Homework M4: Array-Based Implementation

Due Feb 28 at 11:59pm **Points** 100 **Questions** 26
Available until May 30 at 11:59pm **Time Limit** None

Instructions

Review the Homework FAQ page for details about submitting homework. In this homework, you will:

- trace the behind-the-scenes variables in the ArrayBag, ResizeableArrayBag, and AList classes
- evaluate ArrayList code
- implement methods to go inside the ArrayBag, ResizeableArrayBag, and AList classes

Homework Files

Below is the driver/tester program. I strongly recommend using this to test your code before submitting. You can ignore the *test methods* at the end of the file.

Important Note: Be sure to use the correct files from the 5th edition (included in the zip file below). Do **not** use the interfaces from older editions- there are differences.

HomeworkM4Files.zip (https://ccsf.instructure.com/courses/47904/files/7370538/download?download_frd=1)

Notes

Remember what it means to write from the implementation perspective.

- Writing from the implementation perspective means you are writing code to go **inside** the ArrayBag.java, ResizeableArrayBag.java, or AList.java class.
- Because you are writing code to go inside of these classes, you have access to the private instance data variables and private methods of the classes. Take advantage of that!

Do not use other kinds of data structures.

- Do not create a different data structure inside the code.
- For example, do not create an ArrayList inside of a method in the ArrayBag class.

For full credit, think about efficiency.

- Your answer should be efficient.
- Sometimes, directly accessing instance data variables rather than invoking an existing method can

yield a more efficient solution.

- Take advantage of writing from the implementation perspective by accessing the instance data variables when it will allow you to write a more efficient solution.
- For now, think of an efficient solution as one that **minimizes the number of passes through the array and avoids nested loops when possible**. (We'll eventually learn how to truly measure efficiency!)
    - Often times, a nested loop will occur when a method involves going through the array and that method is then invoked inside of a loop that also goes through the array.
    - Calls to methods like remove(T), getIndexOf(T), contains(T), and getFrequencyOf(T) ***within*** a loop that goes over the whole array likely creates a nested loop and should be avoided.
    - You can often avoid this by directly accessing the instance data variables or invoking different methods.

Account for special cases.

- Unless explicitly stated otherwise, your code should run, produce an expected result, and not crash for an empty list/bag and a singleton list/bag that contains only one element.

This quiz was locked May 30 at 11:59pm.

Attempt History

	Attempt	Time	Score
LATEST	Attempt 1	5,521 minutes	109 out of 100

Score for this quiz: **109** out of 100

Submitted Feb 25 at 1:08pm

This attempt took 5,521 minutes.

Coding Questions

- These questions ask you to write code at the developer level (or implementation perspective).
    - This means you are writing code that goes **inside** the ArrayBag, ResizableArrayBag, or AList class.
- Carefully review the instance data variables and methods provided in those classes.
    - Use the private instance data variables (and sometimes private methods!) to create an efficient or improved solution.

- You will be graded both on syntax (the code should compile) and semantics (the code should accomplish the task).
- Read carefully to determine which ADT is being required.

Write a **removeAll(T)** method for the **ArrayBag** class. The method removes **all** occurrences of a specified entry from a bag and returns the number of elements removed.

For full credit, write an efficient solution.

To do this, try to minimize the number of times you cycle through the array. Remember that some methods in the class cycle through the array. Be careful about invoking these methods that go through the array *inside* a loop- this can create a nested loop, which is inefficient! (See the notes at the top of this page for examples.)

The method header is:

```
public int removeAll(T element)
```

Question 1

20 / 20 pts

Paste your complete ArrayBag removeAll(T) method here.

Your Answer:

```
//Question 1: removeALL method for ArrayBag
public int removeAll(T anEntry) {
    checkInitialization();

    // create a tempbag
    BagInterface<T> tempBag = new ArrayBag<T>();

    // parameter
    int counter = 0;

    // executes if numberOfEntries not equal to zero
    if (numberOfEntries > 0) {
        // loop through bag
        for (int i = 0; i <  numberOfEntries; i++) {
            if (this.bag[i].equals(anEntry)) {    // if match increment cou
nter
                counter++;
            } else {    // add to tempList if not matching element
                tempBag.add(this.bag[i]);
            }
        }
        // clear original bag
        this.clear();
        // set tempList items back into original bag
        while (!tempBag.isEmpty()) {
            this.add(tempBag.remove());
        }
    }
    // return counter
    return counter;
}
```

Write a **trimToSize()** method for the **ResizableArrayBag** class. The method removes empty spaces from the internal, behind-the-scenes "bag" array so that there are no "null" places at the end of the array.

