Project Name: Project 1: Voting System **Team# 02**

Test Stage: Unit _x_   System __
Test Case ID#: EDCVD_1a
Test Description: This will test if the method verifyLineIsDigit(String line) correctly verify if the strings with only numbers pass

Test Date: 3/25/24

Name(s) of Testers: Bethany Freeman

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/test/java/ExtractDataCPLTest.java
verifyLineIsDigit()

Automated: yes__x_   no ___

Results: Pass ____x_   Fail_________

Preconditions for Test: An ExtractDataCPL object must have been created

Step # 1	Test Step Description	Test Data	Expected Result	Actual Result	Notes
2	Create a new ExtractDataCPL object	ExtractDataCPL test01= new ExtractDataCPL(validFile01, "CPL")	Object created	Object created	
3	Create a String line	String line = "";	variable created	variable created	
4	assert that false is given when test01.verifyLineIsDigit(line) is called	false, test01.verifyLineIsDigit(line)	true	true	

Post condition(s) for Test:

VerifyLineIsDigit(String) has been verified to not work with an empty string

Project Name: Project 1: Voting System **Team# 02**

Test Stage: Unit _x_ System __
Test Case ID#: EDCVD_1b
Test Description: This will test if the method verifyLineIsDigit(String line) correctly verify if the strings with only numbers pass

Test Date: 3/25/24

Name(s) of Testers: Bethany Freeman

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/test/java/ExtractDataCPLTest.java
verifyLineIsDigit()

Automated: yes__x_ no ___

Results: Pass ____x_ Fail_________

Preconditions for Test: An ExtractDataCPL object must have been created

Step # 1	Test Step Description	Test Data	Expected Result	Actual Result	Notes
2	Create a new ExtractDataCPL object	ExtractDataCPL test01= new ExtractDataCPL(validFile01, "CPL")	Object created	Object created	
3	Create a String line	String line = "1";	variable created	variable created	
4	assert that true is given when test01.verifyLineIsDigit(line) is called	true, test01.verifyLineIsDigit(line)	true	true	

Post condition(s) for Test:
VerifyLineIsDigit(String) has been verified to work with just a number

Project Name: Project 1: Voting System **Team# 02**

Test Stage: Unit _x_ System __
Test Case ID#: EDCVD_1c
Test Description: This will test if the method verifyLineIsDigit(String line) correctly verify if the strings with only numbers pass

Test Date: 3/25/24

Name(s) of Testers: Bethany Freeman

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/test/java/ExtractDataCPLTest.java
verifyLineIsDigit()

Automated: yes x no ___

Results: Pass ____x_ Fail________

Preconditions for Test: An ExtractDataCPL object must have been created

Step # 1	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
2	Create a new ExtractDataCPL object	ExtractDataCPL test01= new ExtractDataCPL(validFile01, "CPL")	Object created	Object created	
3	Create a String line	String line = " 1"	variable created	variable created	
4	assert that true is given when test01.verifyLineIsDigit(line) is called	true, test01.verifyLineIsDigit(line)	true	true	

Post condition(s) for Test:
VerifyLineIsDigit(String) has been verified to work with a string that is a digit and space

Project Name: Project 1: Voting System **Team# 02**

Test Stage: Unit _x_ System __

Test Date: 3/25/24

Test Case ID#: EDCVD_1d

Test Description: This will test if the method verifyLineIsDigit(String line) correctly verify if the strings with only numbers pass

Name(s) of Testers: Bethany Freeman

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/test/java/ExtractDataCPLTest.java
verifyLineIsDigit()

Automated: yes x no ___

Results: Pass ____x_ Fail________

Preconditions for Test: An ExtractDataCPL object must have been created

Step # 1	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
2	Create a new ExtractDataCPL object	ExtractDataCPL test01= new ExtractDataCPL(validFile01, "CPL")	Object created	Object created	
3	Create a String line	String line = "abc";	variable created	variable created	
4	assert that false is given when test01.verifyLineIsDigit(line) is called	false, test01.verifyLineIsDigit(line)	true	true	

Post condition(s) for Test:

VerifyLineIsDigit(String) has been verified to not work with a string of letters

Project Name: Project 1: Voting System **Team# 02**

Test Stage: Unit _x_ System __ **Test Date: 3/25/24**

Test Case ID#: EDCVD_1e

Test Description: This will test if the method verifyLineIsDigit(String line) correctly verify if the strings with only numbers pass

Name(s) of Testers: Bethany Freeman

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/test/java/ExtractDataCPLTest.java
verifyLineIsDigit()

Automated: yes __x__ no ___

Results: Pass ____x_ Fail________

Preconditions for Test: An ExtractDataCPL object must have been created

Step # 1	Test Step Description	Test Data	Expected Result	Actual Result	Notes
2	Create a new ExtractDataCPL object	ExtractDataCPL test01= new ExtractDataCPL(validFile01, "CPL")	Object created	Object created	
3	Create a String line	String line = "12b";	variable created	variable created	
4	assert that false is given when test01.verifyLineIsDigit(line) is called	false, test01.verifyLineIsDigit(line)	true	true	

Post condition(s) for Test:

VerifyLineIsDigit(String) has been verified to not work with string of digits and letters

Project Name: Project 1: Voting System **Team# 02**

Test Stage: Unit _x_ System __ **Test Date: 3/26/24**

Test Case ID#: EDCFP_2a **Name(s) of Testers: Bethany Freeman**

Test Description: Format parties into partyCandidates with the correct number of parties passed in

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/test/java/ExtractDataCPL
formatPartyInformation()

Automated: yes__x_ no ___

Results: Pass ___x__ Fail________

Preconditions for Test: an initialized partyVotes and candidateVotes, empty. a file that exists. an ExtractDataCPL Object

Step # 1	Test Step Description	Test Data	Expected Result	Actual Result	Notes
2	Create a new BufferedReader Object	validFile01 = new BufferedReader(new FileReader(new File("src/testing/java/InputFiles/CPLPartyInfo01.txt")))	Object Created	Object Created	
3	Create ExtractDataCPL Object	test01 = new ExtractDataCPL(validFile01, "CPL")	Object Created	Object Created	
4	Call formatPartyInformation	partyCandidates = test01.formatPartyInformation(6, partyVotes, candidateVotes);	Object instantiated	Object Instantiated	

5	Create HashMap<String, ArrayList<String>>	HashMap<String, ArrayList<String>> expected = new HashMap<String, ArrayList<String>>();
		expected.put("Democratic", new ArrayList<>(Arrays.asList(" Mary", " Jane", " Kim")));
		expected.put("Republican", new ArrayList<>(Arrays.asList(" Allen", " Joe", " Sarah")));
		expected.put("Green", new ArrayList<>(Arrays.asList(" Sally", " Nikki")));
		expected.put("Independant", new ArrayList<>(Arrays.asList(" Mike")));
		expected.put("Grass", new ArrayList<>(Arrays.asList(" Mars", " Jacob")));
		expected.put("Pluto", new ArrayList<>(Arrays.asList(" Space", " Stars")));
			Object Created	Object Created	

6	Assert Expected equals party candidates	expected, partyCandidates	true	true	

Post condition(s) for Test:

formatPartyInformation() given the correct number of parties will correctly format the parties into a Hashmap

Project Name: Project 1: Voting System **Team# 02**

Test Stage: Unit _x_ System __

Test Date: 3/26/24

Test Case ID#: EDCFP_2b

Name(s) of Testers: Bethany Freeman

Test Description: Format parties into partyCandidates with the wrong number of parties passed in

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/test/java/ExtractDataCPL
formatPartyInformation()

Automated: yes__x_ no ___

Results: Pass ___x__ Fail_________

Preconditions for Test: an initialized partyVotes and candidateVotes, empty. a file that exists. an ExtractDataCPL Object

