Project Name: Project 1: Voting System **Team#17**

Test Stage: Unit _x_ System __ **Test Date: 11/12/2018**

Test Case ID#: Koo_BallotFile_001 **Name(s) of Testers: Justin Koo**

Test Description: Test method is named BallotFileConstructorDoesNotThrowExceptionsSimpleOPL() located in BallotFileTest.java. It tests/uses the BallotFile constructor.

Automated: yes_x__ no ___

Results: Pass ___x__ Fail_____

Preconditions for Test: A valid BallotFile object is passed into the constructor as a parameter

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	A File object is instantiated	simple_opl_ballot_file.csv	A File object is instantiated pointing to simple_opl_ballot_file.csv	A File object is instantiated pointing to simple_opl_ballot_file.csv	
2	A BallotFile object is instantiated	File object from Step #1	A BallotFile object is instantiated using File object from Step #1 and does not throw an exception during construction.	A BallotFile object is instantiated using File object from Step #1 and does not throw an exception during construction.	
3					
4					

Post condition(s) for Test:

A BallotFile object is instantiated without any exceptions being thrown.

Project Name: Project 1: Voting System **Team#17**

Test Stage: Unit _x_ System __ **Test Date: 11/12/2018**

Test Case ID#: Koo_BallotFile_002 **Name(s) of Testers: Justin Koo**

Test Description: Test method is named BallotFileConstructorThrowsIOException() located in BallotFileTest.java. It tests/uses the BallotFile constructor.

Automated: yes_x__ no ___

Results: Pass __x___ Fail_____

Preconditions for Test: An invalid File object is passed into the BallotFile constructor.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	An invalid File object is instantiated	doesnotexist.csv	A File object pointing to doesnotexist.csv is instantiated	A File object pointing to doesnotexist.csv is instantiated	
2	A BallotFile object is instantiated	File object from step #1	BallotFile constructor throws an IOException	BallotFile constructor throws an IOException	
3					
4					

Post condition(s) for Test:

An IOException is thrown indicating that “doesnotexist.csv” cannot be read.

Project Name: Project 1: Voting System **Team#17**

Test Stage: Unit _x_ System __ **Test Date: 11/12/2018**

Test Case ID#: Koo_BallotFile_003 **Name(s) of Testers: Justin Koo**

Test Description: Test method is named getFilenameSimpleOPL() located in BallotFile.java. It tests/uses the getFilename method of the BallotFile class.

Automated: yes_x__ no ___

Results: Pass __x___ Fail_____

Preconditions for Test: A BallotFile has been instantiated without any errors.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Instantiated a BallotFile	simple_opl_ballot_file.csv	A BallotFile object is instantiated.	A BallotFile object is instantiated	
2	call the getFilename method	The BallotFile instance in step #1	returns "simple_opl_ballot_file.csv"	returns "simple_opl_ballot_file.csv"	
3					
4					

Post condition(s) for Test:

"simple_opl_ballot_file.csv" is returned from the getFilename method call.

Project Name: Project 1: Voting System **Team#17**

Test Stage: Unit _x_ System __ **Test Date: 11/12/2018**

Test Case ID#: Koo_BallotFile_004 **Name(s) of Testers: Justin Koo**

Test Description: Test method is named getLine3SimpleOPL() located in BallotFileTest.java. It tests/uses the getLine() method of the class.

Automated: yes_x__ no ___

Results: Pass __x___ Fail_____

Preconditions for Test: A BallotFile has been instantiated without any errors

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Instantiated a BallotFile	simple_opl_ballot_file.csv	A BallotFile object is instantiated.	A BallotFile object is instantiated	
2	call getLine(3)	the BallotFile object in step #1	returns "[Pike,D], [Foster,D],[Deutsch,R], [Borg,R], [Jones,R],[Smith,I]"	returns "[Pike,D], [Foster,D],[Deutsch,R], [Borg,R], [Jones,R],[Smith,I]"	
3					
4					

Post condition(s) for Test:

returns "[Pike,D], [Foster,D],[Deutsch,R], [Borg,R], [Jones,R],[Smith,I]"

Project Name: Project 1: Voting System **Team#17**

Test Stage: Unit _x_ System __ **Test Date: 11/12/2018**

Test Case ID#: Koo_BallotFile_005 **Name(s) of Testers: Justin Koo**

Test Description: Test method is named getLastLineSimpleOPL() located in BallotFileTest.java. It tests/uses the getLine() method of the class.

Automated: yes_x__ no ___

Results: Pass __x___ Fail_____

Preconditions for Test: A BallotFile has been instantiated without any errors

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Instantiated a BallotFile	simple_opl_ballot_file.csv	A BallotFile object is instantiated.	A BallotFile object is instantiated	
2	call getLine() for the last line	the BallotFile object in step #1 at the last line	returns",1,,,,"	returns ",1,,,,"	
3					
4					

Post condition(s) for Test:

returns ",1,,,,"

Project Name: Project 1: Voting System **Team#17**

Test Stage: Unit _x_ System __ **Test Date: 11/12/2018**

Test Case ID#: Koo_BallotFile_006 **Name(s) of Testers: Justin Koo**

Test Description: Test method is named getLineThrowsIndexOutOfBoundsExceptionSimpleOPL() located in BallotFileTest.java. It tests/uses the getLine() method of the class.

Automated: yes_x__ no ___

Results: Pass __x___ Fail_____

Preconditions for Test: A BallotFile has been instantiated without any errors

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Instantiated a BallotFile	simple_opl_ballot_file.csv	A BallotFile object is instantiated.	A BallotFile object is instantiated	
2	call getLine() for a line that does not exist	the BallotFile object in step #1 at last line + 1	throws an IndexOutOfBoundsException	throws an IndexOutOfBoundsException	
3					
4					

Post condition(s) for Test:

throws an IndexOutOfBoundsException

Project Name: Project 1: Voting System **Team#17**

Test Stage: Unit _x_ System __ **Test Date: 11/12/2018**

Test Case ID#: Koo_BallotFile_007 **Name(s) of Testers: Justin Koo**

Test Description: Test method is named getElectionTypeSimpleOPL located in BallotFileTest.java. It tests/uses the getElectionType method of the class.

Automated: yes_x__ no ___

Results: Pass __x___ Fail_____

Preconditions for Test: A BallotFile has been instantiated without any errors

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Instantiated a BallotFile	simple_opl_ballot_file.csv	A BallotFile object is instantiated.	A BallotFile object is instantiated	
2	call getElectionType	the BallotFile object in step #1	returns "OPL"	returns "OPL"	
3					
4					

Post condition(s) for Test:

returns "OPL"

Project Name: Project 1: Voting System **Team#17**

Test Stage: Unit _x_ System __ **Test Date: 11/12/2018**

Test Case ID#: Koo_Candidate_001 **Name(s) of Testers: Justin Koo**

Test Description: Test method is named candidateConstructor() located in Candidate.java. It tests/uses the constructor of the Candidate class.

Automated: yes_x__ no ___

Results: Pass __x___ Fail_____

Preconditions for Test: None

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	instantiate a new Candidate instance	name: "McCain" party: "R"	no exceptions are thrown, i.e. the construction was successful	no exceptions are thrown	
2					
3					
4					

Post condition(s) for Test:

no exceptions are thrown after instantiation of a new candidate

Project Name: Project 1: Voting System **Team#17**

Test Stage: Unit _x_ System __ **Test Date: 11/12/2018**

Test Case ID#: Koo_Candidate_002 **Name(s) of Testers: Justin Koo**

Test Description: Test method is named getName() located in CandidateTest.java. It tests/uses the getName() method of the Candidate class

Automated: yes_x__ no ___

Results: Pass __x___ Fail_____

Preconditions for Test: A Candidate object is instantiated

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	instantiate a new Candidate instance	name: "Borg" party: "D"	a new Candidate is instantiated	a new Candidate is instantiated	
2	call getName()	using the Candidate object in step #1	returns "Borg"	returns "Borg"	
3					
4					

Post condition(s) for Test:

returns "Borg"

Project Name: Project 1: Voting System **Team#17**

Test Stage: Unit _x_ System __ **Test Date: 11/12/2018**

Test Case ID#: Koo_Candidate_003 **Name(s) of Testers: Justin Koo**

Test Description: Test method is named getParty() located in CandidateTest.java. It tests/uses the getParty() method of the Candidate class

Automated: yes_x__ no ___

Results: Pass __x___ Fail_____

Preconditions for Test: A Candidate object is instantiated

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	instantiate a new Candidate instance	name: "Borg" party: "D"	a new Candidate is instantiated	a new Candidate is instantiated	
2	call getParty	using the Candidate object in step #1	returns "D"	returns "D"	
3					
4					

Post condition(s) for Test:

returns "D"

Project Name: Project 1: Voting System **Team#17**

Test Stage: Unit _x_ System __ **Test Date: 11/12/2018**

Test Case ID#: Koo_Candidate_004 **Name(s) of Testers: Justin Koo**

Test Description: Test method is named getNumVotes() located in CandidateTest.java. It tests/uses the getNumVotes() method of the Candidate class

Automated: yes_x__ no ___

Results: Pass __x___ Fail_____

Preconditions for Test: A Candidate object is instantiated

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	instantiate a new Candidate instance	name: "Borg" party: "D"	a new Candidate is instantiated	a new Candidate is instantiated	
2	call getNumVotes()	using the Candidate object in step #1	returns 0	returns 0	
3					
4					

Post condition(s) for Test:

returns 0

Project Name: Project 1: Voting System **Team#17**

Test Stage: Unit _x_ System __ **Test Date: 11/12/2018**

Test Case ID#: Koo_Candidate_005 **Name(s) of Testers: Justin Koo**

Test Description: Test method is named getAcquiredBallotsIsEmpty() located in CandidateTest.java. It tests/uses the getAcquiredBallots() method of the Candidate class

Automated: yes_x__ no ___

Results: Pass __x___ Fail_____

Preconditions for Test: A Candidate object is instantiated

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	instantiate a new Candidate instance	name: "Borg" party: "D"	a new Candidate is instantiated	a new Candidate is instantiated	
2	call getAcquiredBallots()	using the Candidate object in step #1	returns an empty List	returns an empty List	
3					
4					

Post condition(s) for Test:

returns an empty List

Project Name: Project 1: Voting System **Team#17**

Test Stage: Unit _x_ System __ **Test Date: 11/12/2018**

Test Case ID#: Koo_Candidate_006 **Name(s) of Testers: Justin Koo**

Test Description: Test method is named setName() located in CandidateTest.java. It tests/uses the setName(), getName() methods of the Candidate class

Automated: yes_x__ no ___

Results: Pass __x___ Fail_____

Preconditions for Test: A Candidate object is instantiated

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	instantiate a new Candidate instance	name: "Borg" party: "D"	a new Candidate is instantiated	a new Candidate is instantiated	
2	call setName(newname)	newname = "Pike"	see step #3	see step #3	
3	call getName()	Candidate object from step #1 after step #2	returns "Pike"	returns "Pike"	
4					

