ESAME DI FONDAMENTI DI INFORMATICA T-2 del 10/02/2017
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: CognomeNome-matricola (es. RossiMario-0000123456)

Avendo recentemente deliberato l'adozione di un sistema elettorale maggioritario, il Parlamento dell'Elbonia ha richiesto un'applicazione per simulare i risultati elettorali in base alla quantità e dimensione dei collegi elettorali.

DESCRIZIONE DEL DOMINIO DEL PROBLEMA

Ai fini elettorali, il territorio di un Paese viene diviso in zone, chiamate *collegi,* ciascuno dei quali elegge N≥1 deputati. Nel caso in cui N=1 il sistema si dice *uninominale*; se invece N>1 si dice *plurinominale.* Se l'intero Paese costituisce un unico grande collegio, esso elegge N= N_{MAX} deputati, ossia tutti i deputati del Parlamento.
Nel sistema maggioritario, la lista che, in un dato collegio, prende più voti ottiene tutti gli N deputati in palio: i voti delle altre liste in quel collegio vanno persi. Tale effetto distorsivo è tanto maggiore quanto maggiore è la dimensione dei collegi: esso è minimo nel caso uninominale, mentre diventa massimo nel caso, estremo e quindi puramente astratto, in cui l'intero Paese sia un unico collegio, in cui chi vince ottiene tutti i deputati, lasciando tutti gli altri a becco asciutto.

L'applicazione richiesta ha l'obiettivo di simulare il risultato (seggi ottenuti da ciascun partito, posto che i voti espressi nelle diverse zone del Paese siano noti) al variare del numero di collegi fra 1 e $N_{MAX.}$ Tuttavia, poiché ogni collegio deve eleggere un numero intero di deputati, sussiste il vincolo che, detto C il numero di collegi (variabile fra N_{MAX} e 1), il rapporto N_{MAX}/C sia intero: pertanto, N e C possono assumere solo valori compatibili con tale vincolo.

ESEMPIO: siano i seggi da assegnare N_{MAX} = 180.
Il numero N di deputati eletti in ogni collegio varia fra 1 e 180: corrispondentemente, il numero C di collegi varia fra 180 e 1, essendo i due legati dal vincolo di essere interi e tali che N×C=180. Pertanto, i valori possibili per N (e dualmente per C) non sono tutti i naturali fra 1 e 180, ma soltanto 1, 2, 3, 4, 5, 6, 9, 10, 12, 15, 18, 20, 30, 36, 45, 90, 180.

Per poter simulare collegi di diversa ampiezza, sono disponibili i voti espressi dai cittadini in N_{MAX} mini-collegi: nel caso uninominale essi costituiscono direttamente i collegi da utilizzare nella simulazione, mentre nel caso plurinominale l'applicazione dovrà *aggregarne due o più* per creare collegi via via più grandi.

ESEMPIO: siano i seggi da assegnare N_{MAX} = 180.
Sono forniti i voti ottenuti dai vari partiti in 180 mini-collegi da 1 deputato ciascuno. Nel caso N=1 (C=180), i collegi coincidono con i mini-collegi; nel caso N=2 l'applicazione dovrà invece considerare 90 collegi, ciascuno fatto di due mini-collegi adiacenti. Analogamente, per N=3 l'applicazione dovrà considerare 60 collegi, ottenuti aggregando tre mini-collegi adiacenti; e così via fino al caso limite di N=N_{MAX}, in cui tutti i 180 mini-collegi sono aggregati in un unico maxi-collegio nazionale.

Il file di testo Voti.csv contiene i voti riportati dai quattro partiti esistenti in Elbonia nei 180 mini-collegi.

