Week 3 worksheet: Collection Classes, Wrapper Classes, and Linked Lists

Total points: 10

Out: February 4 (Tuesday)

Due: February 8 (Saturday, 1159 PM) … this is a long one, good practice for midterm 1, so extra time

*No late submissions will be accepted*

## What to download?

Download the following four classes from the D2L site Materials 🡪 Content 🡪 Worksheets 🡪 Worksheet 3 Files: BagTest.java, IntArrayBag.java, IntLinkedBag.java, IntNode.java.

## What to have bookmarked?

To do this worksheet, you will need to work with some classes from Main’s textbook. I provide the source code and the API. The source code is on the class D2L page. The APIs are linked here.

|  |  |
| --- | --- |
| Class | API |
| IntNode | [Main's IntNode API](https://home.cs.colorado.edu/~main/docs/edu/colorado/nodes/IntNode.html) |
| IntArrayBag | [Main's IntArrayBag API](https://home.cs.colorado.edu/~main/docs/edu/colorado/collections/IntArrayBag.html) |
| IntLinkedBag | [Main's IntLinkedBag API](https://home.cs.colorado.edu/~main/docs/edu/colorado/collections/IntLinkedBag.html) |

## What to submit?

Upload only one zipped file to the designated D2L folder. It should contain the IntArrayBag.java and IntLinkedBag.java classes, and this worksheet with your answer to Exercise 1, Question 2 and your output from Exercise 3.

Although it’s OK to zip your entire archive of .java files (e.g., from Eclipse), **don’t modify the IntNode.java or BagTest.java classes!**

# Exercise 1: IntArrayBag [4 pts]

1. [1 pt] Implement an instance method called “replace” in the **IntArrayBag** class that takes two input parameters, **int oldVal** and **Integer newVal**. The method replaces each occurrence of **oldVal** in the bag with **newVal**. Don’t manually convert the Integer object to an int primitive, let the autoboxing/auto-unboxing do its thing. Remember to write a full method, including the method header:

**public void replace(int oldVal, Integer newVal) {**

**for (int i = 0; i < manyItems; i++) {**

**if (data[i] == oldVal) {**

**data[i] = newVal;**

**}**

**}**

**}**

1. [2 pt] If you did it right, this worked. Explain what autoboxing and/or auto-unboxing the program did in order to get newVal into the array.

In this program we are converting a primitive number to an object with the Integer class Wrapper. Using the Number class methods we can auto Wrap the primitive number with the obj.intValue() method to box or Integer.valueOf() to unbox

1. [1 pt] Implement an instance method, called **sameAs**, in the **IntArrayBag** class that takes one input parameter and returns a **boolean** value. The input parameter is of type **IntArrayBag**. The method returns true if the input bag and the bag that activates the method have exactly the same elements. Otherwise, the method returns false. Note that the locations of the elements in the data arrays are not necessarily be the same, it is only the number of occurrences of each element that must be the same. The method header should be:

**public boolean sameAs(IntArrayBag bagA) {**

**// If the sizes are different, the bags cannot be the same**

**if (this.manyItems != bagA.manyItems) {**

**return false;**

**}**

**int[] thisDataCopy = Arrays.copyOf(this.data, this.manyItems);**

**int[] bagADataCopy = Arrays.copyOf(bagA.data, bagA.manyItems);**

**Arrays.sort(thisDataCopy);**

**Arrays.sort(bagADataCopy);**

**return Arrays.equals(thisDataCopy, bagADataCopy);**

**}**

# Exercise 2: IntLinkedBag [2 pts]

Top of Form

Bottom of Form

1. [1 pt] Implement an instance method called “replace” in the **IntLinkedBag** class that takes two input parameters, **int oldVal** and **Integer newVal**. The method replaces each occurrence of **oldVal** in the bag with **newVal**. Don’t manually convert the Integer object to an int primitive, let the autoboxing/auto-unboxing do its thing. Remember to write a full method, including the method header:

**public void replace(int oldVal, Integer newVal) {**

**IntNode current = head;**

**while (current != null) {**

**if (current.getData() == oldVal) {**

**current.setData(newVal);**

**}**

**current = current.getLink();**

**}**

**}**

1. [1 pt] Implement an instance method, called **sameAs**, in the **IntLinkedBag** class that takes one input parameter and returns a **boolean** value. The input parameter is of type **IntLinkedBag**. The method returns true if the input bag and the bag that activates the method have exactly the same elements. Otherwise, the method returns false. Note that the locations of the elements in the data arrays are not necessarily be the same, it is only the number of occurrences of each element that must be the same. The method header should be:

**public boolean sameAs(IntLinkedBag bagL) {**

**if (this.manyNodes != bagL.manyNodes) {**

**return false;**

**}**

**IntNode current = this.head;**

**while (current != null) {**

**int thisBag = this.countOccurrences(current.getData());**

**int otherBag = bagL.countOccurrences(current.getData());**

**if (thisBag != otherBag) {**

**return false;**

**}**

**current = current.getLink();**

**}**

**return true;**

**}**

# Exercise 3: Testing [4 pts]

Copy the BagTest class from the D2L site and run it with your modified IntArrayBag.java, IntLinkedBag.java, IntNode.java classes. Do not change that class! Show the output that you get below. Notice that to get the 4 points for this exrecise, all you need to do is correctly write the replace() and sameAs() methods in the IntArrayBag and IntLinkedBag classes.

*Remember to show the output you get from running the BagTest class below. Remember to include this worksheet with your solution to this and to Exercise 1, part 2 in your zip archive!*

*Output:*

*bagA1 and bagA2 have the same elements: true*

*bagL1 and bagL2 have the same elements: true*

*bagA1 and bagA2 have the same elements: false*

*bagL1 and bagL2 have the same elements: false*