Project 2 - Voting Application (Team 4)

Generated by Doxygen 1.8.13

Contents

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

Class Documentation

3.1 Ballot Class Reference

The Ballot class provides the necessary functionality to represent a voter's ballot in an election. It is able to represent ballots in both plurality and single-tranferrable vote elections. A ballot had an Id, a list of the Candidates in the order they were ranked on the ballot, and a variable indicating the current highest ranked Candidate choice remaining on the ballot.

`#include <ballot.h>`

Public Member Functions

- Ballot (int id, std::vector< Candidate * > choices)
- int getId ()
- void setId (int id)
- std::vector< Candidate * > getCandidates ()
- int getCurrentChoice ()
- void nextChoice ()

3.1.1 Detailed Description

The Ballot class provides the necessary functionality to represent a voter's ballot in an election. It is able to represent ballots in both plurality and single-tranferrable vote elections. A ballot had an Id, a list of the Candidates in the order they were ranked on the ballot, and a variable indicating the current highest ranked Candidate choice remaining on the ballot.

3.1.2 Constructor & Destructor Documentation

3.1.2.1 Ballot()

```
Ballot::Ballot (
            int id,
            std::vector< Candidate *> choices )
```

The constructor for the Ballot class which sets the initial id of the ballot and the Candidates that were ranked on the Ballot. It defaults the current choice index to 0.

Parameters

id	an int that the Id of the Ballot object will be set to
choices	a vector of pointers to Candidates ranked on the Ballot object (in order)

Author

Brendan Ritchie (ritch167)

3.1.3 Member Function Documentation

3.1.3.1 getCandidates()

```
std::vector<Candidate*> Ballot::getCandidates ( )
```

This method returns the vector of pointers to Candidates ranked on the Ballot object (in order)

Author

Brendan Ritchie (ritch167)

3.1.3.2 getCurrentChoice()

```
int Ballot::getCurrentChoice ( )
```

This method returns the index of the current highest ranked Candidate choice in the vector of Candidates ranked on the Ballot object

Author

Brendan Ritchie (ritch167)

3.1.3.3 getId()

```
int Ballot::getId ( )
```

This method returns the Id of the Ballot object

Author

Brendan Ritchie (ritch167)

3.1.3.4 nextChoice()

```
void Ballot::nextChoice ( )
```

This method sets the currentChoice to the next Candidate in order

Author

Yiwen Xu (xu000515)

3.1.3.5 setId()

```
void Ballot::setId (
            int id )
```

This method sets the Id of the Ballot object

Parameters

id	an int that the Id of the Ballot object will be set to

Author

Brendan Ritchie (ritch167)

The documentation for this class was generated from the following file:

- /home/ritch167/csci5801/repo-Team4/Project2/src/ballot.h

3.2 Candidate Class Reference

The Candidate class provides the necessary functionality to represent a candidate in an election. It is able to represent candidates in both plurality and single-tranferrable vote elections. A Candidate had an Id, a name, a forward-linked list of Ballot objects assigned to them, a int representing the count of how many Ballots are assigned to them, double representing the time of when they received their first Ballot in an election (needed for STV elections), and boolean indicating whether the Candidate has been declared a winner or a loser yet.

`#include <candidate.h>`

Public Member Functions

- Candidate (int id, std::string name)
- int getId ()
- std::string getName ()
- std::list< Ballot * > getBallotsFor ()
- void addBallot (Ballot *new_ballot)
- Ballot * removeBallot ()
- int getBallotsForSize ()
- double getWhenGotFirstBallot ()
- void setWhenGotFirstBallot (double time)
- bool getAssignedStatus ()
- void setAssignedStatus (bool status)

3.2.1 Detailed Description

The Candidate class provides the necessary functionality to represent a candidate in an election. It is able to represent candidates in both plurality and single-tranferrable vote elections. A Candidate had an Id, a name, a forward-linked list of Ballot objects assigned to them, a int representing the count of how many Ballots are assigned to them, double representing the time of when they received their first Ballot in an election (needed for STV elections), and boolean indicating whether the Candidate has been declared a winner or a loser yet.