Step # 1	Test Step Description	Test Data	Expected Result	Actual Result	Notes
2	Create a new BufferedReader Object	validFile01 = new BufferedReader(new FileReader(new File("src/testing/java/InputFiles/CPLPartyInfo02.txt")))	Object Created	Object Created	
3	Create ExtractDataCPL Object	test01 = new ExtractDataCPL(validFile01, "CPL")	Object Created	Object Created	
4	Call formatPartyInformation	partyCandidates = test01.formatPartyInformation(0, partyVotes, candidateVotes);	Object instantiated	Object Instantiated	
5	Create HashMap<String, ArrayList<String>>	expected = new HashMap<String, ArrayList<String>>();	Object Created	Object Created	
6	Assert Expected equals party candidates	expected, partyCandidates	true	true	

Post condition(s) for Test:

formatPartyInformation() given the wrong number of parties will incorrectly format the parties into a Hashmap

Project Name: Project 1: Voting System **Team# 02**

Test Stage: Unit _x_ System __ **Test Date: 3/26/24**

Test Case ID#: EDCFP_2c **Name(s) of Testers: Bethany Freeman**

Test Description: Format parties into partyCandidates when a party has no candidates

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/test/java/ExtractDataCPL
formatPartyInformation()

Automated: yes__x_ no ___

Results: Pass ___x__ Fail________

Preconditions for Test: an initialized partyVotes and candidateVotes, empty. a file that exists. an ExtractDataCPL Object

Step # 1	Test Step Description	Test Data	Expected Result	Actual Result	Notes
2	Create a new BufferedReader Object	validFile01 = new BufferedReader(new FileReader(new File("src\\testing\\java\\InputFiles\\CPLPartyInfo03.txt")))	Object Created	Object Created	
3	Create ExtractDataCPL Object	test01 = new ExtractDataCPL(validFile01, "CPL")	Object Created	Object Created	
4	Call formatPartyInformation	partyCandidates = test01.formatPartyInformation(6, partyVotes, candidateVotes);	Object instantiated	Object Instantiated	

5	Create HashMap<String, ArrayList<String>>	expected = new HashMap<String, ArrayList<String>>();
		expected.put("Democratic", new ArrayList<>(Arrays.asList(" Mary", " Jane", " Kim")));
		expected.put("Republican", new ArrayList<>(Arrays.asList(" Allen", " Joe", " Sarah")));
		expected.put("Green", new ArrayList<>(Arrays.asList(" Sally", " Nikki")));
		expected.put("Independant", new ArrayList<>());
		expected.put("Grass", new ArrayList<>(Arrays.asList(" Mars", " Jacob")));
		expected.put("Pluto", new ArrayList<>(Arrays.asList(" Space", " Stars")));
			Object Created	Object Created	

6	Assert Expected equals party candidates	expected, partyCandidates	true	true	

Post condition(s) for Test:
formatPartyInformation() given a party that has no candidates will format the parties into a Hashmap where that party has an empty arraylist attached to it

Project Name: Project 1: Voting System **Team# 02**

Test Stage: Unit _x_ System __

Test Date: 3/26/24

Test Case ID#: EDCFBI_3a

Name(s) of Testers: Bethany Free.am

Test Description: Format Balloting into partyVotes for a file with correct Ballot formatting

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/test/java/ExtractDataCPLTest
FormatBallotInformation()

Automated: yes_x__ no ___

Results: Pass ___x__ Fail_________

Preconditions for Test: formatPartyInformation() already proven to work, a file that exists in the relative path given
An ExtractDataCPL Object, partyVotes is initialized to an empty ArrayList<>, candidateVotes is initialized to an empty arrayList

Step # 1	Test Step Description	Test Data	Expected Result	Actual Result	Notes
2	Create a new BufferedReader Object	validFile01 = new BufferedReader(new FileReader(new File("src/testing/java/InputFiles/CPLBallotTest01.txt")))	Object created	Object Created	
3	Create a new ExtractDataCPL Object	test01 = new ExtractDataCPL(validFile01, "CPL")	Object Created	Object Created	
4	Instantiate partyCandidates using formatPartyInformation()	partyCandidates = test01.formatPartyInformation(3, partyVotes, candidateVotes)	Object Instantiated	Object Instantiated	
5	Call formatBallotInformation	test01.formatBallotInformation(partyVotes, candidateVotes, partyCandidates);	Nothing	Nothing	

