

Voting System

Generated by Doxygen 1.8.17

1 Bug List	1
2 Hierarchical Index	3
2.1 Class Hierarchy	3
3 Class Index	5
3.1 Class List	5
4 File Index	7
4.1 File List	7
5 Class Documentation	9
5.1 AuditFile Class Reference	9
5.1.1 Detailed Description	10
5.1.2 Constructor & Destructor Documentation	10
5.1.2.1 AuditFile()	10
5.1.3 Member Function Documentation	10
5.1.3.1 getFile()	10
5.1.3.2 labelFile()	10
5.1.3.3 setFile()	11
5.1.3.4 setOutputResult()	11
5.1.3.5 write()	11
5.2 Ballot Class Reference	11
5.2.1 Detailed Description	12
5.2.2 Constructor & Destructor Documentation	12
5.2.2.1 Ballot() [1/2]	12
5.2.2.2 Ballot() [2/2]	12
5.2.3 Member Function Documentation	13
5.2.3.1 getIndex()	13
5.2.3.2 getMapping()	13
5.2.3.3 getRank()	13
5.2.3.4 setMapping()	13
5.2.3.5 setRank()	14
5.3 Candidate Class Reference	14
5.3.1 Detailed Description	15
5.4 CPLLottery Class Reference	15
5.5 CPLProcessing Class Reference	16
5.5.1 Constructor & Destructor Documentation	17
5.5.1.1 CPLProcessing()	17
5.5.2 Member Function Documentation	18
5.5.2.1 output()	18
5.5.2.2 read()	18
5.5.2.3 setUp()	18
5.6 CPLVoteBallot Class Reference	19

5.6.1 Constructor & Destructor Documentation	20
5.6.1.1 CPLVoteBallot()	20
5.6.2 Member Function Documentation	21
5.6.2.1 getMapAllocatedSeat()	21
5.6.2.2 getMapBallot()	21
5.6.2.3 getMapRemainSeat()	21
5.6.2.4 getNumParties()	22
5.6.2.5 getParties()	22
5.6.2.6 getQuota()	22
5.6.2.7 getSeats()	23
5.6.2.8 setMapAllocatedSeat()	23
5.6.2.9 setMapRemainSeat()	23
5.6.2.10 setParties()	25
5.6.2.11 setSeats()	25
5.7 CPLVoteSystem Class Reference	26
5.7.1 Constructor & Destructor Documentation	27
5.7.1.1 CPLVoteSystem()	27
5.7.2 Member Function Documentation	27
5.7.2.1 conductElection()	27
5.7.2.2 getWinner()	27
5.7.2.3 startElection()	28
5.8 Display Class Reference	28
5.8.1 Detailed Description	29
5.8.2 Constructor & Destructor Documentation	29
5.8.2.1 Display()	29
5.8.3 Member Function Documentation	29
5.8.3.1 getOutputTerminal()	29
5.8.3.2 overWrite()	29
5.8.3.3 print()	30
5.8.3.4 setOutputTerminal()	30
5.8.3.5 write()	30
5.9 IBallotProcessing Class Reference	31
5.10 IElectionStratgey Class Reference	32
5.11 IRBallot Class Reference	32
5.11.1 Detailed Description	33
5.11.2 Constructor & Destructor Documentation	33
5.11.2.1 IRBallot()	33
5.11.3 Member Function Documentation	34
5.11.3.1 getCandidates()	34
5.11.3.2 getMapPercentage()	34
5.11.3.3 getNumCandidates()	34
5.11.3.4 setCandidates()	34

5.11.3.5 setMapPercentage()	35
5.11.3.6 setNumCandidates()	35
5.12 IRCandidate Class Reference	35
5.12.1 Detailed Description	37
5.12.2 Constructor & Destructor Documentation	37
5.12.2.1 IRCandidate() [1/2]	37
5.12.2.2 IRCandidate() [2/2]	37
5.12.3 Member Function Documentation	37
5.12.3.1 addBallot()	37
5.12.3.2 getBallotList()	38
5.12.3.3 getNumBallots()	38
5.12.3.4 setBallotList()	38
5.12.3.5 setNumBallots()	38
5.13 IRProcessing Class Reference	39
5.13.1 Detailed Description	40
5.13.2 Constructor & Destructor Documentation	40
5.13.2.1 IRProcessing() [1/2]	40
5.13.2.2 IRProcessing() [2/2]	41
5.13.3 Member Function Documentation	41
5.13.3.1 getCandidates()	41
5.13.3.2 getMapPercentage()	41
5.13.3.3 getNumCandidates()	42
5.13.3.4 nextLine()	42
5.13.3.5 output()	42
5.13.3.6 read()	42
5.13.3.7 redistribute()	42
5.13.3.8 setCandidates()	43
5.13.3.9 setMapPercentage()	43
5.13.3.10 setNumCandidates()	43
5.13.3.11 setUp()	44
5.14 IRVoteSystem Class Reference	44
5.14.1 Detailed Description	46
5.14.2 Constructor & Destructor Documentation	46
5.14.2.1 IRVoteSystem() [1/2]	46
5.14.2.2 IRVoteSystem() [2/2]	46
5.14.3 Member Function Documentation	46
5.14.3.1 conductElection()	46
5.14.3.2 getBallotSystem()	47
5.14.3.3 getCandidates()	47
5.14.3.4 getElimnation()	47
5.14.3.5 getProcessedBallot()	48
5.14.3.6 getWinner()	48

5.14.3.7 setBallotSystem()	48
5.14.3.8 setCandidates()	48
5.14.3.9 setProcessedBallot()	49
5.14.3.10 setWinner()	49
5.14.3.11 startElection()	49
5.15 ISpecialCase Class Reference	50
5.16 IVoteBallot Class Reference	50
5.16.1 Detailed Description	51
5.16.2 Member Function Documentation	51
5.16.2.1 getMapBallot()	51
5.16.2.2 getTotalBallot()	51
5.16.2.3 setMapBallot()	51
5.16.2.4 setTotalBallot()	52
5.17 IVotingSystem Class Reference	52
5.17.1 Detailed Description	54
5.17.2 Member Function Documentation	54
5.17.2.1 conductElection()	54
5.17.2.2 getAuditFile()	54
5.17.2.3 getAuditing()	55
5.17.2.4 getDisplayScreen()	55
5.17.2.5 getSpecialCase()	55
5.17.2.6 getStatus()	55
5.17.2.7 setAuditFile()	55
5.17.2.8 setAuditing()	56
5.17.2.9 setDisplayScreen()	56
5.17.2.10 setSpecialCase()	56
5.17.2.11 setStatus()	57
5.17.2.12 startElection()	57
5.18 Party Class Reference	57
5.18.1 Detailed Description	58
5.18.2 Constructor & Destructor Documentation	58
5.18.2.1 Party() [1/2]	58
5.18.2.2 Party() [2/2]	58
5.18.3 Member Function Documentation	58
5.18.3.1 getCandidate()	59
5.18.3.2 getMaxSeat()	59
5.18.3.3 getName()	59
5.18.3.4 setCandidate()	59
5.18.3.5 setMaxSeat()	60
5.18.3.6 setName()	60
5.19 PopularityCase Class Reference	60
5.19.1 Detailed Description	61

5.19.2 Constructor & Destructor Documentation	61
5.19.2.1 PopularityCase()	61
5.19.3 Member Function Documentation	62
5.19.3.1 getCandidates()	62
5.19.3.2 helper()	62
5.19.3.3 run()	62
5.19.3.4 setCandidates()	62
5.20 TieBreaker Class Reference	63
5.20.1 Detailed Description	64
5.20.2 Constructor & Destructor Documentation	64
5.20.2.1 TieBreaker()	64
5.20.3 Member Function Documentation	64
5.20.3.1 run() [1/2]	64
5.20.3.2 run() [2/2]	64
6 File Documentation	67
6.1 src/CPLProcessing.cpp File Reference	67
6.1.1 Detailed Description	67
6.2 src/CPLVoteBallot.cpp File Reference	68
6.2.1 Detailed Description	68
6.3 src/CPLVoteSystem.cpp File Reference	68
6.3.1 Detailed Description	69
6.4 src/include/AuditFile.h File Reference	69
6.4.1 Detailed Description	70
6.5 src/include/Ballot.h File Reference	70
6.5.1 Detailed Description	71
6.6 src/include/CPLProcessing.h File Reference	72
6.6.1 Detailed Description	73
6.7 src/include/CPLVoteBallot.h File Reference	73
6.7.1 Detailed Description	74
6.8 src/include/CPLVoteSystem.h File Reference	74
6.8.1 Detailed Description	75
6.9 src/include/Display.h File Reference	76
6.9.1 Detailed Description	76
6.10 src/include/IRBallot.h File Reference	77
6.10.1 Detailed Description	78
6.11 src/include/IRCandidate.h File Reference	79
6.11.1 Detailed Description	80
6.12 src/include/IRProcessing.h File Reference	80
6.12.1 Detailed Description	81
6.13 src/include/IRVoteSystem.h File Reference	81
6.13.1 Detailed Description	82

6.14 src/include/IVoteBallot.h File Reference	82
6.14.1 Detailed Description	83
6.15 src/include/Party.h File Reference	83
6.15.1 Detailed Description	84
6.16 src/include/PopularityCase.h File Reference	85
6.16.1 Detailed Description	85
Index	87

Chapter 1

Bug List

File [CPLProcessing.cpp](#)

No known bugs.

File [CPLProcessing.h](#)

No known bugs.

File [CPLVoteBallot.cpp](#)

No known bugs.

File [CPLVoteSystem.cpp](#)

No known bugs.

File [CPLVoteSystem.h](#)

No known bugs.

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

AuditFile	9
Ballot	11
Candidate	14
IRCandidate	35
Display	28
IBallotProcessing	31
CPLProcessing	16
IRProcessing	39
IElectionStratgey	32
ISpecialCase	50
CPLLottery	15
PopularityCase	60
TieBreaker	63
IVoteBallot	50
CPLVoteBallot	19
IRBallot	32
IVotingSystem	52
CPLVoteSystem	26
IRVoteSystem	44
Party	57

Chapter 3

Class Index

3.1 Class List

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

AuditFile	File that logs information about a voting audit	9
Ballot	A Ballot class to hold an individual ballot found in a Instant Runoff election. Each IR Candidate will hold a list of this Ballot	11
Candidate	An output class from the IBallotProcessing interface. This class holds objects/information for an election Candidate including their name and party. These attributes will then be displayed when a candidate wins eiher an IR or CPL election	14
CPLLottery	15
CPLProcessing	16
CPLVoteBallot	19
CPLVoteSystem	26
Display	Display that can be used to output information	28
IBallotProcessing	31
IElectionStratgey	32
IRBallot	An output class from the IR Processing class. This class holds objects/information for the IR Vote System to condut its election and determine who's the winner	32
IRCandidate	A candidate class for an Instant Runoff Candidate . This classs is for holding ballots	35
IRProcessing	IR Processing is to process all ballots in a .csv file and prepare an organized set of information to send to the IRVoteSystem class, conducting and determining the winner	39
IRVoteSystem	The voting system to conduct the election and determine a winning candidate	44
ISpecialCase	50
IVoteBallot	The abstract class where all VoteBallots are inherited from	50
IVotingSystem	The abstract class for each voting election system will inherit from	52
Party	A class representing a political party with candidate information and seat limits	57
PopularityCase	A class representing a popularity special case implementation	60
TieBreaker	A class representing a tiebreaker special case implementation	63

Chapter 4

File Index

4.1 File List

Here is a list of all documented files with brief descriptions:

src/ CPLProcessing.cpp	
Process CPL ballot file and save the information of parties, candidates and seats	67
src/ CPLVoteBallot.cpp	
Initiate CPLVoteBallot	68
src/ CPLVoteSystem.cpp	
Conduct election according to CPL rules and show the winners	68
src/include/ AuditFile.h	69
src/include/ Ballot.h	70
src/include/ Candidate.h	??
src/include/ CPLProcessing.h	
Process CPL ballot file and save the information of parties, candidates and seats	72
src/include/ CPLVoteBallot.h	
CPLVoteBallot saves information of election	73
src/include/ CPLVoteSystem.h	
Conduct election and show winners	74
src/include/ Display.h	76
src/include/ IBallotProcessing.h	??
src/include/ ICPLLottery.h	??
src/include/ IElectionStrategy.h	??
src/include/ IRBallot.h	77
src/include/ IRCandidate.h	79
src/include/ IRProcessing.h	80
src/include/ IRVoteSystem.h	81
src/include/ ISpecialCase.h	??
src/include/ IVoteBallot.h	82
src/include/ IVotingSystem.h	??
src/include/ Party.h	83
src/include/ PopularityCase.h	85
src/include/ TieBreaker.h	??

Chapter 5

Class Documentation

5.1 AuditFile Class Reference

The [AuditFile](#) class represents a file that logs information about a voting audit.

```
#include <AuditFile.h>
```

Public Member Functions

- [AuditFile](#) ()
Default constructor for the [AuditFile](#) class.
- [AuditFile](#) (string output_result)
Constructor for the [AuditFile](#) class that takes a file name as an argument.
- void **open** ()
- void **close** ()
- void [labelFile](#) (string name)
A function to label a file with a specific name.
- void [produceFile](#) ()
A function to create a new file.
- void [write](#) (string output_result, bool newLine=true)
A function to write the output into the file.
- FILE * [getFile](#) () const
Getter function for the file.
- void [setFile](#) (FILE *file_)
Setter function for the file.
- string [outputResult](#) () const
Getter function for the output.
- void [setOutputResult](#) (const string &outputResult)
Setter function for the output.

