ESAME DI FONDAMENTI DI INFORMATICA T-2 dell' 11/07/2017
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 e CARTELLA : CognomeNome-matricola (es. RossiMario-0000123456)
NOME ZIP DA CONSEGNARE : CognomeNome-matricola.zip (es. RossiMario-0000123456.zip)

Nella Repubblica di Dentinia dev'essere scelto un nuovo sistema elettorale: per questo è stata richiesta un'applicazione che simuli l'attribuzione dei seggi, al variare del livello di *sbarramento* fra 0 (min) e 10% (max).

DESCRIZIONE DEL DOMINIO DEL PROBLEMA

Nei sistemi **puramente proporzionali**, i seggi sono attribuiti in proporzione ai voti ottenuti da ciascuno, secondo diverse formule. In particolare, nei ***sistemi a quoziente*** l'attribuzione dei seggi avviene determinando in primo luogo il numero di voti necessari per ottenere un seggio (*quoziente elettorale*), che nel caso più semplice (metodo Hare) è il risultato della divisione fra il totale dei voti espressi e il numero di seggi da assegnare (tipicamente non sarà un valore intero).
I seggi spettanti a ogni partito si ottengono poi dividendo per tale quoziente i voti da ciascuno ottenuti e arrotondando all'intero inferiore. Se, come è probabile, le divisioni danno luogo a resto, non tutti i seggi in palio vengono assegnati in prima fase: i seggi rimasti sono attribuiti ai partiti che hanno i resti (voti inutilizzati) più alti.
In pratica, con N seggi, detto Q il quoziente elettorale = Vtot / N:

1. i seggi Si da attribuire al partito Pi si ottengono dividendo i suoi voti per Q: Si = int [Vi / Q]
   I voti residui (resti) del partito Pi non utilizzati per assegnare tali Si seggi sono: Ri = Vi % Q
2. i seggi rimasti ancora da assegnare si assegnano a quei partiti a cui corrispondono i resti (voti rimasti) più alti.

Per attenuare l'effetto di parcellizzazione che i sistemi puramente proporzionali tendono a determinare, spesso si introduce uno **sbarramento,** che esclude dal riparto i partiti che non raggiungono una data percentuale di voti.

ESEMPIO
Si supponga di dover assegnare 600 seggi ai partiti A,B,C,D,E,F e che essi ottengano rispettivamente 9.100.000, 7.200.000, 4.880.000, 4.100.000, 2.320.000, 1.000.000 voti. I seggi da assegnare sono N=600 e i voti totali 28.600.000. Il quoziente elettorale risulta perciò Q = 47666.66. I seggi attribuiti in prima istanza sono quindi A=190, B=151, C=102, D=86, E=48, F=20, per un totale di 597: ne restano da assegnare 3. A tal fine si considerano i resti (voti inutilizzati), qui pari rispettivamente a 43334.60, 2434.34, 18000.68, 667.24, 32000.32, 46666.80: i tre resti più alti sono A,E,F che quindi ricevono un seggio in più. Pertanto il risultato finale è A=191, B=151, C=102, D=86, E=49, F= 21 (totale: 600).

In caso di **sbarramento** (ad esempio, al **5%**), i partiti che non raggiungono tale quota (nel caso del 5%, 1.430.000 voti), non partecipano al riparto: è il caso del partito F, che viene quindi escluso. Il calcolo deve quindi essere ripetuto con N=600 e voti totali = 27.600.000, cui corrisponde un quoziente di 46.000. I seggi attribuiti in prima istanza risultano A=197, B=156, C=106, D=89, E=50 (totale: 598), con 2 da assegnare ai resti, che qui sono 38000, 24000, 4000, 20000, 0: ergo i due seggi extra vanno ad A e B. Il risultato finale è perciò A=198, B=157, C=106, D=89, E=50 (totale: 600).