3.2.2 Constructor & Destructor Documentation

3.2.2.1 Candidate()

```
Candidate::Candidate (
            int id,
            std::string name )
```

The constructor for the Candidate class sets the initial id of the ballot and the Candidate object's name. It defaults the count of its Ballot objects for to 0, when they got their first ballot to -1.0, and their winner/loser assigned status to false.

Parameters

id	an int that the Id of the Candidate object will be set to
name	a string that the name of the Candidate will be set to

Author

Brendan Ritchie (ritch167)

3.2.3 Member Function Documentation

3.2.3.1 addBallot()

```
void Candidate::addBallot (
            Ballot * new_ballot )
```

This method is responsible for adding new Ballot to the end of the list ballotsFor of the Candidate.

Parameters

new_ballot	a pointer to the new Ballot which will be added to the list

Author

Yiwen Xu (xu000515)

3.2.3.2 getAssignedStatus()

```
bool Candidate::getAssignedStatus ( )
```

This method returns the boolean indicating whether the Candidate object has been declared a winner or a loser yet.

Author

Brendan Ritchie (ritch167)

3.2.3.3 getBallotsFor()

```
std::list<Ballot *> Candidate::getBallotsFor ( )
```

This method returns the list of pointers to Ballot objects assigned to the Candidate object.

Author

Brendan Ritchie (ritch167)

3.2.3.4 getBallotsForSize()

```
int Candidate::getBallotsForSize ( )
```

This method returns the the count of how many Ballots objects are assigned to the Candidate objects.

Author

Brendan Ritchie (ritch167)

3.2.3.5 getId()

```
int Candidate::getId ( )
```

This method returns the Id of the Candidate object.

Author

Brendan Ritchie (ritch167)

3.2.3.6 getName()

```
std::string Candidate::getName ( )
```

This method returns the name of the Candidate object.

Author

Brendan Ritchie (ritch167)

3.2.3.7 getWhenGotFirstBallot()

```
double Candidate::getWhenGotFirstBallot ( )
```

This method returns the double representing the time of when the Candidate object received their first Ballot in an election (needed for STV elections).

Author

Brendan Ritchie (ritch167)

3.2.3.8 removeBallot()

```
Ballot* Candidate::removeBallot ( )
```

This method removes a Ballot from the front of the list ballotsFor for the Candidate and returns that Ballot.

Author

Yiwen Xu (xu000515)

3.2.3.9 setAssignedStatus()

```
void Candidate::setAssignedStatus (
            bool status )
```

This method sets the status indicating whether the Candidate object has been declared a winner or a loser yet.

Parameters

status	a boolean indicating whether the Candidate has been assigned or not

Author

Brendan Ritchie (ritch167)

3.2.3.10 setWhenGotFirstBallot()

```
void Candidate::setWhenGotFirstBallot (
            double time )
```

This method sets the time when the Candidate object received their first Ballot in an election (needed for STV elections).

Parameters

time	an double representing the number of seconds since the election algorithm started processing

Author

Brendan Ritchie (ritch167)

The documentation for this class was generated from the following file:

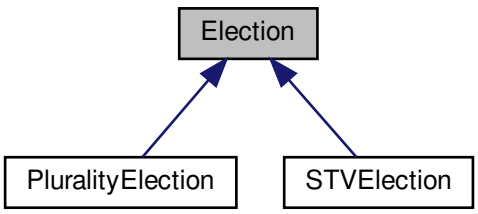
- /home/ritch167/csci5801/repo-Team4/Project2/src/candidate.h

3.3 Election Class Reference

The Election class is an abstract class that provides the necessary functionality to represent an election. While non-abstract child classes are required to implement their own Ballot tallying algorithms and results, the Election class provides a template of necessary methods and properties that all types of elections will utilize.

```
#include <election.h>
```

Inheritance diagram for Election:



Public Member Functions

- Election (std::string type, int seats, std::vector< Candidate *> cands, std::vector< Ballot *> bals)
- std::string getType ()
- int getNumSeats ()
- std::vector< Candidate * > getCandidates ()
- std::vector< Ballot * > getBallots ()
- std::vector< Candidate * > getWinners ()
- std::vector< Candidate * > getLosers ()
- std::string getAuditFilePath ()
- virtual void runAlgorithm ()=0
- virtual std::string getResults ()=0