Protected Attributes

- FILE * **file**
- ofstream **fileStream**
- string **output_result**
- string **name**
- string **fileName**

5.1.1 Detailed Description

The [AuditFile](#) class represents a file that logs information about a voting audit.

5.1.2 Constructor & Destructor Documentation

5.1.2.1 AuditFile()

```
AuditFile::AuditFile (
    string output_result )
```

Constructor for the [AuditFile](#) class that takes a file name as an argument.

Parameters

<i>output_result</i>	is the parameter
----------------------	------------------

5.1.3 Member Function Documentation

5.1.3.1 getFile()

```
FILE* AuditFile::getFile ( ) const [inline]
```

Getter function for the file.

5.1.3.2 labelFile()

```
void AuditFile::labelFile (
    string name )
```

A function to label a file with a specific name.

Parameters

<i>name</i>	The name of the file to be labeled
-------------	------------------------------------

5.1.3.3 setFile()

```
void AuditFile::setFile (
    FILE * file_ ) [inline]
```

Setter function for the file.

Parameters

<i>output_result</i>	The output to be written to the file
----------------------	--------------------------------------

5.1.3.4 setOutputResult()

```
void AuditFile::setOutputResult (
    const string & outputResult ) [inline]
```

Setter function for the output.

Parameters

<i>outputResult</i>	The output to be set.
---------------------	-----------------------

5.1.3.5 write()

```
void AuditFile::write (
    string output_result,
    bool newLine = true )
```

A function to write the output into the file.

Parameters

<i>output_result</i>	The output to be written to the file
----------------------	--------------------------------------

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

- src/include/[AuditFile.h](#)
- src/AuditFile.cpp

5.2 Ballot Class Reference

A [Ballot](#) class to hold an individual ballot found in a Instant Runoff election. Each IR [Candidate](#) will hold a list of this [Ballot](#).

```
#include <Ballot.h>
```

Public Member Functions

- [Ballot](#) (int rank, vector< int > mapping)
Construct a new [Ballot](#) object.
- [Ballot](#) ()
Construct a new [Ballot](#) object.
- void [increaseRank](#) ()
Is used to increase the ranking variable.
- int [getIndex](#) ()
Get the index of ranking in the map.
- int [getRank](#) ()
Get the ranking.
- void [setRank](#) (int rank_)
Set the ranking. Error checks if the ranking goes below 0 and if the ranking is beyond the size of the mapping.
- vector< int > [getMapping](#) ()
Get the mapping.
- void [setMapping](#) (vector< int > &mapping)
Set the mapping.

5.2.1 Detailed Description

A [Ballot](#) class to hold an individual ballot found in a Instant Runoff election. Each IR [Candidate](#) will hold a list of this [Ballot](#).

5.2.2 Constructor & Destructor Documentation

5.2.2.1 [Ballot](#)() [1/2]

```
Ballot::Ballot (
    int rank,
    vector< int > mapping )
```

Construct a new [Ballot](#) object.

Parameters

<i>rank</i>	int the current preference (rank).
<i>mapping</i>	vector<int> the mapping of preferences to candidates.

5.2.2.2 [Ballot](#)() [2/2]

```
Ballot::Ballot ( )
```

Construct a new [Ballot](#) object.

5.2.3 Member Function Documentation

5.2.3.1 getIndex()

```
int Ballot::getIndex ( )
```

Get the index of ranking in the map.

Returns

int

5.2.3.2 getMapping()

```
vector<int> Ballot::getMapping ( ) [inline]
```

Get the mapping.

Returns

vector<int>

5.2.3.3 getRank()

```
int Ballot::getRank ( ) [inline]
```

Get the ranking.

Returns

int

5.2.3.4 setMapping()

```
void Ballot::setMapping (
    vector< int > & mapping ) [inline]
```

Set the mapping.

Parameters

<i>mapping</i>	
----------------	--

5.2.3.5 setRank()

```
void Ballot::setRank (
    int rank_ ) [inline]
```

Set the ranking. Error checks if the ranking goes below 0 and if the ranking is beyond the size of the mapping.

Parameters

<i>rank</i> ↔	int
—	

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

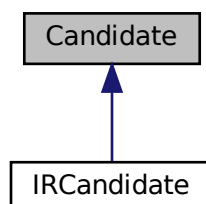
- src/include/[Ballot.h](#)
- src/Ballot.cpp

5.3 Candidate Class Reference

An output class from the [IBallotProcessing](#) interface. This class holds objects/information for an election [Candidate](#) including their name and party. These attributes will then be displayed when a candidate wins either an IR or CPL election.

```
#include <Candidate.h>
```

Inheritance diagram for Candidate:



Public Member Functions

- **Candidate** (string name, string party)
- string **getName** ()
- void **setName** (string name_)
- string **getParty** ()
- void **setParty** (string party_)

Protected Attributes

- string **name**
- string **party**

5.3.1 Detailed Description

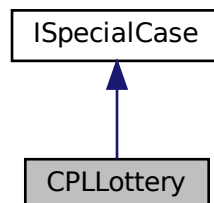
An output class from the [IBallotProcessing](#) interface. This class holds objects/information for an election [Candidate](#) including their name and party. These attributes will then be displayed when a candidate wins either an IR or CPL election.

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

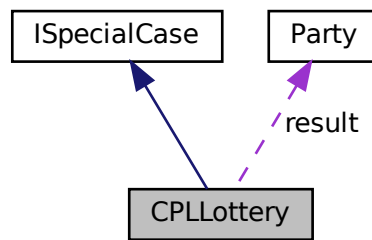
- src/include/Candidate.h
- src/Candidate.cpp

5.4 CPLLottery Class Reference

Inheritance diagram for CPLLottery:



Collaboration diagram for CPLLottery:



Public Member Functions

- **CPLLottery** ([Party](#) [] parties)
- string **result1000** () const
- void **setResult1000** (const string &result1000)

Protected Attributes

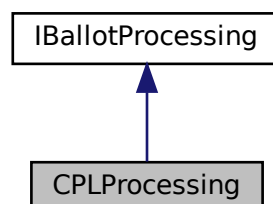
- [Party](#) **result**
- std::vector< [Party](#) > **parties**
- string **result_1000**

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

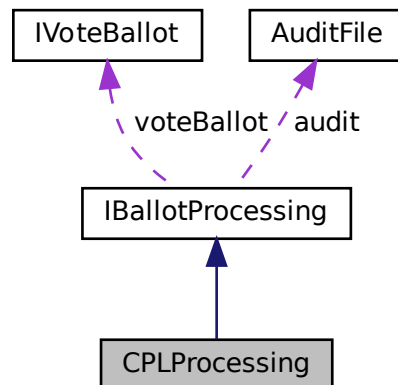
- src/include/ICPLLottery.h

5.5 CPLProcessing Class Reference

Inheritance diagram for CPLProcessing:



Collaboration diagram for CPLProcessing:



Public Member Functions

- [CPLProcessing](#) (FILE *file_)
Create a new [CPLProcessing](#) with an input file CPL election.
- [CPLVoteBallot * output](#) ()
Output a [CPLVoteBallot](#) that contains all ballot information.
- bool [read](#) ()
Start to read files/.
- bool [setUp](#) ()
Parse files and save ballot information/.

Additional Inherited Members

5.5.1 Constructor & Destructor Documentation

5.5.1.1 CPLProcessing()

```

CPLProcessing::CPLProcessing (
    FILE * file_ )
  
```

Create a new [CPLProcessing](#) with an input file CPL election.

Parameters

<i>file</i> ↔	CPL ballot file
—	

Returns

void

5.5.2 Member Function Documentation

5.5.2.1 output()

```
CPLVoteBallot * CPLProcessing::output ( ) [virtual]
```

Output a [CPLVoteBallot](#) that contains all ballot information.

Parameters

<i>no</i>	parameter
-----------	-----------

Returns

CPLVoteBallot*

Implements [IBallotProcessing](#).

5.5.2.2 read()

```
bool CPLProcessing::read ( ) [inline], [virtual]
```

Start to read files/.

Parameters

<i>no</i>	parameter
-----------	-----------

Returns

bool

Implements [IBallotProcessing](#).

5.5.2.3 setUp()

```
bool CPLProcessing::setUp ( ) [virtual]
```

Parse files and save ballot infomration/.

Parameters

<i>no</i>	parameter
-----------	-----------

Returns

bool

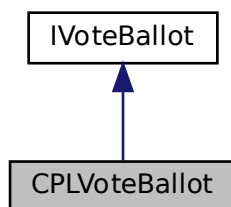
Implements [IBallotProcessing](#).

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

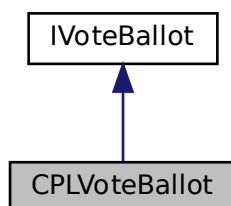
- [src/include/CPLProcessing.h](#)
- [src/CPLProcessing.cpp](#)

5.6 CPLVoteBallot Class Reference

Inheritance diagram for CPLVoteBallot:



Collaboration diagram for CPLVoteBallot:



Public Member Functions

- [CPLVoteBallot](#) (std::vector< [Party](#) * > parties, std::vector< int > mapBallot, std::vector< int > mapAllocated, std::vector< int > mapRemain, int numSeats, int parttNum, int quotaNum)
Create a new [CPLVoteBallot](#).
- std::vector< [Party](#) * > [getParties](#) () const
Get list of parties that participate election.
- void [setParties](#) (const std::vector< [Party](#) * > &parties_)
Set a list of parties that participate election.
- std::vector< int > [getMapAllocatedSeat](#) () const
Get the map of allocated seat.
- void [setMapAllocatedSeat](#) (const std::vector< int > &mapAllocatedSeat_)
Get the map of allocated seat.
- std::vector< int > [getMapRemainSeat](#) () const
Get the map of remaining seats.
- void [setMapRemainSeat](#) (const std::vector< int > &mapRemainSeat_)
Set the map of remaining seats.
- int [getSeats](#) () const
Get the total number of seats.
- void [setSeats](#) (int seats_)
Set the total number of seats.
- std::vector< int > [getMapBallot](#) () const
Get the map of ballots.
- int [getNumParties](#) () const
Get the total number of parties.
- int [getQuota](#) () const
Get the quota value.

Additional Inherited Members

5.6.1 Constructor & Destructor Documentation

5.6.1.1 CPLVoteBallot()

```
CPLVoteBallot::CPLVoteBallot (
    std::vector< Party * > parties,
    std::vector< int > mapBallot,
    std::vector< int > mapAllocated,
    std::vector< int > mapRemain,
    int numSeats,
    int parttNum,
    int quotaNum )
```

Create a new [CPLVoteBallot](#).

Parameters

<i>parties</i>	list of parties that participate election
<i>mapBallot</i>	map of parties and their total number of ballots received
<i>mapAllocated</i>	map of parties and the number of seats they received
<i>mapRemain</i>	map of parties and the number of remaining ballots
<i>numSeat</i>	total number of seats
<i>partyNum</i>	total number of parties

Returns

void

5.6.2 Member Function Documentation

5.6.2.1 getMapAllocatedSeat()

```
std::vector<int> CPLVoteBallot::getMapAllocatedSeat ( ) const [inline]
```

Get the map of allocated seat.

Parameters

<i>parties</i> ↔	the new list of parties
—	

Returns

std::vector<int>

5.6.2.2 getMapBallot()

```
std::vector<int> CPLVoteBallot::getMapBallot ( ) const [inline]
```

Get the map of ballots.

Parameters

<i>no</i>	parameters
-----------	------------

Returns

std::vector<int>

5.6.2.3 getMapRemainSeat()

```
std::vector<int> CPLVoteBallot::getMapRemainSeat ( ) const [inline]
```

Get the map of remaining seats.

Parameters

<i>no</i>	parameters
-----------	------------

Returns

std::vector<int>

5.6.2.4 getNumParties()

```
int CPLVoteBallot::getNumParties ( ) const [inline]
```

Get the total number of parties.

Parameters

<i>no</i>	parameters
-----------	------------

Returns

int

5.6.2.5 getParties()

```
std::vector<Party*> CPLVoteBallot::getParties ( ) const [inline]
```

Get list of parties that participate election.

Parameters

<i>no</i>	parameters
-----------	------------

Returns

std::vector<Party*>

5.6.2.6 getQuota()

```
int CPLVoteBallot::getQuota ( ) const [inline]
```

Get the quota value.

Parameters

<i>no</i>	parameters
-----------	------------

Returns

int

5.6.2.7 getSeats()

```
int CPLVoteBallot::getSeats ( ) const [inline]
```

Get the total number of seats.

Parameters

<i>no</i>	parameters
-----------	------------

Returns

int

5.6.2.8 setMapAllocatedSeat()

```
void CPLVoteBallot::setMapAllocatedSeat (
    const std::vector< int > & mapAllocatedSeat_ ) [inline]
```

Get the map of allocated seat.