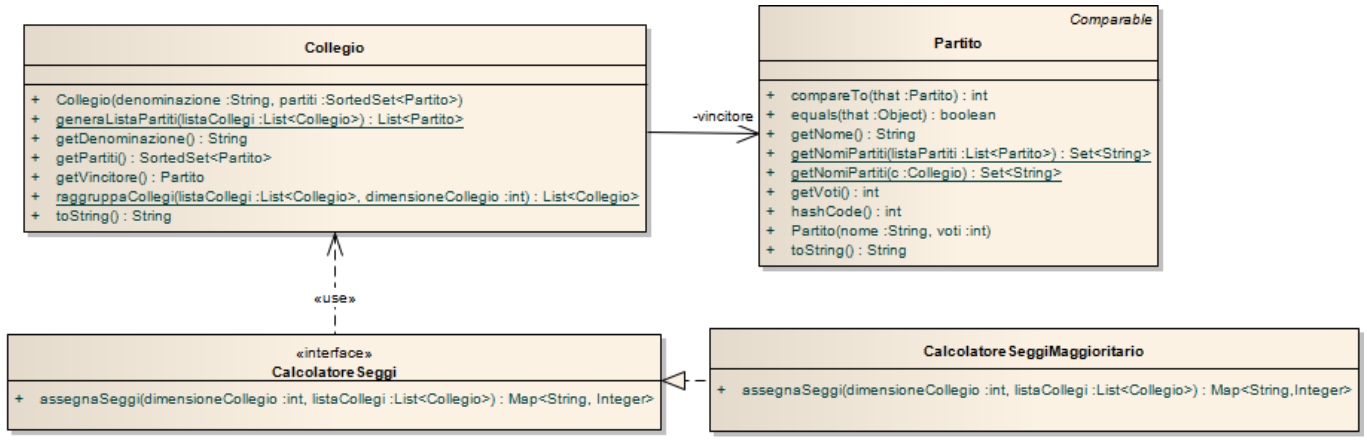
Parte 1 (punti: 16)

Dati (package elbonia.model) *(punti: 10)*

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

SEMANTICA:

a) la classe ***Partito*** (fornita nello start kit) rappresenta un partito coi suoi voti (coppia nome/voti) nel contesto di un certo collegio ed è ***Comparable*** rispetto al numero di voti (più voti = più "grande"). Offre anche due metodi statici di utilità, entrambi denominati ***getNomiPartiti***, che restituiscono un ***Set<String>*** contenenti i nomi dei partiti – rispettivamente, presenti in un singolo collegio o in una intera lista di ***Partito.***



b) la classe ***Collegio*** (fornita) rappresenta un collegio elettorale con i corrispondenti partiti; i partiti sono contenuti in un set ordinato (***Partito*** è ***Comparable***) in modo che i partiti più votati vengano prima degli altri. La classe contiene anche due metodi statici di utilità, ***generaListaPartiti*** e ***raggruppaCollegi.*** Il primo, data una lista di ***Collegio***, restituisce una lista di ***Partito***, in cui ciascun partito è associato al totale dei voti ottenuti in quei collegi; il secondo invece, data una lista di mini-***Collegio*** e la dimensione K del collegio, restituisce una nuova lista di ***Collegio*** raggruppati a K a K.

c) l'interfaccia ***CalcolatoreSeggi*** (fornita) rappresenta il generico calcolatore seggi indipendente dallo specifico algoritmo di assegnazione; il metodo astratto ***assegnaSeggi*** prende in ingresso la dimensione del collegio e una lista di ***Collegio*** che rappresenta i mini-collegi letti dal file: restituisce una mappa (nome partito, seggi) di C collegi, ognuno con N deputati, adeguatamente popolata.

d) la classe ***CalcolatoreSeggiMaggioritario*** **(da realizzare)** concretizza ***CalcolatoreSeggi*** nel caso del metodo maggioritario; perciò, la sua implementazione di ***assegnaSeggi*** effettua l'assegnazione secondo il criterio spiegato all'inizio. [Ad esempio, se la dimensione del collegio è N=3 e la lista contiene i dati di 180 mini-collegi, la mappa dovrà essere costruita a partire da una lista di C=60 collegi, costruiti ciascuno aggregando 3 mini-collegi consecutivi e sommandone i relativi voti. In altre parole, i voti dei quattro partiti P0,P1,P2,P3 riportati nei tre mini-collegi n.0,1,2 dovranno essere sommati: $V_{P0} = V_{P0,0} + V_{P0,1} + V_{P0,2}$; $V_{P1} = V_{P1,0} + V_{P1,1} + V_{P1,2}$; etc].

SUGGERIMENTO: è conveniente costruirsi dapprima una *lista collegi raggruppati*, aggregando a N a N gli elementi della lista dei mini-collegi data con l'apposito metodo di ***Collegio***; si potrà quindi lavorare con questa, sapendo che ogni collegio raggruppato elegge N deputati, che il sistema maggioritario assegna tutti all'unico partito che prende più voti.

