Project Name: Project 1: Voting System
Team# 02

Test Stage: Unit _x_ System __

Test Case ID#: OPL_ARV_1A

Test Description: Testing a valid index to ensure method properly updates remainingVotes array

Test Date: 3/25/2024

Name(s) of Testers: Derrick

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

Automated: Yes _X_ No ___

Results: Pass _X_ Fail ___

Preconditions for Test: An OPL object has been created

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Define a new int, index01, which will be a valid index to modify within the created OPL object's remainingVotes array	index01 = 1	int was created successfully	int was created successfully	
2	Initialize a new object of type ArrayList<ArrayList<Object>> object, which will represent the ArrayList that should result	ArrayList<ArrayList<Object>> expectedA = new ArrayList<>(); expectedA.add(new ArrayList<>(Arrays.asList("Sarah", 50))); expectedA.add(new ArrayList<>(Arrays.asList("Bob", -26))); expectedA.add(new ArrayList<>(Arrays.asList("Jon", 20))); expectedA.add(new ArrayList<>(Arrays.asList("Craig", 34))); expectedA.add(new ArrayList<>(Arrays.asList("Klein", 2))); expectedA.add(new ArrayList<>(Arrays.asList("Rain", 15))); expectedA.add(new ArrayList<>(Arrays.asList("Water", 61))); expectedA.add(new ArrayList<>(Arrays.asList("Grass", 23))); expectedA.add(new ArrayList<>(Arrays.asList("Ash", 10))); expectedA.add(new ArrayList<>(Arrays.asList("Matt", 65)));	Object was created successfully	Object was created successfully	
3	Call the adjustRemainingVotes method on the opl01 at index01	opl01.adjustRemainingVotes(index01);	remainingVotes was successfully modified	remainingVotes was successfully modified	

4	Assert that the opl01 object's remainingVotes field has been properly updated	assertEquals(opl01.remainingVotes, expectedA);	True	True	

Post condition(s) for Test:

The OPL object has a correctly updated remainingVotes field, which equals the ArrayList<ArrayList<Object>> expectedA

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

Test Stage: Unit _x_ System __ **Test Date: 3/25/2024**

Test Case ID#: OPL_ARV_1B **Name(s) of Testers:** Derrick

Test Description: Testing an invalid (negative) index to ensure method does not alter the opl01's remainingVotes array

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

Automated: Yes _X_ No ___

Results: Pass _X_ Fail ___

Preconditions for Test: An OPL object has been created

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Define a new int, index02, which will be an invalid index that will be used to test whether the remainingVotes array gets incorrectly modified or not	index02 = -1	int was created successfully	int was created successfully	
2	Initialize a new object of type ArrayList<ArrayList<Object>> object, which will represent the ArrayList that should result	ArrayList<ArrayList<Object>> expectedA = new ArrayList<>(); expectedA.add(new ArrayList<>(Arrays.asList("Sarah", 50))); expectedA.add(new ArrayList<>(Arrays.asList("Bob", -26))); expectedA.add(new ArrayList<>(Arrays.asList("Jon", 20))); expectedA.add(new ArrayList<>(Arrays.asList("Craig", 34))); expectedA.add(new ArrayList<>(Arrays.asList("Klein", 2))); expectedA.add(new ArrayList<>(Arrays.asList("Rain", 15))); expectedA.add(new ArrayList<>(Arrays.asList("Water", 61))); expectedA.add(new ArrayList<>(Arrays.asList("Grass", 23))); expectedA.add(new ArrayList<>(Arrays.asList("Ash", 10))); expectedA.add(new ArrayList<>(Arrays.asList("Matt", 65)));	Object was created successfully	Object was created successfully	
3	Call the adjustRemainingVotes method on the opl01 at index02	assertThrows(IOException.class, () -> opl01.adjustRemainingVotes(index02));	IOException	IOException	

Post condition(s) for Test:

An IOException is thrown

Project Name: Project 1: Voting System

Team# 02

Test Stage: Unit _x_ System __ **Test Date:** 3/25/2024

Test Case ID#: OPL_ARV_1C **Name(s) of Testers:** Derrick

Test Description: Testing an invalid (too large) index to ensure method does not alter the opl01's remainingVotes array

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

Automated: Yes _X_ No ___

Results: Pass _X_ Fail ___

Preconditions for Test: An OPL object has been created

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Define a new int, index03, which will be an invalid index that will be used to test whether the remainingVotes array gets incorrectly modified or not	index03 = 15	int was created successfully	int was created successfully	
2	Initialize a new object of type ArrayList<ArrayList<Object>> object, which will represent the ArrayList that should result	ArrayList<ArrayList<Object>> expectedA = new ArrayList<>(); expectedA.add(new ArrayList<>(Arrays.asList("Sarah", 50))); expectedA.add(new ArrayList<>(Arrays.asList("Bob", -26))); expectedA.add(new ArrayList<>(Arrays.asList("Jon", 20))); expectedA.add(new ArrayList<>(Arrays.asList("Craig", 34))); expectedA.add(new ArrayList<>(Arrays.asList("Klein", 2))); expectedA.add(new ArrayList<>(Arrays.asList("Rain", 15))); expectedA.add(new ArrayList<>(Arrays.asList("Water", 61))); expectedA.add(new ArrayList<>(Arrays.asList("Grass", 23))); expectedA.add(new ArrayList<>(Arrays.asList("Ash", 10))); expectedA.add(new ArrayList<>(Arrays.asList("Matt", 65)));	Object was created successfully	Object was created successfully	
3	Call the adjustRemainingVotes method on the opl01 at index03	assertThrows(IOException.class, () -> opl01.adjustRemainingVotes(index03));	IOException	IOException	

Post condition(s) for Test:

An IOException is thrown

Project Name: Project 1: Voting System

Team# 02

Test Stage: Unit _x_ System __ **Test Date:** 3/25/2024

Test Case ID#: OPL_ARV_1D **Name(s) of Testers:** Derrick

Test Description: Testing a valid index to ensure method subtracts the correct amount from the remainingVotes array at the given index

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

Automated: Yes X No

Results: Pass X Fail

Preconditions for Test: An OPL object has been created

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Define a new int, index04, which will be a valid index to modify within the created OPL object's remainingVotes array	index04 = 3	int was created successfully	int was created successfully	
2	Define a new int, expectedVal, which will represent the value that the opl01's remainingVotes array at index04 should be	expectedVal = (int)opl01.remainingVotes.get(index04).get(1) - ((int)opl01.fileData.getNumberBallots()/(int)opl01.fileData.getNumberSeats());	Object was created successfully	Object was created successfully	
3	Call the adjustRemainingVotes method on the opl01 at index04	opl01.adjustRemainingVotes(index04);	remainingVotes was successfully modified	remainingVotes was successfully modified	
4	Assert that the opl01 object's remainingVotes field at index04 matches the expected result	assertEquals(opl01.remainingVotes.get(index04).get(1), expectedVal);	True	True	