Parameters

<i>mapAllocatedSeat_</i>	the new map of allocated seat
--------------------------	-------------------------------

Returns

void

5.6.2.9 setMapRemainSeat()

```
void CPLVoteBallot::setMapRemainSeat (
    const std::vector< int > & mapRemainSeat_ ) [inline]
```

Set the map of remaining seats.

Parameters

<i>mapRemain↔ Seat_</i>	the new map of remaining seats
-----------------------------	--------------------------------

Returns

void

5.6.2.10 setParties()

```
void CPLVoteBallot::setParties (
    const std::vector< Party * > & parties_ ) [inline]
```

Set a list of parties that participate election.

Parameters

<i>no</i>	parameters
-----------	------------

Returns

void

5.6.2.11 setSeats()

```
void CPLVoteBallot::setSeats (
    int seats_ ) [inline]
```

Set the total number of seats.

Parameters

<i>seats↔ _</i>	tota number of seat
---------------------	---------------------

Returns

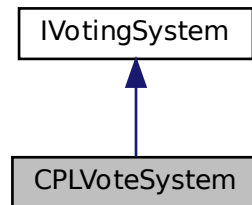
void

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

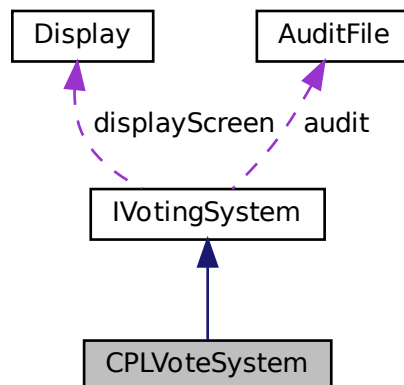
- src/include/CPLVoteBallot.h
- src/CPLVoteBallot.cpp

5.7 CPLVoteSystem Class Reference

Inheritance diagram for CPLVoteSystem:



Collaboration diagram for CPLVoteSystem:



Public Member Functions

- [CPLVoteSystem](#) ([CPLVoteBallot](#) *[CPLVoteBallot](#))
Create a new [CPLVoteSystem](#) with an input [CPLVoteBallot](#).
- bool [startElection](#) ()
Start election with checking the information ballots of each party.
- bool [conductElection](#) ()
Conduct election according to CPL rules.
- std::vector< [Candidate](#) > [getWinner](#) ()
Get the winners in each party.

Additional Inherited Members

5.7.1 Constructor & Destructor Documentation

5.7.1.1 CPLVoteSystem()

```
CPLVoteSystem::CPLVoteSystem (
    CPLVoteBallot * CPLVoteBallot )
```

Create a new [CPLVoteSystem](#) with an input [CPLVoteBallot](#).

Parameters

<i>incomingBallot</i>	CPL ballot file
-----------------------	-----------------

Returns

void

5.7.2 Member Function Documentation

5.7.2.1 conductElection()

```
bool CPLVoteSystem::conductElection ( ) [virtual]
```

Conduct election according to CPL rules.

Parameters

<i>no</i>	parameter
-----------	-----------

Returns

bool

Implements [IVotingSystem](#).

5.7.2.2 getWinner()

```
std::vector<Candidate> CPLVoteSystem::getWinner ( ) [inline]
```

Get the winners in each party.

Parameters

<i>no</i>	parameter
-----------	-----------

Returns

std::vector<Candidate>

5.7.2.3 startElection()

```
bool CPLVoteSystem::startElection ( ) [virtual]
```

Start election with checking the information ballots of each party.

Parameters

<i>no</i>	parameter
-----------	-----------

Returns

bool

Implements [IVotingSystem](#).

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

- src/include/[CPLVoteSystem.h](#)
- src/[CPLVoteSystem.cpp](#)

5.8 Display Class Reference

The [Display](#) class represents a display that can be used to output information.

```
#include <Display.h>
```

Public Member Functions

- [Display](#) (string outputTerminal)
Constructor for the [Display](#) class.
- [Display](#) ()
Default constructor for the [Display](#) class.
- void [write](#) (string outputTerminal)
A function to append to the output terminal.
- void [overWrite](#) (string outputTerminal)
A function to overwrite the output terminal.
- string [print](#) ()
A function to print the output.
- string [getOutputTerminal](#) () const
Getter function for the output terminal.
- void [setOutputTerminal](#) (const string &outputTerminal_)
Setter function for the output terminal.

Protected Attributes

- string `outputTerminal`

5.8.1 Detailed Description

The [Display](#) class represents a display that can be used to output information.

5.8.2 Constructor & Destructor Documentation

5.8.2.1 Display()

```
Display::Display (  
    string outputTerminal )
```

Constructor for the [Display](#) class.

Parameters

<i>outputTerminal</i>	The output terminal to be used.
-----------------------	---------------------------------

5.8.3 Member Function Documentation

5.8.3.1 getOutputTerminal()

```
string Display::getOutputTerminal ( ) const [inline]
```

Getter function for the output terminal.

Returns

The output terminal.

5.8.3.2 overWrite()

```
void Display::overWrite (  
    string outputTerminal )
```

A function to overwrite the output terminal.

Parameters

<i>outputTerminal</i>	The new output to be displayed.
-----------------------	---------------------------------

5.8.3.3 print()

```
string Display::print ( )
```

A function to print the output.

Returns

The output.

5.8.3.4 setOutputTerminal()

```
void Display::setOutputTerminal (
    const string & outputTerminal_ ) [inline]
```

Setter function for the output terminal.

Parameters

<i>outputTerminal_</i>	The output terminal to be set.
------------------------	--------------------------------

5.8.3.5 write()

```
void Display::write (
    string outputTerminal )
```

A function to append to the output terminal.

Parameters

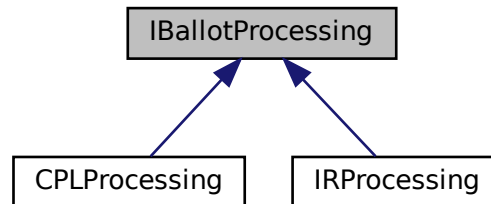
<i>outputTerminal</i>	is the information that will be appended to the terminal.
-----------------------	---

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

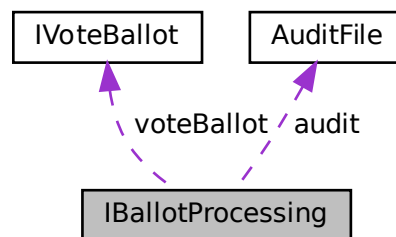
- [src/include/Display.h](#)
- [src/Display.cpp](#)

5.9 IBallotProcessing Class Reference

Inheritance diagram for IBallotProcessing:



Collaboration diagram for IBallotProcessing:



Public Member Functions

- virtual bool **setUp** ()=0
- virtual bool **read** ()=0
- virtual **IVoteBallot** * **output** ()=0
- virtual FILE * **getFile** () const
- virtual void **setFile** (FILE *file_)
- virtual int **getBLinesToRead** () const
- virtual void **setBLinesToRead** (int bLinesToRead_)
- virtual **IVoteBallot** * **getVoteBallot** ()
- virtual void **setVoteBallot** (**IVoteBallot** *voteBallot_)
- virtual std::vector< int > **getMapBallot** ()
- virtual void **setMapBallot** (const std::vector< int > &mapBallot_)
- **AuditFile** * **getAudit** () const
- void **setAudit** (**AuditFile** *audit_)
- int **getAuditing** () const
- void **setAuditing** (int auditing_)

Protected Attributes

- FILE * **file**
- int **bLinesToRead**
- [IVoteBallot](#) * **voteBallot**
- std::vector< int > **mapBallot**
- [AuditFile](#) * **audit**
- int **auditing** = 0

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

- src/include/IBallotProcessing.h

5.10 IElectionStrategy Class Reference

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

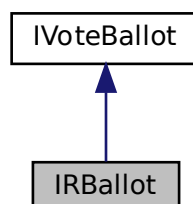
- src/include/IElectionStrategy.h

5.11 IRBallot Class Reference

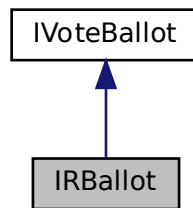
An output class from the IR Processing class. This class holds objects/information for the IR Vote System to conduct its election and determine who's the winner.

```
#include <IRBallot.h>
```

Inheritance diagram for IRBallot:



Collaboration diagram for IRBallot:



Public Member Functions

- [IRBallot](#) (std::vector< [IRCandidate](#) * > candidate, std::vector< int > [mapBallot](#), std::vector< double > mapPercentage)
Construct a new [IRBallot](#) object.
- std::vector< [IRCandidate](#) * > [getCandidates](#) () const
Get the list of Candidates.
- void [setCandidates](#) (const std::vector< [IRCandidate](#) * > &candidates_)
Set the list of Candidates.
- std::vector< double > [getMapPercentage](#) () const
Get the Percentage mapping.
- void [setMapPercentage](#) (const std::vector< double > &mapPercentage_)
Set the Percentage mapping.
- int [getNumCandidates](#) () const
Get the number of candidates.
- void [setNumCandidates](#) (int numCandidates_)
Set the number of candidates.

Additional Inherited Members

5.11.1 Detailed Description

An output class from the IR Processing class. This class holds objects/information for the IR Vote System to conduct its election and determine who's the winner.

5.11.2 Constructor & Destructor Documentation

5.11.2.1 IRBallot()

```

IRBallot::IRBallot (
    std::vector< IRCandidate * > candidate,
    std::vector< int > mapBallot,
    std::vector< double > mapPercentage )
  
```

Construct a new [IRBallot](#) object.

Parameters

<i>candidate</i>	vector<IRCandidate*> holds all the candidates
<i>candidate</i>	vector<IRCandidate*> holds all the candidates
<i>mapBallot</i>	vector<int> maps the amount of ballot each candidate has
<i>mapPercentage</i>	vector<double> maps the percentage each candidate has overall

5.11.3 Member Function Documentation

5.11.3.1 getCandidates()

```
std::vector<IRCandidate*> IRBallot::getCandidates ( ) const [inline]
```

Get the list of Candidates.

Returns

std::vector<IRCandidate*>

5.11.3.2 getMapPercentage()

```
std::vector<double> IRBallot::getMapPercentage ( ) const [inline]
```

Get the Percentage mapping.

Returns

std::vector<double>

5.11.3.3 getNumCandidates()

```
int IRBallot::getNumCandidates ( ) const [inline]
```

Get the number of candidates.

Returns

int

5.11.3.4 setCandidates()

```
void IRBallot::setCandidates (
    const std::vector< IRCandidate * > & candidates_ ) [inline]
```

Set the list of Candidates.

Parameters

<i>candidates</i> ↔	vector<IRCandidate*>
—	

5.11.3.5 setMapPercentage()

```
void IRBallot::setMapPercentage (
    const std::vector< double > & mapPercentage_ ) [inline]
```

Set the Percentage mapping.

Parameters

<i>map</i> ↔	
Percentage_	

5.11.3.6 setNumCandidates()

```
void IRBallot::setNumCandidates (
    int numCandidates_ ) [inline]
```

Set the number of candidates.

Parameters

<i>num</i> ↔	int
Candidates_	

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

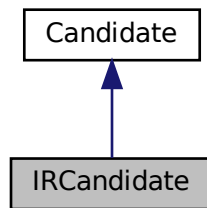
- src/include/IRBallot.h
- src/IRBallot.cpp

5.12 IRCandidate Class Reference

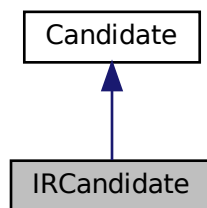
A candidate class for an Instant Runoff [Candidate](#). This class is for holding ballots.

```
#include <IRCandidate.h>
```

Inheritance diagram for IRCandidate:



Collaboration diagram for IRCandidate:



Public Member Functions

- [IRCandidate](#) (string name)
Construct a new [IRCandidate](#) object.
- [IRCandidate](#) ()
Construct a new [IRCandidate](#) object.
- void [addBallot](#) ([Ballot](#) *ballot_)
Add a ballot to the candidate.
- vector< [Ballot](#) * > [getBallotList](#) ()
Get the [Ballot](#) List object.
- void [setBallotList](#) (vector< [Ballot](#) * > list)
Set the [Ballot](#) List object.
- int [getNumBallots](#) () const
Get the number of ballots the [IR Candidate](#) has.
- void [setNumBallots](#) (int numBallots_)
Set the number of ballots the [IR Candidate](#) has.
- [Ballot](#) * [popBallot](#) ()

Protected Attributes

- int `numBallots` = 0
Holds the number of ballots.
- vector< `Ballot` * > `ballotList`
Hold the list of ballots an IR candidate has.

5.12.1 Detailed Description

A candidate class for an Instant Runoff `Candidate`. This class is for holding ballots.

5.12.2 Constructor & Destructor Documentation

5.12.2.1 IRCandidate() [1/2]

```
IRCandidate::IRCandidate (
    string name )
```

Construct a new `IRCandidate` object.

Parameters

<i>name</i>	a string argument for name of <code>Candidate</code> .
-------------	--

5.12.2.2 IRCandidate() [2/2]

```
IRCandidate::IRCandidate ( )
```

Construct a new `IRCandidate` object.