Protected Member Functions

- void setAuditFilePath (std::string name)
- void writeToAuditFile ()
- void addWinner (Candidate *win)
- void addLoser (Candidate *lose)

Protected Attributes

- std::string **type_**
- int **numSeats_**
- std::vector< Candidate * > **candidates_**
- std::vector< Ballot * > **ballots_**
- std::vector< Candidate * > **winners_**
- std::vector< Candidate * > **losers_**
- std::string **auditFilePath_**
- std::stringstream **auditText_**

3.3.1 Detailed Description

The Election class is an abstract class that provides the necessary functionality to represent an election. While non-abstract child classes are required to implement their own Ballot tallying algorithms and results, the Election class provides a template of necessary methods and properties that all types of elections will utilize.

3.3.2 Constructor & Destructor Documentation

3.3.2.1 Election()

```
Election::Election (
            std::string type,
            int seats,
            std::vector< Candidate *> cands,
            std::vector< Ballot *> bals )
```

The constructor for the Election class sets the type of the election, the number of seats in the election, the vector that stores the Candidates who are running in the election, and the vector that stores the Ballots cast in the election. It also defaults the name of the audit file path to the empty string.

Parameters

type	a string that indicates the election type
seats	an int that indicates the number of seats in the election
cands	a vector of Candidates who are up for election
bals	a vector of Ballots that were cast in the election

Author

Brendan Ritchie (ritch167)

3.3.3 Member Function Documentation

3.3.3.1 addLoser()

```
void Election::addLoser (
            Candidate * lose )  [protected]
```

This methods adds a candidate to losers_ vector which is a vector of Candidates.

Author

Yifan Zhang(zhan4372)

3.3.3.2 addWinner()

```
void Election::addWinner (
            Candidate * win )  [protected]
```

This methods adds a candidate to winners_ vector which is a vector of Candidates.

Author

Yifan Zhang(zhan4372)

3.3.3.3 getAuditFilePath()

```
std::string Election::getAuditFilePath ( )
```

This method returns the path for the audit file in the Election object.

Author

Brendan Ritchie (ritch167)

3.3.3.4 getBallots()

```
std::vector<Ballot*> Election::getBallots ( )
```

This method returns the vector of Ballot pointers in the Election object.

Author

Brendan Ritchie (ritch167)

3.3.3.5 getCandidates()

```
std::vector<Candidate*> Election::getCandidates ( )
```

This method returns the vector of Candidate pointers in the Election object.

Author

Brendan Ritchie (ritch167)

3.3.3.6 getLosers()

```
std::vector<Candidate*> Election::getLosers ( )
```

This method returns the losers which is a vector of Candidate pointers in the Election object.

Author

Brendan Ritchie (ritch167)

3.3.3.7 getNumSeats()

```
int Election::getNumSeats ( )
```

This method returns the number of the seats of the Election object.

Author

Brendan Ritchie (ritch167)

3.3.3.8 getResults()

```
virtual std::string Election::getResults ( )  [pure virtual]
```

This pure virtual method will be implemented in the child classes of Election and will be responsible for compiling the results for the specific election type

Implemented in STVElection, and PluralityElection.

3.3.3.9 getType()

```
std::string Election::getType ( )
```

This method returns the type of the Election object.

Author

Brendan Ritchie (ritch167)

3.3.3.10 getWinners()

```
std::vector<Candidate*> Election::getWinners ( )
```

This method returns the winners which is a vector of Candidate pointers in the Election object.

Author

Brendan Ritchie (ritch167)

3.3.3.11 runAlgorithm()

```
virtual void Election::runAlgorithm ( )  [pure virtual]
```

This pure virtual method will be implemented in the child classes of Election and will be responsible for running the speicific election type's vote tallying algorithm.

Implemented in STVElection, and PluralityElection.