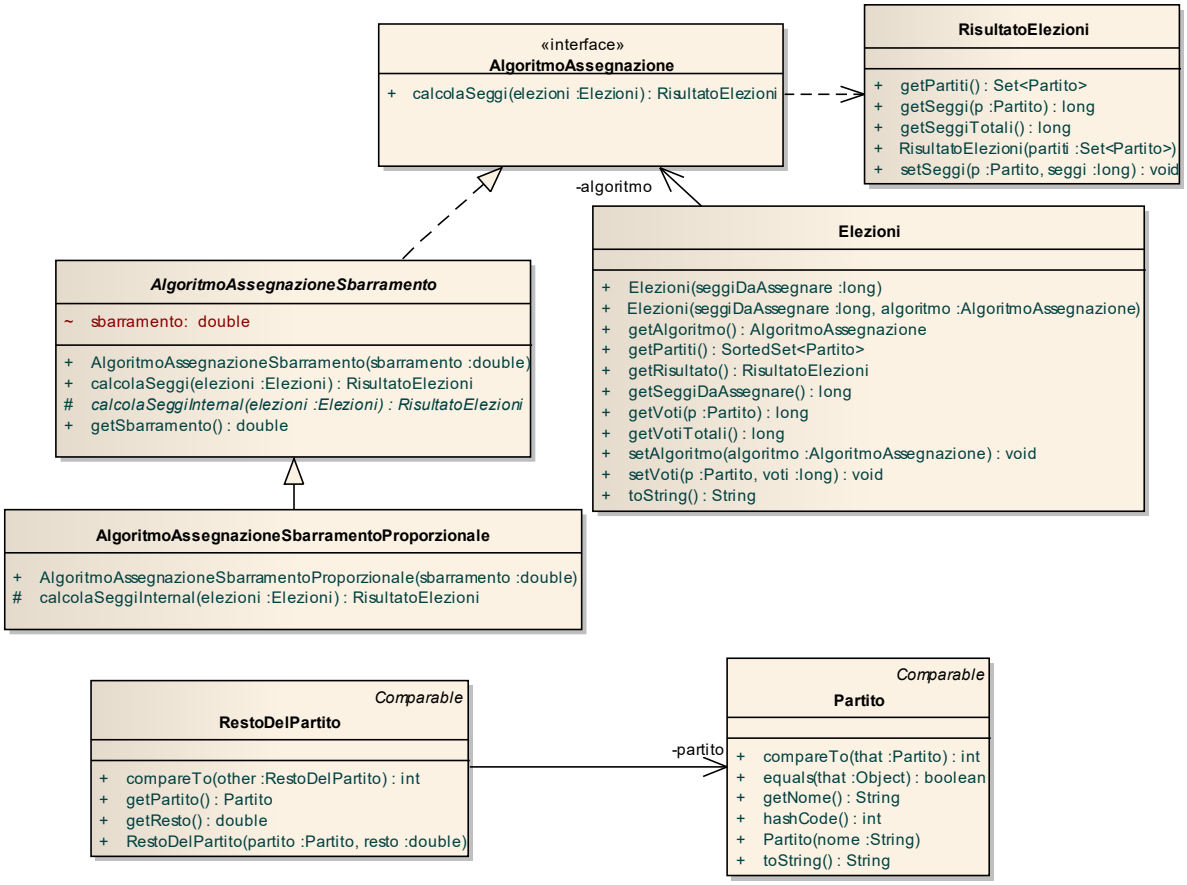
Il file di testo RisultatiElezioni.txt contiene i risultati delle elezioni (nel formato dettagliato più oltre): la prima riga indica il numero di seggi complessivamente da assegnare, le successive riportano i voti ottenuti da ciascun partito.

NELLE PAGINE CHE SEGUONO VENGONO DETTAGLIATE LE TRE PARTI:

- MODELLO DEI DATI
- PERSISTENZA
- INTERFACCIA GRAFICA (Javafx, swing)

Parte 1 (punti: 18)

Dati (package elections.model) *(punti: 11)*



SEMANTICA:

a) la classe ***Partito*** (fornita) rappresenta un partito, caratterizzato dal suo nome;

b) la classe ***Elezioni*** (fornita) mantiene i dati relativi ai voti ottenuti da ogni partito: il costruttore riceve il numero di seggi da assegnare ed eventualmente l'algoritmo da usare per il calcolo del risultato. Gli accessor *setVoti/getVoti* operano sui voti di un dato partito in modo controllato, mentre i metodi *getVotiTotali* e *getSeggiDaAssegnare* restituiscono le proprietà omonime. Il metodo *toString* restituisce una stringa multi-linea con partiti, voti e seggi perfettamente formattata.