Note that you are not changing the forward-facing contents of the array bag. You are only changing the behind-the-scenes array storage.

- This method should accomplish the same functionality as the trimToSize method in ArrayList (https://docs.oracle.com/javase/8/docs/api/java/util/ArrayList.html#trimToSize--) .
- You do **not** need to account for any nulls that occur as "values" in the list. You only need to remove the nulls that represent "unused" space in the array.
- For full credit, make sure your method will work correctly and not cause problems under special cases (including when the method is invoked on an empty bag).
- In order to use the provided tester, make sure you use the provided ResizableArrayBag class. I've added an equals method and made the bag array public for testing purposes.

The method header is:

```
public void trimToSize()
```

Question 2

15 / 15 pts

Paste your complete ResizableArrayBag trimToSize() method here.

Your Answer:

```
// Question 2: trimToSize method for ResizableArrayBag class
public void trimToSize() {
    checkInitialization();

    // if bag is empty or currentSize() equal to bags capacity, don't do an
ything
    if (!this.isEmpty() && this.getCurrentSize() < bag.length) {
        // create a new bag with size equal to current objects in array
        @SuppressWarnings("unchecked")
        T[] tempResizableBag = (T[]) new Object[this.getCurrentSize()];

        // copy old bag to new bag
        for (int i = 0; i < this.getCurrentSize(); i++) {
            tempResizableBag[i] = this.bag[i];
        }

        // replace old bag with new bag
        this.bag = tempResizableBag;
    }
}
```

Write a **removeAll(T)** method for the **AList** class. The method removes **all** occurrences of a specific entry from the list and returns the number of elements removed. The remaining list elements should be left in the same relative order.

The method header is:

```
public int removeAll(T element)
```

Question 3

20 / 20 pts

Paste your complete AList removeAll(T) method here.

Your Answer:

```
// Question 3: removeAll method for Alist class
public int removeAll(T element) {
    checkInitialization();

    // create an array
    AList<T> tempList = new AList<T>();

    // parameter
    int counter = 0;

    // executes if numberOfEntries greater than 0
    if (numberOfEntries > 0) {
        for (int i = 1; i <= this.numberOfEntries; i++) {
            // if match
            if (this.getEntry(i).equals(element)) {
                // increment counter
                counter++;
            } else { // add to tempList if not matching element
                tempList.add(this.list[i]);
            }
        }
        // clear original list
        this.clear();

        // restore list with tempList
        for (int i = 1; i <= tempList.getLength(); i++) {
            this.add(tempList.getEntry(i));
        }
    }
    // return
    return counter;
}
```

note that you can accomplish this task without emptying and rebuilding the list

Write an equals method for the **AList** class to override the method inherited from Object. Two lists should be considered *logically equivalent* ("the same") if they contain **the same (equivalent) elements in the same order**.

For full credit, do not use any other data structures other than the underlying arrays in your answer.

The method header is:

```
@Override
```

```
public boolean equals(Object obj)
```

Question 4

20 / 20 pts

Paste your complete equals method here.

Your Answer:

```
// Question 4: equals method AList class
@Override
public boolean equals(Object obj) {
    checkInitialization();

    if (obj instanceof AList<?>) {
        AList<T> otherAList = (AList<T>) obj;

        // check if both arrays are empty
        if (this.isEmpty() && otherAList.isEmpty()) {
            return true;
        } else if (this.getLength() != otherAList.getLength()) { // checks
if both arrays are the same length;
            return false;
        } else if (this.getLength() == otherAList.getLength()) {
            // compares elements using a loop
            for (int i = 1; i <= this.getLength(); i++) {
                if (!this.getEntry(i).equals(otherAList.getEntry(i))) {
                    return false;
                }
            }
        }
    }
    return true;
}
```

Multiple Choice Questions for ArrayBag

These next set of questions ask about objects of type **ArrayBag**. Refer to the files provided with the homework for details.