Post condition(s) for Test:

member variable name is set to "Pike"

Project Name: Project 1: Voting System **Team#17**

Test Stage: Unit _x_ System __ **Test Date: 11/12/2018**

Test Case ID#: Koo_Candidate_007 **Name(s) of Testers: Justin Koo**

Test Description: Test method is named setParty() located in CandidateTest.java. It tests/uses the setParty(), getParty() methods of the Candidate class

Automated: yes_x__ no ___

Results: Pass __x___ Fail_____

Preconditions for Test: A Candidate object is instantiated

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	instantiate a new Candidate instance	name: "Borg" party: "D"	a new Candidate is instantiated	a new Candidate is instantiated	
2	call setParty(newparty)	newparty = "R"	see step #3	see step #3	
3	call getParty()	Candidate object from step #1 after step #2	returns "R"	returns "R"	
4					

Post condition(s) for Test:

member variable party is set to "R"

Project Name: Project 1: Voting System **Team#17**

Test Stage: Unit _x_ System __ **Test Date: 11/12/2018**

Test Case ID#: Koo_Candidate_008 **Name(s) of Testers: Justin Koo**

Test Description: Test method is named acquireBallotCorrectBallot() located in CandidateTest.java. It tests/uses the acquireBallot() method of the Candidate class

Automated: yes_x__ no ___

Results: Pass __x___ Fail_____

Preconditions for Test: A Candidate object is instantiated

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	instantiate a new Candidate instance	name: "Borg" party: "D"	a new Candidate is instantiated	a new Candidate is instantiated	
2	call acquireBallot(ballot_id)	ballot_id = 0	see step #3	see step #3	
3	call getAcquiredBallots().get(0)	The Candidate object from step #1 after step #2	returns 0	returns 0	
4					

Post condition(s) for Test:

member variable acquired_ballots has 0 at index 0.

Project Name: Project 1: Voting System **Team#17**

Test Stage: Unit _x_ System __ **Test Date: 11/12/2018**

Test Case ID#: Koo_Candidate_009 **Name(s) of Testers: Justin Koo**

Test Description: Test method is named getAcquiredBallotsCorrectNumVotes() located in CandidateTest.java. It tests/uses the getAcquiredBallots(), getNumVotes() methods of the Candidate class

Automated: yes_x__ no ___

Results: Pass __x___ Fail_____

Preconditions for Test: A Candidate object is instantiated

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	instantiate a new Candidate instance	name: "Borg" party: "D"	a new Candidate is instantiated	a new Candidate is instantiated	
2	call acquireBallot(ballot_id)	ballot_id = 0	see step #3	see step #3	
3	call getNumVotes()	Candidate object from step #1 after step #2	returns 1	returns 1	
4					

Post condition(s) for Test:

member variable acquired_ballots has 1 element in the list

Project Name: Project 1: Voting System **Team#17**

Test Stage: Unit _x_ System __ **Test Date: 11/13/2018**

Test Case ID#: Koo_Party_001 **Name(s) of Testers: Justin Koo**

Test Description: Test method is named partyConstructor() located in PartyTest.java. It tests/uses the constructor for the Party class to make sure no errors are thrown during instantiation.

Automated: yes_x__ no ___

Results: Pass __x___ Fail_____

Preconditions for Test: none

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	instantiate a new Party instance	name: "R" num_candidates: 100	no exceptions are thrown, i.e. the construction was successful	no exceptions are thrown	
2					
3					
4					

Post condition(s) for Test:

no exceptions are thrown during construction

Project Name: Project 1: Voting System **Team#17**

Test Stage: Unit _x_ System __ **Test Date: 11/13/2018**

Test Case ID#: Koo_Party_002 **Name(s) of Testers: Justin Koo**

Test Description: Test method is named getName() located in PartyTest.java. It tests/uses the getName() method for the Party class.

Automated: yes_x__ no ___

Results: Pass __x___ Fail_____

Preconditions for Test: A party object is instantiated

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	instantiate a new Party instance	name: "D" num_candidates: 3	a new Party is instantiated	a new Party is instantiated	
2	call getName()	The Party object from step #1	returns "D"	returns "D"	
3					
4					

Post condition(s) for Test:

returns "D"

Project Name: Project 1: Voting System **Team#17**

Test Stage: Unit _x_ System __ **Test Date: 11/13/2018**

Test Case ID#: Koo_Party_003 **Name(s) of Testers: Justin Koo**

Test Description: Test method is named getNumCandidates() located in PartyTest.java. It tests/uses the getNumCandidates() method for the Party class.

Automated: yes_x__ no ___

Results: Pass __x___ Fail_____

Preconditions for Test: A party object is instantiated

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	instantiate a new Party instance	name: "D" num_candidates: 3	a new Party is instantiated	a new Party is instantiated	
2	call getNumCandidates	The Party object from step #1	returns 3	returns 3	
3					
4					

Post condition(s) for Test:

returns 3

Project Name: Project 1: Voting System **Team#17**

Test Stage: Unit _x_ System __ **Test Date: 11/12/2018**

Test Case ID#: Koo_Party_004 **Name(s) of Testers: Justin Koo**

Test Description: Test method is named getNumSeats() located in PartyTest.java. It tests/uses the getNumSeats() method for the Party class.

Automated: yes_x__ no ___

Results: Pass __x___ Fail_____

Preconditions for Test: A party object is instantiated

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	instantiate a new Party instance	name: "D" num_candidates: 3	a new Party is instantiated	a new Party is instantiated	
2	call getNumSeats()	The Party object from step #1	returns 0	returns 0	
3					
4					

Post condition(s) for Test:

returns 0

Project Name: Project 1: Voting System **Team#17**

Test Stage: Unit _x_ System __ **Test Date: 11/12/2018**

Test Case ID#: Koo_Party_005 **Name(s) of Testers: Justin Koo**

Test Description: Test method is named getNumVotes() located in PartyTest.java. It tests/uses the getNumVotes() method for the Party class.

Automated: yes_x__ no ___

Results: Pass __x___ Fail_____

Preconditions for Test: A party object is instantiated

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	instantiate a new Party instance	name: "D" num_candidates: 3	a new Party is instantiated	a new Party is instantiated	
2	call getNumVotes()	The Party object from step #1	returns 0	returns 0	
3					
4					

Post condition(s) for Test:

returns 0

Project Name: Project 1: Voting System **Team#17**

Test Stage: Unit _x_ System __ **Test Date: 11/12/2018**

Test Case ID#: Koo_Party_006 **Name(s) of Testers: Justin Koo**

Test Description: Test method is named getAcquiredBallotsIsEmpty() located in PartyTest.java. It tests/uses the getAcquiredBallots() method for the Party class.

Automated: yes_x__ no ___

Results: Pass __x___ Fail_____

Preconditions for Test: A party object is instantiated

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	instantiate a new Party instance	name: "D" num_candidates: 3	a new Party is instantiated	a new Party is instantiated	
2	call getAcquiredBallots()	The Party object from step #1	returns an empty list	returns an empty list	
3					
4					

Post condition(s) for Test:

returns an empty list

Project Name: Project 1: Voting System **Team#17**

Test Stage: Unit _x_ System __ **Test Date: 11/12/2018**

Test Case ID#: Koo_Party_007 **Name(s) of Testers: Justin Koo**

Test Description: Test method is named setName() located in PartyTest.java. It tests/uses the setName() and getName() methods for the Party class.

Automated: yes_x__ no ___

Results: Pass __x___ Fail_____

Preconditions for Test: A party object is instantiated

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	instantiate a new Party instance	name: "D" num_candidates: 3	a new Party is instantiated	a new Party is instantiated	
2	call setName(newname)	newname = "R"	see step #3	see step #3	
3	call getName()	the Party object from step #1 after step #2	returns "R"	returns "R"	
4					

Post condition(s) for Test:

returns "R"

Project Name: Project 1: Voting System **Team#17**

Test Stage: Unit _x_ System __ **Test Date: 11/12/2018**

Test Case ID#: Koo_Party_008 **Name(s) of Testers: Justin Koo**

Test Description: Test method is named setNumCandidates() located in PartyTest.java. It tests the setNumCandidates(), getNumCandidates method for the Party class.

Automated: yes_x__ no ___

Results: Pass __x___ Fail_____

Preconditions for Test: A party object is instantiated

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	instantiate a new Party instance	name: "D" num_candidates: 3	a new Party is instantiated	a new Party is instantiated	
2	call setNumCandidates(newnum)	newnum= 4	see step #3	see step #3	
3	call getNumCandidates()	the Party object from step #1 after step #2	returns 4	returns 4	
4					

Post condition(s) for Test:

returns 4

Project Name: Project 1: Voting System **Team#17**

Test Stage: Unit _x_ System __ **Test Date: 11/13/2018**

Test Case ID#: Koo_Party_009 **Name(s) of Testers: Justin Koo**

Test Description: Test method is named getAcquiredBallotesCorrectNumVotes() located in PartyTest.java. It tests/uses the acquireBallot() method of the Party class

Automated: yes_x__ no ___

Results: Pass __x___ Fail_____

Preconditions for Test: A Party object is instantiated

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	instantiate a new Party instance	name: "D" party: "3"	a new Candidate is instantiated	a new Candidate is instantiated	
2	call acquireBallot(ballot_id)	ballot_id = 0	see step #3	see step #3	
3	call getAcquiredBallots().get(0)	The Party object from step #1 after step #2	returns 0	returns 0	
4					

Post condition(s) for Test:

member variable acquired_ballots has 0 at index 0.

Project Name: Project 1: Voting System **Team#17**

Test Stage: Unit _x_ System __ **Test Date: 11/13/2018**

Test Case ID#: Koo_Party_010 **Name(s) of Testers: Justin Koo**

Test Description: Test method is named acquireBallotCorrectNumVotes() located in PartyTest.java. It tests/uses the getAcquiredBallots() method of the Party class

Automated: yes_x__ no ___

Results: Pass __x___ Fail_____

Preconditions for Test: A Party object is instantiated

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	instantiate a new Party instance	name: "D" party: "3"	a new Candidate is instantiated	a new Candidate is instantiated	
2	call acquireBallot(ballot_id)	ballot_id = 0	see step #3	see step #3	
3	call getNumVotes()	The Party object from step #1 after step #2	returns 1	returns 1	
4					

Post condition(s) for Test:

member variable acquired_ballots has 1 element in the list

Project Name: Project 1: Voting System **Team#17**

Test Stage: Unit _x_ System __ **Test Date: 11/13/2018**

Test Case ID#: Koo_UserInterface_001 **Name(s) of Testers: Justin Koo**

Test Description: Test method is named UserInterfaceConstructor located in UserInterfaceTest.java. It tests that no exceptions are thrown when an UserInterface object is instantiated.

Automated: yes_x__ no ___

Results: Pass __x___ Fail_____

Preconditions for Test: None

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	A new UserInterface object is instantiated	none	No exceptions are thrown	no exceptions are thrown	
2					
3					
4					

Post condition(s) for Test:

no exceptions are thrown during the constructor call of the UserInterface class.