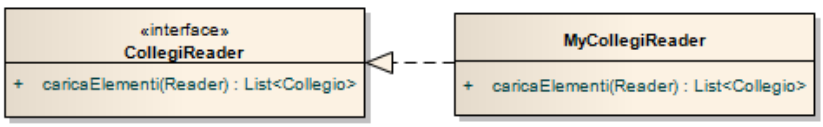
Per costruire la mappa, conviene procurarsi prima la lista dei nomi dei partiti (anche per questo si suggerisce l'uso dei metodi di utilità forniti) e con questa inizializzare la mappa, inserendo inizialmente zero seggi per tutti i partiti; si potrà poi popolare la mappa iterativamente, aggiungendo via via N al numero dei seggi al partito vincitore in ogni collegio.

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

Persistenza (package elbonia.persistence) *(punti 6)*

Come sopra anticipato, il file di testo Voti.csv contiene i voti riportati dai quattro partiti nei 180 mini-collegi, un collegio per riga: la prima riga riporta le intestazioni, mentre le righe successive riportano ciascuna il numero del collegio e i voti ottenuti dai vari partiti. Il separatore è sempre e comunque il carattere `';'` (punto e virgola).

L'architettura software è illustrata nel diagramma UML che segue:



SEMANTICA:

a) l'interfaccia ***CollegiReader*** (fornita) dichiara il metodo ***caricaElementi***;

b) la classe ***MyCollegiReader*** **(da realizzare)** concretizza ***CollegiReader*** prevedendo un metodo ***caricaElementi*** che legga tutto il contenuto del ***Reader*** ricevuto come argomento e restituisca la lista dei mini-***Collegi*** opportunamente popolata; i mini collegi dovranno essere chiamati "collegio0", "collegio1", etc. Il numero di partiti non deve essere predeterminato (ad es. nel file ne sono riportati quattro, nei test ne sono riportati tre). In caso di errore nel formato del file (valori non numerici, mancanza di uno dei dati previsti) dev'essere lanciata una ***BadFileFormatException***.

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

Parte 2 (punti: 14)

L'interfaccia grafica contiene uno *slider* (istanza di ***ElectionSlider***, fornita nello start kit e spiegata più sotto) per variare la dimensione del collegio fra 1 e N_{MAX}. Subito sotto, una tabella (istanza di ***ElectionTablePanel***, anch'essa fornita nello start kit e spiegata più sotto) mostra voti e i seggi ottenuti dai vari partiti.

Inizialmente, la tabella ha tutti i seggi azzerati (Fig. 1): non appena l'utente muove lo slider, l'applicazione aggiorna la tabella con i seggi corrispondenti e fa comparire il corrispondente grafico a torta (Figg. 2, 3, 4 e 5): quest'ultimo è istanza della classe ***ElectionPieChart***, anch'essa fornita nello start kit e spiegata più sotto.

In alto, due campi di testo non modificabili mostrano, per comodità, sia il valore attuale dello *slider* (il numero di collegi, N), sia la corrispondente dimensione del collegio ($C = N_{MAX}/N$).

La classe ***Program*** (fornita nello start kit) contiene il *main* di partenza dell'intera applicazione; la classe ausiliaria ***GUITest*** (pure fornita) consente di far partire l'applicazione anche in caso di malfunzionamento del reader o di reader assente, fornendo al controller alcuni dati fissi di prova.

Controller (package elbonia.ui.controller) *(punti 5)*

L'interfaccia ***UserInteractor*** (fornita), implementata dalla classe ***SwingUserInteractor*** (fornita), consente di mostrare un messaggio all'utente (metodo ***showMessage***) o di terminare l'applicazione (metodo ***shutDownApplication***).

La classe astratta ***Controller*** (fornita) presenta un costruttore con un argomento di tipo ***UserInteractor*** e definisce/dichiara i seguenti metodi, ognuno riportato con la sua semantica:

- ***loadData*** (definito) prende in ingresso un ***Reader*** e un ***CollegiReader*** ed effettua il caricamento dei dati;
- ***getListaCollegi*** (definito) restituisce la lista dei ***Collegio*** caricati da ***loadData*** **e** gestite dal controller;
- ***getUserInteractor*** (definito) restituisce l'oggetto che implementa ***UserInteractor***;
- ***calcola*** (unico metodo astratto) prende come argomento *la dimensione del collegio* ed opera sulla lista dei collegi gestita dal controller, assegnando i seggi sulla base dell'algoritmo selezionato; il risultato è una mappa che associa a ogni nome di partito (una stringa) il numero dei rispettivi seggi.

