

ESAME DI FONDAMENTI DI INFORMATICA T-2 del 18/06/2013

Proff. E. Denti – G. Zannoni

Tempo a disposizione: 4 ore MAX

NB: il candidato troverà nell'archivio ZIP scaricato da Esamix anche il software "Start Kit"

NOME PROGETTO ECLIPSE: CognomeNome-matricola (es. RossiMario-0000123456)

Nella repubblica di Dentinia si deve eleggere la Camera Grande, con sistema elettorale proporzionale. È stata richiesta un'applicazione che calcoli l'attribuzione dei seggi con i più noti meccanismi proporzionali esistenti.

DESCRIZIONE DEL DOMINIO DEL PROBLEMA

Sebbene tutti i meccanismi proporzionali mirino ad attribuire i seggi "in proporzione" ai voti ottenuti da ciascuno, esistono diversi metodi di calcolo, con caratteristiche differenti, che spesso portano a risultati finali diversi.

I sistemi a quoziente

Nei sistemi a quoziente, l'attribuzione dei seggi avviene determinando in primo luogo il *quoziente elettorale*, ossia il numero di voti necessari per ottenere un seggio: nel caso più semplice, si adotta il **quoziente naturale**, ossia il risultato della divisione (intera) fra il totale dei voti espressi e il numero di seggi da assegnare. I seggi spettanti a ogni partito si ottengono poi dividendo (con divisione intera) per tale quoziente i voti da ciascuno ottenuti. Se, come è probabile, le divisioni danno luogo a resto, non tutti i seggi in palio vengono assegnati: i seggi rimasti sono attribuiti ai partiti che hanno i resti (voti inutilizzati) più alti. In pratica, detto Q il quoziente elettorale:

- i seggi S_i da attribuire al partito P_i si ottengono dividendo i suoi voti per Q : $S_i = V_i / Q$
i voti residui (resti) del partito P_i non utilizzati per assegnare tali S_i seggi sono: $R_i = V_i \% Q$
- i seggi rimasti ancora da assegnare si assegnano a quei partiti che hanno ottenuto i resti più alti.

I sistemi a divisore

Nei sistemi a divisore, invece, i seggi si attribuiscono seguendo una *graduatoria*, ottenuta dividendo i voti presi da ogni partito per una *serie prestabilita di divisori*. Nel sistema più noto (**D'Hondt**) i divisori sono i numeri naturali 1, 2, 3, 4, 5, ecc., fino (al limite, se necessario) al numero di seggi da assegnare; i seggi sono quindi assegnati ai vari partiti partendo dal valore più alto in graduatoria e via via scendendo, fino ad assegnare tutti i seggi in palio. In pratica:

- si divide il numero dei voti presi da ciascun partito per una *successione di divisori* $\mathcal{D} = D_1, D_2, D_3, \dots, D_S$;
- si assegnano i seggi ai vari partiti secondo la graduatoria di tali valori, partendo dal più alto.

ESEMPIO: la figura sottostante illustra il procedimento di assegnazione di 15 seggi con i due sistemi.

Nel sistema a quoziente naturale (a destra) si calcola il quoziente elettorale dividendo il totale dei voti espressi (30.652) per i seggi da assegnare (15), ottenendo $Q=2043$. Si dividono perciò i voti ottenuti da ogni partito per 2043: trattandosi di divisioni intere, risultano così assegnati solo 12 seggi su 15, con parecchi "resti" (ad esempio, per i Gialli 7460/2043 dà 3 seggi, con resto di 1331 voti). I rimanenti 3 seggi sono quindi assegnati a quei partiti che hanno riportato i 3 resti più alti: in questo caso Neri, Blu, Gialli (in grassetto).

Nel sistema a divisore D'Hondt (a sinistra), si dividono invece i voti di ogni partito per 1,2,3,4,5.. (al più, se necessario, fino a 15) e si prendono i 15 risultati più alti (in grassetto in figura), dando a ogni partito tanti seggi quanti valori ha fra i primi 15 (ad esempio, i Gialli prendono 4 seggi perché hanno 4 valori fra i primi 15; i Blu soltanto 1).

SEGGI: 15		Metodo a divisore D'Hondt					Quoziente naturale			
	Divisori:	1	2	3	4	5	segg	segg ai		
partito	voti	V/D1	V/D2	V/D3	V/D4	V/D5	segg	diretti	resti	segg ai
Gialli	7.460	7460	3730	2487	1865	1492	4	3	1331	1
Neri	2.040	2040	1020	680	510	408	1	0	2040	1
Blu	3.482	3482	1741	1161	871	696	1	1	1439	1
Rossi	8.748	8748	4374	2916	2187	1750	4	4	576	0
Verdi	8.922	8922	4461	2974	2231	1784	5	4	750	0
	30.652						15	12	Q = 2043	15

Il file di testo [RisultatiElezioni.txt](#) contiene i risultati delle elezioni (dettagliato più oltre): la prima riga indica il numero di seggi complessivamente da assegnare, le successive riportano i voti ottenuti da ciascun partito.