Project Name: Project 1: Voting System **Team#17**

Test Stage: Unit _x_ System __ **Test Date: 11/13/2018**

Test Case ID#: Koo_UserInterface_002 **Name(s) of Testers: Justin Koo**

Test Description: Test whether the method requestBallotFilename() of the UserInterface class displays to the console a prompt for the user to enter a ballot file name.

Automated: yes___ no _x__

Results: Pass __x___ Fail_____

Preconditions for Test: A UserInterface object is instantiated

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Tester runs VotingSystem.java	none	The console displays a prompt for users to enter a ballot filename	The console displays a prompt for users to enter a ballot filename.	
2					
3					
4					

Post condition(s) for Test:

The user can see a prompt from the console to enter a ballot filename.

Project Name: Project 1: Voting System **Team#17**

Test Stage: Unit _x_ System __ **Test Date: 11/13/2018**

Test Case ID#: Koo_UserInterface_003 **Name(s) of Testers: Justin Koo**

Test Description: Test whether the method displayInvalidElectionType() of the UserInterface class displays to the console information informing the user about an invalid election type encountered.

Automated: yes___ no _x__

Results: Pass __x___ Fail_____

Preconditions for Test: A UserInterface object is instantiated

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Tester runs VotingSystem.java	none	The console displays a prompt for users to enter a ballot filename	The console displays a prompt for users to enter a ballot filename.	
2	Tester enters a filename	invalid_election_type_ballot_file.csv	The console informs the user that the election type encountered, "CPL" is unexpected.	The console informs the user that the election type encountered, "CPL" is unexpected.	
3					
4					

Post condition(s) for Test:

The console informs the user that the election type encountered "CPL" is unexpected. The console displays a prompt for a second time for users to enter a ballot filename.

Project Name: Project 1: Voting System **Team#17**

Test Stage: Unit _x_ System __ **Test Date: 11/13/2018**

Test Case ID#: Koo_UserInterface_004 **Name(s) of Testers: Justin Koo**

Test Description: Test whether the method displayExceptionMessage() of the UserInterface class displays to the console the exception message of any exception that is thrown when reading the given ballot file.

Automated: yes___ no _x__

Results: Pass __x___ Fail_____

Preconditions for Test: A UserInterface object is instantiated

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Tester runs VotingSystem.java	none	The console displays a prompt for users to enter a ballot filename	The console displays a prompt for users to enter a ballot filename.	
2	Tester enters a filename string	any string that denotes a filename that does not exist within the src/ directory	The console informs the user that the system could not find the file specified.	The console informs the user that the system could not find the file specified.	
3					
4					

Post condition(s) for Test:

The console informs the user that the system could not find the file specified. The console displays a prompt for users a second time to enter a ballot filename.

Project Name: Project 1: Voting System **Team#17**

Test Stage: Unit _x_ System __ **Test Date: 11/13/2018**

Test Case ID#: Koo_UserInterface_005 **Name(s) of Testers: Justin Koo**

Test Description: Test whether the method displayResults() of the UserInterface class displays to the console the summarized results of the election.

Automated: yes___ no _x__

Results: Pass __x___ Fail_____

Preconditions for Test: A UserInterface object is instantiated

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Tester runs VotingSystem.java	none	The console displays a prompt for users to enter a ballot filename	The console displays a prompt for users to enter a ballot filename.	
2	Tester enters a filename string	simple_opl_ballot_file.csv	The console displays some result to the user (may not be correct, but nonempty))	The console displays some result to the user (may not be correct, but nonempty)	
3					
4					

Post condition(s) for Test:

The console displays some information informing the user about the election within “simple_opl_ballot_file.csv”. These informations may not be necessarily correct, but it should be nonempty.

Project Name: Project 1: Voting System **Team#17**

Test Stage: Unit _x_ System __ **Test Date: 11/13/2018**

Test Case ID#: Koo_OPLElection_001 **Name(s) of Testers: Justin Koo**

Test Description: Test method is named OPLElectionConstructorSimpleOPL located in OPLElectionTest.java. It tests that no exceptions are thrown during the constructor call if a valid BallotFile object is provided as an argument.

Automated: yes_x__ no ___

Results: Pass __x___ Fail_____

Preconditions for Test: A valid BallotFile object is passed into the OPLElection constructor.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	A valid BallotFile object is instantiated	simple_opl_ballot_file.csv	A valid BallotFile object is created. No exceptions thrown.	A valid BallotFile object is created. No Exceptions thrown	
2	Instantiate a new OPLElection object	pass the BallotFile object in step #1 into the constructor	No exceptions are thrown	No exceptions thrown.	
3					
4					

Post condition(s) for Test:

A new OPLElection object is instantiated without any errors being thrown.

Project Name: Project 1: Voting System **Team#17**

Test Stage: Unit _x_ System __ **Test Date: 11/13/2018**

Test Case ID#: Koo_OPLElection_002 **Name(s) of Testers: Justin Koo**

Test Description: Test method is getNumCandidatesSimpleOPL() located in OPLElectionTest.java. It tests the getNumCandidates() method of the OPLElection class.

Automated: yes_x__ no ___

Results: Pass __x___ Fail_____

Preconditions for Test: A valid BallotFile object is passed into the OPLElection constructor.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	A valid BallotFile object is instantiated	simple_opl_ballot_file.csv	A valid BallotFile object is created. No exceptions thrown.	A valid BallotFile object is created. No Exceptions thrown	
2	Instantiate a new OPLElection object	pass the BallotFile object in step #1 into the constructor	No exceptions are thrown	No exceptions thrown.	
3	call getNumCandidates()	using the OPLElection object in step #2	returns 6	returns 6	
4					

Post condition(s) for Test:

returns 6

Project Name: Project 1: Voting System **Team#17**

Test Stage: Unit _x_ System __ **Test Date: 11/13/2018**

Test Case ID#: Koo_OPLElection_003 **Name(s) of Testers: Justin Koo**

Test Description: Test method is getNumSeatsSimpleOPL() located in OPLElectionTest.java. It tests the getNumSeats() method of the OPLElection class.

Automated: yes_x__ no ___

Results: Pass __x___ Fail_____

Preconditions for Test: A valid BallotFile object is passed into the OPLElection constructor.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	A valid BallotFile object is instantiated	simple_opl_ballot_file.csv	A valid BallotFile object is created. No exceptions thrown.	A valid BallotFile object is created. No Exceptions thrown	
2	Instantiate a new OPLElection object	pass the BallotFile object in step #1 into the constructor	No exceptions are thrown	No exceptions thrown.	
3	call getNumSeats()	using the OPLElection object in step #2	returns 3	returns 3	
4					

Post condition(s) for Test:

returns 6

Project Name: Project 1: Voting System **Team#17**

Test Stage: Unit _x_ System __ **Test Date: 11/13/2018**

Test Case ID#: Koo_OPLElection_004 **Name(s) of Testers: Justin Koo**

Test Description: Test method is getNumBallotsSimpleOPL() located in OPLElectionTest.java. It tests the getNumBallots() method of the OPLElection class.

Automated: yes_x__ no ___

Results: Pass __x___ Fail_____

Preconditions for Test: A valid BallotFile object is passed into the OPLElection constructor.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	A valid BallotFile object is instantiated	simple_opl_ballot_file.csv	A valid BallotFile object is created. No exceptions thrown.	A valid BallotFile object is created. No Exceptions thrown	
2	Instantiate a new OPLElection object	pass the BallotFile object in step #1 into the constructor	No exceptions are thrown	No exceptions thrown.	
3	call getNumBallots()	using the OPLElection object in step #2	returns 9	returns 9	
4					

Post condition(s) for Test:

returns 9

Project Name: Project 1: Voting System **Team#17**

Test Stage: Unit _x_ System __ **Test Date: 11/13/2018**

Test Case ID#: Koo_OPLElection_005 **Name(s) of Testers: Justin Koo**

Test Description: Test method is getQuotaSimpleOPL() located in OPLElectionTest.java. It tests the getQuota() method of the OPLElection class.

Automated: yes_x__ no ___

Results: Pass __x___ Fail_____

Preconditions for Test: A valid BallotFile object is passed into the OPLElection constructor.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	A valid BallotFile object is instantiated	simple_opl_ballot_file.csv	A valid BallotFile object is created. No exceptions thrown.	A valid BallotFile object is created. No Exceptions thrown	
2	Instantiate a new OPLElection object	pass the BallotFile object in step #1 into the constructor	No exceptions are thrown	No exceptions thrown.	
3	call getQuota()	using the OPLElection object in step #2	returns 3	returns 3	
4					

Post condition(s) for Test:

returns 3

Project Name: Project 1: Voting System **Team#17**

Test Stage: Unit _x_ System __ **Test Date: 11/13/2018**

Test Case ID#: Koo_OPLElection_006 **Name(s) of Testers: Justin Koo**

Test Description: Test method is getCandidatesNamesSimpleOPL() located in OPLElectionTest.java. It tests the getCandidates() method of the OPLElection class.

Automated: yes_x__ no ___

Results: Pass __x___ Fail_____

Preconditions for Test: A valid BallotFile object is passed into the OPLElection constructor.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	A valid BallotFile object is instantiated	simple_opl_ballot_file.csv	A valid BallotFile object is created. No exceptions thrown.	A valid BallotFile object is created. No Exceptions thrown	
2	Instantiate a new OPLElection object	pass the BallotFile object in step #1 into the constructor	No exceptions are thrown	No exceptions thrown.	
3	call getCandidates()	using the OPLElection object in step #2	returns a Candidate[]	returns a Candidate[]	
4	Assert names are correct by calling getName() on each Candidate in the Candidate[]	using the Candidate[] in step #3	Candidate[] from step #3 contains candidates with names {"Pike", "Foster", "Deutsch", "Borg", "Jones", "Smith"}	Candidate[] from step #3 contains candidates with names {"Pike", "Foster", "Deutsch", "Borg", "Jones", "Smith"}	

Post condition(s) for Test:

A Candidate[] is returned containing Candidate objects that have the correct names.

Project Name: Project 1: Voting System **Team#17**

Test Stage: Unit _x_ System __ **Test Date: 11/13/2018**

Test Case ID#: Koo_OPLElection_007 **Name(s) of Testers: Justin Koo**

Test Description: Test method is getCandidatesPartiesSimpleOPL() located in OPLElectionTest.java. It tests the getCandidates() method of the OPLElection class.

Automated: yes_x__ no ___

Results: Pass __x___ Fail_____

Preconditions for Test: A valid BallotFile object is passed into the OPLElection constructor.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	A valid BallotFile object is instantiated	simple_opl_ballot_file.csv	A valid BallotFile object is created. No exceptions thrown.	A valid BallotFile object is created. No Exceptions thrown	
2	Instantiate a new OPLElection object	pass the BallotFile object in step #1 into the constructor	No exceptions are thrown	No exceptions thrown.	
3	call getCandidates()	using the OPLElection object in step #2	returns a Candidate[]	returns a Candidate[]	
4	Assert parties are correct by calling getParty() on each Candidate in the Candidate[]	using the Candidate[] in step #3	Candidate[] from step #3 contains candidates with parties {"D", "D", "R", "R", "R", "I"}	Candidate[] from step #3 contains candidates with parties {"D", "D", "R", "R", "R", "I"}	

Post condition(s) for Test:

A Candidate[] is returned containing Candidate objects that have the correct parties.