Post condition(s) for Test:

The OPL object properly updates the remainingVotes field, and the value of opl01.remainingVotes[index04] equals the expected value

~~**Project Name: Project 1: Voting System**~~

Team# 02

Test Stage: **Unit** _x_ **System** ___

Test Date: **3/25/2024**

Test Case ID#: OPL_DCV_2A

Name(s) of Testers: Derrick

Test Description: Ensuring two array lists are the same after the method is called

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

Automated: **Yes** ___ **No** ___

Results: **Pass** X **Fail** ___

Preconditions for Test: An OPL object has been created

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes

1	Initialize a new object of type ArrayList<ArrayList<Object>> object, expected1, which will be copied	ArrayList<ArrayList<Object>> expected1 = new ArrayList<ArrayList<Object>>();
		expected1.add(new ArrayList<>(Arrays.asList("Sarah", 50)));
		expected1.add(new ArrayList<>(Arrays.asList("Joe", 20)));
		expected1.add(new ArrayList<>(Arrays.asList("Bill", 13)));
		expected1.add(new ArrayList<>(Arrays.asList("Timothey", 50)));
			Object was created successfully	Object was created successfully	

2	Initialize a new object of type ArrayList<ArrayList<Object>> object, copied1, using the deepCopyVotes() method	ArrayList<ArrayList<Object>> copied1 = opl01.deepCopyVotes(expected1);	Object was created successfully	Object was created successfully	
3	Assert that the expected1 ArrayList is equivalent to the copied ArrayList, copied1	assertEquals(copied1, expected1);	True	True	

Post condition(s) for Test:

The copied1 ArrayList should be equivalent to the expected1 ArrayList.

Project Name: Project 1: Voting System

Team# 02

Test Stage: Unit _x_ System __ **Test Date:** 3/25/2024

Test Case ID#: OPL_DCV_2B **Name(s) of Testers:** Derrick

Test Description: Ensure variables that are modified in original ArrayList do not affect the copied ArrayList

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

Automated: Yes _X_ No ___

Results: Pass _X_ Fail ___

Preconditions for Test: An OPL object has been created

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes

1	Initialize a new object of type ArrayList<ArrayList<Object>> object, expected1, which will be copied	ArrayList<ArrayList<Object>> expected1 = new ArrayList<ArrayList<Object>>();
		expected1.add(new ArrayList<>(Arrays.asList("Sarah", 50)));
		expected1.add(new ArrayList<>(Arrays.asList("Joe", 20)));
		expected1.add(new ArrayList<>(Arrays.asList("Bill", 13)));
		expected1.add(new ArrayList<>(Arrays.asList("Timothey", 50)));
			Object was created successfully	Object was created successfully	

2	Initialize a new object of type ArrayList<ArrayList<Object>> object, copied1, using the deepCopyVotes() method	ArrayList<ArrayList<Object>> copied1 = opl01.deepCopyVotes(expected1);	Object was created successfully	Object was created successfully	
3	Assert that the expected1 ArrayList is equivalent to the copied ArrayList, copied1	assertEquals(copied1, expected1);	True	True	

Post condition(s) for Test:

The copied1 ArrayList does not get modified after modifying the expected1 ArrayList

Project Name: Project 1: Voting System

Team# 02

Test Stage: Unit _x_ System __ **Test Date:** 3/25/2024

Test Case ID#: OPL_DVC_2C **Name(s) of Testers:** Derrick

Test Description: Properly copying an empty ArrayList to another

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

Automated: Yes _X_ No ___

Results: Pass _X_ Fail ___

Preconditions for Test: An OPL object has been created

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Initialize a new object of type ArrayList<ArrayList<Object>> object, expected2, which will be copied	ArrayList<ArrayList<Object>> expected2 = new ArrayList<ArrayList<Object>>();	Object was created successfully	Object was created successfully	
2	Initialize a new object of type ArrayList<ArrayList<Object>> object, copied2, using the deepCopyVotes() method	ArrayList<ArrayList<Object>> copied2 = opl01.deepCopyVotes(expected2);	Object was created successfully	Object was created successfully	
3	Assert that the expected2 ArrayList is equivalent to the copied ArrayList, copied2	assertEquals(copied2, expected2);	True	True	

Post condition(s) for Test:

The empty expected2 ArrayList is equal to the copied2 ArrayList

Project Name: Project 1: Voting System
Team# 02

Test Stage: Unit _x_ System __ **Test Date:** 3/25/2024

Test Case ID#: OPL_DCV_2D **Name(s) of Testers:** Derrick

Test Description: Ensure modifying the original empty list will not affect the copy of it

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

Automated: Yes _X_ No ___

Results: Pass _X_ Fail ___

Preconditions for Test: An OPL object has been created

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Initialize a new object of type ArrayList<ArrayList<Object>> object, expected2, which will be copied	ArrayList<ArrayList<Object>> expected2 = new ArrayList<ArrayList<Object>>();	Object was created successfully	Object was created successfully	
2	Initialize a new object of type ArrayList<ArrayList<Object>> object, copied2, using the deepCopyVotes() method	ArrayList<ArrayList<Object>> copied2 = opl01.deepCopyVotes(expected2);	Object was created successfully	Object was created successfully	
3	Add a new item to the expected2 ArrayList	expected2.add(new ArrayList<>(Arrays.asList("Sarah", 50)));	Successfully added item to list	Successfully added item to list	
4	Assert that the expected2 ArrayList is equivalent to the copied ArrayList, copied2	assertEquals(copied2, expected2);	True	True	

Post condition(s) for Test:

The copied1 ArrayList does not get modified after modifying the expected1 ArrayList

Project Name: Project 1: Voting System
Team# 02

Test Stage: Unit _x_ System __

Test Date: 3/25/2024

Test Case ID#: OPL_GR_3A

Name(s) of Testers: Derrick

Test Description: Check if generated random number is within the specified range, 0 to 10, inclusive

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

Automated: Yes _X_ No ___

Results: Pass X Fail

Preconditions for Test: An OPL object has been created

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Define a new float, randomValue, which will be defined using opl01's generateRandom() function	randomValue = opl01.generateRandom();	float was created successfully	float was created successfully	
2	Assert that the randomly generated value is greater than 0	assertTrue("Generated value should be >= 0", randomValue >= 0);	True	True	
3	Assert that the randomly generated value is less than 10	assertTrue("Generated value should be < 10", randomValue < 10);	True	True	

Post condition(s) for Test:

The randomly generated number is within the range 0 to 10, inclusive

Project Name: Project 1: Voting System
Team# 02

Test Stage: Unit _x_ System __ **Test Date:** 3/25/2024

Test Case ID#: OPL_GR_3B **Name(s) of Testers:** Derrick

Test Description: Ensure multiple calls to the method generate different values

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

Automated: Yes _X_ No ___

Results: Pass _X_ Fail ___

Preconditions for Test: An OPL object has been created

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Define a new float, previousRandomValue, which will be set using opl01's generateRandom() method	float previousRandomValue = opl01.generateRandom();	float was created successfully	float was created successfully	
2	Define a new boolean, allDifferent, and initialize it to 'true'. This will be used to determine if multiple calls to generateRandom() generate different values	boolean allDifferent = true;	boolean was created successfully	boolean was created successfully	

3	Call generateRandom() 100 times and ensure that the same value is never generated twice in a row by generating a random value and assigning it to the float newRandomValue and comparing it to previousRandomValue 100 times	for (int i = 0; i < 100; i++) {
		float newRandomValue = opl01.generateRandom();
		if (newRandomValue == previousRandomValue) {
		allDifferent = false;
		break;
		}
		previousRandomValue = newRandomValue;
		}
			allDifferent = true	allDifferent = true	

4	Assert that the allDifferent boolean remains true	assertTrue("generateRandom() should generate different values (almost always)", allDifferent);	True	True	

Post condition(s) for Test:

The generated float previousRandomValue is never equivalent to the newly generated newRandomValue

Project Name: Project 1: Voting System

Team# 02

Test Stage: Unit _x_ System __

Test Date: 3/25/2024

Test Case ID#: OPL_GR_3C

Name(s) of Testers: Derrick

Test Description: Ensure the random numbers are actually distributed evenly over a large set and not pseudorandom, testing for 90% accuracy

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

Automated: Yes X No

Results: Pass X Fail

Preconditions for Test: An OPL object has been created

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes

1	Define a 10 integers, each used to find if a randomly generated number falls within each of the ranges	int between0and1 = 0;
		int between1and2 = 0;
		int between2and3 = 0;
		int between3and4 = 0;
		int between4and5 = 0;
		int between5and6 = 0;
		int between6and7 = 0;
		int between7and8 = 0;
		int between8and9 = 0;
		int between9and10 = 0;
			integers were created successfully	integers were created successfully	

2	Generate 10,000 random numbers and add one to each range that they fall under	for (int i = 0; i < 10000; i++) {
		float rv = opl01.generateRandom();
		if (rv < 1) {
		between0and1++;
		}else if(rv < 2){
		between1and2++;
		}else if(rv < 3){
		between2and3++;
		}else if(rv < 4){
		between3and4++;
		}else if(rv < 5){
		between4and5++;
		}else if(rv < 6){
		between5and6++;
		}else if(rv < 7){
		between6and7++;
		}else if(rv < 8){
		between7and8++;
		}else if(rv < 9){
		between8and9++;
		}else if(rv < 10){
		between9and10++;
		}
		}
			Loop finished successfully	Loop finished successfully	

3	Ensure that each of the 10 integers created are at minimum 900	assertTrue("at least 900 should be in the range 0 to 1", between0and1>=900); assertTrue("at least 900 should be in the range 1 to 2", between1and2>=900); assertTrue("at least 900 should be in the range 2 to 3", between2and3>=900); assertTrue("at least 900 should be in the range 3 to 4", between3and4>=900); assertTrue("at least 900 should be in the range 4 to 5", between4and5>=900); assertTrue("at least 900 should be in the range 5 to 6", between5and6>=900); assertTrue("at least 900 should be in the range 6 to 7", between6and7>=900); assertTrue("at least 900 should be in the range 7 to 8", between7and8>=900); assertTrue("at least 900 should be in the range 8 to 9", between8and9>=900); assertTrue("at least 900 should be in the range 9 to 10", between9and10>=900);	True (* 10)	True (* 10)	

Post condition(s) for Test:

Each of the ranges are generated at minimum 900 times, ensuring even distribution of randomly generated numbers

Project Name: Project 1: Voting System
Team# 02

Test Stage: Unit _x_ System __

Test Date: 3/25/2024

Test Case ID#: OPL_BT_4A

Name(s) of Testers: Derrick

Test Description: Test the breakTie function with a valid number of ties

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

Automated: Yes _X_ No ___

Results: Pass _X_ Fail ___

Preconditions for Test: An OPL object has been created

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Define a new int, numTie01, which will be a valid number of ties which much be broken using the OPL object	numTie01 = 3	int was created successfully	int was created successfully	
2	Create a new int using opl01's breakTie function with numTie01 as the parameter	int result01 = opl01.breakTie(numTie01);	int was created successfully	int was created successfully	
3	Assert that the resulting index should be between 0 and 4	assertTrue("resulting index should be between 0 and 4 inclusive", result01 >= 0 && result01 < numTie01);	True	True	

Post condition(s) for Test:

The OPL object correctly breaks a tie when using a valid number of ties into the breakTie method

Project Name: Project 1: Voting System
Team# 02

Test Stage: Unit _x_ System __ **Test Date:** 3/25/2024

Test Case ID#: OPL_BT_4B **Name(s) of Testers:** Derrick

Test Description: Test the breakTie function with an invalid number of ties (0)

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

Automated: Yes _X_ No ___

Results: Pass _X_ Fail ___

Preconditions for Test: An OPL object has been created

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Define a new int, numTie02, which will be an invalid number of ties	numTie02 = 0	int was created successfully	int was created successfully	
2	Create a new int using opl01's breakTie function with numTie02 as the parameter	int result02 = opl01.breakTie(numTie02);	int was created successfully	int was created successfully	
3	Assert that the produced value is -1, indicating failure	assertEquals(result02, -1);	True	True	

Post condition(s) for Test:

The breakTie method returns -1 indicating the input number of ties was invalid

Project Name: Project 1: Voting System
Team# 02

Test Stage: Unit _x_ System __ **Test Date:** 3/25/2024

Test Case ID#: OPL_BT_4C **Name(s) of Testers:** Derrick

Test Description: Test the breakTie function with an invalid number of ties (5, too large)

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

Automated: Yes _X_ No ___

Results: Pass _X_ Fail ___

Preconditions for Test: An OPL object has been created

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Define a new int, numTie03, which will be an invalid number of ties	numTie02 = 5	int was created successfully	int was created successfully	
2	Create a new int using opl01's breakTie function with numTie03 as the parameter	int result03 = opl01.breakTie(numTie03);	int was created successfully	int was created successfully	
4	Assert that the produced value is -1, indicating failure	assertEquals(result03, -1);	True	True	

Post condition(s) for Test:

The breakTie method returns -1 indicating the input number of ties was invalid

Project Name: Project 1: Voting System
Team# 02

Test Stage: Unit _x_ System __ **Test Date:** 3/25/2024

Test Case ID#: OPL_BT_4D **Name(s) of Testers:** Derrick

Test Description: Test the breakTie function with 1 tie

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

Automated: Yes _X_ No ___

Results: Pass _X_ Fail ___

Preconditions for Test: An OPL object has been created

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Define a new int, numTie04, which is 1 and should cause the breakTie method to return 0	numTie04 = 1	int was created successfully	int was created successfully	
2	Create a new int using opl01's breakTie function with numTie04 as the parameter	int result04 = opl01.breakTie(numTie04);	int was created successfully	int was created successfully	
3	Assert that the resulting int is 0	assertEquals(result04, 0);	True	True	

Post condition(s) for Test:

The breakTie method returns 0

Project Name: Project 1: Voting System
Team# 02

Test Stage: Unit _x_ System __

Test Date: 3/25/2024

Test Case ID#: OPL_AW_5A

Name(s) of Testers: Derrick

Test Description: Tests to see whether the list is empty on start and if calling it with a winner adds the correct winner

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

Automated: Yes X No

Results: Pass _X_ Fail ___

Preconditions for Test: An OPL object has been created

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Initialize a new ArrayList, winOrderExpected, which is empty at first	ArrayList<String> winOrderExcpected = new ArrayList<String>();	ArrayLilst was created successfully	ArrayList was created successfully	
2	Ensure the created opl01 object has an initially empty ArrayList, such as the on which was just initialized	assertEquals(winOrderExcpected, opl01.winOrder);	True	True	
3	Add a winner (party name) to the winOrderExpected ArrayList	winOrderExcpected.add("Lib");	Item successfully added	Item successfully added	
4	Use the addWinner function using the index of the (party name) added in winorderExpected (“Lib”)	opl01.addWinner(3);	Item successfully added	Item successfully added	

5	Assert that the winOrderExpected and opl01's winOrder ArrayLists are equal	assertEquals(winOrderExcpected, opl01.winOrder);	True	True	

Post condition(s) for Test:

The OPL's winOrder is equivalent to the winOrderExpected ArrayList

Project Name: Project 1: Voting System

Test Stage: Unit _x_ System __ **Test Date:** 3/25/2024

Test Case ID#: OPL_AW_5B **Name(s) of Testers:** Derrick

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

Automated: Yes _X_ No ___

Test Description: Ensure when trying to add an index that is too large will cause an exception to be thrown

Results: Pass _X_ Fail ___

Preconditions for Test: An OPL object has been created

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Try to add a winner at an invalid index (5) using the addWinner function, see if an exception is thrown	assertThrows(IOException.class, () -> opl01.addWinner(5));	IOException	IOException	

Post condition(s) for Test:

The winOrder for the opl01 object is unchanged

Project Name: Project 1: Voting System
Team# 02

Test Stage: Unit _x_ System __ **Test Date:** 3/25/2024

Test Case ID#: OPL_AW_5C **Name(s) of Testers:** Derrick

Test Description: Ensure when trying to add an index that is negative will cause an exception to be thrown

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

Automated: Yes X No

Results: Pass X Fail ___

Preconditions for Test: An OPL object has been created

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Try to add a winner at an invalid index (-1) using the addWinner function, see if an exception is thrown	assertThrows(IOException.class, () -> opl01.addWinner(-1));	IOException	IOException	

Post condition(s) for Test:

The winOrder for the opl01 object is unchanged

Project Name: Project 1: Voting System
Team# 02

Test Stage: Unit _x_ System ___

Test Case ID#: OPL_ASA_6A

Test Description: Test whether adding a seat at a specified index updates value to 1 from empty seat allocation array

Test Date: 3/25/2024

Name(s) of Testers: Derrick

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

Automated: Yes _X_ No ___

Results: Pass _X_ Fail ___

Preconditions for Test: An OPL object has been created

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Define a new boolean, 'firstRound' as true	boolean firstRound = true;	boolean was created successfully	boolean was created successfully	
2	Define a new integer, index01, as 1	int index01 = 1;	int was created successfully	int was created successfully	

3	Initialize a new ArrayList with the expected values	ArrayList<ArrayList<Object>> expected1 = new ArrayList<ArrayList<Object>>();
		expected1.add(new ArrayList<>(Arrays.asList("Dem", new int[]{0, 0})));
		expected1.add(new ArrayList<>(Arrays.asList("Rep", new int[]{1, 0})));
		expected1.add(new ArrayList<>(Arrays.asList("Green", new int[]{0, 0})));
		expected1.add(new ArrayList<>(Arrays.asList("Lib", new int[]{0, 0}));
			Object was created successfully	Object was created successfully	

4	Call the adjustSeatAllocation on the opl1 object with the parameters index01 and boolean firstRound	opl01.adjustSeatAllocation(index01, firstRound);	Seats were adjusted successfully	Seats were adjusted successfully	

	Assert that the adjustSeatAllocation function did not change opl01's seatAllocation size	assertEquals(opl01.seatAllocation.size(), expected1.size());	True	True	

	Assert that each index of opl01's seatAllocation is equivalent to the expected1	for (int i = 0; i < expected1.size(); i++) {
		assertEquals(expected1.get(i).size(),
		expected1.get(i).size());
		assertEquals(expected1.get(i).get(0),
		expected1.get(i).get(0));
		assertArrayEquals((int[])expected1.get(i).get(1),
		(int[])expected1.get(i).get(1));
		}
			True	True	

Post condition(s) for Test:

The opl01's seatAllocation is equivalent to the expected1's ArrayList

Project Name: Project 1: Voting System
Team# 02

Test Stage: Unit _x_ System ___

Test Date: 3/25/2024

Test Case ID#: OPL_ASA_6B

Name(s) of Testers: Derrick

Test Description: Test that adding in a second round to the same object works

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

Automated: Yes _X_ No ___

Results: Pass X Fail

Preconditions for Test: An OPL object has been created

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Initialize an int[] array 'array'	int[] array = (int[]) expected1.get(1).get(1);	Array was created successfully	Array was created successfully	
2	Increment the first index of 'array'	array[1]++;	Value incremented successfully	Value incremented successfully	
3	Define a new boolean, 'firstRound' as false	boolean firstRound = false;	boolean was created successfully	boolean was created successfully	
4	Define a new integer, index01, as 1	int index01 = 1;	int was created successfully	int was created successfully	

5	Initialize a new ArrayList with the expected values	ArrayList<ArrayList<Object>> expected1 = new ArrayList<ArrayList<Object>>();
		expected1.add(new ArrayList<>(Arrays.asList("Dem", new int[]{0, 0})));
		expected1.add(new ArrayList<>(Arrays.asList("Rep", new int[]{1, 0})));
		expected1.add(new ArrayList<>(Arrays.asList("Green", new int[]{0, 0}));
			Object was created successfully	Object was created successfully	

		expected1.add(new ArrayList<>(Arrays.asList("Lib", new int[]{0, 0})));			
6	Call the adjustSeatAllocation on the opl1 object with the parameters index01 and boolean firstRound	opl01.adjustSeatAllocation(index01, firstRound);	Seats were adjusted successfully	Seats were adjusted successfully	
7	Assert that the adjustSeatAllocation function did not change opl01's seatAllocation size	assertEquals(opl01.seatAllocation.size(), expected1.size());	True	True	