Question 5

1 / 1 pts

What is the size of the private bag array (bag.length) after the execution of the following code?

```
ArrayBag<Integer> arrayBag = new ArrayBag<>();
```

- ○ 10001
- ○ 10000
- ○ none of these is correct
- ○ 26
- **Correct!** [✓ 25]

Question 6

1 / 1 pts

The private instance data variables for an ArrayBag<Integer> object called arrayBag is below:

```
numberOfEntries = 7
```

0	1	2	3	4	5	6	7	8	9	10
42	52	16	78	42	82	18	null	null	null	null

What is the result of executing the code below?

```
arrayBag.add(43);
```

- none of these is correct
- the add method returns false
- numberOfEntries will be 6

Correct!

- [✓ numberOfEntries will be 8]
- numberOfEntries will be 7
- the add method returns null
- numberOfEntries will be 11
- an exception will be thrown

Question 7

2 / 2 pts

The private instance data variables for an ArrayBag<Integer> object called arrayBag is below:

```
numberOfEntries = 0
```

0	1	2	3	4	5	6	7	8	9	10
null	null	null	null	null	null	null	null	null	null	null

List the index of each element in the bag array (displayed above) after the following statements are executed.

```
    arrayBag.add(43);
    arrayBag.add(17);
    arrayBag.add(92);
    arrayBag.add(84);
```

Correct!	**17**	1
Correct!	**43**	0
Correct!	**84**	3
Correct!	**92**	2

Other Incorrect Match Options:

- 5
- 7
- 8
- 10
- 9
- 4
- 6

Question 8

1 / 1 pts

The private instance data variables for an ArrayBag<Integer> object called arrayBag is below:

```
numberOfEntries = 5
```

0	1	2	3	4	5	6	7	8	9	10
42	52	16	78	39	null	null	null	null	null	null

What is the result of executing the code below?.

```
arrayBag.remove();
```

Correct!

[✓ the bag array will contain [42, 52, 16, 78, null, null, null, null, null, null, null]]

an exception will be thrown

the bag array will contain [52, 16, 78, 39, null, null, null, null, null, null, null]

the bag array will contain [39, 52, 16, 78, null, null, null, null, null, null, null]

none of these is correct

the bag array will contain [null, 52, 16, 78, 39, null, null, null, null, null, null]

Question 9

1 / 1 pts

The private instance data variables for an ArrayBag<Integer> object called arrayBag is below:

```
numberOfEntries = 5
```

0	1	2	3	4	5	6	7	8	9	10
5	4	3	2	1	null	null	null	null	null	null

What is the result of executing the code below?.

```
arrayBag.remove(3);
```

Correct!

[✓ the bag array will contain [5, 4, 1, 2, null, null, null, null, null, null, null]]

the bag array will contain [5, 4, 3, 1, null, null, null, null, null, null, null]

the bag array will contain [5, 4, 2, 1, null, null, null, null, null, null, null]

an exception will be thrown

none of these is correct

Question 10

1 / 1 pts

The private instance data variables for an ArrayBag<Integer> object called arrayBag is below:

```
numberOfEntries = 11
```

0	1	2	3	4	5	6	7	8	9	10

42	52	16	78	39	42	61	82	59	44	18

What is the result of executing the code below?.

```
arrayBag.remove(999);
```

- the remove method will return null
- none of these is correct
- an exception will be thrown
- [✓ the remove method will return false]

Correct!

Multiple Choice Questions for ResizableArrayBag

These next set of questions ask about objects of type **ResizableArrayBag**. Refer to the files provided with the homework for details.

Question 11

1 / 1 pts

The initial capacity of a ResizableArrayBag is 10. What is the length of the private bag array (bag.length) after adding 8 entries to the bag?

Correct!

10

orrect Answers 10 (with margin: 0)

Question 12

1 / 1 pts

The initial capacity of a ResizableArrayBag is 10. What is the length of the private bag array (bag.length) after adding 15 entries to the bag?