Project Name: Project 1: Voting System **Team#17**

Test Stage: Unit _x_ System __ **Test Date: 11/13/2018**

Test Case ID#: Koo_OPLElection_008 **Name(s) of Testers: Justin Koo**

Test Description: Test method is getPartiesNamesSimpleOPL() located in OPLElectionTest.java. It tests the getParties() method of the OPLElection class.

Automated: yes_x__ no ___

Results: Pass __x___ Fail_____

Preconditions for Test: A valid BallotFile object is passed into the OPLElection constructor.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	A valid BallotFile object is instantiated	simple_opl_ballot_file.csv	A valid BallotFile object is created. No exceptions thrown.	A valid BallotFile object is created. No Exceptions thrown	
2	Instantiate a new OPLElection object	pass the BallotFile object in step #1 into the constructor	No exceptions are thrown	No exceptions thrown.	
3	call getParties()	using the OPLElection object in step #2	returns a Party[]	returns a Party[]	
4	Assert parties names are correct by calling getName() on each Party in the Party[]	using the Party[] in step #3	Party[] from step #3 contains parties with party names: {“D”,”R”,”I”}	Party[] from step #3 contains parties with party names: {“D”,”R”,”I”}	

Post condition(s) for Test:

A Party[] is returned containing Party objects that have the correct party names.

Project Name: Project 1: Voting System **Team#17**

Test Stage: Unit _x_ System __ **Test Date: 11/13/2018**

Test Case ID#: Koo_OPLElection_009 **Name(s) of Testers: Justin Koo**

Test Description: Test method is getPartiesNumCandidatesSimpleOPL() located in OPLElectionTest.java. It tests the getParties() method of the OPLElection class.

Automated: yes_x__ no ___

Results: Pass __x___ Fail_____

Preconditions for Test: A valid BallotFile object is passed into the OPLElection constructor.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	A valid BallotFile object is instantiated	simple_opl_ballot_file.csv	A valid BallotFile object is created. No exceptions thrown.	A valid BallotFile object is created. No Exceptions thrown	
2	Instantiate a new OPLElection object	pass the BallotFile object in step #1 into the constructor	No exceptions are thrown	No exceptions thrown.	
3	call getParties()	using the OPLElection object in step #2	returns a Party[]	returns a Party[]	
4	Assert number of candidates of parties names are correct by calling getNumCandidates() on each Party in the Party[]	using the Party[] in step #3	Party[] from step #3 contains parties with num candidates: {2, 3, 1}	Party[] from step #3 contains parties with num candidates: {2, 3, 1}	

Post condition(s) for Test:

A Party[] is returned containing Party objects that have the correct number of candidates.

Project Name: Project 1: Voting System **Team#17**

Test Stage: Unit _x_ System __ **Test Date: 11/13/2018**

Test Case ID#: Koo_OPLElection_010 **Name(s) of Testers: Justin Koo**

Test Description: Test method is getCandidateWinnersNotEmptySimpleOPL() located in OPLElectionTest.java. It tests the getCandidateWinners() method of the OPLElection class.

Automated: yes_x__ no ___

Results: Pass __x___ Fail_____

Preconditions for Test: A valid BallotFile object is passed into the OPLElection constructor.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	A valid BallotFile object is instantiated	simple_opl_ballot_file.csv	A valid BallotFile object is created. No exceptions thrown.	A valid BallotFile object is created. No Exceptions thrown	
2	Instantiate a new OPLElection object	pass the BallotFile object in step #1 into the constructor	No exceptions are thrown	No exceptions thrown.	
3	call getCandidateWinners()	using the OPLElection object in step #2	returns a nonempty Set<Candidate>	returns a nonempty Set<Candidate>	

Post condition(s) for Test:

returns a nonempty Set<Candidate>

Project Name: Project 1: Voting System **Team#17**

Test Stage: Unit _x_ System __ **Test Date: 11/13/2018**

Test Case ID#: Koo_OPLElection_011 **Name(s) of Testers: Justin Koo**

Test Description: Test method is toStringNotEmptySimpleOPL() located in OPLElectionTest.java. It tests the toString() method of the OPLElection class.

Automated: yes_x__ no ___

Results: Pass __x___ Fail_____

Preconditions for Test: A valid BallotFile object is passed into the OPLElection constructor.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	A valid BallotFile object is instantiated	simple_opl_ballot_file.csv	A valid BallotFile object is created. No exceptions thrown.	A valid BallotFile object is created. No Exceptions thrown	
2	Instantiate a new OPLElection object	pass the BallotFile object in step #1 into the constructor	No exceptions are thrown	No exceptions thrown.	
3	call toString()	using the OPLElection object in step #2	returns a nonempty String	returns a nonempty String	due to randomness when a tie occurs, the string returned will not always be the same

Post condition(s) for Test:

returns a nonempty String

Project Name: Project 1: Voting System **Team#17**

Test Stage: Unit _x_ System __ **Test Date: 11/13/2018**

Test Case ID#: Koo_OPLElection_010 **Name(s) of Testers: Justin Koo**

Test Description: Test method is getPartyWinnersNotEmptySimpleOPL() located in OPLElectionTest.java. It tests the getPartyWinners() method of the OPLElection class.

Automated: yes_x__ no ___

Results: Pass __x___ Fail_____

Preconditions for Test: A valid BallotFile object is passed into the OPLElection constructor.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	A valid BallotFile object is instantiated	simple_opl_ballot_file.csv	A valid BallotFile object is created. No exceptions thrown.	A valid BallotFile object is created. No Exceptions thrown	
2	Instantiate a new OPLElection object	pass the BallotFile object in step #1 into the constructor	No exceptions are thrown	No exceptions thrown.	
3	call getPartyWinners()	using the OPLElection object in step #2	returns a nonempty Set<Party>	returns a nonempty Set<Party>	

Post condition(s) for Test:

returns a nonempty Set<Party>

Project Name: Project 1: Voting System **Team#17**

Test Stage: Unit __ System _x_ **Test Date: 11/13/2018**

Test Case ID#: Koo_System_001 **Name(s) of Testers: Justin Koo**

Test Description: Test method is named VotingSystemIRBallotFile1NoErrors() located in SystemTest.java. It tests whether the program can run without errors using ir_ballot_file_1.csv.

Automated: yes_x__ no ___

Results: Pass __x___ Fail_____

Preconditions for Test: None

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Main method is invoked with "ir_ballot_file_1.csv" as input into System.in	ir_ballot_file_1.csv	No errors/exceptions thrown	No errors/exceptions thrown	
2					
3					

Post condition(s) for Test:

No errors/exceptions thrown

Project Name: Project 1: Voting System **Team#17**

Test Stage: Unit __ System _x_ **Test Date: 11/13/2018**

Test Case ID#: Koo_System_002 **Name(s) of Testers: Justin Koo**

Test Description: Test method is named VotingSystemIRBallotFile2NoErrors() located in SystemTest.java. It tests whether the program can run without errors using ir_ballot_file_2.csv.

Automated: yes_x__ no ___

Results: Pass __x___ Fail_____

Preconditions for Test: None

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Main method is invoked with "ir_ballot_file_2.csv" as input into System.in	ir_ballot_file_2.csv	No errors/exceptions thrown	No errors/exceptions thrown	
2					
3					

Post condition(s) for Test:

No errors/exceptions thrown

Project Name: Project 1: Voting System **Team#17**

Test Stage: Unit __ System _x_ **Test Date: 11/13/2018**

Test Case ID#: Koo_System_003 **Name(s) of Testers: Justin Koo**

Test Description: Test method is named VotingSystemIRBallotFile3NoErrors() located in SystemTest.java. It tests whether the program can run without errors using ir_ballot_file_3.csv.

Automated: yes_x__ no ___

Results: Pass __x___ Fail_____

Preconditions for Test: None

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Main method is invoked with "ir_ballot_file_3.csv" as input into System.in	ir_ballot_file_3.csv	No errors/exceptions thrown	No errors/exceptions thrown	
2					
3					

Post condition(s) for Test:

No errors/exceptions thrown

Project Name: Project 1: Voting System **Team#17**

Test Stage: Unit __ System _x_ **Test Date: 11/13/2018**

Test Case ID#: Koo_System_004 **Name(s) of Testers: Justin Koo**

Test Description: Test method is named VotingSystemIRBallotFile1NoErrors() located in SystemTest.java. It tests whether the program can run without errors using ir_ballot_file_4.csv.

Automated: yes_x__ no ___

Results: Pass __x___ Fail_____

Preconditions for Test: None

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Main method is invoked with "ir_ballot_file_4.csv" as input into System.in	ir_ballot_file_4.csv	No errors/exceptions thrown	No errors/exceptions thrown	
2					
3					

Post condition(s) for Test:

No errors/exceptions thrown

Project Name: Project 1: Voting System **Team#17**

Test Stage: Unit __ System _x_ **Test Date: 11/13/2018**

Test Case ID#: Koo_System_005 **Name(s) of Testers: Justin Koo**

Test Description: Test method is named VotingSystemIRBallotFile5NoErrors() located in SystemTest.java. It tests whether the program can run without errors using ir_ballot_file_5.csv.

Automated: yes_x__ no ___

Results: Pass __x___ Fail_____

Preconditions for Test: None

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Main method is invoked with "ir_ballot_file_5.csv" as input into System.in	ir_ballot_file_5.csv	No errors/exceptions thrown	No errors/exceptions thrown	
2					
3					

Post condition(s) for Test:

No errors/exceptions thrown

Project Name: Project 1: Voting System **Team#17**

Test Stage: Unit __ System _x_ **Test Date: 11/13/2018**

Test Case ID#: Koo_System_006 **Name(s) of Testers: Justin Koo**

Test Description: Test method is named VotingSystemSimpleIRBallotFileNoErrors() located in SystemTest.java. It tests whether the program can run without error using simple_ir_ballot_file.csv.

Automated: yes_x__ no ___

Results: Pass __x___ Fail_____

Preconditions for Test: None

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Main method is invoked with "simple_ir_ballot_file.csv" as input into System.in	simple_ir_ballot_file.csv	No errors/exceptions thrown	No errors/exceptions thrown	
2					
3					

Post condition(s) for Test:

No errors/exceptions thrown

Project Name: Project 1: Voting System **Team#17**

Test Stage: Unit __ System _x_ **Test Date: 11/13/2018**

Test Case ID#: Koo_System_007 **Name(s) of Testers: Justin Koo**

Test Description: Test method is named VotingSystemLargeIRBallotFileNoErrors() located in SystemTest.java. It tests whether the program can run without error using large_ir_ballot_file.csv.