6	Created ArrayList<ArrayList<Object>>	ArrayList<ArrayList<Object>> expectedPartyVotes = new ArrayList<>();
		expectedPartyVotes.add(new ArrayList<>(Arrays.asList("Grass", 26646)));
		expectedPartyVotes.add(new ArrayList<>(Arrays.asList("Pluto", 26742)));
		expectedPartyVotes.add(new ArrayList<>(Arrays.asList("Republican", 26612)));
			Object Created	Object Created	

7	Assert expectedPartyVotes equals PartyVotes	expectedPartyVotes, partyVotes	true	true	

Post condition(s) for Test:

PartyVotes will have been correctly created based on ballots

Project Name: Project 1: Voting System **Team# 02**

Test Stage: Unit _x_ System __ **Test Date: 3/26/24**

Test Case ID#: EDCFBI_3b **Name(s) of Testers: Bethany Free.am**

Test Description: Format Balloting into partyVotes for a file with incorrect Ballot formatting on at least one of the votes

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/test/java/ExtractDataCPLTest
FormatBallotInformation()

Automated: yes_x__ no ___

Results: Pass ___x__ Fail_________

Preconditions for Test: formatPartyInformation() already proven to work, a file that exists in the relative path given
An ExtractDataCPL Object, partyVotes is initialized to an empty ArrayList<>, candidateVotes is initialized to an empty arrayList

Step # 1	Test Step Description	Test Data	Expected Result	Actual Result	Notes
2	Create a new BufferedReader Object	validFile01 = new BufferedReader(new FileReader(new File("src\\testing\\java\\InputFiles\\CPLBallotTest02.txt")))	Object created	Object Created	
3	Create a new ExtractDataCPL Object	test01 = new ExtractDataCPL(validFile01, "CPL")	Object Created	Object Created	
4	Instantiate partyCandidates using formatPartyInformation()	partyCandidates = test01.formatPartyInformation(3, partyVotes, candidateVotes)	Object Instantiated	Object Instantiated	
6	Assert that IOException is thrown	IOException, test01.formatBallotInformation(partyVotes, candidateVotes, partyCandidates)	true	true	

Post condition(s) for Test:
formatBallotInformation will have thrown an exception with the message "File format is not in the correct format"

Project Name: Project 1: Voting System **Team# 02**

Test Stage: Unit _x_ System __

Test Date: 3/26/24

Test Case ID#: EDCFBI_3c

Test Description: Format Balloting into partyVotes for a file where the line is null

Name(s) of Testers: Bethany Free.am

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/test/java/ExtractDataCPLTest
FormatBallotInformation()

Automated: yes_x__ no ___

Results: Pass ___x__ Fail________

Preconditions for Test: formatPartyInformation() already proven to work, a file that exists in the relative path given An ExtractDataCPL Object, partyVotes is initialized to an empty ArrayList<>, candidateVotes is initialized to an empty arrayList

Step # 1	Test Step Description	Test Data	Expected Result	Actual Result	Notes
2	Create a new BufferedReader Object	validFile01 = new BufferedReader(new FileReader(new File("src\\testing\\java\\InputFiles\\CPLPartyInfo01.txt")))	Object created	Object Created	
3	Create a new ExtractDataCPL Object	test01 = new ExtractDataCPL(validFile01, "CPL")	Object Created	Object Created	
4	Instantiate partyCandidates using formatPartyInformation()	partyCandidates = test01.formatPartyInformation(3, partyVotes, candidateVotes)	Object Instantiated	Object Instantiated	
6	Assert that IOException is thrown	IOException, test01.formatBallotInformation(partyVotes, candidateVotes, partyCandidates)	true	true	

Post condition(s) for Test:

formatBallotInformation will have thrown an exception with the message "File format is not in the correct format"

Project Name: Project 1: Voting System **Team# 02**

Test Stage: Unit _x_ System __ **Test Date: 3/26/24**

Test Case ID#: EDCFBI_3d
Test Description: Format Balloting into partyVotes for a file where all the votes have no '1'