3.3.3.12 setAuditFilePath()

```
void Election::setAuditFilePath (
            std::string name )  [protected]
```

This method sets the auditFilePath of the Election object

Parameters

name	a string that the auditFilePath of the Election object will be set to

Author

Brendan Ritchie (ritch167)

3.3.3.13 writeToAuditFile()

```
void Election::writeToAuditFile ( )  [protected]
```

This methods writes the content of stingstream auditText_ to audit files with file path as auditFilePath_.

Author

Yifan Zhang(zhan4372)

The documentation for this class was generated from the following file:

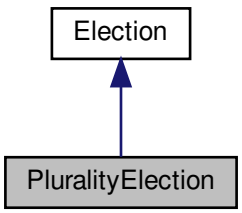
- /home/ritch167/csci5801/repo-Team4/Project2/src/election.h

3.4 PluralityElection Class Reference

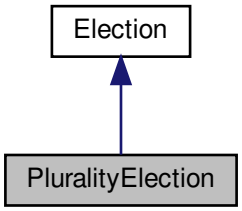
The PluralityElection class is a child class of the abstract Election class that provides the necessary functionality to represent a Plurality election.

```
#include <plurality_election.h>
```

Inheritance diagram for PluralityElection:



Collaboration diagram for PluralityElection:



Public Member Functions

- PluralityElection (std::string type, int seats, std::vector< Candidate *> cands, std::vector< Ballot *> bals)
- void runAlgorithm () override
- std::string getResults () override

Additional Inherited Members

3.4.1 Detailed Description

The PluralityElection class is a child class of the abstract Election class that provides the necessary functionality to represent a Plurality election.

3.4.2 Constructor & Destructor Documentation

3.4.2.1 PluralityElection()

```
PluralityElection::PluralityElection (
            std::string type,
            int seats,
            std::vector< Candidate *> cands,
            std::vector< Ballot *> bals )
```

The constructor for the PluralityElection class utilizes the constructor of the Election class to set the type, number of seats, the Candidates in the election, and the Ballots in the election.

Parameters

type	a string that indicates the election type
seats	an int that indicates the number of seats in the election
cands	a vector of Candidates who are up for election
bals	a vector of Ballots that were cast in the election

Author

Brendan Ritchie (ritch167)

3.4.3 Member Function Documentation

3.4.3.1 getResults()

```
std::string PluralityElection::getResults ( )  [override], [virtual]
```

This method gets the result of running the plurality algorithm. It returns a string variable that contains the results of the plurality election, as well as the Candidates and their vote percentage. Also, the string contains who won and who lost.

Author

Yiwen Xu (xu000515)

Implements Election.

3.4.3.2 runAlgorithm()

```
void PluralityElection::runAlgorithm ( )  [override], [virtual]
```

This method is responsible for running the election based on the plurality algorithm.

Author

Yiwen Xu (xu000515)

Implements Election.

The documentation for this class was generated from the following file:

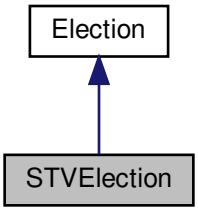
- /home/ritch167/csci5801/repo-Team4/Project2/src/plurality_election.h

3.5 STVElection Class Reference

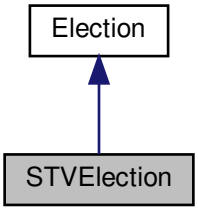
The STVElection class is a child class of the abstract Election class that provides the necessary functionality to represent a Single-Transferrable Vote (ranked choice) election using the Droop algorithm. On top of all of the properties of the Election class, the STVElection class also has a value for the Droop quota, a value indicating whether to turn the Ballot shuffle on/off, and a vector that stores the order of the shuffled Ballot objects for the election.

`#include <stv_election.h>`

Inheritance diagram for STVElection:



Collaboration diagram for STVElection:



Public Member Functions

- STVElection (std::string type, int seats, std::vector< Candidate *> cands, std::vector< Ballot *> bals, bool shuffle)
- void runAlgorithm () override
- int getDroop ()
- bool getShuffleStatus ()
- std::vector< int > getShuffledBallots ()
- std::string getResults () override

Additional Inherited Members

3.5.1 Detailed Description

The STVElection class is a child class of the abstract Election class that provides the necessary functionality to represent a Single-Transferrable Vote (ranked choice) election using the Droop algorithm. On top of all of the properties of the Election class, the STVElection class also has a value for the Droop quota, a value indicating whether to turn the Ballot shuffle on/off, and a vector that stores the order of the shuffled Ballot objects for the election.