5.12.3 Member Function Documentation

5.12.3.1 addBallot()

```
void IRCandidate::addBallot (
    Ballot * ballot_ )
```

Add a ballot to the candidate.

Parameters

<i>ballot</i> ↔	A Ballot class input
—	

5.12.3.2 getBallotList()

```
vector<Ballot*> IRCandidate::getBallotList ( ) [inline]
```

Get the [Ballot](#) List object.

Returns

```
vector<Ballot*>
```

5.12.3.3 getNumBallots()

```
int IRCandidate::getNumBallots ( ) const [inline]
```

Get the number of ballots the IR [Candidate](#) has.

Returns

```
int
```

5.12.3.4 setBallotList()

```
void IRCandidate::setBallotList (
    vector< Ballot * > list ) [inline]
```

Set the [Ballot](#) List object.

Parameters

<i>list</i>	vector< Ballot *>
-------------	-----------------------------------

5.12.3.5 setNumBallots()

```
void IRCandidate::setNumBallots (
    int numBallots_ ) [inline]
```

Set the number of ballots the IR [Candidate](#) has.

Parameters

<i>num</i> ↔ <i>Ballots_</i>	int
---------------------------------	-----

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

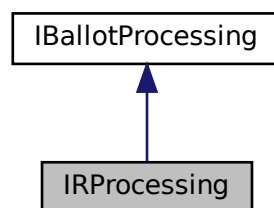
- `src/include/IRCandidate.h`
- `src/IRCandidate.cpp`

5.13 IRProcessing Class Reference

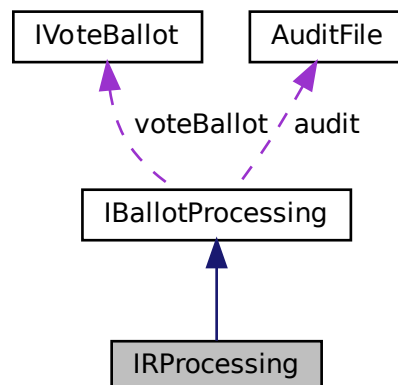
IR Processing is to process all ballots in a .csv file and prepare an organized set of information to send to the [IRVoteSystem](#) class, conducting and determining the winner.

```
#include <IRProcessing.h>
```

Inheritance diagram for IRProcessing:



Collaboration diagram for IRProcessing:



Public Member Functions

- [IRProcessing](#) ()
Construct a new [IRProcessing](#) object.
- [IRProcessing](#) (FILE *file)
- [IRProcessing](#) ([IRBallot](#) *ballot)
Construct a new [IRProcessing](#) object given the [IRBallot](#) (A class containing the organized information to send to vote election).
- char * [nextLine](#) ()
Helper function to read the next line and return the content separated by ','.
- bool [setUp](#) ()
Function to read the header parts of the .csv file and fill out the necessary information. Does not read the ballots in the .csv file.
- bool [read](#) ()
Function to read all the ballots in the .csv and distribute to each candidate.
- [IRBallot](#) * [output](#) ()
Create an object holding required information for the voting election in order to conduct.
- void [calculate](#) ()
Calculate the map percentage. Each candidate's number of ballots is divided by the total ballots found in the .csv file.
- bool [redistribute](#) ([IRCandidate](#) *cand)
Function to remove all the ballots in a eliminated candidate and distribute the ballot to the next preference candidate.
- int [getNumCandidates](#) () const
Get the number of candidates left in election.
- void [setNumCandidates](#) (int numCandidates_)
Set the number of candidates left in election.
- std::vector< [IRCandidate](#) * > [getCandidates](#) () const
Get the list of IRCandidates.
- void [setCandidates](#) (const std::vector< [IRCandidate](#) * > &candidates_)
Set the list of IRCandidates.
- std::vector< double > [getMapPercentage](#) () const
Get the mapping of percentage.
- void [setMapPercentage](#) (const std::vector< double > &mapPercentage_)
Set the percentage mapping.

Additional Inherited Members

5.13.1 Detailed Description

IR Processing is to process all ballots in a .csv file and prepare an organized set of information to send to the [IRVoteSystem](#) class, conducting and determining the winner.

The processing will also handle redistribution if a candidate is eliminated.

5.13.2 Constructor & Destructor Documentation

5.13.2.1 [IRProcessing](#)() [1/2]

```
IRProcessing::IRProcessing ( ) [inline]
```

Construct a new [IRProcessing](#) object.

Parameters

<i>file</i>	FILE (the .csv file)
-------------	----------------------

5.13.2.2 IRProcessing() [2/2]

```
IRProcessing::IRProcessing (
    IRBallot * ballot )
```

Construct a new [IRProcessing](#) object given the [IRBallot](#) (A class containing the organized information to send to vote election).

Parameters

<i>ballot</i>	IRBallot*
---------------	-----------

5.13.3 Member Function Documentation**5.13.3.1 getCandidates()**

```
std::vector<IRCandidate*> IRProcessing::getCandidates ( ) const [inline]
```

Get the list of IRCandidates.

Returns

```
std::vector<IRCandidate*>
```

5.13.3.2 getMapPercentage()

```
std::vector<double> IRProcessing::getMapPercentage ( ) const [inline]
```

Get the mapping of percentage.

Returns

```
std::vector<double>
```

5.13.3.3 getNumCandidates()

```
int IRProcessing::getNumCandidates ( ) const [inline]
```

Get the number of candidates left in election.

Returns

int

5.13.3.4 nextLine()

```
char * IRProcessing::nextLine ( )
```

Helper function to read the next line and return the content separated by ','.

Returns

char*

5.13.3.5 output()

```
IRBallot * IRProcessing::output ( ) [virtual]
```

Create an object holding required information for the voting election in order to conduct.

Returns

IRBallot* the class containing information.

Implements [IBallotProcessing](#).

5.13.3.6 read()

```
bool IRProcessing::read ( ) [virtual]
```

Function to read all the ballots in the .csv and distribute to each candidate.

Returns

true if the reading was successful
false if the reading failed

Implements [IBallotProcessing](#).

5.13.3.7 redistribute()

```
bool IRProcessing::redistribute (
    IRCandidate * cand )
```

Function to remove all the ballots in a eliminated candidate and distribute the ballot to the next preference candidate.

Parameters

<i>cand</i>	IRCandidate* the eliminated candidate.
-------------	--

Returns

true if redistribution was sucessful.

false if an error has occured.

5.13.3.8 setCandidates()

```
void IRProcessing::setCandidates (
    const std::vector< IRCandidate * > & candidates_ ) [inline]
```

Set the list of IRCandidates.

Parameters

<i>candidates</i> ↔	vector<IRCandidate*>
—	

5.13.3.9 setMapPercentage()

```
void IRProcessing::setMapPercentage (
    const std::vector< double > & mapPercentage_ ) [inline]
```

Set the percentage mapping.

Parameters

<i>map</i> ↔ <i>Percentage_</i>	vector<double>
------------------------------------	----------------

5.13.3.10 setNumCandidates()

```
void IRProcessing::setNumCandidates (
    int numCandidates_ ) [inline]
```

Set the number of candidates left in election.

Parameters

<i>num</i> ↔ <i>Candidates_</i>	int
------------------------------------	-----

5.13.3.11 setUp()

```
bool IRProcessing::setUp ( ) [virtual]
```

Function to read the header parts of the .csv file and fill out the necessarily information. Does not read the ballots in the .csv file.

Returns

true if the set up was suceessful
false if the set up fail

Implements [IBallotProcessing](#).

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

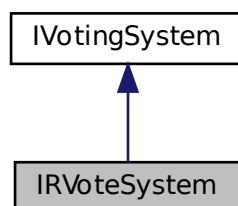
- [src/include/IRProcessing.h](#)
- [src/IRProcessing.cpp](#)

5.14 IRVoteSystem Class Reference

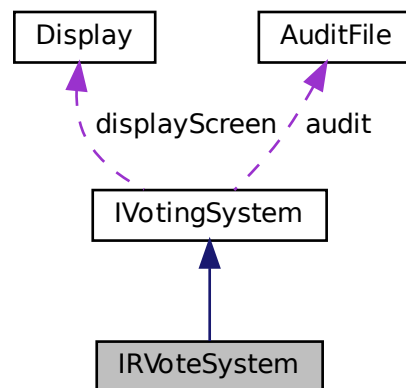
The voting system to conduct the election and determine a winning candidate.

```
#include <IRVoteSystem.h>
```

Inheritance diagram for IRVoteSystem:



Collaboration diagram for IRVoteSystem:



Public Member Functions

- [IRVoteSystem](#) ()
Construct a new [IRVoteSystem](#) object.
- [IRVoteSystem](#) ([IRProcessing](#) *ballotSystem_, [IRBallot](#) *ballot)
Construct a new [IRVoteSystem](#) object given the [IR Ballot](#) Processing system and the [IR Ballot](#).
- bool [startElection](#) ()
Start the election process and determine a winner.
- bool [conductElection](#) ()
Helper function for [startElection\(\)](#) where it compares the percentage mapping and if there's a majority winner.
- std::vector< [IRCandidate](#) * > [getElimination](#) ()
Get the eliminated candidate. It is the candidate with the lowest amount of ballots.
- [IRBallot](#) * [getProcessedBallot](#) () const
Get the information used for the vote election.
- void [setProcessedBallot](#) ([IRBallot](#) *processedBallot_)
Set the information used for the vote election.
- [IRProcessing](#) * [getBallotSystem](#) () const
Get the ballot processing system used for the vote election.
- void [setBallotSystem](#) ([IRProcessing](#) *ballotSystem_)
Set the ballot processing system used for the vote election.
- [IRCandidate](#) [getWinner](#) () const
Get the winner.
- void [setWinner](#) (const [IRCandidate](#) &winner_)
Set the winner.
- std::vector< [IRCandidate](#) * > [getCandidates](#) () const
Get the list of Candidates in the election.
- void [setCandidates](#) (const std::vector< [IRCandidate](#) * > &candidates_)
Set the list of Candidates in the election.

Additional Inherited Members

5.14.1 Detailed Description

The voting system to conduct the election and determine a winning candidate.

For redistribution, the IR Vote System will use the ballot processing system associating with it.

5.14.2 Constructor & Destructor Documentation

5.14.2.1 IRVoteSystem() [1/2]

```
IRVoteSystem::IRVoteSystem ( ) [inline]
```

Construct a new [IRVoteSystem](#) object.

5.14.2.2 IRVoteSystem() [2/2]

```
IRVoteSystem::IRVoteSystem (
    IRProcessing * ballotSystem_,
    IRBallot * ballot )
```

Construct a new [IRVoteSystem](#) object given the IR [Ballot](#) Processing system and the IR [Ballot](#).

Parameters

<i>ballotSystem_</i>	IRProcessing* the processing system used in order to redistribute
<i>ballot</i>	IRBallot* contains all the information to conduct the election

5.14.3 Member Function Documentation

5.14.3.1 conductElection()

```
bool IRVoteSystem::conductElection ( ) [virtual]
```

Helper function for [startElection\(\)](#) where it compares the percentage mapping and if there's a majority winner.

Returns

true if the winner has been decided
false if somehow the winner has not been decided

Implements [IVotingSystem](#).

5.14.3.2 getBallotSystem()

```
IRProcessing* IRVoteSystem::getBallotSystem ( ) const [inline]
```

Get the ballot processing system used for the vote election.

Returns

IRProcessing*

5.14.3.3 getCandidates()

```
std::vector<IRCandidate*> IRVoteSystem::getCandidates ( ) const [inline]
```

Get the list of Candidates in the election.

Returns

std::vector<IRCandidate*>

5.14.3.4 getElimnation()

```
std::vector< IRCandidate * > IRVoteSystem::getElimnation ( )
```

Get the eliminated candidate. It is the candidate with the lowest amount of ballots.

It is a vector incase there is more than 1 candidate with the same amount of lowest ballots.

Returns

std::vector<IRCandidate*> List of candidate with the lowest amount of ballots. Often times, there's only 1 candidate.

5.14.3.5 getProcessedBallot()

```
IRBallot* IRVoteSystem::getProcessedBallot ( ) const [inline]
```

Get the information used for the vote election.

Returns

IRBallot*

5.14.3.6 getWinner()

```
IRCandidate IRVoteSystem::getWinner ( ) const [inline]
```

Get the winner.

Returns

IRCandidate the winner

5.14.3.7 setBallotSystem()

```
void IRVoteSystem::setBallotSystem (
    IRProcessing * ballotSystem_ ) [inline]
```

Set the ballot processing system used for the vote election.

Parameters

<i>ballot↔ System_</i>	IRProcessing*
----------------------------	---------------

5.14.3.8 setCandidates()

```
void IRVoteSystem::setCandidates (
    const std::vector< IRCandidate * > & candidates_ ) [inline]
```

Set the list of Candidates in the election.

Parameters

<i>candidates↔ _</i>	vector<IRCandidate*>
--------------------------	----------------------

5.14.3.9 setProcessedBallot()

```
void IRVoteSystem::setProcessedBallot (
    IRBallot * processedBallot_ ) [inline]
```

Set the information used for the vote election.

