaco collegata;
    private static final int MIN_LINK_IN_LISTA_CONSIGLIO = 1;
    public ListaConsiglio(String n, String s) {
        super(n,s);
        insieme_link = new LinkedList<TipoLinkInListaConsiglio>();
        collegata = null;
    }
    public int quantiCandidati() {
        return insieme_link.size();
    }
    public void inserisciLinkInListaConsiglio(AssociazioneInListaConsiglio a) {
        if ((a != null) && !insieme_link.contains(a.getLink()))
          insieme_link.add(a.getLink());
    }
```

```
    public void eliminaLinkInListaConsiglio(AssociazioneInListaConsiglio a) {
        if (a != null) insieme_link.remove(a.getLink());
    }
    public List<TipoLinkInListaConsiglio> getLinkInListaConsiglio()
                                                    throws EccezioneMoltMin{
      if (insieme_link.size() < MIN_LINK_IN_LISTA_CONSIGLIO)
          throw new EccezioneMoltMin("Molteplicita' minima violata");
      return (LinkedList<TipoLinkInListaConsiglio>)insieme_link.clone();
    }
    public void setCollegata(ListaSindaco l) {
      collegata = l;
    }
    public ListaSindaco getCollegata() {
      return collegata;
    }
}
```

La classe Java ListaSindaco

```
// File AppElezioni/ListaSindaco/ListaSindaco.java
package AppElezioni.ListaSindaco;

import AppElezioni.Lista.*;
import AppElezioni.*;
import java.util.*;

public class ListaSindaco extends Lista {
    protected TipoLinkInListaSindaco link;
    private static final int MIN_LINK_IN_LISTA_SINDACO = 1;

    public ListaSindaco(String n, String s) {
        super(n,s);
        link = null;
    }
    public int quantiSindaco() {
        if (link == null) return 0;
        else return 1;
    }
    public void inserisciLinkInListaSindaco(AssociazioneInListaSindaco a) {
        if (a != null) link = a.getLink();
    }
    public void eliminaLinkInListaSindaco(AssociazioneInListaSindaco a) {
        if (a != null) link = null;
```

```
    }
    public TipoLinkInListaSindaco getLinkInListaSindaco()
                                    throws EccezioneMoltMin{
        if (quantiSindaco() < MIN_LINK_IN_LISTA_SINDACO)
            throw new EccezioneMoltMin("Molteplicita' minima violata");
        return link;
    }
}
```

La classe Java Candidato

```
// File AppElezioni/Candidato.java
package AppElezioni;

import AppElezioni.ListaConsiglio.*;
import AppElezioni.ListaSindaco.*;
import AppElezioni.*;
import java.util.*;

public class Candidato  {
    private String nome;
    private String blog;
    private TipoLinkInListaConsiglio link_consiglio;
    private TipoLinkInListaSindaco link_sindaco;
    public Candidato(String n, String b) {
        nome = n;
        blog = b;
        link_consiglio = null;
        link_sindaco = null;
    }
    public String getNome() { return nome; }
    public void setBlog(String b) { blog = b; }
    public String getBlog() { return blog; }
```

```
    public void inserisciLinkInListaConsiglio(AssociazioneInListaConsiglio a) {
        if (a != null) link_consiglio = a.getLink();
    }
    public void eliminaLinkInListaConsiglio(AssociazioneInListaConsiglio a) {
        if (a != null) link_consiglio = null;
    }
    public TipoLinkInListaConsiglio getLinkInListaConsiglio() {
      return link_consiglio;
    }
    public void inserisciLinkInListaSindaco(AssociazioneInListaSindaco a) {
        if (a != null) link_sindaco = a.getLink();
    }
    public void eliminaLinkInListaSindaco(AssociazioneInListaSindaco a) {
        if (a != null) link_sindaco = null;
    }
    public TipoLinkInListaSindaco getLinkInListaSindaco(){
        return link_sindaco;
    }
}
```

La classe Java TipoLinkInListaConsiglio

```
// File AppElezioni/TipoLinkInListaConsiglio.java
package AppElezioni;

import AppElezioni.ListaConsiglio.*;
import AppElezioni.Candidato.*;
import AppElezioni.*;

public class TipoLinkInListaConsiglio {
    private final ListaConsiglio laListaConsiglio;
    private final Candidato ilCandidato;
    public TipoLinkInListaConsiglio(Candidato c, ListaConsiglio l)
        throws EccezionePrecondizioni {
        if (l == null || c == null)  // CONTROLLO PRECONDIZIONI
            throw new EccezionePrecondizioni
                ("Gli oggetti devono essere inizializzati");
        laListaConsiglio = l;
        ilCandidato = c;
    }
    public boolean equals(Object o) {
        if (o != null && getClass().equals(o.getClass())) {
            TipoLinkInListaConsiglio a = (TipoLinkInListaConsiglio) o;
            return a.laListaConsiglio == laListaConsiglio &&
                a.ilCandidato == ilCandidato;
        }
```