Name(s) of Testers: Bethany Free.am

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/test/java/ExtractDataCPLTest
FormatBallotInformation()

Automated: yes_x__ no ___

Results: Pass ___x__ Fail_________

Preconditions for Test: formatPartyInformation() already proven to work, a file that exists in the relative path given An ExtractDataCPL Object, partyVotes is initialized to an empty ArrayList<>, candidateVotes is initialized to an empty arrayList

Step # 1	Test Step Description	Test Data	Expected Result	Actual Result	Notes
2	Create a new BufferedReader Object	validFile01 = new BufferedReader(new FileReader(new File("src\\testing\\java\\InputFiles\\CPLBallotTest03.txt")))	Object created	Object Created	
3	Create a new ExtractDataCPL Object	test01 = new ExtractDataCPL(validFile01, "CPL")	Object Created	Object Created	
4	Instantiate partyCandidates using formatPartyInformation()	partyCandidates = test01.formatPartyInformation(3, partyVotes, candidateVotes)	Object Instantiated	Object Instantiated	
6	Assert that IOException is thrown	IOException, test01.formatBallotInformation(partyVotes, candidateVotes, partyCandidates)	true	true	

Post condition(s) for Test:
formatBallotInformation will have thrown an exception with the message "File format is not in the correct format"

Project Name: Project 1: Voting System **Team# 02**

Test Stage: Unit _x_ System __ **Test Date: 3/26/24**

Test Case ID#: EDCFBI_3e **Name(s) of Testers: Bethany Free.am**

Test Description: Format Balloting into partyVotes for a file with correct Ballot formatting

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/test/java/ExtractDataCPLTest
FormatBallotInformation()

Automated: yes_x__ no ___

Results: Pass ___x__ Fail________

Preconditions for Test: formatPartyInformation() already proven to work, a file that exists in the relative path given An ExtractDataCPL Object, partyVotes is initialized to an empty ArrayList<>, candidateVotes is initialized to an empty arrayList

Step # 1	Test Step Description	Test Data	Expected Result	Actual Result	Notes
2	Create a new BufferedReader Object	validFile01 = new BufferedReader(new FileReader(new File("src\\testing\\java\\InputFiles\\CPLBallotTest04.txt")))	Object created	Object Created	
3	Create a new ExtractDataCPL Object	test01 = new ExtractDataCPL(validFile01, "CPL")	Object Created	Object Created	
4	Instantiate partyCandidates using formatPartyInformation()	partyCandidates = test01.formatPartyInformation(4, partyVotes, candidateVotes)	Object Instantiated	Object Instantiated	
5	Call formatBallotInformation	test01.formatBallotInformation(partyVotes, candidateVotes, partyCandidates);	Nothing	Nothing	

6	Created ArrayList<ArrayList<Object>>	expectedPartyVotes = new ArrayList<>();
		expectedPartyVotes.add(new ArrayList<>(Arrays.asList("Green", 22648)));
		expectedPartyVotes.add(new ArrayList<>(Arrays.asList("Independant", 22613)));
		expectedPartyVotes.add(new ArrayList<>(Arrays.asList("Grass", 22231)));
		expectedPartyVotes.add(new ArrayList<>(Arrays.asList("Pluto", 22508)))
			Object Created	Object Created	

7	Assert expectedPartyVotes equals PartyVotes	expectedPartyVotes, partyVotes	true	true	

Post condition(s) for Test:

PartyVotes will have been correctly created based on ballots

Project Name: Project 1: Voting System **Team# 02**

Test Stage: Unit _x_ System __

Test Date: 3/26/2024

Test Case ID#: EDCEF_4

Test Description: This checks to see if ExtractFromFile() will throw an IOException since there are not enough digits in the given file

Name(s) of Testers: Bethany Freeman

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/test/java/ExtractDataCPL.java
extractFromFile()

Automated: yes __x__ no ___

Results: Pass __x___ Fail________

Preconditions for Test: A file with the header “CPL” has been passed in, the file exists, a ExtractDataCPL object was created