Automated: yes_x__ no ___

Results: Pass __x___ Fail_____

Preconditions for Test: None

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Main method is invoked with "large_ir_ballot_file.csv" as input into System.in	large_ir_ballot_file.csv	No errors/exceptions thrown	No errors/exceptions thrown	
2					
3					

Post condition(s) for Test:

No errors/exceptions thrown

Project Name: Project 1: Voting System **Team#17**

Test Stage: Unit __ System _x_ **Test Date: 11/13/2018**

Test Case ID#: Koo_System_008 **Name(s) of Testers: Justin Koo**

Test Description: Test method is named VotingSystemOPLBallotFile0Candidate0Seat0BallotNoErrors() located in SystemTest.java. It tests whether the program can run without errors using opl_ballot_file_0_candidate_0_seat_0_ballot.csv

Automated: yes_x__ no ___

Results: Pass __x___ Fail_____

Preconditions for Test: None

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Main method is invoked with "opl_ballot_file_0_candidate_0_seat_0_ballot.csv" as input into System.in	opl_ballot_file_0_candidate_0_seat_0_ballot.csv	No errors/exceptions thrown	No errors/exceptions thrown	
2					
3					

Post condition(s) for Test:

No errors/exceptions thrown

Project Name: Project 1: Voting System **Team#17**

Test Stage: Unit __ System _x_ **Test Date: 11/13/2018**

Test Case ID#: Koo_System_009 **Name(s) of Testers: Justin Koo**

Test Description: Test method is named VotingSystemOPLBallotFile0Candidate1Seat0BallotNoErrors() located in SystemTest.java. It tests whether the program can run without errors using opl_ballot_file_0_candidate_1_seat_0_ballot.csv

Automated: yes_x__ no ___

Results: Pass __x___ Fail_____

Preconditions for Test: None

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Main method is invoked with "opl_ballot_file_0_candidate_1_seat_0_ballot.csv" as input into System.in	opl_ballot_file_0_candidate_1_seat_0_ballot.csv	No errors/exceptions thrown	No errors/exceptions thrown	
2					
3					

Post condition(s) for Test:

No errors/exceptions thrown

Project Name: Project 1: Voting System **Team#17**

Test Stage: Unit __ System _x_ **Test Date: 11/13/2018**

Test Case ID#: Koo_System_010 **Name(s) of Testers: Justin Koo**

Test Description: Test method is named VotingSystemOPLBallotFile1Candidate0Seat0BallotNoErrors() located in SystemTest.java. It tests whether the program can run without errors using opl_ballot_file_1_candidate_0_seat_0_ballot.csv

Automated: yes_x__ no ___

Results: Pass __x___ Fail_____

Preconditions for Test: None

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Main method is invoked with "opl_ballot_file_1_candidate_0_seat_0_ballot.csv" as input into System.in	opl_ballot_file_1_candidate_0_seat_0_ballot.csv	No errors/exceptions thrown	No errors/exceptions thrown	
2					
3					

Post condition(s) for Test:

No errors/exceptions thrown

Project Name: Project 1: Voting System **Team#17**

Test Stage: Unit __ System _x_ **Test Date: 11/13/2018**

Test Case ID#: Koo_System_011 **Name(s) of Testers: Justin Koo**

Test Description: Test method is named VotingSystemOPLBallotFile1Candidate0Seat1BallotNoErrors() located in SystemTest.java. It tests whether the program can run without errors using opl_ballot_file_1_candidate_0_seat_1_ballot.csv

Automated: yes_x__ no ___

Results: Pass __x___ Fail_____

Preconditions for Test: None

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Main method is invoked with "opl_ballot_file_1_candidate_0_seat_1_ballot.csv" as input into System.in	opl_ballot_file_1_candidate_0_seat_1_ballot.csv	No errors/exceptions thrown	No errors/exceptions thrown	
2					
3					

Post condition(s) for Test:

No errors/exceptions thrown

Project Name: Project 1: Voting System **Team#17**

Test Stage: Unit __ System _x_ **Test Date: 11/13/2018**

Test Case ID#: Koo_System_012 **Name(s) of Testers: Justin Koo**

Test Description: Test method is named VotingSystemOPLBallotFile1Candidate1Seat0BallotNoErrors() located in SystemTest.java. It tests whether the program can run without errors using opl_ballot_file_1_candidate_1_seat_0_ballot.csv

Automated: yes_x__ no ___

Results: Pass __x___ Fail_____

Preconditions for Test: None

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Main method is invoked with "opl_ballot_file_1_candidate_1_seat_0_ballot.csv" as input into System.in	opl_ballot_file_1_candidate_1_seat_0_ballot.csv	No errors/exceptions thrown	No errors/exceptions thrown	
2					
3					

Post condition(s) for Test:

No errors/exceptions thrown

Project Name: Project 1: Voting System **Team#17**

Test Stage: Unit __ System _x_ **Test Date: 11/13/2018**

Test Case ID#: Koo_System_013 **Name(s) of Testers: Justin Koo**

Test Description: Test method is named VotingSystemOPLBallotFile1Candidate1Seat1BallotNoErrors() located in SystemTest.java. It tests whether the program can run without errors using opl_ballot_file_1_candidate_1_seat_1_ballot.csv

Automated: yes_x__ no ___

Results: Pass __x___ Fail_____

Preconditions for Test: None

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Main method is invoked with "opl_ballot_file_1_candidate_1_seat_1_ballot.csv" as input into System.in	opl_ballot_file_1_candidate_1_seat_1_ballot.csv	No errors/exceptions thrown	No errors/exceptions thrown	
2					
3					

Post condition(s) for Test:

No errors/exceptions thrown

Project Name: Project 1: Voting System **Team#17**

Test Stage: Unit __ System _x_ **Test Date: 11/13/2018**

Test Case ID#: Koo_System_014 **Name(s) of Testers: Justin Koo**

Test Description: Test method is named VotingSystemSimpleOPLBallotFileNoErrors() located in SystemTest.java. It tests whether the program can run without errors using simple_opl_ballot_file.csv

Automated: yes_x__ no ___

Results: Pass __x___ Fail_____

Preconditions for Test: None

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Main method is invoked with "simple_opl_ballot_file" as input into System.in	simple_opl_ballot_file.csv	No errors/exceptions thrown	No errors/exceptions thrown	
2					
3					

Post condition(s) for Test:

No errors/exceptions thrown

Project Name: Project 1: Voting System **Team#17**

Test Stage: Unit __ System _x_ **Test Date: 11/13/2018**

Test Case ID#: Koo_System_015 **Name(s) of Testers: Justin Koo**

Test Description: Test method is named VotingSystemLargeOPLBallotFileNoErrors() located in SystemTest.java. It tests whether the program can run without errors using large_opl_ballot_file.csv

Automated: yes_x__ no ___

Results: Pass __x___ Fail_____

Preconditions for Test: None

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Main method is invoked with "large_opl_ballot_file.csv" as input into System.in	large_opl_ballot_file.csv	No errors/exceptions thrown	No errors/exceptions thrown	
2					
3					

Post condition(s) for Test:

No errors/exceptions thrown

Project Name: Project 1: Voting System **Team#17**

Test Stage: Unit __ System _x_ **Test Date: 11/13/2018**

Test Case ID#: Koo_System_016 **Name(s) of Testers: Justin Koo**

Test Description: Test method is named VotingSystemLargeOPLBallotFileUnder8Minutes() located in SystemTest.java. It tests whether the program can run in under 8 minutes using large_opl_ballot_file.csv (100,000 ballots)

Automated: yes_x__ no ___

Results: Pass __x___ Fail_____

Preconditions for Test: None

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Main method is invoked with "large_opl_ballot_file.csv" as input into System.in	large_opl_ballot_file.csv	Finishes in under 8 minutes	Finishes in under 8 minutes	
2					
3					

Post condition(s) for Test:

Finishes in under 8 minutes

Project Name: Project 1: Voting System **Team#17**

Test Stage: Unit __ System _x_ **Test Date: 11/13/2018**

Test Case ID#: Koo_System_017 **Name(s) of Testers: Justin Koo**

Test Description: Test method is named VotingSystemLargeIRBallotFileUnder8Minutes() located in SystemTest.java. It tests whether the program can run in under 8 minutes using large_ir_ballot_file.csv (100,000 ballots)

Automated: yes_x__ no ___

Results: Pass __x___ Fail_____

Preconditions for Test: None

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Main method is invoked with "large_ir_ballot_file.csv" as input into System.in	large_irl_ballot_file.csv	Finishes in under 8 minutes	Finishes in under 8 minutes	
2					
3					

Post condition(s) for Test:

Finishes in under 8 minutes

Project Name: Project 1: Voting System **Team#17**

Test Stage: Unit __ System _x_ **Test Date: 11/13/2018**

Test Case ID#: Koo_System_018 **Name(s) of Testers: Justin Koo**

Test Description: Test to see if console output and auditfile contents are correct given input file "simple_opl_ballot_file.csv"

Automated: yes___ no _x__

Results: Pass __x___ Fail_____

Preconditions for Test: None

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Main method is invoked with "simple_opl_ballot_file.csv" as input into System.in	simple_opl_ballot_file.csv	Console display correct summary results of winners. Audit file generated with correct contents, allowing one to track the election step by step.	Console display correct summary results of winners.. Audit file generated with correct contents, allowing one to track the election step by step.	
2					
3					

Post condition(s) for Test:

Console displays correct results. Audit file is generated. Audit file contains information allowing one to track election step by step.

Project Name: Project 1: Voting System **Team#17**

Test Stage: Unit __ System _x_ **Test Date: 11/13/2018**

Test Case ID#: Koo_System_019 **Name(s) of Testers: Justin Koo**

Test Description: Test to see if console output and auditfile contents are correct given input file "simple_ir_ballot_file.csv"

Automated: yes___ no _x__

Results: Pass __x___ Fail_____

Preconditions for Test: None

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Main method is invoked with "simple_ir_ballot_file.csv" as input into System.in	simple_ir_ballot_file.csv	Console display correct summary results of winners. Audit file generated with correct contents, allowing one to track the election step by step.	Console display correct summary results of winners.. Audit file generated with correct contents, allowing one to track the election step by step.	
2					
3					

Post condition(s) for Test:

Console displays correct results. Audit file is generated. Audit file contains information allowing one to track election step by step.

Project Name: Project 1: Voting System **Team#17**

Test Stage: Unit _x_ System __ **Test Date: 11/13/2018**

Test Case ID#: Zhang_IRElection_001 **Name(s) of Testers: Xiaochen Zhang**

Test Description: Test method is named IRElectionConstructorSimpleIR() located in IRElectionTest.java. It tests that no exceptions are thrown during the constructor call if a valid BallotFile object is provided as an argument.

Automated: yes_x__ no ___

Results: Pass ___x__ Fail_____

Preconditions for Test: A valid BallotFile object is passed into the IRElection constructor.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	A valid BallotFile object is instantiated	simple_ir_ballot_file.csv	A valid BallotFile object is created. No exceptions thrown.	A valid BallotFile object is created. No Exceptions thrown	
2	Instantiate a new IRElection object	pass the BallotFile object in step #1 into the constructor	No exceptions are thrown	No exceptions thrown.	
3					
4					

Post condition(s) for Test:

A new IRElection object is instantiated without any errors being thrown.