Correct! 20

orrect Answers 20 (with margin: 0)

Question 13

1 / 1 pts

The initial capacity of a ResizableArrayBag is 10. What is the length of the private bag array (bag.length) after adding 55 entries to the bag?

Correct! 80

orrect Answers 80 (with margin: 0)

Multiple Choice Questions for AList

These next set of questions ask about objects of type **AList**. Refer to the files provided with the homework for details.

Question 14

2 / 2 pts

The private instance data variables for an AList<Integer> object called aList is below:

```
numberOfEntries = 0
```

0	1	2	3	4	5	6	7	8	9	10
null	null	null	null	null	null	null	null	null	null	null

List the index of each element in the list array (displayed above) after the following statements are executed.

```
aList.add(62);
aList.add(35);
aList.add(1, 19);
aList.add(2, 57);
```

Correct!	**19**	1
Correct!	**35**	4
Correct!	**57**	2
Correct!	**62**	3

Other Incorrect Match Options:

- 5
- 9
- 0
- 6
- 7

- 8
- 10

Question 15

1 / 1 pts

Using the code above, what is the value of numberOfEntries after the code is executed?

Correct!

4

orrect Answers

4 (with margin: 0)

Question 16

2 / 2 pts

The private instance data variables for an AList<Integer> object called aList is below:

```
numberOfEntries = 0
```

0	1	2	3	4	5	6	7	8	9	10
null	null	null	null	null	null	null	null	null	null	null

List the index of each element in the list array (displayed above) after the following statements are executed.

```
aList.add(43);
aList.add(17);
aList.add(92);
aList.add(84);
```

Correct!	**17**	2
Correct!	**43**	1
Correct!	**84**	4
Correct!	**92**	3

Other Incorrect Match Options:

- 6
- 5
- 9
- 0
- 10
- 7
- 8

Question 17

1 / 1 pts

Using the code above, what is the value of numberOfEntries after the code is executed?

Correct! 4

orrect Answers 4 (with margin: 0)

Question 18

2 / 2 pts

The private instance data variables for an AList<Integer> object called aList is below:

```
numberOfEntries = 5
```

0	1	2	3	4	5	6	7	8	9	10
nu ll	4	19	39	17	2	nu ll	nu ll	nu ll	nu ll	nul l

List the index of each element in the list array (displayed above) after the following statements are executed.

```
aList.replace(4, 23);
aList.add(55);
aList.remove(2);
aList.add(77);
```

Correct!	2	4
Correct!	4	1
Correct!	17	not in the list array
Correct!	19	not in the list array
Correct!	23	3
Correct!	39	2
Correct!	55	

5

Correct! **77** 6

Other Incorrect Match Options:

- 9
- 0
- 8
- 7
- 10

Question 19

1 / 1 pts

The private instance data variables for an AList<Integer> object called aList is below:

```
numberOfEntries = 5
```

0	1	2	3	4	5	6	7	8	9	10
null	18	63	14	19	22	null	null	null	null	null

What will be the value of list[3] after the following statement is executed?

```
aList.remove(3);
```

○ 14

○ 18

○ none of these is correct because an exception will be thrown

Correct!

- [✓ 19]
- 63
- null
- 22

Question 20

1 / 1 pts

The private instance data variables for an AList<Integer> object called aList is below:

```
numberOfEntries = 5
```

0	1	2	3	4	5	6	7	8	9	10
null	42	16	23	35	62	null	null	null	null	null

Which of the following is true after the following statement is executed?

```
aList.remove(0);
```

- ○ the remove method will return null
- ○ no exception will be thrown and the list array contents will not be changed
- **Correct!** [✓ an exception will be thrown]
- ○ the remove method will return false
- ○ none of these is correct

Question 21

1 / 1 pts

The private instance data variables for an AList<Integer> object called aList is below:

```
numberOfEntries = 5
```

0	1	2	3	4	5	6	7	8	9	10
null	5	4	3	2	1	null	null	null	null	null

What will be the value of list[4] after the following statement is executed?

```
aList.remove(4);
```

- ○ 5
- ○ 4
- ○ 2
- ○ 3
- **Correct!** [✓ 1]
- ○ null
- ○ none of these is correct because an exception will be thrown

Question 22

0 / 1 pts

The private instance data variables for an AList<Integer> object called aList is below:

```
numberOfEntries = 5
```

0	1	2	3	4	5	6	7	8	9	10

null	45	32	71	52	19	null	null	null	null	null

What is the values of the array returned by invoking aList.toArray()?