Parameters

<i>processedBallot_</i>	IRBallot*
-------------------------	-----------

5.14.3.10 setWinner()

```
void IRVoteSystem::setWinner (
    const IRCandidate & winner_ ) [inline]
```

Set the winner.

Parameters

<i>winner_</i>	IRCandidate
----------------	-------------

5.14.3.11 startElection()

```
bool IRVoteSystem::startElection ( ) [virtual]
```

Start the election process and determine a winner.

Returns

true if the winner has been decided
false if somehow the winner has not been decided

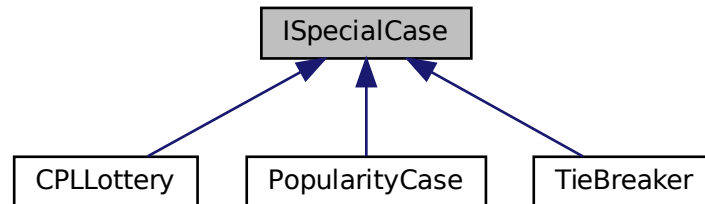
Implements [IVotingSystem](#).

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

- [src/include/IRVoteSystem.h](#)
- [src/IRVoteSystem.cpp](#)

5.15 ISpecialCase Class Reference

Inheritance diagram for ISpecialCase:



Public Member Functions

- virtual int **run** ()=0

Protected Attributes

- std::vector< int > **mapBiasTracker**

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

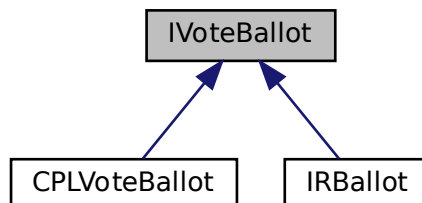
- src/include/ISpecialCase.h

5.16 IVoteBallot Class Reference

The abstract class where all VoteBallots are inherited from.

```
#include <IVoteBallot.h>
```

Inheritance diagram for IVoteBallot:



Public Member Functions

- `std::vector< int > getMapBallot () const`
get the mapping of ballots to <candidates / parties / ...>
- `void setMapBallot (const std::vector< int > &mapBallot_)`
Set the mapping of ballots to <candidates / parties / ...>
- `int getTotalBallot () const`
Get the total amount of ballots.
- `void setTotalBallot (int totalBallot_)`
Set the total amount of ballots.

Protected Attributes

- `std::vector< int > mapBallot`
Holds the mapping of ballots.
- `int totalBallot`
total amount of ballots

5.16.1 Detailed Description

The abstract class where all VoteBallots are inherited from.

VoteBallots contains all necessarily information and is produced by a [Ballot](#) Processing class.

VoteBallots are used in its pertaining Vote Election to conduct an election

5.16.2 Member Function Documentation

5.16.2.1 [getMapBallot\(\)](#)

```
std::vector<int> IVoteBallot::getMapBallot ( ) const [inline]
```

get the mapping of ballots to <candidates / parties / ...>

Returns

`std::vector<int>`

5.16.2.2 [getTotalBallot\(\)](#)

```
int IVoteBallot::getTotalBallot ( ) const [inline]
```

Get the total amount of ballots.

Returns

`int`

5.16.2.3 [setMapBallot\(\)](#)

```
void IVoteBallot::setMapBallot (
    const std::vector< int > & mapBallot_ ) [inline]
```

Set the mapping of ballots to <candidates / parties / ...>

Parameters

<i>map</i> ↔ <i>Ballot_</i>	
--------------------------------	--

5.16.2.4 setTotalBallot()

```
void IVoteBallot::setTotalBallot (
    int totalBallot_ ) [inline]
```

Set the total amount of ballots.

Parameters

<i>total</i> ↔ <i>Ballot_</i>	int
----------------------------------	-----

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

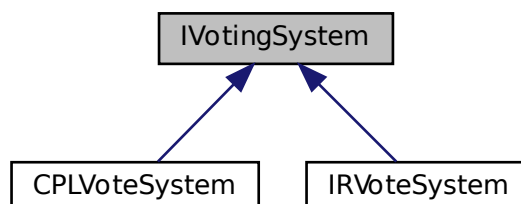
- [src/include/IVoteBallot.h](#)

5.17 IVotingSystem Class Reference

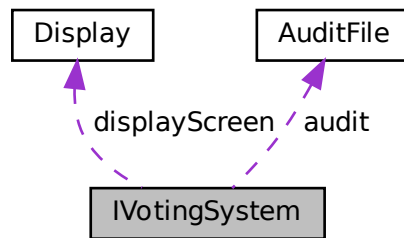
The abstract class for each voting election system will inherit from.

```
#include <IVotingSystem.h>
```

Inheritance diagram for IVotingSystem:



Collaboration diagram for IVotingSystem:



Public Member Functions

- virtual bool [startElection](#) ()=0
To start the election process and handle special cases.
- virtual bool [conductElection](#) ()=0
A helper function to startElection. Is used to determine a winner.
- [AuditFile](#) * [getAuditFile](#) () const
Get the Audit File object.
- void [setAuditFile](#) ([AuditFile](#) *auditFile_)
Set the Audit File object.
- [Display](#) * [getDisplayScreen](#) () const
Get the [Display](#) Screen object.
- void [setDisplayScreen](#) ([Display](#) *displayScreen_)
Set the [Display](#) Screen object.
- int [getStatus](#) () const
get the status of the election.
- void [setStatus](#) (int status_)
Set the status of the election. 1 if complete. -1 if incomplete.
- int [getAuditing](#) () const
Get the write to audit option int.
- void [setAuditing](#) (int auditing_)
Set the write to audit option int.
- vector< [ISpecialCase](#) * > [getSpecialCase](#) () const
Get the Special Cases vector.
- void [setSpecialCase](#) (const vector< [ISpecialCase](#) * > &specialCase_)
Set the Special Cases vector.

Protected Attributes

- vector< [ISpecialCase](#) * > [specialCase](#)
Hold the methods of special cases handler.
- [AuditFile](#) * [audit](#)
the audit file that will be produced

- [Display](#) * [displayScreen](#)
the display object
- int [status](#) = -1
1 if election is complete and winner is found.
- int [auditing](#) = 0
1(true) if should write to audit. 0(False) to avoid audit.

5.17.1 Detailed Description

The abstract class for each voting election system will inherit from.

5.17.2 Member Function Documentation

5.17.2.1 [conductElection\(\)](#)

```
virtual bool IVotingSystem::conductElection ( ) [pure virtual]
```

A helper function to startElection. Is used to determine a winner.

Returns

true if winner was found.
false

Implemented in [IRVoteSystem](#), and [CPLVoteSystem](#).

5.17.2.2 [getAuditFile\(\)](#)

```
AuditFile* IVotingSystem::getAuditFile ( ) const [inline]
```

Get the Audit File object.

Returns

[AuditFile](#)*

5.17.2.3 getAuditing()

```
int IVotingSystem::getAuditing ( ) const [inline]
```

Get the write to audit option int.

Returns

int

5.17.2.4 getDisplayScreen()

```
Display* IVotingSystem::getDisplayScreen ( ) const [inline]
```

Get the [Display](#) Screen object.

Returns

Display*

5.17.2.5 getSpecialCase()

```
vector<ISpecialCase*> IVotingSystem::getSpecialCase ( ) const [inline]
```

Get the Special Cases vector.

Returns

vector<ISpecialCase*>

5.17.2.6 getStatus()

```
int IVotingSystem::getStatus ( ) const [inline]
```

get the status of the election.

Returns

int

5.17.2.7 setAuditFile()

```
void IVotingSystem::setAuditFile (
    AuditFile * auditFile_ ) [inline]
```

Set the Audit File object.

Parameters

<i>audit</i> ↔ <i>File_</i>	AuditFile
--------------------------------	---------------------------

5.17.2.8 setAuditing()

```
void IVotingSystem::setAuditing (
    int auditing_ ) [inline]
```

Set the write to audit option int.

Parameters

<i>auditing</i> ↔ _	int
------------------------	-----

5.17.2.9 setDisplayScreen()

```
void IVotingSystem::setDisplayScreen (
    Display * displayScreen_ ) [inline]
```

Set the [Display](#) Screen object.

Parameters

<i>display</i> ↔ <i>Screen_</i>	Display
------------------------------------	-------------------------

5.17.2.10 setSpecialCase()

```
void IVotingSystem::setSpecialCase (
    const vector< ISpecialCase * > & specialCase_ ) [inline]
```

Set the Special Cases vector.

Parameters

<i>special</i> ↔ <i>Case_</i>	
----------------------------------	--

5.17.2.11 setStatus()

```
void IVotingSystem::setStatus (
    int status_ ) [inline]
```

Set the status of the election. 1 if complete. -1 if incomplete.

Parameters

<i>status</i> ↔	int
—	

5.17.2.12 startElection()

```
virtual bool IVotingSystem::startElection ( ) [pure virtual]
```

To start the election process and handle special cases.

Returns

true if successful.

false

Implemented in [IRVoteSystem](#), and [CPLVoteSystem](#).

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

- src/include/IVotingSystem.h

5.18 Party Class Reference

A class representing a political party with candidate information and seat limits.

```
#include <Party.h>
```

Public Member Functions

- [Party](#) (vector< [Candidate](#) > candidates, int maxSeat, string name)
Construct a new [Party](#) object with the given candidates, maximum number of seats, and name.
- [Party](#) ()
Construct a new, empty [Party](#) object.
- string [getName](#) ()
Get the name of the party.
- void [setName](#) (string newName)
Set the name of the party.
- vector< [Candidate](#) > [getCandidate](#) ()
Get the vector of [Candidate](#) objects representing the candidates in the party.
- void [setMaxSeat](#) (int newMaxSeat)
Set the maximum number of seats the party can have.
- void [setCandidate](#) ([Candidate](#) candidate)
Add a new [Candidate](#) to the party.
- int [getMaxSeat](#) ()
Get the maximum number of seats the party can have.

Protected Attributes

- vector< [Candidate](#) > **candidates**
- int **maxSeat**
- string **name**

5.18.1 Detailed Description

A class representing a political party with candidate information and seat limits.

This class stores a vector of [Candidate](#) objects, an integer maximum number of seats, and a string name for the party. It provides methods to get and set these values, as well as add candidates to the party.

5.18.2 Constructor & Destructor Documentation

5.18.2.1 Party() [1/2]

```
Party::Party (
    vector< Candidate > candidates,
    int maxSeat,
    string name )
```

Construct a new [Party](#) object with the given candidates, maximum number of seats, and name.

Parameters

<i>candidates</i>	A vector of Candidate objects representing the candidates in the party.
<i>maxSeat</i>	An integer representing the maximum number of seats the party can have.
<i>name</i>	A string representing the name of the party.

5.18.2.2 Party() [2/2]

```
Party::Party ( )
```

Construct a new, empty [Party](#) object.

5.18.3 Member Function Documentation

5.18.3.1 getCandidate()

```
vector<Candidate> Party::getCandidate ( ) [inline]
```

Get the vector of [Candidate](#) objects representing the candidates in the party.

Returns

A vector of [Candidate](#) objects representing the candidates in the party.

5.18.3.2 getMaxSeat()

```
int Party::getMaxSeat ( ) [inline]
```

Get the maximum number of seats the party can have.

Returns

An integer representing the maximum number of seats the party can have.

5.18.3.3 getName()

```
string Party::getName ( ) [inline]
```

Get the name of the party.

Returns

A string representing the name of the party.

5.18.3.4 setCandidate()

```
void Party::setCandidate (
    Candidate candidate ) [inline]
```

Add a new [Candidate](#) to the party.

Parameters

<i>candidate</i>	A Candidate object representing the new candidate to add to the party.
------------------	--

5.18.3.5 setMaxSeat()

```
void Party::setMaxSeat (
    int newMaxSeat ) [inline]
```

Set the maximum number of seats the party can have.

Parameters

<i>newMaxSeat</i>	An integer representing the new maximum number of seats for the party.
-------------------	--

5.18.3.6 setName()

```
void Party::setName (
    string newName ) [inline]
```

Set the name of the party.

Parameters

<i>newName</i>	A string representing the new name of the party.
----------------	--

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

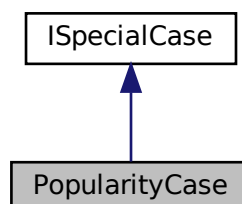
- src/include/[Party.h](#)
- src/Party.cpp

5.19 PopularityCase Class Reference

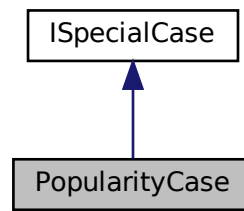
A class representing a popularity special case implementation.

```
#include <PopularityCase.h>
```

Inheritance diagram for PopularityCase:



Collaboration diagram for PopularityCase:



Public Member Functions

- `PopularityCase` (`std::vector< IRCandidate * >` candidates)
Construct a new `PopularityCase` object with the given vector of `IRCandidate` pointers.
- `int run ()`
Run the popularity special case with the current set of candidates.
- `int helper ()`
A helper method used internally by the class.
- `vector< IRCandidate * > getCandidates () const`
Get the current set of candidates.
- `void setCandidates (const vector< IRCandidate * > &candidates_)`
Set the current set of candidates to the given vector of `IRCandidate` pointers.