3.5.2 Constructor & Destructor Documentation

3.5.2.1 STVElection()

```
STVElection::STVElection (
            std::string type,
            int seats,
            std::vector< Candidate *> cands,
            std::vector< Ballot *> bals,
            bool shuffle )
```

The constructor for the STVElection class utilizes the constructor of the Election class to set the type, number of seats, the Candidates in the election, and the Ballots in the election. It the sets the shuffle status of the STVElection to the value in shuffle. It also calculates the Droop quota here and defaults the vector that stores the order of the shuffled Ballot objects for the election to hold the ints 0 to the number of Ballotsin the election.

Parameters

type	a string that indicates the election type
seats	an int that indicates the number of seats in the election
cands	a vector of Candidates who are up for election
bals	a vector of Ballots that were cast in the election
shuffle	a boolean that indicates whether to turn the Ballot shuffle on/off

Author

Brendan Ritchie (ritch167)

3.5.3 Member Function Documentation

3.5.3.1 getDroop()

```
int STVElection::getDroop ( )
```

This method returns the droop quota of the STVElection object

Author

Brendan Ritchie (ritch167)

3.5.3.2 getResults()

```
std::string STVElection::getResults ( )  [override], [virtual]
```

This method returns the result of an election with STV algorithm

Author

Yifan Zhang (zhan4372)

Implements Election.

3.5.3.3 getShuffledBallots()

```
std::vector<int> STVElection::getShuffledBallots ( )
```

This method returns the vector of shullfed index of ballots of the STVElection object

Author

Brendan Ritchie (ritch167)

3.5.3.4 getShuffleStatus()

```
bool STVElection::getShuffleStatus ( )
```

This method returns the status of the shullfle option of the STVElection object

Author

Brendan Ritchie (ritch167)

3.5.3.5 runAlgorithm()

```
void STVElection::runAlgorithm ( )  [override], [virtual]
```

This is the primary function which is used to run the algorithm to determine winners and losers for an STV Election. It shuffles the ballots if necessary, initially distributes them to candidates, and then relies on the redistribute() function to handle ballot redistribution from there on out. It also makes use of the writeToAuditFile() to write the necessary information for election auditing to the designated text file.

Author

Brendan Ritchie (ritch167)

Implements Election.

The documentation for this class was generated from the following file:

- /home/ritch167/csci5801/repo-Team4/Project2/src/stv_election.h

3.6 VotingApp Class Reference

The VotingApp class provides the necessary functionality to run the user interface for the application. It handles all interactions with the user in the interface and initiates the process of running an election after being given all the necessary information by the user. It also provides the necessary functionality to run in a test mode for debugging purposes.

```
#include <voting_app.h>
```

Public Member Functions

- VotingApp (bool test_mode)
- void run ()

3.6.1 Detailed Description

The VotingApp class provides the necessary functionality to run the user interface for the application. It handles all interactions with the user in the interface and initiates the process of running an election after being given all the necessary information by the user. It also provides the necessary functionality to run in a test mode for debugging purposes.

3.6.2 Constructor & Destructor Documentation

3.6.2.1 VotingApp()

```
VotingApp::VotingApp (
            bool test_mode )
```

The constructor for the VotingApp class sets test mode status of the object. It defaults the Election member variable to NULL.

Parameters

test_mode	a boolean that indicates whether the application is running in test mode or not

Author

Brendan Ritchie (ritch167)

3.6.3 Member Function Documentation

3.6.3.1 run()

```
void VotingApp::run ( )
```

This is the run method for the voting system. No inputs and returns nothing.

Author

Sara Nelson (nels8907)

The documentation for this class was generated from the following file:

- /home/ritch167/csci5801/repo-Team4/Project2/src/voting_app.h

Index