c) la classe ***RisultatoElezioni*** (fornita) mantiene i dati relativi ai seggi ottenuti da ciascun partito: il costruttore riceve un set di ***Partito,*** mentre gli accessor *setSeggi/getSeggi*, *getSeggiTotali* permettono di recuperare le corrispondenti proprietà.

d) l'interfaccia ***AlgoritmoAssegnazione*** (fornita) dichiara il metodo astratto *calcolaSeggi,* che prende in ingresso una istanza di ***Elezioni***, restituendo la corrispondente istanza di ***RisultatoElezioni***. Se tale istanza è *null*, dev'essere lanciata ***IllegalArgumentException*** con idoneo messaggio d'errore.

e) La classe astratta ***AlgoritmoAssegnazioneSbarramento*** (fornita) implementa *calcolaSeggi* incapsulando il meccanismo dello sbarramento e delegando il calcolo al metodo astratto *calcolaSeggiInternal,* cui passa una istanza di ***Elezioni*** in cui ai partiti sotto soglia sono assegnati zero voti. Se tale istanza è *null*, dev'essere lanciata ***IllegalArgumentException*** con idoneo messaggio d'errore.

f) la classe ***AlgoritmoAssegnazioneSbarramentoProporzionale*** (**da realizzare**) estende ***AlgoritmoAssegnazioneSbarramento*** *e implementa* *calcolaSeggiInternal* usando il metodo proporzionale a quoziente, con lo sbarramento dell'entità specificata. A tal fine può utilizzare la classe ***RestoDelPartito*** (sotto specificata) per rappresentare i resti di ogni partito.

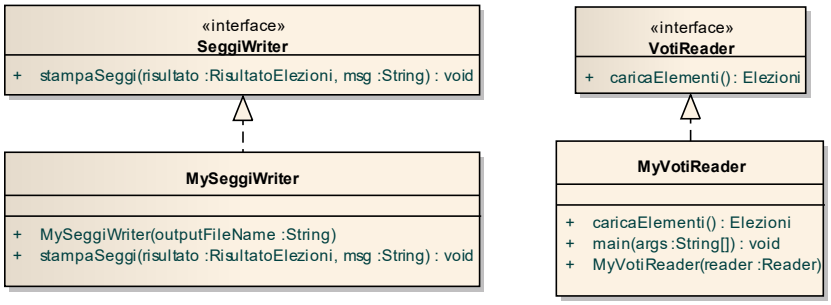
SUGGERIMENTO: calcolare i seggi spettanti a ciascun partito, indi memorizzare in una lista di ***RestoDelPartito*** i voti non utilizzati. Dopo averla ordinata in senso decrescente rispetto ai resti, scandirla assegnando ai partiti coi più alti resti gli eventuali seggi ancora non assegnati: nel farlo, occorre ricordare di non assegnare seggi a quei partiti che non hanno ottenuto voti o i cui voti sono stati cancellati per sbarramento.

g) la classe ***RestoDelPartito*** (fornita) rappresenta la coppia <***Partito***, resto> (ossia i voti residui, non utilizzati per l'assegnazione seggi in prima fase, di un dato ***Partito***), comparabile in senso decrescente sul valore double del resto: il costruttore riceve i due elementi della coppia, recuperabili tramite i due accessor.

Persistenza (elections.persistence) *(punti 7)*

Il file di testo RisultatiElezioni.txt contiene i risultati delle elezioni: la prima riga indica il numero di seggi totali da assegnare, nella forma "SEGGI IN PALIO" seguita da un intero, mentre le successive riportano i voti ottenuti da ciascun partito, formattati secondo l'uso italiano con il "." come separatore delle migliaia. Ogni riga contiene nell'ordine il nome del partito (che può contenere spazi), indi il numero di voti da esso ottenuti.

```
SEGGI IN PALIO 600
Partito del cioccolato fondente      9.100.000
Movimento vaniglia democratica       7.200.000
Insieme per il salume nostrano       4.880.000
Pizza è progresso                    4.100.000
Vegetarianesimo domani               2.320.000
Unità nel formaggio                  1.000.000
```