Additional Inherited Members

5.19.1 Detailed Description

A class representing a popularity special case implementation.

This class inherits from the `ISpecialCase` interface and provides an implementation for the `run()` method. It takes a vector of `IRCandidate` pointers as input and stores it in a private member variable. It also provides a `helper()` method and getter/setter methods for the candidates member variable.

5.19.2 Constructor & Destructor Documentation

5.19.2.1 PopularityCase()

```
PopularityCase::PopularityCase (
    std::vector< IRCandidate * > candidates )
```

Construct a new `PopularityCase` object with the given vector of `IRCandidate` pointers.

Parameters

<i>candidates</i>	A vector of IRCandidate pointers.
-------------------	---

5.19.3 Member Function Documentation

5.19.3.1 getCandidates()

```
vector<IRCandidate*> PopularityCase::getCandidates ( ) const [inline]
```

Get the current set of candidates.

Returns

A vector of [IRCandidate](#) pointers.

5.19.3.2 helper()

```
int PopularityCase::helper ( )
```

A helper method used internally by the class.

Returns

An integer representing the result of the helper method.

5.19.3.3 run()

```
int PopularityCase::run ( ) [virtual]
```

Run the popularity special case with the current set of candidates.

Returns

An integer representing the result of the popularity special case.

Implements [ISpecialCase](#).

5.19.3.4 setCandidates()

```
void PopularityCase::setCandidates (
    const vector< IRCandidate * > & candidates_ ) [inline]
```

Set the current set of candidates to the given vector of [IRCandidate](#) pointers.

Parameters

<i>candidates</i> ↔	A vector of IRCandidate pointers.
—	

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

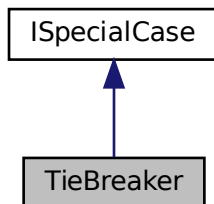
- `src/include/PopularityCase.h`
- `src/PopularityCase.cpp`

5.20 TieBreaker Class Reference

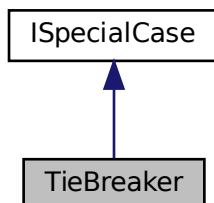
A class representing a tiebreaker special case implementation.

```
#include <TieBreaker.h>
```

Inheritance diagram for TieBreaker:



Collaboration diagram for TieBreaker:



Public Member Functions

- [TieBreaker](#) ()
Construct a new [TieBreaker](#) object.
- int [run](#) ()
Run the tiebreaker special case with default parameters.
- int [run](#) (int size)
Run the tiebreaker special case with the given size parameter.

Additional Inherited Members

5.20.1 Detailed Description

A class representing a tiebreaker special case implementation.

This class inherits from the [ISpecialCase](#) interface and provides implementations for the [run\(\)](#) and [run\(int\)](#) methods. It also has a default constructor.

5.20.2 Constructor & Destructor Documentation

5.20.2.1 TieBreaker()

```
TieBreaker::TieBreaker ( ) [inline]
```

Construct a new [TieBreaker](#) object.

5.20.3 Member Function Documentation

5.20.3.1 run() [1/2]

```
int TieBreaker::run ( ) [virtual]
```

Run the tiebreaker special case with default parameters.

Returns

An integer representing the result of the tiebreaker.

Implements [ISpecialCase](#).

5.20.3.2 run() [2/2]

```
int TieBreaker::run (  
    int size )
```

Run the tiebreaker special case with the given size parameter.

Parameters

<i>size</i>	An integer representing the size parameter for the tiebreaker.
-------------	--

Returns

An integer representing the result of the tiebreaker.

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

- src/include/TieBreaker.h
- src/TieBreaker.cpp

Chapter 6

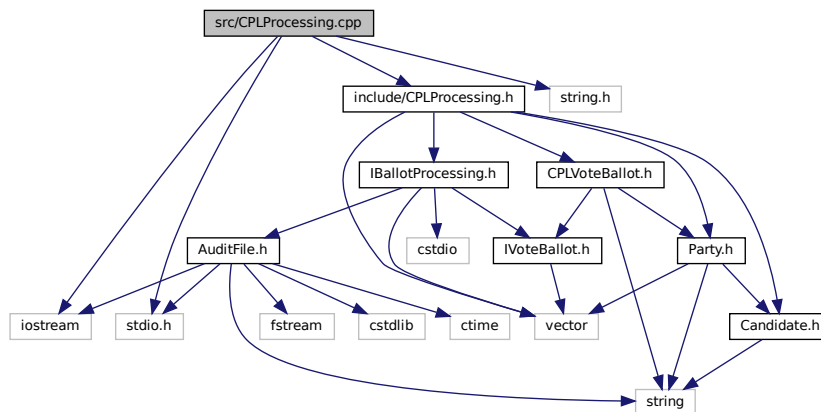
File Documentation

6.1 src/CPLProcessing.cpp File Reference

Process CPL ballot file and save the information of parties, candidates and seats.

```
#include <iostream>
#include <stdio.h>
#include <string.h>
#include "include/CPLProcessing.h"
```

Include dependency graph for CPLProcessing.cpp:



6.1.1 Detailed Description

Process CPL ballot file and save the information of parties, candidates and seats.

Implementation of the functions of [CPLProcessing](#)

Author

Wenjing Jiang

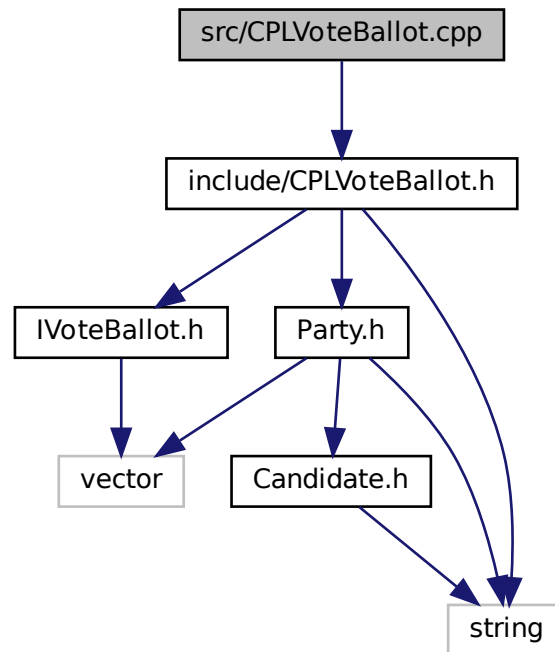
Bug No known bugs.

6.2 src/CPLVoteBallot.cpp File Reference

Initiate [CPLVoteBallot](#).

```
#include "include/CPLVoteBallot.h"
```

Include dependency graph for CPLVoteBallot.cpp:



6.2.1 Detailed Description

Initiate [CPLVoteBallot](#).

Author

Wenjing Jiang

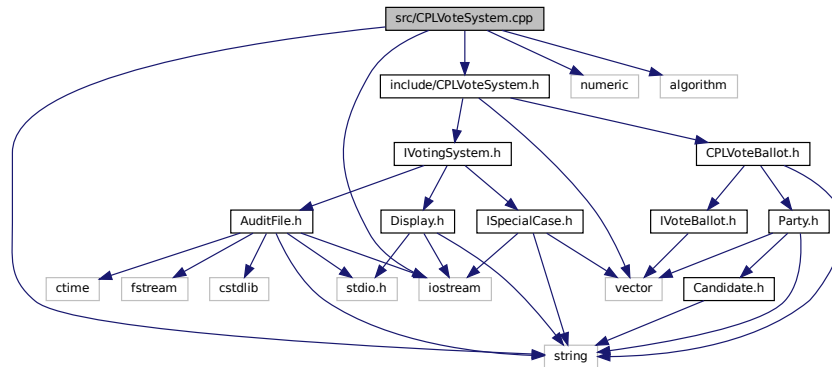
Bug No known bugs.

6.3 src/CPLVoteSystem.cpp File Reference

Conduct election according to CPL rules and show the winners.

```
#include <iostream>
#include <string>
```

```
#include <numeric>
#include <algorithm>
#include "include/CPLVoteSystem.h"
Include dependency graph for CPLVoteSystem.cpp:
```



6.3.1 Detailed Description

Conduct election according to CPL rules and show the winners.

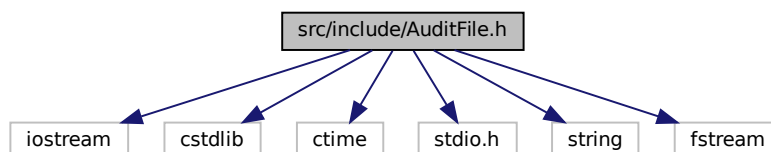
Author

Wenjing Jiang

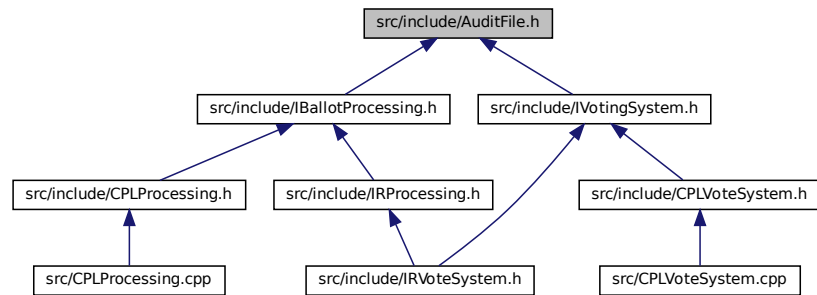
Bug No known bugs.

6.4 src/include/AuditFile.h File Reference

```
#include <iostream>
#include <cstdlib>
#include <ctime>
#include <stdio.h>
#include <string>
#include <fstream>
Include dependency graph for AuditFile.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [AuditFile](#)

The [AuditFile](#) class represents a file that logs information about a voting audit.

6.4.1 Detailed Description

Author

Matin Horri (horri031@umn.edu)

Version

0.1

Date

2023-03-25

Copyright

Copyright (c) 2023

6.5 src/include/Ballot.h File Reference

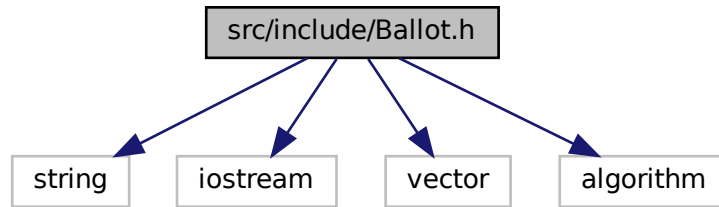
```

#include <string>
#include <iostream>
#include <vector>

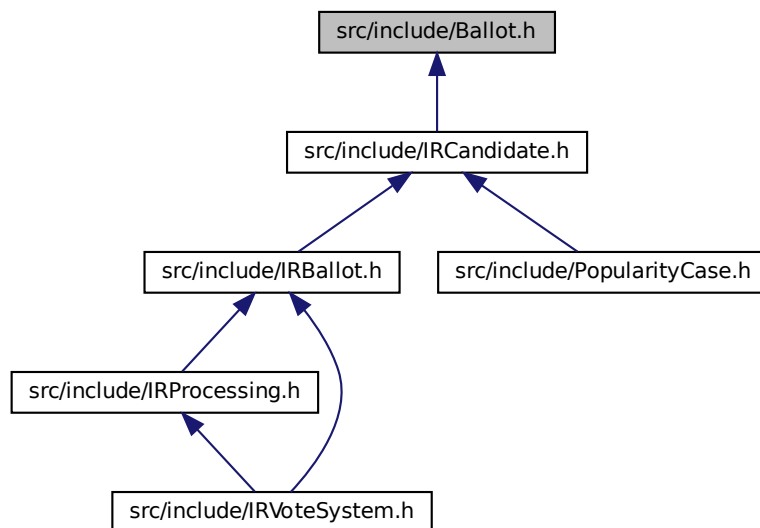
```

```
#include <algorithm>
```

Include dependency graph for Ballot.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [Ballot](#)

A [Ballot](#) class to hold an individual ballot found in a Instant Runoff election. Each IR [Candidate](#) will hold a list of this [Ballot](#).

6.5.1 Detailed Description

Author

your name ([you@domain.com](#))

Version

0.1

Date

2023-03-25

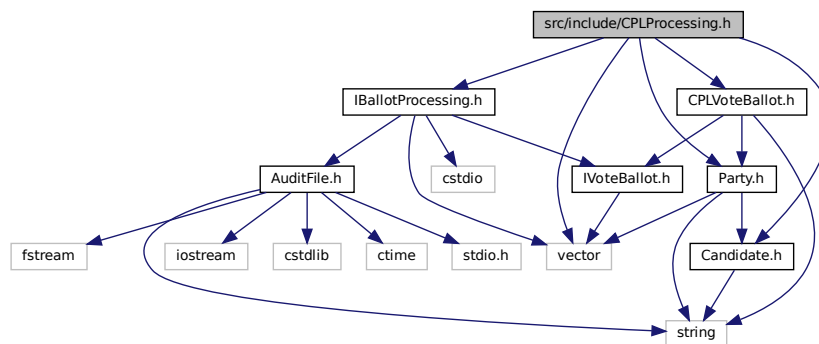
Copyright

Copyright (c) 2023

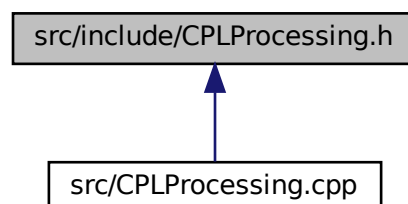
6.6 src/include/CPLProcessing.h File Reference

Process CPL ballot file and save the information of parties, candidates and seats.