8	Assert that each index of opl01's seatAllocation is equivalent to the expected1	for (int i = 0; i < expected1.size(); i++) {
		assertEquals(expected1.get(i).size(), expected1.get(i).size());
		assertEquals(expected1.get(i).get(0), expected1.get(i).get(0));
		assertArrayEquals((int[])expected1.get(i).get(1), (int[])expected1.get(i).get(1));
		}
			True	True	

Post condition(s) for Test:

The opl01's seatAllocation is equivalent to the expected1's ArrayList after adding a second round

Project Name: Project 1: Voting System
Team# 02

Test Stage: Unit _x_ System ___

Test Case ID#: OPL_ASA_6C

Test Description: Test that the same index can be incremented twice

Test Date: 3/25/2024

Name(s) of Testers: Derrick

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

Automated: Yes _X_ No ___

Results: Pass X Fail

Preconditions for Test: An OPL object has been created

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Initialize an int[] array 'array'	int[] array = (int[]) expected1.get(1).get(1);	Array was created successfully	Array was created successfully	
2	Increment the first index of 'array' twice	array[1]++; array[1]++	Value incremented successfully	Value incremented successfully	
3	Define a new boolean, 'firstRound' as false	boolean firstRound = false;	boolean was created successfully	boolean was created successfully	
4	Define a new integer, index01, as 1	int index01 = 1;	int was created successfully	int was created successfully	

5	Initialize a new ArrayList with the expected values	ArrayList<ArrayList<Object>> expected1 = new ArrayList<ArrayList<Object>>();
		expected1.add(new ArrayList<>(Arrays.asList("Dem", new int[]{0, 0})));
		expected1.add(new ArrayList<>(Arrays.asList("Rep", new int[]{1, 0})));
		expected1.add(new ArrayList<>(Arrays.asList("Green", new int[]{0, 0}));
			Object was created successfully	Object was created successfully	

		expected1.add(new ArrayList<>(Arrays.asList("Lib", new int[]{0, 0})));			
6	Call the adjustSeatAllocation on the opl1 object with the parameters index01 and boolean firstRound	opl01.adjustSeatAllocation(index01, firstRound);	Seats were adjusted successfully	Seats were adjusted successfully	
7	Assert that the adjustSeatAllocation function did not change opl01's seatAllocation size	assertEquals(opl01.seatAllocation.size(), expected1.size());	True	True	

8	Assert that each index of opl01's seatAllocation is equivalent to the expected1	for (int i = 0; i < expected1.size(); i++) {
		assertEquals(expected1.get(i).size(), expected1.get(i).size());
		assertEquals(expected1.get(i).get(0), expected1.get(i).get(0));
		assertArrayEquals((int[])expected1.get(i).get(1), (int[])expected1.get(i).get(1));
		}
			True	True	

Post condition(s) for Test:

The opl01's seatAllocation is equivalent to the expected1's ArrayList after incrementing the same index twice

Project Name: Project 1: Voting System
Team# 02

Test Stage: Unit _x_ System ___

Test Date: 3/25/2024

Test Case ID#: OPL_ASA_6D

Name(s) of Testers: Derrick

Test Description: Testing with an invalid index (-1). IOException should be thrown

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

Automated: Yes _X_ No ___

Results: Pass _X_ Fail ___

Preconditions for Test: An OPL object has been created

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Define a new integer, index02, as -1	int index02 = -1;	int was created successfully	int was created successfully	

2	Initialize a new ArrayList with the expected values	ArrayList<ArrayList<Object>> expected1 = new ArrayList<ArrayList<Object>>();
		expected1.add(new ArrayList<>(Arrays.asList("Dem", new int[]{0, 0})));
		expected1.add(new ArrayList<>(Arrays.asList("Rep", new int[]{1, 0})));
		expected1.add(new ArrayList<>(Arrays.asList("Green", new int[]{0, 0})));
		expected1.add(new ArrayList<>(Arrays.asList("Lib", new int[]{0, 0})));
			Object was created successfully	Object was created successfully	

3	Call the adjustSeatAllocation on the opl1 object with the parameters index02 and boolean firstRound	assertThrows(IOException.class, () -> opl01.adjustSeatAllocation(index03, true));	IOException	IOException	

Post condition(s) for Test:

An IOException is thrown

Project Name: Project 1: Voting System
Team# 02

Test Stage: Unit _x_ System ___

Test Case ID#: OPL_ASA_6E

Test Description: Testing with an invalid index (10, too large). seatAllocation array should remain unchanged

Test Date: 3/25/2024

Name(s) of Testers: Derrick

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

Automated: Yes _X_ No ___

Results: Pass _X_ Fail ___

Preconditions for Test: An OPL object has been created

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
4	Define a new integer, index03, as -1	int index03 = 10;	int was created successfully	int was created successfully	

5	Initialize a new ArrayList with the expected values	ArrayList<ArrayList<Object>> expected1 = new ArrayList<ArrayList<Object>>();
		expected1.add(new ArrayList<>(Arrays.asList("Dem", new int[]{0, 0})));
		expected1.add(new ArrayList<>(Arrays.asList("Rep", new int[]{1, 0})));
		expected1.add(new ArrayList<>(Arrays.asList("Green", new int[]{0, 0})));
		expected1.add(new ArrayList<>(Arrays.asList("Lib", new int[]{0, 0}));
			Object was created successfully	Object was created successfully	

6	Call the adjustSeatAllocation on the opl1 object with the parameters index02 and boolean firstRound	assertThrows(IOException.class, () -> opl01.adjustSeatAllocation(index03, true));	IOException	IOException	

Post condition(s) for Test:

An IOException is thrown

Project Name: Project 1: Voting System
Team# 02

Test Stage: Unit _x_ System ___

Test Case ID#: OPL_ISA_7A

Test Description: Ensure the resulting array has the correct default values in all fields

Test Date: 3/25/2024

Name(s) of Testers: Derrick

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

Automated: Yes _X_ No ___

Results: Pass _X_ Fail ___

Preconditions for Test: An OPL object has been created

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes

1	Initialize a new object of type ArrayList<ArrayList<Object>> object, expected1	ArrayList<ArrayList<Object>> expected1 = new ArrayList<ArrayList<Object>>();
		expected1.add(new ArrayList<>(Arrays.asList("Dem", new int[]{0, 0})));
		expected1.add(new ArrayList<>(Arrays.asList("Rep", new int[]{0, 0})));
		expected1.add(new ArrayList<>(Arrays.asList("Green", new int[]{0, 0})));
		expected1.add(new ArrayList<>(Arrays.asList("Lib", new int[]{0, 0})));
			Object was created successfully	Object was created successfully	