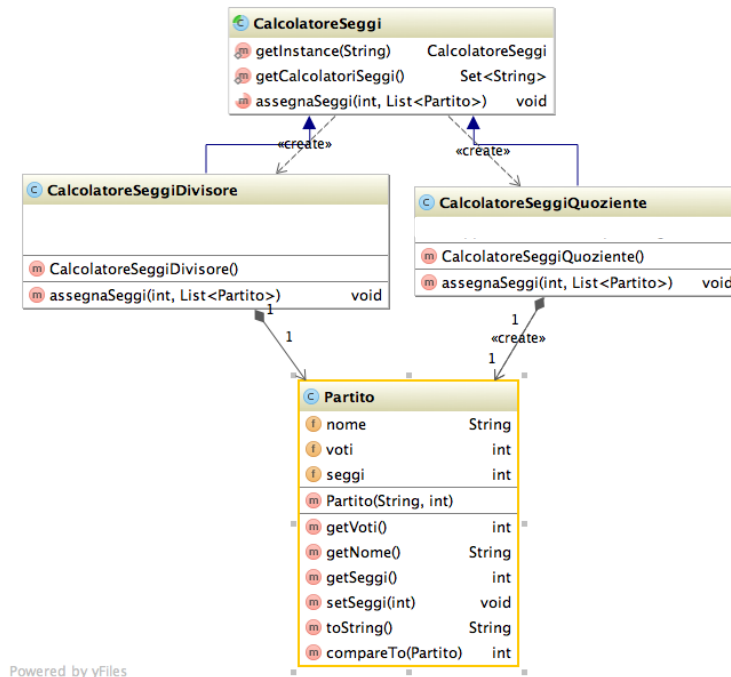
Parte 1

(punti: 20)

Dati (package *dentinia.model*)

(punti: 12)

Il modello dei dati deve essere organizzato secondo il diagramma UML più sotto riportato.



SEMANTICA:

- la classe **Partito** (fornita nello start kit) rappresenta un partito coi suoi voti (e i seggi assegnati);
- la classe astratta **CalcolatoreSeggi** (fornita) rappresenta il generico calcolatore seggi indipendente dallo specifico algoritmo di assegnazione; il metodo astratto **assegnaSeggi** prende in ingresso il numero di seggi da assegnare e una lista di **Partiti**, e assegna i seggi richiesti secondo il proprio algoritmo. La classe contiene inoltre il metodo statico **getInstance** che istanzia e restituisce una specifica istanza di **CalcolatoreSeggi** in base al nome dell'algoritmo passato come argomento: i nomi leciti sono recuperabili tramite il metodo statico **getCalcolatoriSeggi**.
- la classe **CalcolatoreSeggiQuoziente** (da realizzare) concretizza **CalcolatoreSeggi** nel caso dell'algoritmo di assegnazione basato sul quoziente naturale; il nome da usare per richiedere una istanza di **CalcolatoreSeggiQuoziente** è "Metodo del quoziente".
- la classe **CalcolatoreSeggiDivisore** (da realizzare) concretizza **CalcolatoreSeggi** nel caso dell'algoritmo di assegnazione basato sul divisore D'Hondt; il nome da usare per richiedere una istanza di **CalcolatoreSeggiDivisore** è "Metodo D'Hondt".

Suggerimento (non obbligatorio): in entrambi gli algoritmi, al fine di assegnare i seggi correttamente, nel caso di sistema a quoziente occorre ordinare le coppie [resto, partito] in ordine decrescente, mentre nel caso di sistema a divisore occorre ordinare le coppie [quoziente, partito] in ordine decrescente. A tal proposito è fornita nello start kit la classe **ValorePartito** (si veda nel codice) che rappresenta una coppia [valore intero, partito] che, poiché implementa l'interfaccia **Comparable**, può essere opportunamente ordinata.

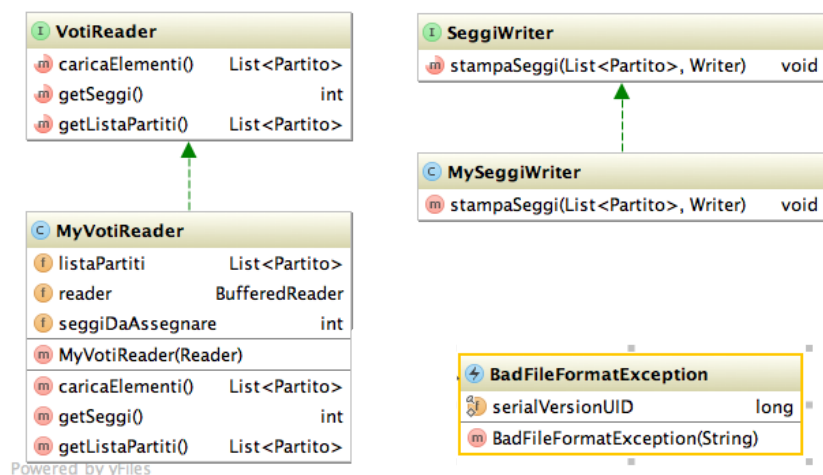
Lo Start Kit contiene anche i test (da includere nel progetto) per verificare il funzionamento di queste classi.