SEMANTICA:

a) L'interfaccia ***VotiReader*** (fornita) dichiara il metodo *caricaElementi* che restituisce una istanza di ***Voti*** popolata con partiti e voti, oltre che seggi totali da assegnare;

b) L'interfaccia ***SeggiWriter*** (fornita) dichiara il metodo *stampaSeggi* che stampa un ***RisultatoElezioni*** utilizzando la *toString*, preceduta dalla data e ora corrente e dal messaggio ricevuto come argomento;

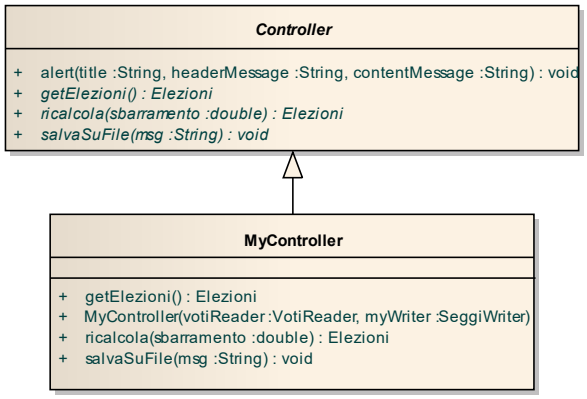
c) La classe ***MyVotiReader*** (**da realizzare**) implementa ***VotiReader***: il costruttore riceve in ingresso il Reader da cui leggere i dati. In caso di problemi di I/O viene propagata l'opportuna ***IOException,*** mentre eventuali problemi nel formato del file devono essere incapsulati in un'opportuna ***BadFileFormatException***.

d) la classe ***MySeggiWriter*** **(da realizzare)** implementa ***SeggiWriter***: il costruttore riceve in ingresso il nome del file su cui scrivere (il main dell'applicazione usa "Report.txt"). Il metodo *stampaSeggi* deve produrre una tabella analoga a quella mostrata in calce, in cui la prima riga contiene data e ora (nel formato indicato), la seconda una frase con l'indicazione del livello di sbarramento applicato, mentre dalla terza riga in poi sono riportati, uno per riga, i partiti coi rispettivi voti e seggi, separati da tabulazioni e formattati secondo l'uso italiano per le migliaia (vedi esempio). In caso di problemi di I/O viene propagata l'opportuna ***IOException***.

```
6-giu-2017 17.07.38
Proporzionale con sbarramento del 4.0%
Insieme per il salume nostrano      Voti:    4.880.000   Seggi:  106
Movimento vaniglia democratica      Voti:    7.200.000   Seggi:  157
Partito del cioccolato fondente     Voti:    9.100.000   Seggi:  198
Pizza è progresso                   Voti:    4.100.000   Seggi:   89
Unità nel formaggio                 Voti:    1.000.000   Seggi:    0
Vegetarianesimo domani              Voti:    2.320.000   Seggi:   50
TOTALE                              Voti:   28.600.000   Seggi:  600
```

Parte 2 (punti: 12)

Il Controller (**completamente fornito**) è organizzato secondo il diagramma UML in figura.



SEMANTICA:

a) La classe astratta ***Controller*** (fornita) dichiara l'interfaccia del controller (metodi *ricalcola, getVoti* e *salvaSuFile*) e implementa il metodo ausiliario *alert*, utile per mostrare avvisi all'utente.

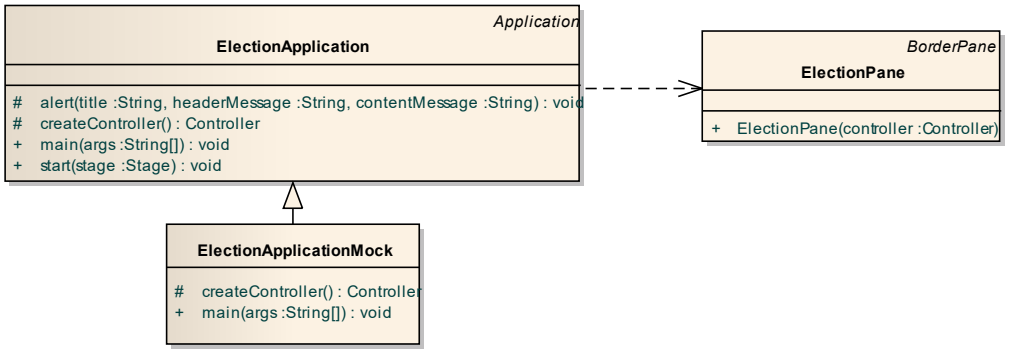
b) La classe ***MyController*** (**pure fornita**) completa l'implementazione realizzando:

- il costruttore a tre argomenti (***VotiReader***, ***SeggiWriter*** e ***AlgoritmoAssegnazione***) che effettua la lettura da file, memorizza gli argomenti in opportuni campi e predispone le istanze interne di ***Voti*** e ***RisultatoElezioni***; tale istanza di ***Voti*** è quella che viene restituita dal metodo *getVoti*;
- il metodo *ricalcola* riceve in ingresso lo sbarramento da applicare e delega all'***AlgoritmoAssegnazione*** il calcolo effettivo, memorizzando il risultato – un'istanza di ***RisultatoElezioni*** – nel proprio campo dati, prima di restituirlo a propria volta;
- il metodo *salvaSuFile* opera solo se il controller dispone internamente di un ***RisultatoElezioni*** valido: in tal caso, semplicemente delega all'opportuno ***SeggiWriter*** la stampa effettiva, passandogli anche il messaggio personalizzato ricevuto come argomento.

Interfaccia Utente JavaFX (elections.ui.javafx) **per studenti A.A. 2016/17** *(punti 12)*

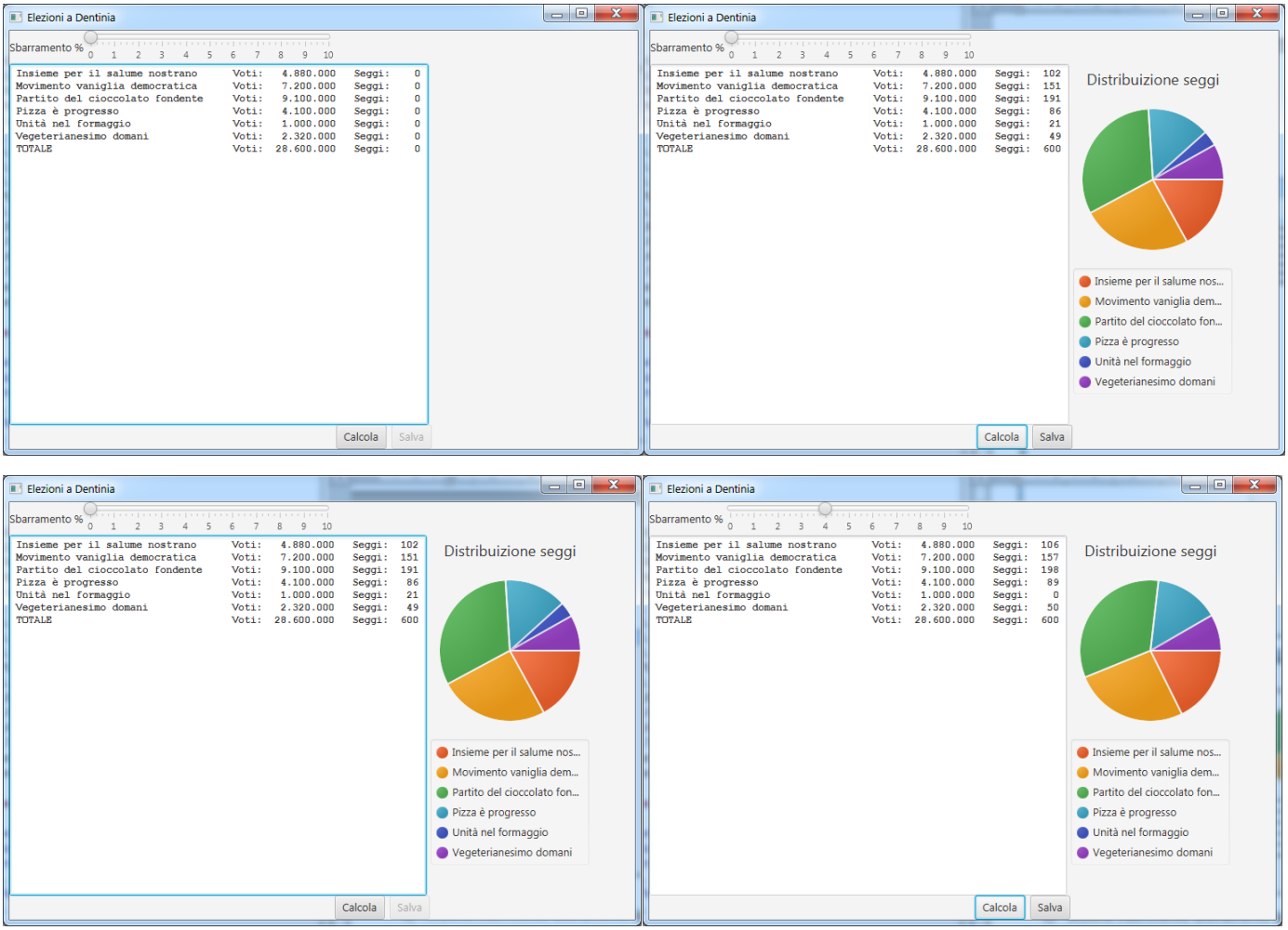
L'interfaccia utente deve essere simile (non necessariamente identica) all'esempio mostrato nelle figura seguenti.