2	Initialize a new object of type ArrayList<ArrayList<Object>> object, result1, using the initializeSeatAllocation() method	ArrayList<ArrayList<Object>> result1 = opl01.initializeSeatAllocation();	Object was created successfully	Object was created successfully	
3	Compare the sizes of expected1 and results1	assertEquals(expected1.size(), result1.size());	True	True	

4	Manually compare each array is equal at each index because assertEquals compares memory addresses for the int[]s	for (int i = 0; i < expected1.size(); i++) {
		assertEquals(expected1.get(i).size(), result1.get(i).size());
		assertEquals(expected1.get(i).get(0), result1.get(i).get(0));
		assertArrayEquals((int[])expected1.get(i).get(1),
		(int[])result1.get(i).get(1));
		}
			True	True	

Post condition(s) for Test:

The expected ArrayList should be equal to the result1 ArrayList

Project Name: Project 1: Voting System
Team# 02

Test Stage: Unit _x_ System __

Test Date: 3/25/2024

Test Case ID#: OPL_FA_8A

Name(s) of Testers: Derrick

Test Description: First allocation test where only 1 seat is available. Ensure it breaks after 1 loop adding no winners as they are all below the remainder

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

Automated: Yes _X_ No ___

Results: Pass _X_ Fail ___

Preconditions for Test: An OPL object has been created

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Call firstAllocation on the opl02 object	opl02.firstAllocation();	firstAllocation was successfully called	firstAllocation was successfully called	
2	Initialize a new object of type ArrayList<ArrayList<Object>> object, which will be the expectedWinOrder	ArrayList<String> expectedWinOrder = new ArrayList<String>();	Object was created successfully	Object was created successfully	
3	Assert that the opl02 object's winOrder field equals the expectedWinOrder	assertEquals(expectedWinOrder, opl02.winOrder);	True	True	

Post condition(s) for Test:

The OPL object's winOrder field equals the expectedWinOrder ArrayList

Project Name: Project 1: Voting System
Team# 02

Test Stage: Unit _x_ System __ **Test Date:** 3/25/2024

Test Case ID#: OPL_FA_8B **Name(s) of Testers:** Derrick

Test Description: Test where one seat is allocated in the round of allocation

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

Automated: Yes X__ No ___

Results: Pass _X_ Fail ___

Preconditions for Test: An OPL object has been created containing a valid remainingVotes array

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Call firstAllocation on the opl04 object	opl04.firstAllocation();	firstAllocation was successfully called	firstAllocation was successfully called	
2	Initialize a new object of type ArrayList , which will be the expectedWinOrder	expectedWinOrder = new ArrayList<>();	Object was created successfully	Object was created successfully	
3	Add a party to expectedWinOrder	expectedWinOrder.add("Dem");	Item was successfully added	Item was successfully added	
4	Assert that the opl04 object's winOrder field equals the expectedWinOrder	assertEquals(expectedWinOrder, opl04.winOrder);	True	True	

Post condition(s) for Test:

The OPL object's winOrder field equals the expectedWinOrder ArrayList

Project Name: Project 1: Voting System
Team# 02

Test Stage: Unit _x_ System __ **Test Date: 3/25/2024**

Test Case ID#: OPL_FA_8C **Name(s) of Testers:** Derrick

Test Description: Two seats are allocated in the first round of allocation

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

Automated: Yes _X_ No ___

Results: Pass X Fail

Preconditions for Test: An OPL object has been created containing a valid remainingVotes array

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Call firstAllocation on the opl03 object	opl03.firstAllocation();	firstAllocation was successfully called	firstAllocation was successfully called	
2	Initialize a new object of type ArrayList , which will be the expectedWinOrder	expectedWinOrder = new ArrayList<>();	Object was created successfully	Object was created successfully	
3	Add a party to expectedWinOrder	expectedWinOrder.add("Dem");	Item was successfully added	Item was successfully added	
4	Add a party to expectedWinOrder	expectedWinOrder.add("Dem");	Item was successfully added	Item was successfully added	

5	Initialize a new int[], expected	int[] expected = new int[]{2,0};	Array was successfully created	Array was successfully created	
6	Assert that the opl03 object's winOrder field equals the expectedWinOrder	assertEquals(expectedWinOrder, opl03.winOrder);	True	True	
7	Initialize a new int[], temp	int[] temp = (int[]) opl03.seatAllocation.get(0).get(1);	Array was successfully created	Array was successfully created	
8	Assert that the tostring of expected equals the tostring of temp	assertEquals(Arrays.toString(expected), Arrays.toString(temp));	True	True	

Post condition(s) for Test:

The OPL object's winOrder field equals the expectedWinOrder ArrayList

Project Name: Project 1: Voting System
Team# 02

Test Stage: Unit _x_ System __ **Test Date:** 3/25/2024

Test Case ID#: OPL_FA_8D **Name(s) of Testers:** Derrick

Test Description: All seats allocated in first allocation, should end

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

Automated: Yes _X_ No ___

Results: Pass _X_ Fail ___

Preconditions for Test: An OPL object has been created containing a valid remainingVotes array

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Call firstAllocation on the opl05 object	opl05.firstAllocation();	firstAllocation was successfully called	firstAllocation was successfully called	
2	Initialize a new object of type ArrayList , which will be the expectedWinOrder	expectedWinOrder = new ArrayList<>();	Object was created successfully	Object was created successfully	
3	Add a party to expectedWinOrder	expectedWinOrder.add("Dem");	Item was successfully added	Item was successfully added	
4	Assert that the opl05 object's winOrder field equals the expectedWinOrder	assertEquals(expectedWinOrder, opl05.winOrder);	True	True	

Post condition(s) for Test:

The OPL object's winOrder field equals the expectedWinOrder ArrayList

Project Name: Project 1: Voting System
Team# 02

Test Stage: Unit _x_ System __ **Test Date:** 3/25/2024

Test Case ID#: OPL_FA_8E **Name(s) of Testers:** Derrick

Test Description: 2 seats aren't allocated to winning party if there's only 1 member

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

Automated: Yes _X_ No ___

Results: Pass _X_ Fail ___

Preconditions for Test: An OPL object has been created containing a valid remainingVotes array

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Call firstAllocation on the opl06 object	opl06.firstAllocation();	firstAllocation was successfully called	firstAllocation was successfully called	
2	Initialize a new object of type ArrayList , which will be the expectedWinOrder	expectedWinOrder = new ArrayList<>();	Object was created successfully	Object was created successfully	
3	Add a party to expectedWinOrder	expectedWinOrder.add("Dem");	Item was successfully added	Item was successfully added	
4	Assert that the opl06 object's winOrder field equals the expectedWinOrder	assertEquals(expectedWinOrder, opl06.winOrder);	True	True	