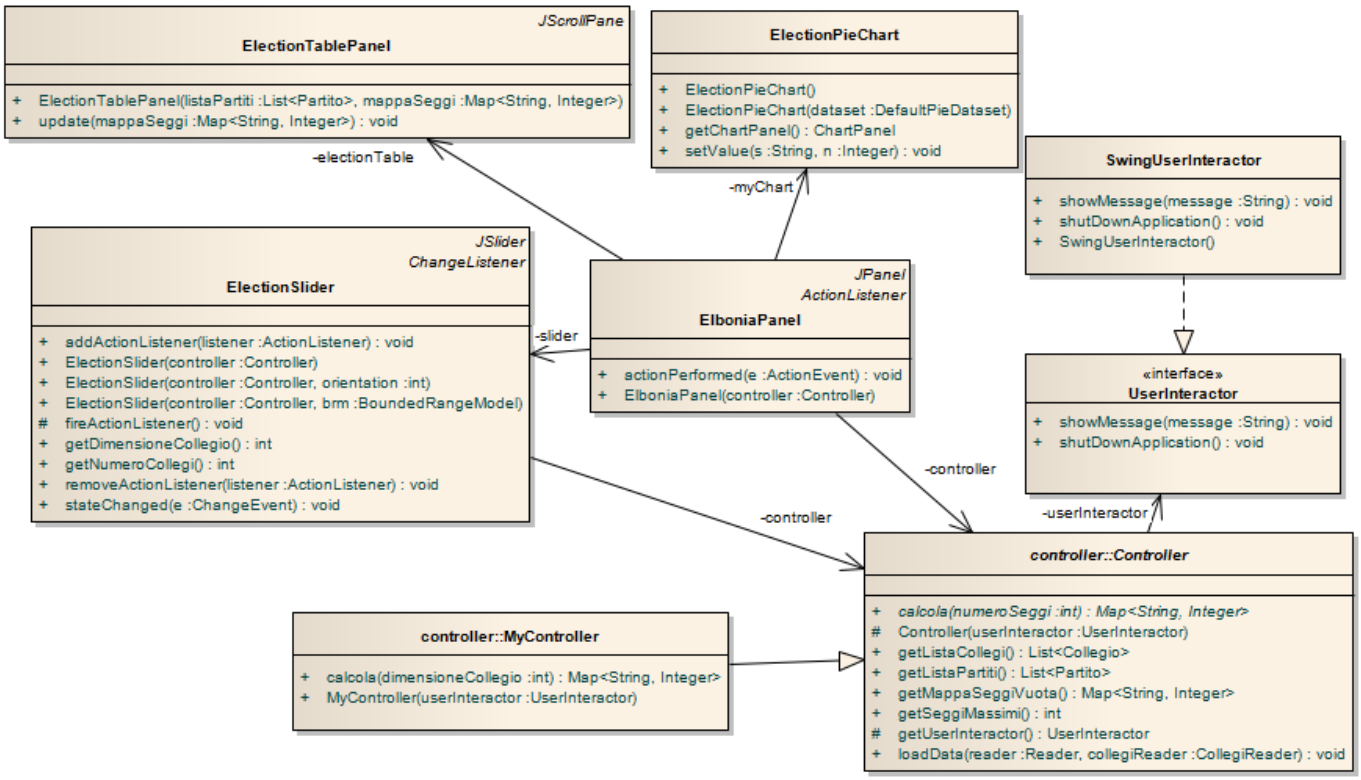
La classe ***MyController*** **(da realizzare)** estende la classe ***Controller***: viene costruita con gli stessi argomenti di ***Controller*** richiamando il costruttore della classe base e definisce il metodo ***calcola*** in modo che:

- verifichi che la dimensione del collegio ricevuta come argomento sia compresa fra 1 e il massimo numero di seggi assegnabili: se così non è, mostra all'utente un messaggio adeguato tramite l'istanza di ***UserInteractor*** posseduta e restituendo *null*;
- istanzi un ***CalcolatoreSeggiMaggioritario*** e lo usi per effettuare l'assegnazione dei seggi nei collegi (ottenibili tramite il metodo ereditato ***getListaCollegi***) tenendo conto della dimensione collegio desiderata. Restituisce la mappa ottenuta come risultato dal calcolatore.