L'architettura segue il modello sotto illustrato:



La classe ***ElectionApplication*** (fornita) costituisce l'applicazione JavaFX che si occupa di aprire il file, il controller e incorporare l'***ElectionPane*** (**da realizzare**). Per consentire di collaudare la GUI anche in assenza della parte di persistenza, è possibile avviare l'applicazione mediante la classe ***ElectionApplicationMock***.

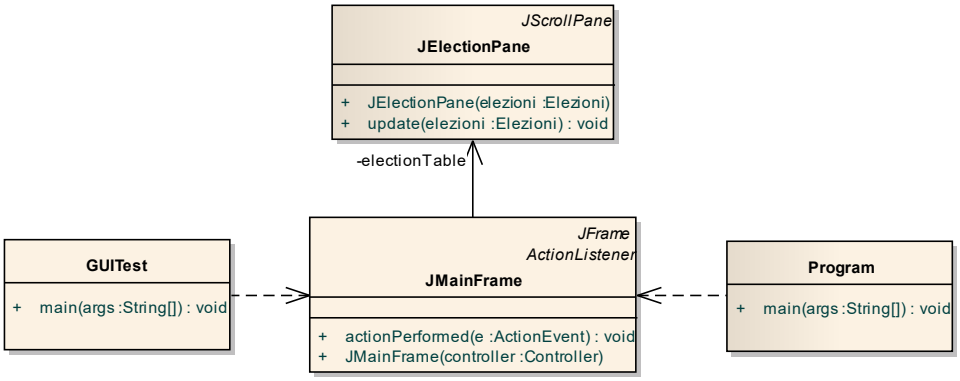
L'interfaccia utente deve essere simile (non necessariamente identica) all'esempio mostrato nella figura seguente.



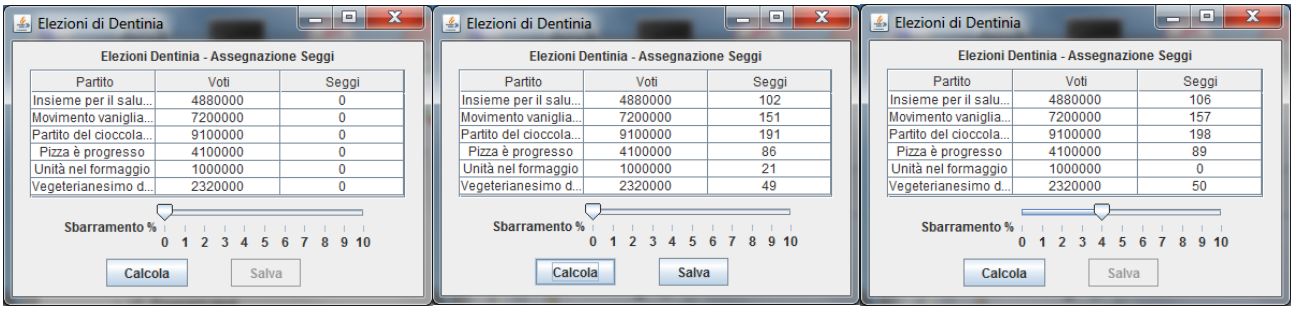
La classe ***ElectionPane*** (**da realizzare**) deve estendere ***BorderPane***.