```
#include <vector>
#include "IBallotProcessing.h"
#include "Party.h"
#include "Candidate.h"
#include "CPLVoteBallot.h"
Include dependency graph for CPLProcessing.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [CPLProcessing](#)

6.6.1 Detailed Description

Process CPL ballot file and save the information of parties, candidates and seats.

Author

Wenjing Jiang

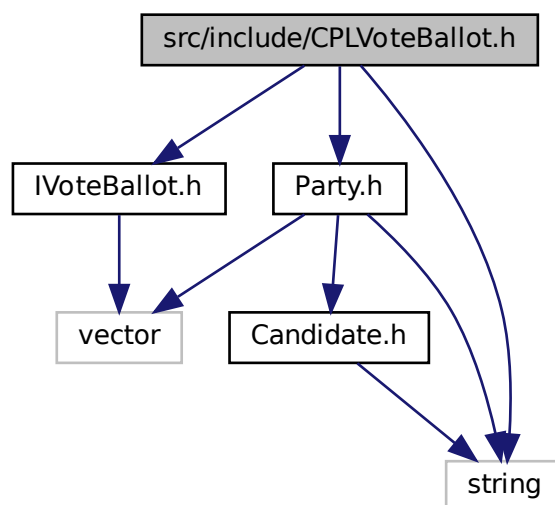
Bug No known bugs.

6.7 src/include/CPLVoteBallot.h File Reference

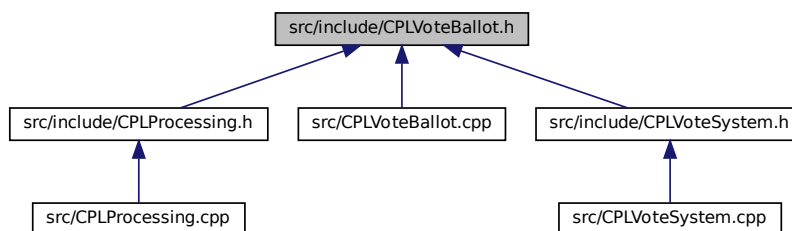
[CPLVoteBallot](#) saves information of election.

```
#include "IVoteBallot.h"
#include "Party.h"
#include <string>
```

Include dependency graph for CPLVoteBallot.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [CPLVoteBallot](#)

6.7.1 Detailed Description

[CPLVoteBallot](#) saves information of election.

Author

Wenjing Jiang (you@domain.com)

Version

0.1

Date

2023-03-26

Copyright

Copyright (c) 2023

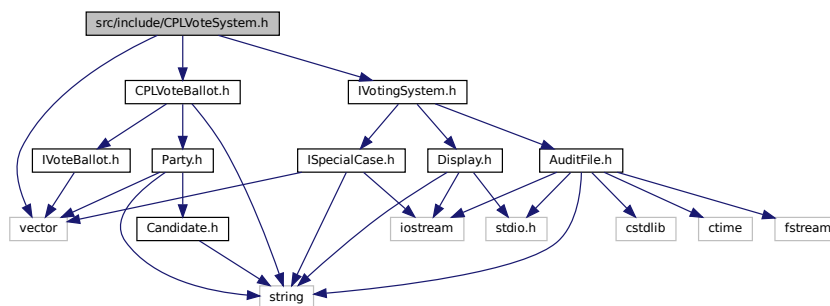
6.8 src/include/CPLVoteSystem.h File Reference

Conduct election and show winners.

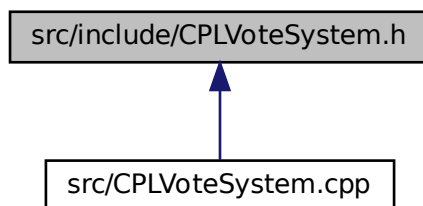
```
#include "CPLVoteBallot.h"
#include "IVotingSystem.h"
```

```
#include <vector>
```

Include dependency graph for CPLVoteSystem.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [CPLVoteSystem](#)

6.8.1 Detailed Description

Conduct election and show winners.

Author

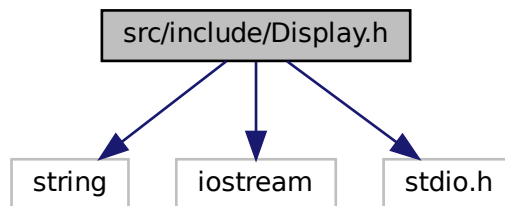
Wenjing Jiang

Bug No known bugs.

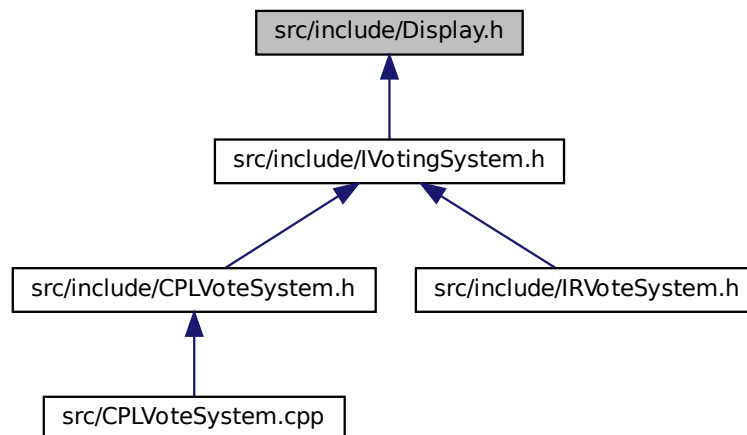
6.9 src/include/Display.h File Reference

```
#include <string>
#include <iostream>
#include <stdio.h>
```

Include dependency graph for Display.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [Display](#)

The [Display](#) class represents a display that can be used to output information.

6.9.1 Detailed Description

Author

Matin Horri (horri031@umn.edu)

Version

0.1

Date

2023-03-25

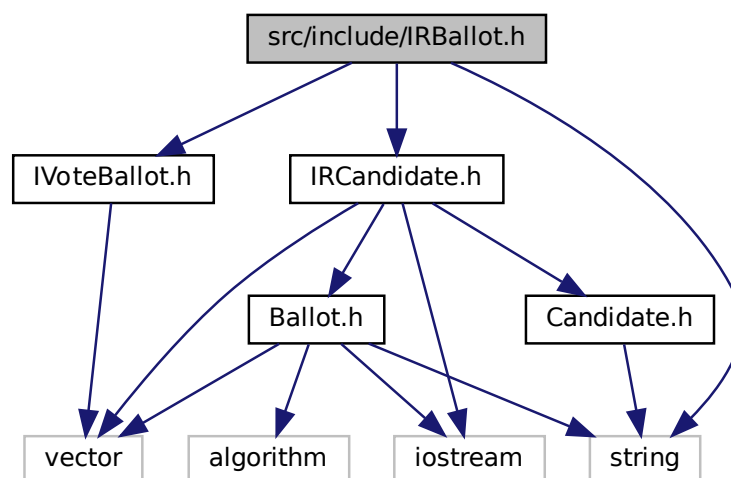
Copyright

Copyright (c) 2023

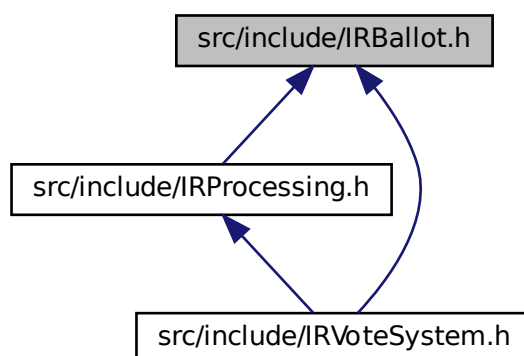
6.10 src/include/IRBallot.h File Reference

```
#include "IVoteBallot.h"  
#include "IRCandidate.h"  
#include <string>
```

Include dependency graph for IRBallot.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [IRBallot](#)

An output class from the IR Processing class. This class holds objects/information for the IR Vote System to conduct its election and determine who's the winner.

6.10.1 Detailed Description

Author

Michael Vang (vang2891@umn.edu)

Version

0.1

Date

2023-03-24

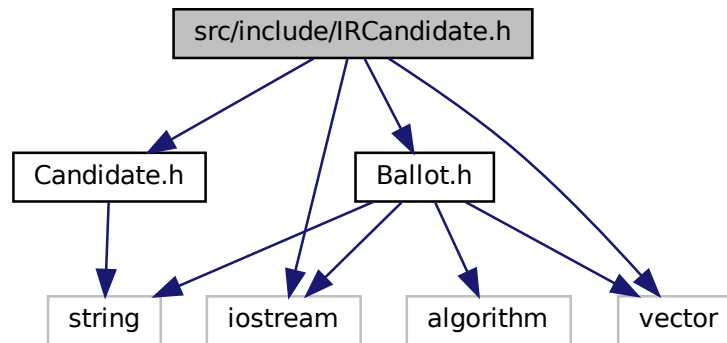
Copyright

Copyright (c) 2023

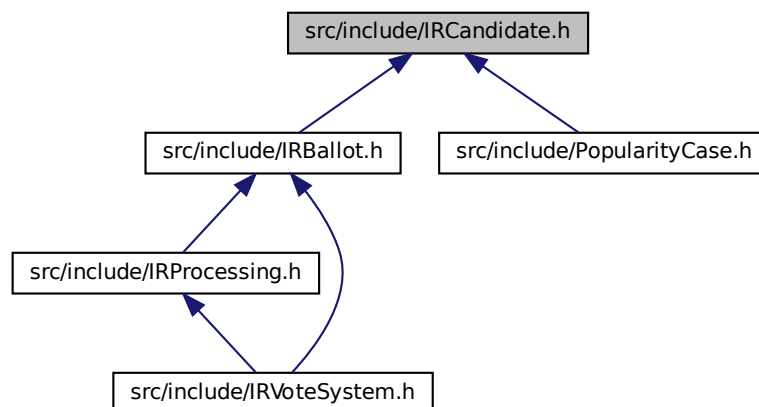
6.11 src/include/IRCandidate.h File Reference

```
#include "Ballot.h"  
#include "Candidate.h"  
#include <iostream>  
#include <vector>
```

Include dependency graph for IRCandidate.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [IRCandidate](#)

A candidate class for an Instant Runoff [Candidate](#). This class is for holding ballots.

6.11.1 Detailed Description

Author

your name (you@domain.com)

Version

0.1

Date

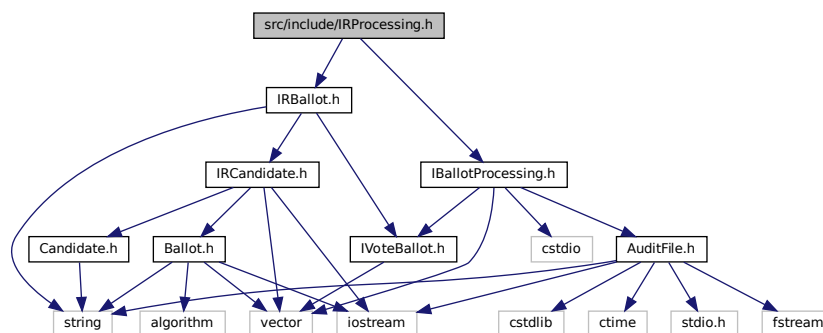
2023-03-24

Copyright

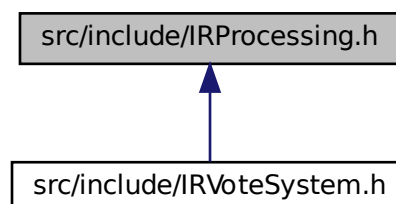
Copyright (c) 2023

6.12 src/include/IRProcessing.h File Reference

```
#include "IRBallot.h"
#include "IBallotProcessing.h"
Include dependency graph for IRProcessing.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [IRProcessing](#)

IR Processing is to process all ballots in a .csv file and prepare an organized set of information to send to the [IRVoteSystem](#) class, conducting and determining the winner.

6.12.1 Detailed Description

Author

Michael Vang (vang2891@umn.edu)

Version

0.1

Date

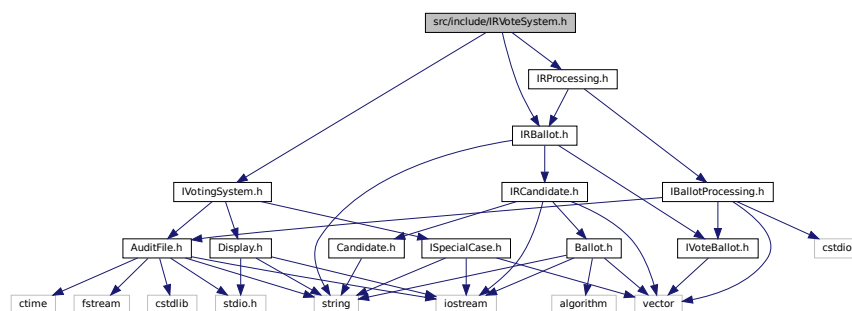
2023-03-24

Copyright

Copyright (c) 2023

6.13 src/include/IRVoteSystem.h File Reference