Project Name: Project 1: Voting System **Team#17**

Test Stage: Unit _x_ System __ **Test Date: 11/13/2018**

Test Case ID#: Zhang_IRElection_002 **Name(s) of Testers: Xiaochen Zhang**

Test Description: Test method is named testToStringSimpleIR() located in IRElectionTest.java. It tests the toString() method of the IRElection class.

Automated: yes_x__ no ___

Results: Pass __x___ Fail_____

Preconditions for Test: A valid File object is passed into the BallotFile constructor.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	A valid BallotFile object is instantiated	simple_ir_ballot_file.csv	A valid BallotFile object is created. No exceptions thrown.	A valid BallotFile object is created. No Exceptions thrown	
2	Instantiate a new IRElection object	pass the BallotFile object in step #1 into the constructor	No exceptions are thrown	No exceptions thrown.	
3	call toString()	using the IRElection object in step #2	returns the correct string	returns the correct string	
4					

Post condition(s) for Test:

Returns a string with correct election type, number of candidates, list of candidates and parties, ballots number, and winning candidate.

Project Name: Project 1: Voting System **Team#17**

Test Stage: Unit _x_ System __ **Test Date: 11/13/2018**

Test Case ID#: Zhang_IRElection_003 **Name(s) of Testers: Xiaochen Zhang**

Test Description: Test method is named testCandidatesSimpleIR() located in IRElectionTest.java. It tests the getCandidates() method of the IRElection class.

Automated: yes_x__ no ___

Results: Pass __x___ Fail_____

Preconditions for Test: A valid File object is passed into the BallotFile constructor.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	A valid BallotFile object is instantiated	simple_ir_ballot_file.csv	A valid BallotFile object is created. No exceptions thrown.	A valid BallotFile object is created. No Exceptions thrown	
2	Instantiate a new IRElection object	pass the BallotFile object in step #1 into the constructor	No exceptions are thrown	No exceptions thrown.	
3	call getCandidates()	using the IRElection object in step #2	returns a list of all candidates	returns a list of all candidates	
4					

Post condition(s) for Test:

Returns a list of all candidates with names “Rosen”, “Kleinberg”, “Chou” and “Royce”

Project Name: Project 1: Voting System **Team#17**

Test Stage: Unit _x_ System __ **Test Date: 11/13/2018**

Test Case ID#: Zhang_IRElection_004 **Name(s) of Testers: Xiaochen Zhang**

Test Description: Test method is testNumCandidatesSimpleIR() located in IRElectionTest.java. It tests the getNumCandidates() method of the IRElection class.

Automated: yes_x__ no ___

Results: Pass __x___ Fail_____

Preconditions for Test: A valid BallotFile object is passed into the IRElection constructor.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	A valid BallotFile object is instantiated	simple_ir_ballot_file.csv	A valid BallotFile object is created. No exceptions thrown.	A valid BallotFile object is created. No Exceptions thrown	
2	Instantiate a new IRElection object	pass the BallotFile object in step #1 into the constructor	No exceptions are thrown	No exceptions thrown.	
3	call getNumCandidates()	using the IRElection object in step #2	returns 4	returns 4	
4					

Post condition(s) for Test:

returns 4

Project Name: Project 1: Voting System **Team#17**

Test Stage: Unit _x_ System __ **Test Date: 11/13/2018**

Test Case ID#: Zhang_IRElection_005 **Name(s) of Testers: Xiaochen Zhang**

Test Description: Test method is testNumBallotsSimpleIR() located in IRElectionTest.java. It tests the getNumBallots() method of the IRElection class.

Automated: yes_x__ no ___

Results: Pass __x___ Fail_____

Preconditions for Test: A valid BallotFile object is passed into the IRElection constructor.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	A valid BallotFile object is instantiated	simple_ir_ballot_file.csv	A valid BallotFile object is created. No exceptions thrown.	A valid BallotFile object is created. No Exceptions thrown	
2	Instantiate a new IRElection object	pass the BallotFile object in step #1 into the constructor	No exceptions are thrown	No exceptions thrown.	
3	call getNumBallots()	using the IRElection object in step #2	returns 6	returns 6	
4					

Post condition(s) for Test:

returns 6

Project Name: Project 1: Voting System **Team#17**

Test Stage: Unit _x_ System __ **Test Date: 11/13/2018**

Test Case ID#: Zhang_IRElection_006 **Name(s) of Testers: Xiaochen Zhang**

Test Description: Test method is testWinningCandidateSimpleIR() located in IRElectionTest.java. It tests the getCandidateWinners() method of the IRElection class.

Automated: yes_x__ no ___

Results: Pass __x___ Fail_____

Preconditions for Test: A valid BallotFile object is passed into the IRElection constructor.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	A valid BallotFile object is instantiated	simple_ir_ballot_file.csv	A valid BallotFile object is created. No exceptions thrown.	A valid BallotFile object is created. No Exceptions thrown	
2	Instantiate a new IRElection object	pass the BallotFile object in step #1 into the constructor	No exceptions are thrown	No exceptions thrown.	
3	call getCandidateWinners()	using the IRElection object in step #2	returns candidate with name "Rosen"	returns candidate with name "Rosen"	

Post condition(s) for Test:

returns candidate with name "Rosen"

Project Name: Project 1: Voting System **Team#17**

Test Stage: Unit _x_ System __ **Test Date: 11/13/2018**

Test Case ID#: Zhang_IRElection_007 **Name(s) of Testers: Xiaochen Zhang**

Test Description: Test method is testWinningPartySimpleIR() located in IRElectionTest.java. It tests the getPartyWinners() method of the IRElection class.

Automated: yes_x__ no ___

Results: Pass __x___ Fail_____

Preconditions for Test: A valid BallotFile object is passed into the IRElection constructor.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	A valid BallotFile object is instantiated	simple_ir_ballot_file.csv	A valid BallotFile object is created. No exceptions thrown.	A valid BallotFile object is created. No Exceptions thrown	
2	Instantiate a new IRElection object	pass the BallotFile object in step #1 into the constructor	No exceptions are thrown	No exceptions thrown.	
3	call getPartyWinners()	using the IRElection object in step #2	returns party with name “D”	returns party with name “D”	

Post condition(s) for Test:

returns party with name “D”

Project Name: Project 1: Voting System **Team#17**

Test Stage: Unit _x_ System __ **Test Date: 11/13/2018**

Test Case ID#: Zhang_IRElection_008 **Name(s) of Testers: Xiaochen Zhang**

Test Description: Test method is testQuota() located in IRElectionTest.java. It tests the getQuota() method of the IRElection class.

Automated: yes_x__ no ___

Results: Pass __x___ Fail_____

Preconditions for Test: A valid BallotFile object is passed into the IRElection constructor.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	A valid BallotFile object is instantiated	simple_ir_ballot_file.csv	A valid BallotFile object is created. No exceptions thrown.	A valid BallotFile object is created. No Exceptions thrown	
2	Instantiate a new IRElection object	pass the BallotFile object in step #1 into the constructor	No exceptions are thrown	No exceptions thrown.	
3	call getQuota()	using the IRElection object in step #2	returns 4	returns 4	

Post condition(s) for Test:

returns 4

Project 2: Agile Scrum

Project 2: Agile Scrum	
PBI	Record and remove invalid ballots
Task Description	Track the number of invalid ballots present in a ballot file.
Testing Number	001
Team Member(s) Responsible	Carlos Alvarenga
Input	Ballot file name
Tests	1. Test getNumInvalidatedBallots() method in IRElection class for ballot file “ir_ballot_file_3” as input where there are no invalid ballots. 2. Test getNumInvalidatedBallots() method in IRElection class for ballot file “ir_ballot_file_1” as input where there’s exactly one invalid ballot. 3. Test getNumInvalidatedBallots() method in IRElection class for ballot file “ir_ballot_file_6” as input where all ballots are invalid. 4. Test getNumInvalidatedBallots() method in IRElection class for ballot file “ir_ballot_file_7” as input where, at most, half of the votes are missing in a given ballot.
Output	1. Output for test #1 is 0. 2. Output for test #2 is 1. 3. Output for test #3 is 2. 4. Output for test #4 is 0.
Passed/Failed	All 4 tests passed
Date	12/10/18

Project 2: Agile Scrum	
PBI	Record and remove invalid ballots
Task Description	Remove invalid ballots from election calculations.
Testing Number	002
Team Member(s) Responsible	Carlos Alvarenga
Input	Ballot file name
Tests	1. Test for ballot file “ir_ballot_file_3” as input where there are no invalid ballots and check generated audit file for correctness and omission of invalid ballots. 2. Test for ballot file “ir_ballot_file_1” as input where there’s exactly one invalid ballot and check generated audit file for correctness and omission of invalid ballots. 3. Test for ballot file “ir_ballot_file_6” as input where all ballots are invalid and check generated audit file for correctness and omission of invalid ballots. 4. Test for ballot file “ir_ballot_file_7” as input where, at most, half of the votes are missing in a given ballot and check generated audit file for correctness and omission of invalid ballots.
Output	1. Either candidate Kleinberg wins with 5 votes or Rosen wins with 6 votes and no ballots are omitted from election calculations. 2. Candidate Rosen wins with 4 votes and ballot #6 is removed from election calculations. 3. Either of the four candidates can win with 0 votes since all ballots are invalidated and removed from election calculations. 4. Candidate Rosen wins with 5 votes and no ballots are omitted from election calculations.
Passed/Failed	All 4 tests passed
Date	12/10/18

Project 2: Agile Scrum	
PBI	Record and remove invalid ballots
Task Description	Record invalid ballot(s) on file named invalidated_dateofelection.txt.
Testing Number	003
Team Member(s) Responsible	Carlos Alvarenga
Input	Ballot file name
Tests	1. Test for ballot file “ir_ballot_file_3” as input where there are no invalid ballots and check that the generated invalid ballots file is correctly named with a unique timestamp and contains all invalid ballots. 2. Test for ballot file “ir_ballot_file_1” as input where there’s exactly one invalid ballot and check that the generated invalid ballots file is correctly named with a unique timestamp and contains all invalid ballots. 3. Test for ballot file “ir_ballot_file_6” as input where all ballots are invalid and check that the generated invalid ballots file is correctly named with a unique timestamp and contains all invalid ballots. 4. Test for ballot file “ir_ballot_file_7” as input where, at most, half of the votes are missing in a given ballot and check that the generated invalid ballots file is correctly named with a unique timestamp and contains all invalid ballots.
Output	1. Invalid ballots file is generated with unique name and no ballots are recorded in the file. 2. Invalid ballots file is generated with unique name and ballot #6 at line number 10 is recorded in the file. 3. Invalid ballots file is generated with unique name and ballot #1 (at line number 5) and ballot #2 (at line number 6) are recorded in the file. 4. Invalid ballots file is generated with unique name and no ballots are recorded in the file.
Passed/Failed	All 4 tests passed
Date	12/10/18