1) In alto, uno ***Slider*** configurato da 0 a 10, passo 1, inizialmente a 0, permette di impostare lo sbarramento%
2) A sinistra, una ***TextArea*** mostra la situazione corrente, in forma testuale: all'inizio essa deve mostrare un risultato elezioni "iniziale", con tutti i seggi a zero. A tal fine conviene costruire al volo un ***RisultatoElezioni*** temporaneo, recuperando i voti dal controller tramite il metodo *getVoti*.
3) A destra, un ***PieChart*** (senza etichette) mostra anch'esso la situazione corrente, in forma grafica.
4) In basso, due pulsanti *Calcola* e *Salva,* di cui il secondo inizialmente disabilitato, controllano il funzionamento dell'applicazione. Più precisamente, *Calcola* scatena il calcolo dei seggi, tenendo conto dello sbarramento prescelto e aggiornando tabella e grafico a torta: dopo ogni ricalcolo, esso ri-abilita il pulsante *Salva* per consentire il salvataggio su file dei nuovi risultati. Il pulsante *Salva* effettua il salvataggio dei risultati sul file, auto-disabilitandosi subito dopo.

Interfaccia Utente Swing (elections.ui.swing) **per studenti A.A. 2015/16 e precedenti** *(punti 12)*



L'interfaccia utente deve essere simile (non necessariamente identica) all'esempio mostrato nella figura seguente:



La classe ***Program*** (fornita) contiene il *main* di partenza dell'intera applicazione, che si occupa di aprire il file e creare tutto il necessario. Per consentire di collaudare la GUI anche in assenza della parte di persistenza, è possibile avviare l'applicazione mediante la classe ***GUITest***.

La classe ***JMainFrame*** (***da realizzare***) deve organizzare l'interfaccia come sopra illustrato, ovvero:

1) in alto, una label che funge da titolo e una ***JElectionPane*** (fornita nello Start Kit e descritta sotto)
2) più sotto, preceduto da apposita label, un ***JSlider*** configurato da 0 a 10 (vedere suggerimento sotto) che permette all'utente di scegliere la soglia di sbarramento fra 0 (default) e 10%.
3) in basso, due pulsanti *Calcola* e *Salva,* di cui il secondo inizialmente disabilitato: il pulsante *Calcola* scatena il calcolo dei seggi, tenendo conto dello sbarramento prescelto e aggiornando la tabella: dopo ogni ricalcolo, esso ri-abilita il pulsante *Salva* per consentire il salvataggio su file dei nuovi risultati. Il pulsante *Salva* effettua il salvataggio dei risultati sul file, auto-disabilitandosi subito dopo.

Il componente ***JElectionPane*** (fornito) implementa una tabella a tre colonne, adatta a mostrare i voti e i seggi.

- Il costruttore riceve un'istanza di ***Elezioni*** che incapsula tutti i dati da mostrare inizialmente, ossia i partiti con i relativi voti (non ancora i seggi, che sono inizialmente tutti a zero);
- Il metodo ***update*** riceve una nuova istanza di ***Elezioni*** che incapsula i nuovi dati da mostrare, che sostituiscono i precedenti.

SUGGERIMENTO: Il componente ***JSlider*** rappresenta una barra di scorrimento. Può essere facilmente costruito col seguente frammento di codice:

```
slider = new JSlider(JSlider.HORIZONTAL, 0 , 10, 0);
slider.setMajorTickSpacing(1);
slider.setPaintTicks(true);
slider.setPaintLabels(true);
```