```
        else return false;
    }
    public int hashCode() {
        return laListaConsiglio.hashCode() + ilCandidato.hashCode();
    }
    public ListaConsiglio getListaConsiglio() { return laListaConsiglio; }
    public Candidato getCandidato() { return ilCandidato; }
    public String toString() {
        return "<" + laListaConsiglio + ", " + ilCandidato + ">";
    }
}
```

La classe Java **AssociazioneInListaConsiglio**

```
// File AppElezioni/AssociazioneInListaConsiglio.java
package AppElezioni;

import AppElezioni.ListaConsiglio.*;

public final class AssociazioneInListaConsiglio {
    private AssociazioneInListaConsiglio(TipoLinkInListaConsiglio x) { link = x; }
    private TipoLinkInListaConsiglio link;
    public TipoLinkInListaConsiglio getLink() { return link; }
    public static void inserisci(TipoLinkInListaConsiglio y) {
        if (y != null &&
            y.getCandidato().getLinkInListaConsiglio() == null) {
            AssociazioneInListaConsiglio k = new AssociazioneInListaConsiglio(y);
            y.getCandidato().inserisciLinkInListaConsiglio(k);
            y.getListaConsiglio().inserisciLinkInListaConsiglio(k);
        }
    }
    public static void elimina(TipoLinkInListaConsiglio y) {
        if (y != null && y.getCandidato().getLinkInListaConsiglio().equals(y)) {
            AssociazioneInListaConsiglio k = new AssociazioneInListaConsiglio(y);
            y.getCandidato().eliminaLinkInListaConsiglio(k);
            y.getListaConsiglio().eliminaLinkInListaConsiglio(k);
        }
    }
}
```

La classe Java TipoLinkInListaSindaco

```
// File AppElezioni/TipoLinkAssegnatoInListaSindaco.java
package AppElezioni;

import AppElezioni.ListaSindaco.*;
import AppElezioni.Candidato.*;
import AppElezioni.*;

public class TipoLinkInListaSindaco {
    private final Candidato ilCandidato;
    private final ListaSindaco laListaSindaco;
    public TipoLinkInListaSindaco(Candidato c, ListaSindaco l)
        throws EccezionePrecondizioni {
        if (c == null || l == null)  // CONTROLLO PRECONDIZIONI
            throw new EccezionePrecondizioni
                ("Gli oggetti devono essere inizializzati");
        ilCandidato = c;
        laListaSindaco = l;
    }
    public boolean equals(Object o) {
        if (o != null && getClass().equals(o.getClass())) {
            TipoLinkInListaSindaco a = (TipoLinkInListaSindaco) o;
            return a.ilCandidato == ilCandidato &&
                a.laListaSindaco == laListaSindaco;
        }
```



```
        else return false;
    }
    public int hashCode() {
        return ilCandidato.hashCode() + laListaSindaco.hashCode();
    }
    public Candidato getCandidato() { return ilCandidato; }
    public ListaSindaco getListaSindaco() { return laListaSindaco; }
    public String toString() {
        return "<" + ilCandidato + ", " + laListaSindaco + ">";
    }
}
```

La classe Java AssociazioneInListaSindaco

```
// File AppElezioni/AssociazioneInListaSindaco.java
package AppElezioni;

import AppElezioni.ListaSindaco.*;

public final class AssociazioneInListaSindaco {
    private AssociazioneInListaSindaco(TipoLinkInListaSindaco x) { link = x; }
    private TipoLinkInListaSindaco link;
    public TipoLinkInListaSindaco getLink() { return link; }
    public static void inserisci(TipoLinkInListaSindaco y) {
        if (y != null &&
            y.getCandidato().getLinkInListaSindaco() == null &&
            y.getListaSindaco().quantiSindaco() == 0) {
            AssociazioneInListaSindaco k = new AssociazioneInListaSindaco(y);
            y.getCandidato().inserisciLinkInListaSindaco(k);
            y.getListaSindaco().inserisciLinkInListaSindaco(k);
        }
    }
    public static void elimina(TipoLinkInListaSindaco y) {
        if (y != null && y.getCandidato().getLinkInListaSindaco().equals(y)) {
            AssociazioneInListaSindaco k = new AssociazioneInListaSindaco(y);
            y.getCandidato().eliminaLinkInListaSindaco(k);
            y.getListaSindaco().eliminaLinkInListaSindaco(k);
        }
```