Step # 1	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
2	set validFile01 to a new BufferedReader object	validFile01 = new BufferedReader(new File("src\\testing\\java\\InputFiles\\CPLInput01.txt")))	Object created	Object Created	
3	read the first line to get to the correct spot in the file and to get the header	String header = validFile01.readLine();	variable created	variable created	
4	create a new extractDataCPL object	test01 = new ExtractDataCPL(validFile01, header);	Object created	Object created	
5	Create a new FileData Object	FileData test = test01.extractFromFile();	Object created	Object created	
6	Assert that test.getElectionType() is “CPL”	"CPL", test.getElectionType()	true	true	
7	Assert that test.getNumberSeats() is 3	3, test.getNumberSeats()	true	true	
8	Assert that test.getNumberBallots() is 100000	100000, test.getNumberBallots()	true	true	
9	Assert that test.getNumberParties is 4	4, test.getNumberParties()	true	true	
10	Create HashMap<String, ArrayList<String>> Object	partyCandidates = new HashMap<>(); partyCandidates.put("Democratic", new ArrayList<>(Arrays.asList("Mary", "Jane", "Kim")));	Object created	Object Created	

		partyCandidates.put("Republican", new ArrayList<>(Arrays.asList("Allen, Joe, Sarah")));
		partyCandidates.put("Green", new ArrayList<>(Arrays.asList("Sally", "Nikki"))); partyCandidates.put("Independent", new ArrayList<>(Arrays.asList("Mike")));			

11	Assert that test.getPartyCandidates().containsKey("Democratic") is true	true, test.getPartyCandidates().containsKey("Democratic")	true	true	
12	Assert that test.getPartyCandidates().containsKey("Republican") is true	true, test.getPartyCandidates().containsKey("Republican")	true	true	
13	Assert that test.getPartyCandidates().containsKey("Green") is true	true, test.getPartyCandidates().containsKey("Green")	true	true	
14	Assert that test.getPartyCandidates().containsKey("Independant") is true	true, test.getPartyCandidates().containsKey("Independant")	true	true	
15	Assert that test.getPartyCandidates().get("Democratic").toString() is "[Mary, Jane, Kim]"	"[Mary, Jane, Kim]", test.getPartyCandidates().get("Democratic").toString()	true	true	
16	Assert that test.getPartyCandidates().get("Republican").toString() is "[Allen, Joe, Sarah]"	"[Allen, Joe, Sarah]", test.getPartyCandidates().get("Republican").toString()	true	true	
17	Assert that test.getPartyCandidates().get("Green").toString() is "[Sally, Nikki]"	"[Sally, Nikki]", test.getPartyCandidates().get("Green").toString()	true	true	
18	Assert that test.getPartyCandidates().get("Independant").toString() is "[Mike]"	"[Mike]", test.getPartyCandidates().get("Independant").toString()	true	true	

19	create ArrayList<ArrayList<Object>> Object	partyVotes = new ArrayList<>();
		partyVotes.add(new ArrayList<>(Arrays.asList("Democratic", 24936)));
		partyVotes.add(new ArrayList<>(Arrays.asList("Republican", 25093)));
		partyVotes.add(new ArrayList<>(Arrays.asList("Green", 25067)));
		partyVotes.add(new ArrayList<>(Arrays.asList("Independant", 24904)));
			Object created	Object Created	

20	Assert that partyVotes equals test.getPartyVotes	partyVotes, test.getPartyVotes()	true	true	

21	create ArrayList<ArrayList<Object>> Object	candidateVotes = new ArrayList<>();
		candidateVotes.add(new ArrayList<>(Arrays.asList(" Mary", 0)));
		candidateVotes.add(new ArrayList<>(Arrays.asList(" Jane", 0)));
		candidateVotes.add(new ArrayList<>(Arrays.asList(" Kim", 0)));
		candidateVotes.add(new ArrayList<>(Arrays.asList(" Allen", 0)));
		candidateVotes.add(new ArrayList<>(Arrays.asList(" Joe", 0)));
		candidateVotes.add(new ArrayList<>(Arrays.asList(" Sarah", 0)));
		candidateVotes.add(new ArrayList<>(Arrays.asList(" Sally", 0)));
		candidateVotes.add(new ArrayList<>(Arrays.asList(" Nikki", 0)));
		candidateVotes.add(new ArrayList<>(Arrays.asList(" Mike", 0)));
			Object created	Object Created	