Persistenza (package *dentinia.persistence*)

(punti 8)

Come già anticipato, il file di testo **RisultatiElezioni.txt** contiene i risultati delle elezioni: la prima riga indica il numero di seggi complessivamente da assegnare, nella forma "**SEGGI:** " seguita da un intero, mentre le successive riportano i voti ottenuti da ciascun partito. A tal fine, ognuna di queste righe contiene nell'ordine il nome del partito (che può contenere spazi), il carattere ":" e il numero (intero) di voti da esso ottenuti.

Il modello dei dati deve essere organizzato secondo il diagramma UML seguente:



SEMANTICA:

- l'interfaccia **VotiReader** (fornita nello start kit) dichiara i quattro metodi **getSeggi**, **getListaPartiti**, **caricaElementi**;
- la classe **MyVotiReader** (da realizzare) concretizza **VotiReader** prevedendo:
 - un costruttore a un argomento che, a partire dal nome del file, apra il file e inizializzi le necessarie strutture dati interne, *senza però effettuare alcuna lettura*;
 - un metodo **caricaElementi** senza argomenti che legga tutto il file di testo e memorizzi internamente il numero di seggi da assegnare e la lista dei partiti con i relativi dati (che verrà anche restituita), lanciando **BadFormatException** nel caso di errore nel formato del file oltre alla "naturale" **IOException**.
- l'interfaccia **SeggiWriter** (fornita nello start kit) dichiara il metodo **stampaSeggi** che, data una lista di **Partito** con seggi assegnati, stampa l'attribuzione seggi sul file di testo il cui nome è passato come secondo argomento;
- la classe **MySeggiWriter** (da realizzare) concretizza **SeggiWriter** lanciando **IOException** nel caso il file non possa essere aperto in scrittura.

Lo Start Kit contiene anche i test (da includere nel progetto) per verificare il funzionamento di questa classe.

Parte 2

(punti: 10)

L'interfaccia grafica deve mostrare una tabella coi voti e i seggi ottenuti dai vari partiti (simile, ma non necessariamente identica, all'esempio mostrato in figura): più esattamente, all'inizio (Fig. 1) deve mostrare solo i voti, effettuando poi l'assegnazione dei seggi in base all'algoritmo scelto dall'utente fra quelli disponibili (Fig. 2).

La classe **Program** (fornita nello start kit) contiene il *main* di partenza dell'intera applicazione.

Controller (package dentinia.ui)

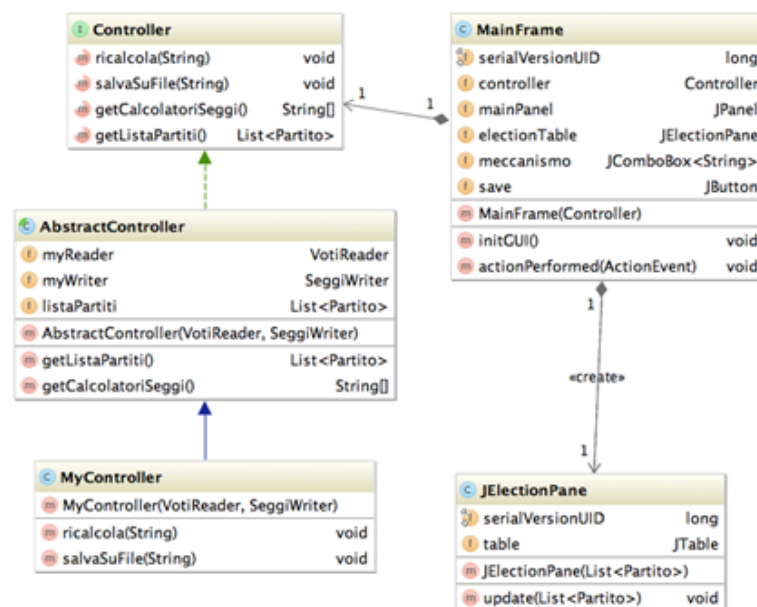
(punti 4)

L'interfaccia **Controller** (fornita) dichiara i seguenti metodi, ognuno riportato con la sua semantica:

- getCalcolatoriSeggi** restituisce un array di stringhe, contenente i nomi dei **CalcolatoreSeggi** disponibili;
- getListaPartiti** restituisce la lista delle istanze di **Partito** gestite dal controller;
- ricalcola** prende come argomento il nome dell'algoritmo di calcolo da impiegare (uno dei nomi restituiti dal metodo **getListaPartiti**) ed opera sulla lista dei partiti gestita dal controller, riassegnando i seggi sulla base dell'algoritmo selezionato;
- salvaSuFile** prende come argomento il nome di un file da creare con i risultati dell'assegnazione dei seggi ed effettua il salvataggio dei dati dei seggi assegnati nell'ultima elaborazione da parte del controller stesso.