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

Interfaccia utente (package elbonia.ui) *(punti 9)*

La classe ***ElectionSlider*** (fornita) estende ***JSlider*** e incapsula la funzionalità di selezione del numero dei mini-collegi che compongono un collegio consentendo di selezionare solo valori ammissibili; il costruttore di tale *slider* prende in ingresso un ***Controller*** e si imposta automaticamente; i metodi ***getDimensioneCollegio***, ***getNumeroCollegi*** restituiscono, rispettivamente, la dimensione del collegio e il numero totale di collegi. Ogni volta che lo slider viene mosse, si genera un evento di azione (ActionEvent), gestibile come ogni altro evento del genere.



La classe ***ElectionTablePanel*** (fornita) estende ***JScrollPane*** e incapsula una tabella a tre colonne specializzata per i risultati elettorali: il suo costruttore accetta una lista di ***Partito*** (coi seggi ancora tutti zero) e una mappa (nomi partiti, seggi), usate per popolare inizialmente la tabella; successivamente, i dati visualizzati possono essere modificati invocando il metodo ***update***, a cui va fornita la mappa (nomi partiti, seggi) aggiornata.

La classe ***ElectionPieChart*** (fornita) rappresenta un grafico a torta: non è esso stesso un pannello, ma i costruttori provvedono a crearne uno opportuno, di tipo ***ChartPanel,*** recuperabile tramite il metodo ***getChartPanel***: in tal modo sarà possibile aggiungere il grafico al proprio pannello aggiungendo il ***ChartPanel.*** il costruttore di default crea un grafico standard, inizialmente vuoto: per popolare i dati (ossia, aggiungere spicchi alla torta) si usa il metodo ***setValue***, i cui argomenti sono una stringa e un intero, che rappresentano rispettivamente la descrizione e il valore di uno spicchio del grafico a torta: ogni volta che si invoca ***setValue***, il grafico acquisisce un nuovo spicchio e *ricalcola automaticamente le dimensioni dei precedenti*. Pertanto, alla prima invocazione il grafico sarà una torta di un unico colore (come in Fig. 4 sotto): poi, via via che si aggiungono spicchi, il grafico verrà aggiornato. Per modificare una voce già presente basta reinvocare ***setValue*** con la stringa già usata in precedenza per quello spicchio.

La classe ***ElboniaPanel*** **(da realizzare)** rappresenta il pannello centrale dell'applicazione (deriva da ***JPanel***); gestisce gli eventi relativi allo *slider*, recuperando la dimensione del collegio e il numero di collegi dallo *slider* stesso, poi inserendo tali dati nei corrispondenti *textfield*, indi richiedendo al controller il calcolo della mappa seggi (passando la dimensione del collegio): con tale mappa, aggiorna infine sia l'***ElectionTablePanel***, sia l'***ElectionPieChart***.