- [45, 32, 71, 52, 19, null, null, null, null, null]

ou Answered

- [✓ none of these is correct]

orrect Answer

- [45, 32, 71, 52, 19]
- [null, 45, 32, 71, 52, 19, null, null, null, null, null]
- [0, 45, 32, 71, 52, 19]
- [0, 45, 32, 71, 52, 19, null, null, null, null, null]

Multiple Choice Questions for ArrayList

These next set of questions ask about objects of type **ArrayList** (in the Java standard library). Refer to the API page (https://docs.oracle.com/javase/8/docs/api/java/util/ArrayList.html) for details.

Question 23

1 / 1 pts

What is true after the execution of the following code?

```
ArrayList<String> arrayList = new ArrayList<String>();
arrayList.add("a");
arrayList.add("b");
arrayList.add("c");
arrayList.remove("b");
```

Correct!

- [✓ the arrayList will contain a, c]
- the arrayList will contain a, null, c
- none of these is correct
- an exception is thrown

Question 24

1 / 1 pts

What is true after the execution of the following code?

```
ArrayList<Integer> arrayList = new ArrayList<>(3);
arrayList.add(45);
arrayList.add(62);
arrayList.add(33);
arrayList.add(19);
```

- there will be a compiler error
- arrayList will contain 3 values

Correct!

- [✓ arrayList will contain 4 values]
- none of these is correct
- an exception is thrown

Question 25

1 / 1 pts

What is true after the execution of the following code?

```
ArrayList<String> arrayList = new ArrayList<String>();
arrayList.add("a");
arrayList.add("b");
arrayList.add("c");
System.out.println(arrayList.remove("x"));
```

- ○ an exception is thrown
- ○ -1 is printed
- **Correct!** [✓ false is printed]
- ○ the code will not compile
- ○ none of these is correct

Optional Extra Credit (10 points)

Modify the AList class to implement the Comparable interface. This will allow you to compare two AList objects to determine which list is greater/smaller.

Write the compareTo method to compare lists this way:

- first by size: the list that has fewer elements is smaller
- second, if the lists are the same size:
  - the comparison of the lists is the comparison of those first **non-matching** elements
  - compare the two lists element-by element; the first time you find an index that contains elements that are not the same, compare

those elements;

Note that using this definition, if the two lists are different sizes, you never look at the elements in the lists.

Using the Driver

To test your AList class, be sure to use the provided AList class. The generics, class header, and various parts of the code have been updated so that the compareTo method can run. Specifically, the version includes the modified lines below.

This change to AList restricts AList objects such that they can *only* hold elements that are Comparable. What this means is that you can invoke compareTo on any element in the list.

```
public class AList<T extends Comparable<? super T>> imp
lements ListInterface<T>, Comparable<AList<T>>

public int compareTo(AList<T> otherList)

T[] tempList = (T[]) new Comparable[initialCapacity +
1]; // in constructor

T[] result = (T[]) new Comparable[numberOfEntries]; //
in toArray
```

Question 26

10 / 0 pts

Paste your compareTo method here.

Your Answer:

```
// Extra credit: implement comparable interface
@Override
public int compareTo(AList<T> otherList) {
    checkInitialization();

    // compare list length
    if (this.getLength() < otherList.getLength()) { // return -1 if this.li
st < otherList
        return -1;
    } else if (this.getLength() > otherList.getLength()) { // return 1 if t
his.list > otherList
```

```
        return 1;
    } else if (this.getLength() == otherList.getLength()) { // compare elem
ents if equal
        for (int i = 1; i <= this.getLength(); i++) {
            if (this.getEntry(i).compareTo(otherList.getEntry(i)) != 0) {
// returns 0 on first element that aren't

// "the same"
                return (this.getEntry(i).compareTo(otherList.getEntry(i)));
            }
        }
    }
    return 0;
}
```

Quiz Score: **109** out of 100