```
    }
}
```

La classe Java Seggio

```
// File AppElezioni/Seggio.java
package AppElezioni;

import AppElezioni.Lista.*;
import AppElezioni.*;
import java.util.*;

public class Seggio  {
    private String codice;
    private int iscritti;
    private HashSet<TipoLinkPreferenze> insieme_link;

    public Seggio(String c, int i) {
        if (i<=0)
            throw new EccezionePrecondizioni("numero iscritti non corretto");
        codice = c;
        iscritti = i;
        insieme_link = new HashSet<TipoLinkPreferenze>();
    }
    public String getCodice() { return codice; }
    public int getIscritti() { return iscritti; }
    public void setIscritti(int i) throws EccezionePrecondizioni {
        if (i<=0)
            throw new EccezionePrecondizioni("numero iscritti non corretto");
```



```
        iscritti = i;
    }

    public void inserisciLinkPreferenze(TipoLinkPreferenze t) {
        if (t != null && t.getSeggio() == this)
           insieme_link.add(t);
    }
    public void eliminaLinkPreferenze(TipoLinkPreferenze t) {
        if (t != null && t.getSeggio() == this)
           insieme_link.remove(t);
    }
    public Set<TipoLinkPreferenze> getLinkPreferenze() {
        return (HashSet<TipoLinkPreferenze>)insieme_link.clone();
    }
}
```

La classe Java TipoLinkPreferenze

```
// File AppElezioni/TipoLinkPreferenze.java
package AppElezioni;

import AppElezioni.Lista.*;
import AppElezioni.Seggio.*;
import AppElezioni.*;

public class TipoLinkPreferenze {
    private final Seggio ilSeggio;
    private final Lista laLista;
    private final int voti;
    public TipoLinkPreferenze(Seggio s, Lista l, int v)
        throws EccezionePrecondizioni {
        if (s == null || l == null || v <= 0)  // CONTROLLO PRECONDIZIONI
            throw new EccezionePrecondizioni
                ("Gli oggetti devono essere inizializzati ed il numero"+
                                " di voti deve essere un intero positivo");
        ilSeggio = s;
        laLista = l;
        voti = v;
    }
    public boolean equals(Object o) {
        if (o != null && getClass().equals(o.getClass())) {
            TipoLinkPreferenze a = (TipoLinkPreferenze) o;
```



```
            return a.ilSeggio == ilSeggio &&
                a.laLista == laLista;
        }
        else return false;
    }
    public int hashCode() {
        return ilSeggio.hashCode() + laLista.hashCode();
    }
    public Seggio getSeggio() { return ilSeggio; }
    public Lista getLista() { return laLista; }
    public int getVoti() { return voti; }
    public String toString() {
        return "<" + ilSeggio + ", " + laLista + ", " + voti + ">";
    }
}
```

Realizzazione in Java dello use case

```
// File AppElezioni/Controlli.java
package AppElezioni;
import java.util.*;
import AppElezioni.Lista.*;
import AppElezioni.ListaSindaco.*;
import AppElezioni.ListaConsiglio.*;

public final class Controlli {
    private Controlli() { }
    public static List<Candidato> candidatiSindaco(Lista l)
           throws EccezioneMoltMin, EccezioneMoltMax {
        LinkedList<Candidato> res = new LinkedList<Candidato>();
        if (l.getClass().equals(ListaSindaco.class)){
          ListaSindaco ll = (ListaSindaco) l;
          res.add(ll.getLinkInListaSindaco().getCandidato());
          return res;
        }
        else {
          ListaConsiglio lc = (ListaConsiglio) l;
          List<TipoLinkInListaConsiglio> s = lc.getLinkInListaConsiglio();
          Iterator<TipoLinkInListaConsiglio> it = s.iterator();
          while (it.hasNext()) {
             TipoLinkInListaConsiglio t = it.next();
             if (t.getCandidato().getLinkInListaSindaco() != null)
```



```
                res.add(t.getCandidato());
          }
          return res;
       }
    }
}
```

La realizzazione del secondo use case è lasciata per esercizio

Realizzazione in Java delle classi per eccezioni

```
// File AppElezioni/EccezioneMoltMin.java
package AppElezioni;

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

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



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

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;
  }
}
```