Test 1

@Test

void testSortIntArrayEquals() {

assertTrue(Arrays.equals(ASorted,B));

}

This test passes because Array B was cloned to array A then sorted matching the array “Asorted.” Using Arrays.equals lets us look inside each element in the array making asserttrue true

Test 2

@Test

void testSortIntArrayDeepEquals() {

assertTrue(Objects.deepEquals(ASorted,B));

}

This test passed because Array A&B do not have any nested arrays and after sorting Array b it equals Array “Asorted.” Using Objects.deepEquals looks at the elements in the array not what they are stored as so asserttrue is true

Test 3

@Test

void testSortIntAssertArrayEquals() {

assertArrayEquals(ASorted,B);

}

This test passes because Array B was cloned to array A then sorted matching the array “Asorted” also this is like the first test just using the assertArrayEquals instead of putting Array.Equals in asserttrue

Test 4

@Test

void testSortIntEquals() {

assertFalse(B.equals(ASorted));

}

This test passes because B has a different reference than A sorted so in the computer they are stored as different values making the assert False ture

Test 5

@Test

void testSortIntAssertEquals() {

assertNotEquals(ASorted, B);

}

This test passes because B has a different reference than A sorted so in the computer they are stored as different values making the assert not equals true. Notice for the first test you had to put Arrays.equals to actually look inside the elements in the array.

Test 6

@Test

void testSortIntAssertIterableEquals() {

//if you are not familiar with autoboxing, the following line convert int to Integer

//see https://stackoverflow.com/questions/2607289/converting-array-to-list-in-java for the various

//methods to create an Iterable object

List<Integer> AS = Arrays.stream(ASorted).boxed().toList();

List<Integer> BS = Arrays.stream(B).boxed().toList();

assertIterableEquals(AS, BS);

}

This test passes you are doing the same thing to both arrays. And both the new list have the same values in them, so when we call assertIterableEquals each part of the list equals each other.

Test 7

@Test

void testSortIntAssertDeepEquals() {

//convert to Object[] to use deepEquals

Integer[] ASortedo = new Integer[ASorted.length];

for (int i=0; i < ASorted.length ; i++) ASortedo[i] = ASorted[i];

Integer[] Bo = new Integer[B.length];

for (int i=0; i < B.length ; i++) Bo[i] = B[i];

assertTrue(Arrays.deepEquals(ASortedo,Bo));

}

This test passes because we are deep copying both Array Asorted and B to Asortedo and Bo respectively. Calling assertTrue with Arrays.deepEquals we look inside both new arrays and compare the values inside making it true.

<https://github.com/COSC-222/lab1-Evanjager/commits/master>

<https://github.com/COSC-222/lab1-Evanjager>