Project 2: Agile Scrum	
PBI	Table displaying election progression is outputted to the screen
Task Description	The total number of votes in the election are recorded.
Testing Number	004
Team Member(s) Responsible	Carlos Alvarenga
Input	Ballot file name
Tests	1. Test getTotalNumVotes() method in IRElection class for ballot file “ir_ballot_file_1” as input where one ballot is invalid. 2. Test getTotalNumVotes() method in IRElection class for ballot file “ir_ballot_file_3” as input where all 6 ballots aren’t invalid. 3. Test getTotalNumVotes() method in IRElection class for ballot file “ir_ballot_file_6” as input where all 2 ballots are invalid.
Output	1. Output for test #1 is 5. 2. Output for test #2 is 6. 3. Output for test #3 is 0.
Passed/Failed	All 3 tests passed
Date	12/10/18

Project 2: Agile Scrum	
PBI	Table displaying election progression is outputted to the screen
Task Description	Track the total number of exhausted ballots through each election round.
Testing Number	005
Team Member(s) Responsible	Carlos Alvarenga
Input	Ballot file name
Tests	1. Test getExhaustedPileTotals() method in IRElection class for ballot file “ir_ballot_file_1” as input where an exhausted ballot only appears in the last (i.e. 4th) round. 2. Test getExhaustedPileTotals() method in IRElection class for ballot file “ir_ballot_file_2” as input where there aren’t any exhausted ballots in the whole election. 3. Test getExhaustedPileTotals() method in IRElection class for ballot file “ir_ballot_file_4” as input where 2 exhausted ballots appear in the last (i.e. 4th) round. 4. Test getExhaustedPileTotals() method in IRElection class for ballot file “ir_ballot_file_5” as input where a winner appears in the first round and there are no exhausted ballots. 5. Test getExhaustedPileTotals() method in IRElection class for ballot file “ir_ballot_file_7” as input where an exhausted ballot appears in round 3. 6. Test getExhaustedPileTotals() method in IRElection class for ballot file “ir_ballot_file_8” as input where one exhausted ballot appears in round 3 and another one appears in the last (i.e. 4th) round.
Output	1. ArrayList [0, 0, 0, 1] where each index corresponds to a round and the value corresponds to the number of total exhausted votes in that round. 2. ArrayList [0, 0, 0, 0] 3. ArrayList [0, 0, 0, 2] 4. ArrayList [0] 5. ArrayList [0, 0, 1, 1] 6. ArrayList [0, 0, 1, 2]
Passed/Failed	All 6 tests passed
Date	12/10/18

Project 2: Agile Scrum	
PBI	Table displaying election progression is outputted to the screen
Task Description	Track the number of exhausted ballots updated/added through each election round.
Testing Number	006
Team Member(s) Responsible	Carlos Alvarenga
Input	Ballot file name
Tests	1. Test getExhaustedPileUpdates() method in IRElection class for ballot file “ir_ballot_file_1” as input where an exhausted ballot only appears in the last (i.e. 4th) round. 2. Test getExhaustedPileUpdates() method in IRElection class for ballot file “ir_ballot_file_2” as input where there aren’t any exhausted ballots in the whole election. 3. Test getExhaustedPileUpdates() method in IRElection class for ballot file “ir_ballot_file_4” as input where 2 exhausted ballots appear in the last (i.e. 4th) round. 4. Test getExhaustedPileUpdates() method in IRElection class for ballot file “ir_ballot_file_5” as input where a winner appears in the first round and there are no exhausted ballots. 5. Test getExhaustedPileUpdates() method in IRElection class for ballot file “ir_ballot_file_7” as input where an exhausted ballot appears in round 3. 6. Test getExhaustedPileUpdates() method in IRElection class for ballot file “ir_ballot_file_8” as input where one exhausted ballot appears in round 3 and another one appears in the last (i.e. 4th) round.
Output	1. ArrayList [0, 0, 0, 1] where each index corresponds to a round and the value corresponds to the number of exhausted votes in that round. 2. ArrayList [0, 0, 0, 0] 3. ArrayList [0, 0, 0, 2] 4. ArrayList [0] 5. ArrayList [0, 0, 1, 0] 6. ArrayList [0, 0, 1, 1]
Passed/Failed	All 6 tests passed
Date	12/10/18

Project 2: Agile Scrum	
PBI	Table displaying election progression is outputted to the screen
Task Description	Populate the election table with the correct number of exhausted ballots and votes through each election round.
Testing Number	007
Team Member(s) Responsible	Carlos Alvarenga
Input	The following components of an IRElection instance: - Array of candidates - ArrayList of total exhausted votes through each round - ArrayList of exhausted votes added through each round - Integer representing the total number of votes in the whole election
Tests	1. Test populate() and toString() methods in Table class for relevant components of an IRElection instance corresponding to ballot file “ir_ballot_file_1” as input, where the ballots from a losing candidate are exhausted or redistributed to another candidate. 2. Test populate() and toString() methods in Table class for relevant components of an IRElection instance corresponding to ballot file “ir_ballot_file_2” as input, where all ballots from a losing candidate are redistributed to another candidate. 3. Test populate() and toString() methods in Table class for relevant components of an IRElection instance corresponding to ballot file “ir_ballot_file_3” as input, where there’s a tie between candidates. 4. Test populate() and toString() methods in Table class for relevant components of an IRElection instance corresponding to ballot file “ir_ballot_file_4” as input, where all ballots from a losing candidate are exhausted. 5. Test populate() and toString() methods in Table class for relevant components of an IRElection instance corresponding to ballot file “ir_ballot_file_5” as input, where there’s a winner in the first round and no ballots are exhausted. 6. Test populate() and toString() methods in Table class for relevant components of an IRElection instance corresponding to ballot file “ir_ballot_file_6” as input, where all ballots are invalid. 7. Test populate() and toString() methods in Table class for relevant components of an IRElection instance corresponding to ballot file “ir_ballot_file_7” as input, where all ballots from a losing candidate are exhausted and all ballots from another losing candidate are redistributed to the winning candidate. 8. Test populate() and toString() methods in Table class for relevant components of an IRElection instance corresponding to ballot file “ir_ballot_file_8” as input, where all ballots from a losing candidate are exhausted and ballots from another losing candidate are either exhausted or redistributed to the winning candidate. 9. Test populate() and toString() methods in Table class for relevant components of an IRElection instance corresponding to table example provided in the project write up as input.

Output	1. The text table generated from the toString() method is correctly formatted with accurate information denoting how candidate Rosen wins with 4 votes in 4 rounds.. 2. The text table generated from the toString() method is correctly formatted with accurate information denoting how candidate Rosen wins with 4 votes in 4 rounds.. 3. Two possible text tables can be generated from the toString() method. Both are correctly formatted with accurate information. One denotes how candidate Kleinberg wins with 5 votes in 4 rounds and the other denotes how candidate Rosen wins with 6 votes in 4 rounds. 4. The text table generated from the toString() method is correctly formatted with accurate information denoting how candidate Rosen wins with 3 votes in 4 rounds. 5. The text table generated from the toString() method is correctly formatted with accurate information denoting how candidate Rosen wins with 6 votes in 1 round.. 6. The text table has 0's populated across all fields and 4 rounds since all ballots are invalid. So, any candidate can win due to random chance when ties occur. 7. The text table generated from the toString() method is correctly formatted with accurate information denoting how candidate Rosen wins with 5 votes in 4 rounds. 8. The text table generated from the toString() method is correctly formatted with accurate information denoting how candidate Rosen wins with 4 votes in 4 rounds. 9. The text table generated from the toString() method is correctly formatted with accurate information denoting how candidate Bob Kiss wins with 4313 votes in 3 rounds.
Passed/Failed	All 9 tests passed
Date	12/10/18

Project 2: Agile Scrum	
PBI	Table displaying election progression is outputted to the screen
Task Description	The election table is outputted to the screen with the correct information regarding the number of votes and exhausted ballots across the election rounds.
Testing Number	008
Team Member(s) Responsible	Carlos Alvarenga
Input	Ballot file name
Tests	1. Test getTable() method in IRElection class for ballot file “ir_ballot_file_1” as input, where the ballots from a losing candidate are exhausted or redistributed to another candidate. 2. Test getTable() method in IRElection class for ballot file “ir_ballot_file_2” as input, where all ballots from a losing candidate are redistributed to another candidate. 3. Test getTable() method in IRElection class for ballot file “ir_ballot_file_3” as input, where there’s a tie between candidates. 4. Test getTable() method in IRElection class for ballot file “ir_ballot_file_4” as input, where all ballots from a losing candidate are exhausted. 5. Test getTable() method in IRElection class for ballot file “ir_ballot_file_5” as input, where there’s a winner in the first round and no ballots are exhausted. 6. Test getTable() method in IRElection class for ballot file “ir_ballot_file_6” as input, where all ballots are invalid. 7. Test getTable() method in IRElection class for ballot file “ir_ballot_file_7” as input, where all ballots from a losing candidate are exhausted and all ballots from another losing candidate are redistributed to the winning candidate. 8. Test getTable() method in IRElection class for ballot file “ir_ballot_file_8” as input, where all ballots from a losing candidate are exhausted and ballots from another losing candidate are either exhausted or redistributed to the winning candidate.
Output	1. The returned text table is correctly formatted with accurate information denoting how candidate Rosen wins with 4 votes in 4 rounds.. 2. The returned text table is correctly formatted with accurate information denoting how candidate Rosen wins with 4 votes in 4 rounds.. 3. Two possible text tables can be returned. Both are correctly formatted with accurate information. One denotes how candidate Kleinberg wins with 5 votes in 4 rounds and the other denotes how candidate Rosen wins with 6 votes in 4 rounds. 4. The returned text table is correctly formatted with accurate information denoting how candidate Rosen wins with 3 votes in 4 rounds.