La classe **AbstractController** (fornita) implementa parzialmente l'interfaccia **Controller** implementando il metodo **getCalcolatoriSeggi**. Essa presenta inoltre un costruttore a tre argomenti (un **VotiReader** tramite cui recuperare i dati di una elezione, un **SeggiWriter** tramite cui serializzare il risultato di una assegnazione di seggi e una **List<Partito>** che rappresenta l'insieme dei partiti gestiti dal controller) che legge dal **VotiReader** i partiti in lista, che inizialmente hanno tutti zero seggi assegnati.

La classe **MyController** (da realizzare) estende la classe **AbstractController** implementando i due metodi mancanti (**ricalcola** e **salvaSuFile**): viene costruita con gli stessi argomenti dell'AbstractController richiamando il costruttore della classe base. Da notare che in un ricalcolo la lista dei partiti e dei voti non cambia, mentre viene aggiornato il numero dei seggi assegnati a ciascun partito sulla base del metodo di calcolo selezionato. Il salvataggio dei risultati (ossia dei seggi assegnati a ciascun partito) deve essere effettuato scrivendoli sul **SeggiWriter** passato a **MyController** all'atto della sua costruzione.

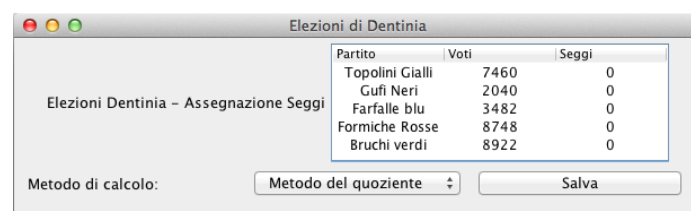


Lo Start Kit contiene anche i test (da includere nel progetto) per verificare il funzionamento di questa classe.

Interfaccia utente (package dentinia.ui)

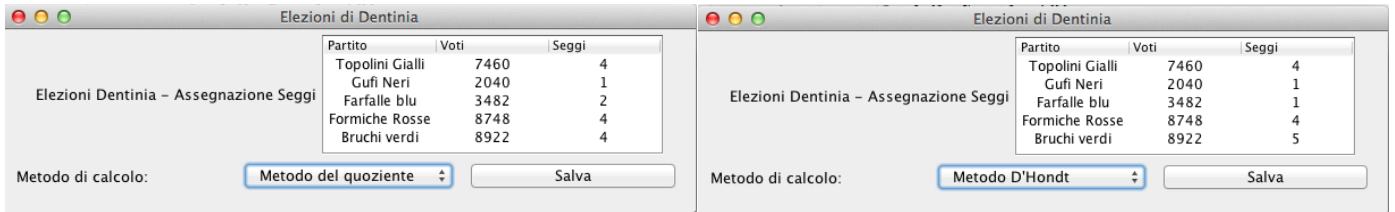
(punti 6)

La classe **JElectionPane** (fornita già pronta nello start kit) incapsula una tabella a tre colonne specializzata per i risultati elettorali: il suo costruttore accetta una lista di **Partito** (coi seggi ancora tutti nulli), usata per popolare inizialmente la tabella; successivamente, i dati visualizzati possono essere modificati invocando il metodo **update**, che accetta anch'esso una lista di **Partito**, ma coi seggi assegnati (cioè non più nulli).



La classe **MainFrame** (da realizzare) realizza la finestra principale:

- la casella a discesa mostra gli algoritmi disponibili: scegliendone uno, deve scattare l'attribuzione dei seggi conseguente, mostrata nella tabella sottostante;



- premendo il pulsante Salva, l'attribuzione dei seggi ai vari partiti dev'essere salvata su file di testo tramite il metodo `stampaSeggi` di `SeggiWriter` : a tal fine è richiesto di utilizzare la classe Swing `JFileChooser` sfruttando in particolare il metodo statico `showSaveDialog` (vedere esempio sotto)

ESEMPIO

```
JFileChooser chooser = new JFileChooser((String)null);
int res = chooser.showSaveDialog(parent); // parent can be this or null
if (res == JFileChooser.APPROVE_OPTION) {
    ...
} // else CANCEL_OPTION, if interested
```

