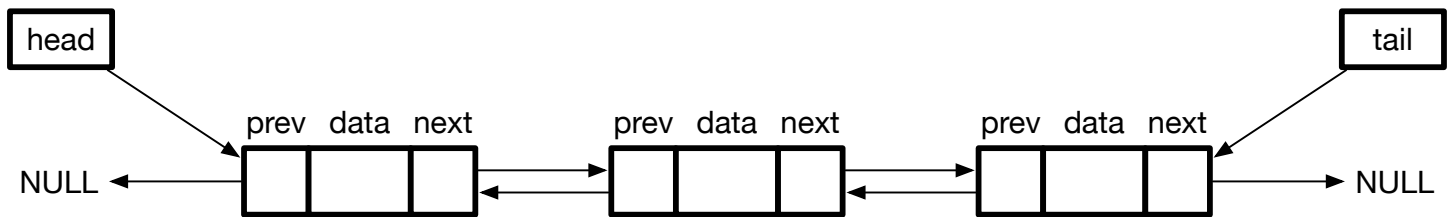


CMSC204

Assignment #3

Doubly Linked Lists



Assignment Description

Your assignment is to write a generic doubly-linked list class with an iterator, and a generic sorted doubly-linked list class with an iterator that inherits from your generic doubly-linked list class. The GUI has been provided for you for this assignment to help you visualize your linked list. Your list classes will also be tested with JUnit tests. Upload the initial files from Blackboard and your working files in a directory into the repository in GitHub you created in Lab 1 and take a screen shot of the files.

More information about doubly-linked lists can be found here: <https://www.geeksforgeeks.org/difference-between-singly-linked-list-and-doubly-linked-list/>

Concepts tested by this assignment

- Exception handling
- Generic Classes
- Double Linked List
- Ordered Double Linked List
- Iterators
- Comparators

Classes

BasicDoubleLinkedList

This generic doubly-linked list relies on a head (reference to first element of the list) and tail (reference to the last element of the list) where the last node points to the first element of the list. Both the head and the tail are set to null when the list is empty. Both point to the same element when there is only one element in the list, and now the element's "next" reference points to itself. A node structure has only two fields: data and the next references. The class must only define the following entities: an inner class Node, an inner class that implements ListIterator (for the iterator method), head and tail references and an integer representing the list

size. However only the `next()`, `hasNext()`, `previous()` and `hasPrevious()` methods of the `ListIterator` are you required to implement. The rest of the methods can throw the `UnsupportedOperationException`, such as:

```
public void remove() throws UnsupportedOperationException{
    throw new UnsupportedOperationException();}
```

All the entities are defined as protected so they can be accessed by the subclass. Follow the Javadoc that is provided.

SortedDoubleLinkedList

A generic sorted double linked list will be constructed using a provided `Comparator` to determine how the list is to be sorted. It extends `BasicDoubleLinkedList` class. The **`addToFront`** and the **`addToEnd`** methods will not be supported and an **`add`** method will be added that inserts to the double linked list in sorted order dependent on the `Comparator`. Follow the Javadoc that is provided.

Exception Handling

- `UnsupportedOperationException` – this exception is a Java library exception and will be returned by the `addToFront` and `addToEnd` implementations of the `SortedDoubleLinkedList` class and by the `remove` method of the iterator.
- `NoSuchElementException` – this exception is a Java library exception and will be returned by the `next` function within the iterator class when there are no more elements in the linked list.

GUI driver (provided for you)

A GUI driver has been provided for you to help you visualize your doubly-linked lists. Here is the minimum that must be in place to start using the GUI driver effectively.

- All methods in your `BasicDoubleLinkedList` and `SortedDoubleLinkedList` must be stubbed.
- The **`addToFront`** or **`addToEnd`** method of the `BasicDoubleLinkedList` must be implemented to create a basic double singly-linked list.
- The **`add`** method of the `SortedDoubleLinkedList` must be implemented to create a sorted double singly-linked list.
- The **`toArrayList`** method in both the `BasicDoubleLinkedList` and `SortedDoubleLinkedList`, which returns an `arraylist` of the items in the list from the head of list to the tail of list. This method is used to display the contents of the lists.

Testing

1. Your code should cause the `BasicDoubleLinkedList_Test` tests to succeed.
2. Your code should cause the `SortedDoubleLinkedList_Test` tests to succeed.
3. Create a JUnit Test – `BasicDoubleLinkedList_STUDENT_Test`.
4. Create a JUnit Test – `SortedDoubleLinkedList_STUDENT_Test`.

Examples using GUI driver

Adding to a Basic List

The screenshot shows the 'Doubly Linked List' application window. The 'Type of lists' section has the 'Basic' radio button selected and circled in red. The 'Add to List' section has 'Fifth' entered in the 'Element to Add' field, and the 'Add End' button is highlighted. The 'Retrieve from List' section has an empty 'Retrieved:' field. The 'Get from List' section has an empty 'Returned:' field. The 'Remove from List' section has an empty 'To be Removed:' field. The 'Iterator' section has an empty 'Returns:' field. The 'Contents of lists' section shows two lists: 'Basic List' with elements First, Second, Third, Fourth, Fifth and 'Sorted List' which is empty. An 'Exit' button is at the bottom.