	5. The returned text table is correctly formatted with accurate information denoting how candidate Rosen wins with 6 votes in 1 round.. 6. The returned text table has 0's populated across all fields and 4 rounds since all ballots are invalid. So, any candidate can win due to random chance when ties occur. 7. The returned text table is correctly formatted with accurate information denoting how candidate Rosen wins with 5 votes in 4 rounds. 8. The returned text table is correctly formatted with accurate information denoting how candidate Rosen wins with 4 votes in 4 rounds.
Passed/Failed	All 8 tests passed
Date	12/10/18

Project 2: Agile Scrum	
PBI	Graphical User Interface (GUI) prompt for filename
Task Description	A GUI window appears only when requested by the user.
Testing Number	009
Team Member(s) Responsible	Carlos Alvarenga
Input	None
Tests	1. Test that GUI window appears when running VotingSystem class when not providing an argument in the terminal command-line interface as a ballot filename parameter. 2. Test that GUI window doesn't appear when running VotingSystem class when providing an argument in the terminal command-line interface as a ballot filename parameter.
Output	1. GUI window appears with the following fields and buttons: a. "Enter Filename" where user can type in filename. b. "Search for File" and "Search" button for user to search for a file on disk. c. "Cancel" button to exit the window. d. "OK" button to confirm file selection. 2. No GUI window appears
Passed/Failed	All 2 tests passed
Date	12/10/18

Project 2: Agile Scrum	
PBI	Graphical User Interface (GUI) prompt for filename
Task Description	GUI accurately capture user input in the text field
Testing Number	010
Team Member(s) Responsible	Carlos Alvarenga
Input	Ballot file name
Tests	1. Test requestBallotFilename() method in UserInterface class (which calls getUserInput() method in GUI class) by selecting “Enter Filename” option in GUI window and providing string “ir_ballot_file_1.csv” as input in the text field. 2. Test requestBallotFilename() method in UserInterface class (which calls getUserInput() method in GUI class) by selecting “Enter Filename” option in GUI window and providing no input in the text field.
Output	1. Returned string is “ir_ballot_file_1.csv” 2. Returned string is “”
Passed/Failed	All 2 tests passed
Date	12/10/18

Project 2: Agile Scrum	
PBI	Graphical User Interface (GUI) prompt for filename
Task Description	GUI accurately capture user input through the search-for-file feature that allows the user to select a file on disk
Testing Number	011
Team Member(s) Responsible	Carlos Alvarenga
Input	Ballot file
Tests	1. Test requestBallotFilename() method in UserInterface class (which calls getUserInput() method in GUI class) by selecting “Search for File” option in GUI window and selecting the file named “ir_ballot_file_1.csv” 2. Test requestBallotFilename() method in UserInterface class (which calls getUserInput() method in GUI class) by selecting “Search for File” option in GUI window and not selecting a file on disk.
Output	1. Returned string is “ir_ballot_file_1.csv” 2. Returned string is “”
Passed/Failed	All 2 tests passed
Date	12/10/18

Project 2: Agile Scrum	
PBI	Graphical User Interface (GUI) prompt for filename
Task Description	GUI window is closed when the user specified it by clicking on the “Cancel”
Testing Number	012
Team Member(s) Responsible	Carlos Alvarenga
Input	User presses “Cancel” button
Tests	1. Test cancelButtonPressed() method in UserInterface class by selecting “Cancel” button in GUI window 2. Test cancelButtonPressed() method in UserInterface class by selecting “OK” button in GUI window
Output	1. Returned value is true and the GUI window is closed 2. Returned value is false since the user didn’t provide a file, but didn’t cancel the GUI window option
Passed/Failed	All 2 tests passed
Date	12/10/18

Project 2: Agile Scrum	
PBI	Graphical User Interface (GUI) prompt for filename
Task Description	GUI window doesn't reappear after being closed or after a valid file is provided
Testing Number	013
Team Member(s) Responsible	Carlos Alvarenga
Input	Ballot filename or user presses "Cancel" button
Tests	1. Test getHasBeenRun() method in GUI when the following events occur: 1.1. Run VotingSystem program without command-line argument 1.2. "Enter Filename" option is selected 1.3. "ir_ballot_file_1.csv" is inputted in the text field 1.4. "OK" button is selected 2. Test getHasBeenRun() method in GUI when the following events occur: 2.1. Run VotingSystem program without command-line argument 2.2. "Enter Filename" option is selected 2.3. "ir_ballot_file_1.csv" is inputted in the text field 2.4. "Cancel" button is selected 3. Test getHasBeenRun() method in GUI when the following events occur: 3.1. Run VotingSystem program without command-line argument 3.2. "Search for File" option is selected 3.3. "Search" button is selected 3.4. "ir_ballot_file_1.csv" is selected 3.5. "Open" button is selected to confirm selection 3.6. "OK" button is selected 4. Test getHasBeenRun() method in GUI when the following events occur: 4.1. Run VotingSystem program without command-line argument 4.2. "Search for File" option is selected 4.3. "Search" button is selected 4.4. "ir_ballot_file_1.csv" is selected 4.5. "Open" button is selected to confirm selection 4.6. "Cancel" button is selected 5. Test getHasBeenRun() method in GUI when the following events occur: 5.1. Run VotingSystem program without command-line argument 5.2. "Cancel" button is selected

Output	1. Returned value is true, VotingSystem program terminates gracefully and the GUI window is closed 2. Returned value is true, VotingSystem program terminates gracefully and the GUI window is closed 3. Returned value is true, VotingSystem program terminates gracefully and the GUI window is closed 4. Returned value is true, VotingSystem program terminates gracefully and the GUI window is closed 5. Returned value is true, VotingSystem program terminates gracefully and the GUI window is closed
Passed/Failed	All 5 tests passed
Date	12/10/18

Project 2: Agile Scrum	
PBI	Graphical User Interface (GUI) prompt for filename
Task Description	GUI window reappears if an invalid file is provided
Testing Number	014
Team Member(s) Responsible	Carlos Alvarenga
Input	Ballot filename
Tests	1. Test that GUI windows reappears if the following events occur: 1.1. Run VotingSystem program without command-line argument 1.2. “Enter Filename” option is selected 1.3. “invalidname” is inputted in the text field 1.4. “OK” button is selected 2. Test that GUI windows reappears if the following events occur: 2.1. Run VotingSystem program without command-line argument 2.2. “Enter Filename” option is selected 2.3. Nothing is inputted in the text field 2.4. “OK” button is selected 3. Test that GUI windows reappears if the following events occur: 3.1. Run VotingSystem program without command-line argument 3.2. “Search for File” option is selected 3.3. “Search” button is selected 3.4. “GUI.java” is selected 3.5. “Open” button is selected to confirm selection 3.6. “OK” button is selected 4. Test that GUI windows reappears if the following events occur: 4.1. Run VotingSystem program without command-line argument 4.2. “Search for File” option is selected 4.3. “Search” button is selected 4.4. “Cancel” button is selected to exit out of this option 4.5. “OK” button is selected 5. Test that GUI windows reappears if the following events occur: 5.1. Run VotingSystem program without command-line argument 5.2. “OK” button is selected
Output	1. Exception message “(No such file or directory)” is outputted to the console log and GUI window reappears, displaying the “main menu” prompting the user for the file

	2. Exception message “(No such file or directory)” is outputted to the console log and GUI window reappears, displaying the “main menu” prompting the user for the file 3. Exception message “(No such file or directory)” is outputted to the console log and GUI window reappears, displaying the “main menu” prompting the user for the file 4. Exception message “(No such file or directory)” is outputted to the console log and GUI window reappears, displaying the “main menu” prompting the user for the file 5. Exception message “(No such file or directory)” is outputted to the console log and GUI window reappears, displaying the “main menu” prompting the user for the file
Passed/Failed	All 5 tests passed
Date	12/10/18

Project 2: Agile Scrum	
PBI	Graphical User Interface (GUI) prompt for filename
Task Description	GUI functionality is correctly integrated with the VotingSystem program to accurately run elections
Testing Number	015
Team Member(s) Responsible	Carlos Alvarenga
Input	Ballot filename
Tests	1. Test GUI functionality and integration with VotingSystem class when the following events occur: a. Run VotingSystem program without command-line argument b. “Enter Filename” option is selected c. “ir_ballot_file_1.csv” is inputted in the text field d. “OK” button is selected 2. Test GUI functionality and integration with VotingSystem class when the following events occur: a. Run VotingSystem program without command-line argument b. “Enter Filename” option is selected c. “Search for File” option is selected d. “Enter Filename” option is selected e. “simple_opl_ballot_file” is inputted in the text field f. “OK” button is selected 3. Test GUI functionality and integration with VotingSystem class when the following events occur: a. Run VotingSystem program without command-line argument b. “Enter Filename” option is selected c. “testing/ir_ballot_file_1.csv”, where “testing” is a subdirectory of the one that holds the program files, is inputted in the text field d. “OK” button is selected 4. Test GUI functionality and integration with VotingSystem class when the following events occur: a. Run VotingSystem program without command-line argument b. “Search for File” option is selected c. “Search” button is selected d. “simple_opl_ballot_file” in the current directory is selected selected e. “Open” button is selected to confirm selection f. “OK” button is selected 5. Test GUI functionality and integration with VotingSystem class when the following events occur: a. Run VotingSystem program without command-line argument b. “Search for File” option is selected c. “Search” button is selected

	d. “ir_ballot_file_1.csv” in the current directory is selected selected e. “Open” button is selected to confirm selection f. “Enter Filename” option is selected g. “Search for File” option is selected h. “Search” button is selected i. “ir_ballot_file_1.csv” in the current directory is selected selected j. “Open” button is selected to confirm selection k. “OK” button is selected 6. Test GUI functionality and integration with VotingSystem class when the following events occur: a. Run VotingSystem program without command-line argument b. “Search for File” option is selected c. “Search” button is selected d. “ir_ballot_file_1.csv” in the current directory is selected selected e. “Cancel” button is selected to exit out of the search-for-file option f. “Enter Filename” option is selected g. “Search for File” option is selected h. “Search” button is selected i. “ir_ballot_file_1.csv” in the current directory is selected selected j. “Open” button is selected to confirm selection k. “OK” button is selected 7. Test GUI functionality and integration with VotingSystem class when the following events occur: a. Run VotingSystem program without command-line argument b. “Search for File” option is selected c. “Search” button is selected d. I navigate to the “testing” directory which isn’t a subdirectory of the folder where the program files resides e. “ir_ballot_file_1.csv” is selected in the “testing” folder f. “Open” button is selected to confirm selection g. “OK” button is selected
Output	1. Election is run correctly, with the appropriate output files generated with the correct content and accurate results being displayed in the console long, and the program gracefully terminates 2. Election is run correctly, with the appropriate output files generated with the correct content and accurate results being displayed in the console long, and the program gracefully terminates 3. Election is run correctly, with the appropriate output files generated with the correct content and accurate results being displayed in the console long, and the program gracefully terminates 4. Election is run correctly, with the appropriate output files generated with the correct content and accurate results being displayed in the console long, and the program gracefully terminates 5. Election is run correctly, with the appropriate output files generated with the correct content and accurate results being displayed in the console long, and the program gracefully terminates

	6. Election is run correctly, with the appropriate output files generated with the correct content and accurate results being displayed in the console long, and the program gracefully terminates 7. Election is run correctly, with the appropriate output files generated with the correct content and accurate results being displayed in the console long, and the program gracefully terminates
Passed/Failed	All 7 tests passed
Date	12/10/18

Project 2: Agile Scrum	
PBI	Command Line Argument for Filename
Task Description	Remove the system prompt for the filename of the ballotfile if a command line argument was specified.
Testing Number	016
Team Member(s) Responsible	Justin Koo
Input	String representing the filename of the ballotfile.
Tests	1. Test that system prompt does not show up if a command line argument is supplied 1.1. Run VotingSystem with command line argument “simple_ir_ballot_file.csv” 2. Test that the system prompt (GUI) does show up if a command line argument is not supplied. 2.2 Run VotingSystem without command line argument.
Output	1. Command line displays results of the election along with a table of how the election progressed. No prompt for a filename appears. 2. The GUI appears allowing users to select a file from a directory or to enter one manually.
Passed/Failed	All 2 tests passed
Date	12/14/18