22	Assert that candidateVotes equals test.getCandidateVotes()	candidateVotes, test.getCandidateVotes()	true	true	

Post condition(s) for Test:

A FileData Object will have been successfully created with the contents of the ballot file

Project Name: Project 1: Voting System **Team# 02**

Test Stage: Unit _x_ System __

Test Date: 3/26/2024

Test Case ID#: EDCEF_5

Name(s) of Testers: Bethany Freeman

Test Description: This checks to see if ExtractFromFile() will throw an IOException since there are not enough digits in the given file

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/test/java/ExtractDataCPL.java
extractFromFile()

Automated: yes __x__ no ___

Results: Pass __x___ Fail________

Preconditions for Test: A file with the header "CPL" has been passed in, the file exists, a ExtractDataCPL object was created

Step # 1	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
2	set validFile01 to a new BufferedReader object	validFile01 = new BufferedReader(new FileReader(new File("src\\testing\\java\\InputFiles\\CPLInput02.txt")));	Object created	Object Created	
3	read the first line to get to the correct spot in the file and to get the header	String header = validFile01.readLine();	variable created	variable created	
4	create a new extractDataCPL object	test01 = new ExtractDataCPL(validFile01, header);	Object created	Object created	
5	Assert that extractFromFile throws an IOException	assertThrows(IOException.class, () -> test01.extractFromFile());	True	True	

Post condition(s) for Test:

An IOException with the message "Not enough digits" will be thrown. This is handled later by main.

Project Name: Project 1: Voting System **Team# 02**

Test Stage: Unit _x_ System __

Test Date: 3/26/2024

Test Case ID#: EDCEF_6

Name(s) of Testers: Bethany Freeman

Test Description: This checks to see if ExtractFromFile() will throw an IOException since there are not enough digits in the given file

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/test/java/ExtractDataCPL.java
extractFromFile()

Automated: yes__ x__ no ___

Results: Pass __x___ Fail________

Preconditions for Test: A file with the header "CPL" has been passed in, the file exists, a ExtractDataCPL object was created

Step # 1	Test Step Description	Test Data	Expected Result	Actual Result	Notes
2	set validFile01 to a new BufferedReader object	validFile01 = new BufferedReader(new FileReader(new File("src\\testing\\java\\Input Files\\CPLInput03.txt")))	Object created	Object Created	
3	read the first line to get to the correct spot in the file and to get the header	String header = validFile01.readLine();	variable created	variable created	
4	create a new extractDataCPL object	test01 = new ExtractDataCPL(validFile01, header);	Object created	Object created	
5	Assert that extractFromFile throws an IOException	assertThrows(IOException.class, () -> test01.extractFromFile());	True	True	

Post condition(s) for Test:

An IOException with the message "Not enough digits" will be thrown. This is handled later by main.

Project Name: The project #, name of your system, and the team#

Test Stage: Indicate whether it is a unit test or a system test.

Test Date: The date the test was performed.

Test Case ID#: A unique ID is required. Decide on a naming convention and use numbering. Example: Ballot_Shuffle_1

Name(s) of Testers: List the names of anyone involved in running this test case.

Test Description: Describe briefly the test objective.

Automated: Indicate if the test is completely automated or being checked manually. (If you have methods running the tests and checking results, select "yes". If you are manually checking results, indicate manual by selecting the "no.")

Results: Indicate if the test passed or failed.

Step #: You will be listing the test steps in order. This number is the step number in the process.

Test Step Description: Details of the test step.

Test Data: What the test data will be for this step. Be clear on what the input data will be. If using a specific file, be clear on the name.

Expected Result: What result are you expecting from the program component or system.

Actual Result: What result were returned based on the test.

Post condition for Test: What will be true after the test has been run? Has the state of the system changed in any way?

Notes: Comments and notes for you and your team members.