Post condition(s) for Test:

The OPL object's winOrder field equals the expectedWinOrder ArrayList

Project Name: Project 1: Voting System
Team# 02

Test Stage: Unit _x_ System __

Test Date: 3/25/2024

Test Case ID#: OPL_SA_9A

Name(s) of Testers: Derrick

Test Description: No seat given in first allocation, assigns seat in second to highest party vote count

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

Automated: Yes _X_ No ___

Results: Pass X Fail

Preconditions for Test: An OPL object has been created

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Call firstAllocation on the opl02 object	opl02.firstAllocation();	firstAllocation was successfully called	firstAllocation was successfully called	
2	Call secondAllocation on the opl02 object	opl02.secondAllocation();	secondAllocation was successfully called	secondAllocation was successfully called	
3	Initialize a new object of type ArrayList<ArrayList<Object>> object, which will be the expectedWinOrder	ArrayList<String> expectedWinOrder = new ArrayList<>();	Object was created successfully	Object was created successfully	
4	Add a party to expectedWinOrder	expectedWinOrder.add("Dem");	Item was successfully added	Item was successfully added	
5	Assert that the opl02 object's winOrder field equals the expectedWinOrder	assertEquals(expectedWinOrder, opl02.winOrder);	True	True	

Post condition(s) for Test:

The OPL object's winOrder field equals the expectedWinOrder ArrayList

Project Name: Project 1: Voting System
Team# 02

Test Stage: Unit _x_ System __ **Test Date:** 3/25/2024

Test Case ID#: OPL_SA_9B **Name(s) of Testers:** Derrick

Test Description: Test where each party should have a seat

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

Automated: Yes _X_ No ___

Results: Pass _X_ Fail ___

Preconditions for Test: An OPL object has been created containing a valid remainingVotes array

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Call firstAllocation on the opl04 object	opl04.firstAllocation();	firstAllocation was successfully called	firstAllocation was successfully called	
2	Call secondAllocation on the opl04 object	opl04.secondAllocation();	secondAllocation was successfully called	secondAllocation was successfully called	
3	Initialize a new object of type ArrayList<ArrayList<Object>> object, which will be the expectedWinOrder	ArrayList<String> expectedWinOrder = new ArrayList<>();	Object was created successfully	Object was created successfully	
4	Add a party to expectedWinOrder	expectedWinOrder.add("Dem");	Item was successfully added	Item was successfully added	
5	Add a party to expectedWinOrder	expectedWinOrder.add("Rep");	Item was successfully added	Item was successfully added	

6	Assert that the opl04 object's winOrder field equals the expectedWinOrder	assertEquals(expectedWinOrder, opl04.winOrder);	True	True	

Post condition(s) for Test:

The OPL object's winOrder field equals the expectedWinOrder ArrayList

Project Name: Project 1: Voting System
Team# 02

Test Stage: Unit _x_ System __

Test Date: 3/25/2024

Test Case ID#: OPL_SA_9C

Name(s) of Testers: Derrick

Test Description: Two seats are allocated in the first round of allocation

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

Automated: Yes _X_ No ___

Results: Pass _X_ Fail ___

Preconditions for Test: An OPL object has been created containing a valid remainingVotes array

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Call firstAllocation on the opl03 object	opl03.firstAllocation();	firstAllocation was successfully called	firstAllocation was successfully called	
2	Call secondAllocation on the opl03 object	opl03.secondAllocation();	secondAllocation was successfully called	secondAllocation was successfully called	
3	Initialize a new object of type ArrayList<ArrayList<Object>> object, which will be the expectedWinOrder	ArrayList<String> expectedWinOrder = new ArrayList<>();	Object was created successfully	Object was created successfully	
4	Add a party to expectedWinOrder	expectedWinOrder.add("Dem");	Item was successfully added	Item was successfully added	
5	Add a duplicate party to expectedWinOrder	expectedWinOrder.add("Dem");	Item was successfully added	Item was successfully added	

6	Assert that the opl03 object's winOrder field equals the expectedWinOrder	assertEquals(expectedWinOrder, opl03.winOrder);	True	True	

Post condition(s) for Test:

The OPL object's winOrder field equals the expectedWinOrder ArrayList

Project Name: Project 1: Voting System
Team# 02

Test Stage: Unit _x_ System __ **Test Date:** 3/25/2024

Test Case ID#: OPL_SA_9D **Name(s) of Testers:** Derrick

Test Description: Tests that the second seat gets correctly allocated to party that has members available even if they didn't win votes

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

Automated: Yes X No

Results: Pass X Fail ___

Preconditions for Test: An OPL object has been created

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Call firstAllocation on the opl06 object	opl06.firstAllocation();	firstAllocation was successfully called	firstAllocation was successfully called	
2	Call secondAllocation on the opl06 object	opl06.secondAllocation();	secondAllocation was successfully called	secondAllocation was successfully called	
3	Initialize a new object of type ArrayList<ArrayList<Object>> object, which will be the expectedWinOrder	ArrayList<String> expectedWinOrder = new ArrayList<>();	Object was created successfully	Object was created successfully	
4	Add a party to expectedWinOrder	expectedWinOrder.add("Dem");	Item was successfully added	Item was successfully added	
5	Add a party to expectedWinOrder	expectedWinOrder.add("Rep");	Item was successfully added	Item was successfully added	

6	Assert that the opl06 object's winOrder field equals the expectedWinOrder	assertEquals(expectedWinOrder, opl06.winOrder);	True	True	

Post condition(s) for Test:

The OPL object's winOrder field equals the expectedWinOrder ArrayList

Project Name: Project 1: Voting System
Team# 02

Test Stage: Unit _x_ System __ **Test Date:** 3/25/2024

Test Case ID#: OPL_SA_9E **Name(s) of Testers:** Derrick

Test Description: Handles the case where there is a tie in the second allocation

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

Automated: Yes _X_ No ___

Results: Pass _X_ Fail ___

Preconditions for Test: An OPL object has been created

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Initialize int 'results' to 0	int results = 0;	int created successfully	int created successfully	

2	Get the result of the election with a tie, record how often Dem wins before Rep, should be about equal	int results = 0;
		for(int i=0; i<1000; i++){
		OPL opl = new OPL(testFile);
		opl.secondAllocation();
		if(opl.winOrder.get(0).equals("Dem")){
		results++;
		}
		}
			results added to successfully	results added to successfully	

3	Ensure generated value is between 450 and 550	assertTrue("Generated value should be between 450 and 550", results<550); assertTrue("Generated value should be between 450 and 550", 450<results);	True (*2)	True (*2)	

Post condition(s) for Test:

The generated value is between 450 and 550

Project Name: Project 1: Voting System
Team# 02

Test Stage: Unit _x_ System __

Test Date: 3/25/2024

Test Case ID#: OPL_RE_10A

Name(s) of Testers: Derrick

Test Description: Ensures expected results for a normal OPL election with one seat to allocate, no ties, 2 parties, 5 candidates

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

Automated: Yes _X_ No ___

Results: Pass _X_ Fail ___

Preconditions for Test: An OPL object has been created

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	runElection() is called on the opl1 object and the resulting ResultsData object is stored in result1	ResultsData result1 = opl1.runElection();	result1 was successfully created	result1 was successfully created	
2	Initialize a new object of type ArrayList<ArrayList<Object>> object, which will be winOrder	ArrayList<String> winOrder = new ArrayList<>();	Object was created successfully	Object was created successfully	
4	Add a party to winOrder	winOrder.add("Dem");	Item was successfully added	Item was successfully added	
4	Assert that the winOrder ArrayList and the result1 partyWinOrder are equal	assertEquals(winOrder, result1.getPartyWinOrder());	True	True	

5	Create a new ArrayList, remainingVotes	ArrayList<ArrayList<Object>> remainingVotes = new ArrayList<>();
		remainingVotes.add(new ArrayList<Object>(Arrays.asList("Dem", 700)));
		remainingVotes.add(new ArrayList<Object>(Arrays.asList("Rep", 300)));
			Object was created successfully	Object was created successfully	

6	Assert that the created remainingVotes array is equivalent to the remainingVotes array in result1	assertEquals(remainingVotes, result1.getRemainingVotes());	True	True	

7	Create a new ArrayList, seatAllocation	ArrayList<ArrayList<Object>> seatAllocation = new ArrayList<>();
		seatAllocation.add(new ArrayList<Object>(Arrays.asList(" Dem", new int[] {0,1})));
		seatAllocation.add(new ArrayList<Object>(Arrays.asList(" Rep", new int[] {0,0})));
			Object was created successfully	Object was created successfully	

8	Initialize 2 int[] arrays, temp and temp2	int[] temp = (int[]) result1.getSeatAllocation().get(0).get(1); int[] temp2 = (int[]) seatAllocation.get(0).get(1);	Array was created successfully	Array was created successfully	
9	Assert that the toStrings for temp and temp2 are equal	assertEquals(Arrays.toString(temp2), Arrays.toString(temp));	True	True	

Post condition(s) for Test:

runElection works on a normal OPL election with one seat to allocate, with no ties

Project Name: Project 1: Voting System
Team# 02

Test Stage: Unit _x_ System __

Test Date: 3/25/2024

Test Case ID#: OPL_RE_10B

Name(s) of Testers: Derrick

Test Description: Normal OPL election with 3 seats to allocate, with ties, 4 parties, 10 candidates

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

Automated: Yes _X_ No ___

Results: Pass X Fail

Preconditions for Test: An OPL object has been created

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create a new ArrayList, winOrder	winOrder = new ArrayList<>(Arrays.asList("Dem", "Green", "Dem", "Lib"));	Object created successfully	Object created successfully	
2	Define 2 integers to represent how often a party appears before another in the winOrder array in the case of a tie	int libWonTie = 0; int demWonTie = 0;	int created successfully	int created successfully	

2	See which party shows up before the other, 10000 tests are taken, should be roughly even	for(int i = 0; i<10000; i++){
		opl2 = new OPL(testFile03);
		result1 = opl2.runElection();
		if(result1.getPartyWinOrder().get(2) == "Lib"){
		libWonTie++;
		}else if(result1.getPartyWinOrder().get(2) == "Dem"){
		demWonTie++;
		}
		}
			Loop terminates successfully	Loop terminates successfully	

3	Assert that generated values should be roughly even	assertTrue("Generated value should be between 4500 and 5500", libWonTie<5500); assertTrue("Generated value should be between 4500 and 5500", 4500<libWonTie); assertTrue("Generated value should be between 4500 and 5500", demWonTie<5500); assertTrue("Generated value should be between 4500 and 5500", 4500<demWonTie);	True	True	

4	Create a new ArrayList to represent the expected number of remaining votes and add values to it	remainingVotes = new ArrayList<>();
		remainingVotes.add(new ArrayList<Object>(Arrays.asList("Dem", 75)));
		remainingVotes.add(new ArrayList<Object>(Arrays.asList("Rep", 36)));
		remainingVotes.add(new ArrayList<Object>(Arrays.asList("Green", 4)));
		remainingVotes.add(new ArrayList<Object>(Arrays.asList("Lib", 75)));
			Object successfully created	Object successfully created	

5	Assert that results1.getRemainingVotes matches the expected remainingVotes array	assertEquals(remainingVotes, result1.getRemainingVotes());	True	True	

6	Create a new ArrayList to represent the expected seat allocation and add values to it	seatAllocation = new ArrayList<>();
		seatAllocation.add(new ArrayList<Object>(Arrays.asList(" Dem", new int[] {1,1})));
		seatAllocation.add(new ArrayList<Object>(Arrays.asList(" Rep", new int[] {0,0})));
		seatAllocation.add(new ArrayList<Object>(Arrays.asList(" Green", new int[] {1,0})));
		seatAllocation.add(new ArrayList<Object>(Arrays.asList(" Lib", new int[] {0,1})));
			Object successfully created	Object successfully created	

7	Create two int[], temp and temp2, using indices in seatAllocation and result1.getSeatAllocation() to ensure they are equal	temp = (int[]) result1.getSeatAllocation().get(0).get(1); temp2 = (int[]) seatAllocation.get(0).get(1);	int[]'s successfully created	int[]'s successfully created	
8	Assert that the values at temp2 and temp are equal	assertEquals(Arrays.toString(temp2), Arrays.toString(temp));	True	True	

9	Create a new HashMap that is the expected HashMap to be found in result1.fileData.getPartyCandidates()	partyCandidatesTest = new HashMap<>();
		partyCandidatesTest.put("Dem", new ArrayList<>(Arrays.asList("Bob","Sarah", "Jon")));
		partyCandidatesTest.put("Rep", new ArrayList<>(Arrays.asList("Craig", "Klein")));
		partyCandidatesTest.put("Green", new ArrayList<>(Arrays.asList("Water", "Grass", "Rain")));
		partyCandidatesTest.put("Lib", new ArrayList<>(Arrays.asList("Matt", "Ash")));
			Object successfully created	Object successfully created	

10	Assert partyCandidatesTest is equal to result1.fileData.getPartyCandidates()	assertEquals(partyCandidatesTest, result1.fileData.getPartyCandidates());	True	True	

Post condition(s) for Test:

The correct election results are obtained after running an election on an opl2 object with 3 seats to allocate, with ties, 4 parties and 10 candidates

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.