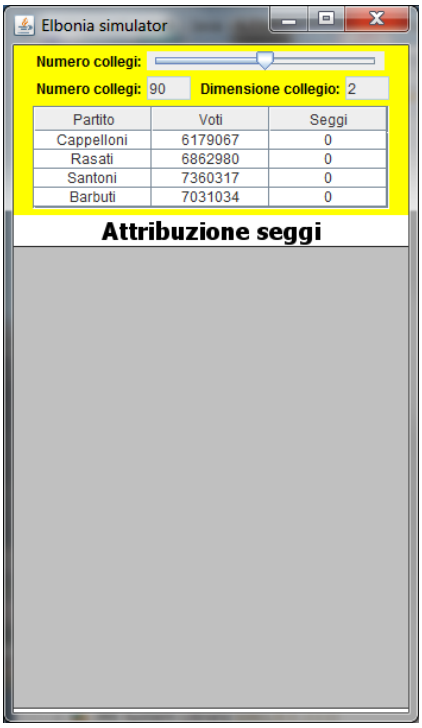


Figura 1

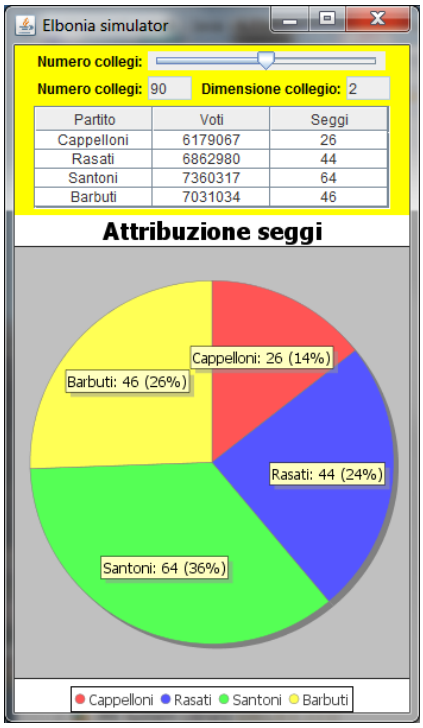


Figura 2

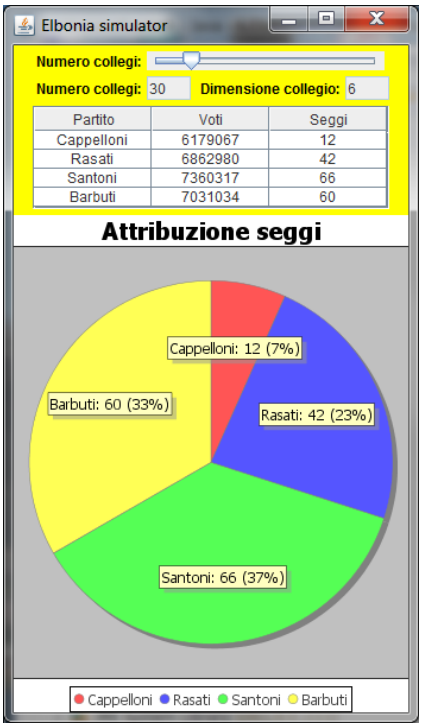


Figura 3

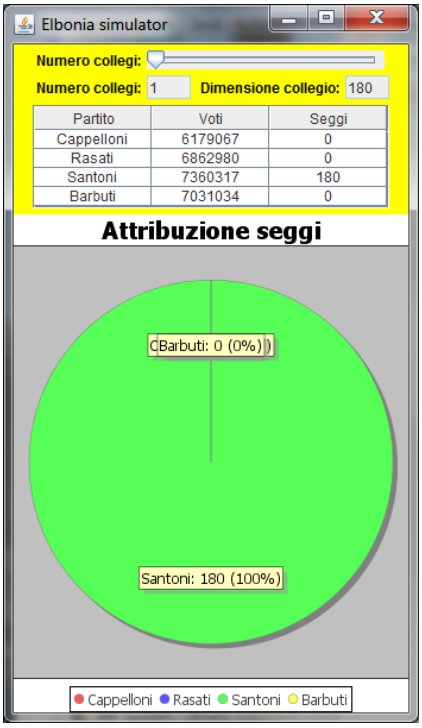


Figura 4

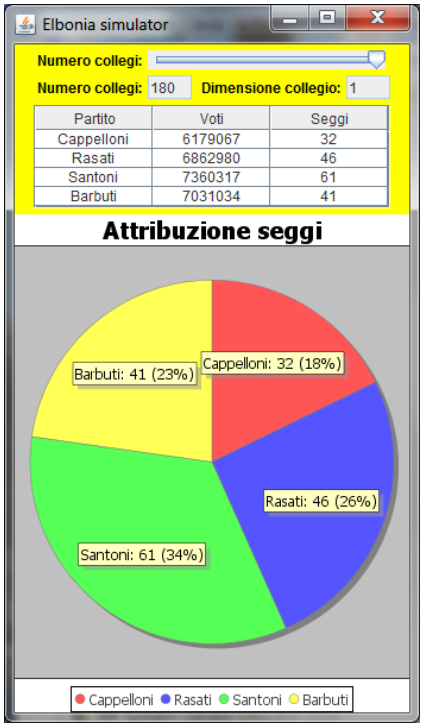


Figura 5