```
#include "IVotingSystem.h"
#include "IRBallot.h"
#include "IRProcessing.h"
Include dependency graph for IRVoteSystem.h:
```



Classes

- class [IRVoteSystem](#)

The voting system to conduct the election and determine a winning candidate.

6.13.1 Detailed Description

Author

Michael Vang (vang2891@umn.edu)

Version

0.1

Date

2023-03-24

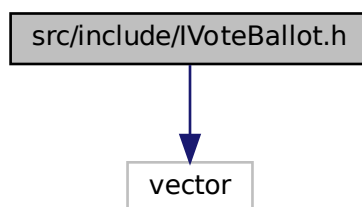
Copyright

Copyright (c) 2023

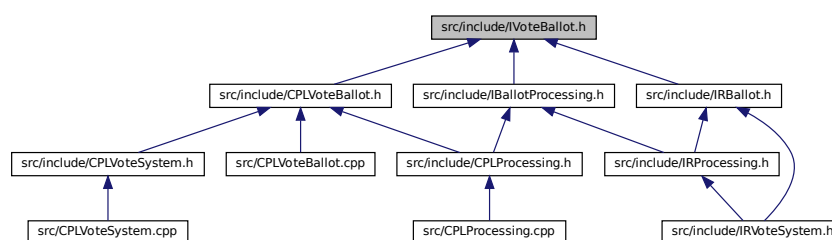
6.14 src/include/IVoteBallot.h File Reference

```
#include <vector>
```

Include dependency graph for IVoteBallot.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [IVoteBallot](#)

The abstract class where all VoteBallots are inherited from.

6.14.1 Detailed Description

Author

Michael Vang (vang2891@umn.edu)

Version

0.1

Date

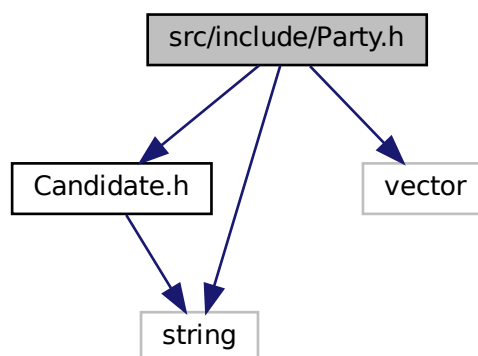
2023-03-25

Copyright

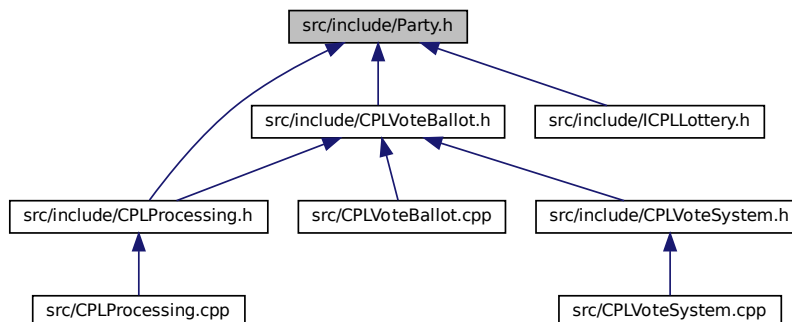
Copyright (c) 2023

6.15 src/include/Party.h File Reference

```
#include "Candidate.h"
#include <string>
#include <vector>
Include dependency graph for Party.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [Party](#)

A class representing a political party with candidate information and seat limits.

6.15.1 Detailed Description

Author

Matin Horri (horri031@umn.edu)

Version

0.1

Date

2023-03-25

Copyright

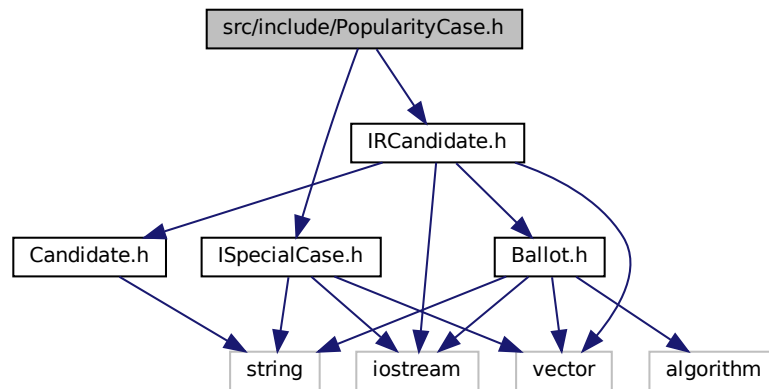
Copyright (c) 2023

6.16 src/include/PopularityCase.h File Reference

```
#include "ISpecialCase.h"
```

```
#include "IRCandidate.h"
```

Include dependency graph for PopularityCase.h:



Classes

- class [PopularityCase](#)
A class representing a popularity special case implementation.

6.16.1 Detailed Description

Author

Matin Horri (horri031@umn.edu)

Version

0.1

Date

2023-03-25

Copyright

Copyright (c) 2023

Index

- addBallot
 - IRCandidate, 37
- AuditFile, 9
 - AuditFile, 10
 - getFile, 10
 - labelFile, 10
 - setFile, 10
 - setOutputResult, 11
 - write, 11
- Ballot, 11
 - Ballot, 12
 - getIndex, 13
 - getMapping, 13
 - getRank, 13
 - setMapping, 13
 - setRank, 14
- Candidate, 14
- conductElection
 - CPLVoteSystem, 27
 - IRVoteSystem, 46
 - IVotingSystem, 54
- CPLLottery, 15
- CPLProcessing, 16
 - CPLProcessing, 17
 - output, 18
 - read, 18
 - setUp, 18
- CPLVoteBallot, 19
 - CPLVoteBallot, 20
 - getMapAllocatedSeat, 21
 - getMapBallot, 21
 - getMapRemainSeat, 21
 - getNumParties, 22
 - getParties, 22
 - getQuota, 22
 - getSeats, 23
 - setMapAllocatedSeat, 23
 - setMapRemainSeat, 23
 - setParties, 25
 - setSeats, 25
- CPLVoteSystem, 26
 - conductElection, 27
 - CPLVoteSystem, 27
 - getWinner, 27
 - startElection, 28
- Display, 28
 - Display, 29
- getOutputTerminal, 29
- overWrite, 29
- print, 30
- setOutputTerminal, 30
- write, 30
- getAuditFile
 - IVotingSystem, 54
- getAuditing
 - IVotingSystem, 54
- getBallotList
 - IRCandidate, 38
- getBallotSystem
 - IRVoteSystem, 47
- getCandidate
 - Party, 58
- getCandidates
 - IRBallot, 34
 - IRProcessing, 41
 - IRVoteSystem, 47
 - PopularityCase, 62
- getDisplayScreen
 - IVotingSystem, 55
- getElimination
 - IRVoteSystem, 47
- getFile
 - AuditFile, 10
- getIndex
 - Ballot, 13
- getMapAllocatedSeat
 - CPLVoteBallot, 21
- getMapBallot
 - CPLVoteBallot, 21
 - IVoteBallot, 51
- getMapPercentage
 - IRBallot, 34
 - IRProcessing, 41
- getMapping
 - Ballot, 13
- getMapRemainSeat
 - CPLVoteBallot, 21
- getMaxSeat
 - Party, 59
- getName
 - Party, 59
- getNumBallots
 - IRCandidate, 38
- getNumCandidates
 - IRBallot, 34
 - IRProcessing, 41

- getNumParties
 - CPLVoteBallot, 22
- getOutputTerminal
 - Display, 29
- getParties
 - CPLVoteBallot, 22
- getProcessedBallot
 - IRVoteSystem, 47
- getQuota
 - CPLVoteBallot, 22
- getRank
 - Ballot, 13
- getSeats
 - CPLVoteBallot, 23
- getSpecialCase
 - IVotingSystem, 55
- getStatus
 - IVotingSystem, 55
- getTotalBallot
 - IVoteBallot, 51
- getWinner
 - CPLVoteSystem, 27
 - IRVoteSystem, 48
- helper
 - PopularityCase, 62
- IBallotProcessing, 31
- IElectionStratgey, 32
- IRBallot, 32
 - getCandidates, 34
 - getMapPercentage, 34
 - getNumCandidates, 34
 - IRBallot, 33
 - setCandidates, 34
 - setMapPercentage, 35
 - setNumCandidates, 35
- IRCandidate, 35
 - addBallot, 37
 - getBallotList, 38
 - getNumBallots, 38
 - IRCandidate, 37
 - setBallotList, 38
 - setNumBallots, 38
- IRProcessing, 39
 - getCandidates, 41
 - getMapPercentage, 41
 - getNumCandidates, 41
 - IRProcessing, 40, 41
 - nextLine, 42
 - output, 42
 - read, 42
 - redistribute, 42
 - setCandidates, 43
 - setMapPercentage, 43
 - setNumCandidates, 43
 - setUp, 44
- IRVoteSystem, 44
 - conductElection, 46
- getBallotSystem, 47
- getCandidates, 47
- getElimnation, 47
- getProcessedBallot, 47
- getWinner, 48
- IRVoteSystem, 46
 - setBallotSystem, 48
 - setCandidates, 48
 - setProcessedBallot, 49
 - setWinner, 49
 - startElection, 49
- ISpecialCase, 50
- IVoteBallot, 50
 - getMapBallot, 51
 - getTotalBallot, 51
 - setMapBallot, 51
 - setTotalBallot, 52
- IVotingSystem, 52
 - conductElection, 54
 - getAuditFile, 54
 - getAuditing, 54
 - getDisplayScreen, 55
 - getSpecialCase, 55
 - getStatus, 55
 - setAuditFile, 55
 - setAuditing, 56
 - setDisplayScreen, 56
 - setSpecialCase, 56
 - setStatus, 56
 - startElection, 57
- labelFile
 - AuditFile, 10
- nextLine
 - IRProcessing, 42
- output
 - CPLProcessing, 18
 - IRProcessing, 42
- overWrite
 - Display, 29
- Party, 57
 - getCandidate, 58
 - getMaxSeat, 59
 - getName, 59
 - Party, 58
 - setCandidate, 59
 - setMaxSeat, 59
 - setName, 60
- PopularityCase, 60
 - getCandidates, 62
 - helper, 62
 - PopularityCase, 61
 - run, 62
 - setCandidates, 62
- print
 - Display, 30

- read
 - CPLProcessing, 18
 - IRProcessing, 42
- redistribute
 - IRProcessing, 42
- run
 - PopularityCase, 62
 - TieBreaker, 64
- setAuditFile
 - IVotingSystem, 55
- setAuditing
 - IVotingSystem, 56
- setBallotList
 - IRCandidate, 38
- setBallotSystem
 - IRVoteSystem, 48
- setCandidate
 - Party, 59
- setCandidates
 - IRBallot, 34
 - IRProcessing, 43
 - IRVoteSystem, 48
 - PopularityCase, 62
- setDisplayScreen
 - IVotingSystem, 56
- setFile
 - AuditFile, 10
- setMapAllocatedSeat
 - CPLVoteBallot, 23
- setMapBallot
 - IVoteBallot, 51
- setMapPercentage
 - IRBallot, 35
 - IRProcessing, 43
- setMapping
 - Ballot, 13
- setMapRemainSeat
 - CPLVoteBallot, 23
- setMaxSeat
 - Party, 59
- setName
 - Party, 60
- setNumBallots
 - IRCandidate, 38
- setNumCandidates
 - IRBallot, 35
 - IRProcessing, 43
- setOutputResult
 - AuditFile, 11
- setOutputTerminal
 - Display, 30
- setParties
 - CPLVoteBallot, 25
- setProcessedBallot
 - IRVoteSystem, 49
- setRank
 - Ballot, 14
- setSeats
 - CPLVoteBallot, 25
- setSpecialCase
 - IVotingSystem, 56
- setStatus
 - IVotingSystem, 56
- setTotalBallot
 - IVoteBallot, 52
- setUp
 - CPLProcessing, 18
 - IRProcessing, 44
- setWinner
 - IRVoteSystem, 49
- src/CPLProcessing.cpp, 67
- src/CPLVoteBallot.cpp, 68
- src/CPLVoteSystem.cpp, 68
- src/include/AuditFile.h, 69
- src/include/Ballot.h, 70
- src/include/CPLProcessing.h, 72
- src/include/CPLVoteBallot.h, 73
- src/include/CPLVoteSystem.h, 74
- src/include/Display.h, 76
- src/include/IRBallot.h, 77
- src/include/IRCandidate.h, 79
- src/include/IRProcessing.h, 80
- src/include/IRVoteSystem.h, 81
- src/include/IVoteBallot.h, 82
- src/include/Party.h, 83
- src/include/PopularityCase.h, 85
- startElection
 - CPLVoteSystem, 28
 - IRVoteSystem, 49
 - IVotingSystem, 57
- TieBreaker, 63
 - run, 64
 - TieBreaker, 64
- write
 - AuditFile, 11
 - Display, 30