Doubly Linked List

Type of lists

☒ Basic ☐ Sorted

Add to List

Element to Add: Fifth Add Front Add End Add

Retrieve from List (deletes from list)

Retrieved: Retrieve First Retrieve Last

Get from List (doesn't delete from list)

Returned: Get First Get Last

Remove from List

To be Removed: Remove

Iterator

Returns: Start Next Previous Has Next Has Previous

Contents of lists

Basic List	Sorted List
First	
Second	
Third	
Fourth	
Fifth	

Exit

Adding to a Sorted List

The screenshot shows the 'Doubly Linked List' application window. The 'Type of lists' section has the 'Sorted' radio button selected and circled in red. The 'Add to List' section has 'Angela' entered in the 'Element to Add' field, and the 'Add' button is highlighted. The 'Retrieve from List' section has an empty 'Retrieved:' field. The 'Get from List' section has an empty 'Returned:' field. The 'Remove from List' section has an empty 'To be Removed:' field. The 'Iterator' section has an empty 'Returns:' field. The 'Contents of lists' section shows two lists: 'Basic List' with elements First, Second, Third, Fourth, Fifth and 'Sorted List' with elements Angela, Maggie, Steven, Thomas, Vicky. An 'Exit' button is at the bottom.

Doubly Linked List

Type of lists

☐ Basic ☒ Sorted

Add to List

Element to Add: Angela Add Front Add End Add

Retrieve from List (deletes from list)

Retrieved: Retrieve First Retrieve Last

Get from List (doesn't delete from list)

Returned: Get First Get Last

Remove from List

To be Removed: Remove

Iterator

Returns: Start Next Previous Has Next Has Previous

Contents of lists

Basic List	Sorted List
First	Angela
Second	Maggie
Third	Steven
Fourth	Thomas
Fifth	Vicky

Exit

Removing Second from basic

Doubly Linked List

Type of lists: ☒ Basic ☐ Sorted

Add to List
Element to Add:

Retrieve from List (deletes from list)
Retrieved:

Get from List (doesn't deletes from list)
Returned:

Remove from List
To be Removed:

Iterator
Returns:

Contents of lists

Basic List	Sorted List
First	Angela
Third	Maggie
Fourth	Steven
Fifth	Thomas
	Vicky

Removing Thomas from sorted

Doubly Linked List

Type of lists: ☐ Basic ☒ Sorted

Add to List
Element to Add:

Retrieve from List (deletes from list)
Retrieved:

Get from List (doesn't deletes from list)
Returned:

Remove from List
To be Removed:

Iterator
Returns:

Contents of lists

Basic List	Sorted List
First	Angela
Third	Maggie
Fourth	Steven
Fifth	Thomas
	Vicky

Start the iterators for Basic and Sorted. Think of iterators being “in between” nodes.

Doubly Linked List

Type of lists

☒ Basic

☐ Sorted

Add to List

Element to Add:

Add Front

Add End

Add

Retrieve from List (deletes from list)

Retrieved:

Retrieve First

Retrieve Last

Get from List (doesn't deletes from list)

Returned:

Get First

Get Last

Remove from List

To be Removed:

Remove

Iterator

Returns:

Start

Next

Previous

Has Next

Has Previous

Contents of lists

↔

Basic List

First

Third

Fourth

Fifth

↔

Sorted List

Angela

Maggie

Steven

Vicky

Exit

Example of selecting “Next” for Basic and then for Sorted list. Think of iterators being “in between” nodes.

The screenshot shows a window titled "Doubly Linked List" with several sections for managing a doubly linked list. The "Type of lists" section has radio buttons for "Basic" and "Sorted", with "Sorted" selected. The "Add to List" section has a text input "Element to Add:" with the value "Angela" and buttons "Add Front", "Add End", and "Add". The "Retrieve from List (deletes from list)" section has a "Retrieved:" input and buttons "Retrieve First" and "Retrieve Last". The "Get from List (doesn't deletes from list)" section has a "Returned:" input and buttons "Get First" and "Get Last". The "Remove from List" section has a "To be Removed:" input with the value "Thomas" and a "Remove" button. The "Iterator" section has a "Returns:" input with the value "Angela" (circled in red), and buttons "Start", "Next" (highlighted with a blue border), "Previous", "Has Next", and "Has Previous". The "Contents of lists" section displays two lists: "Basic List" and "Sorted List". The "Basic List" contains "First", "Third", "Fourth", and "Fifth", with a red circle around the "First" node and a red arrow pointing to the right. The "Sorted List" contains "Angela", "Maggie", "Steven", and "Vicky", with a red circle around the "Angela" node and a red arrow pointing to the right. An "Exit" button is at the bottom.

Basic List	Sorted List
First	Angela
Third	Maggie
Fourth	Steven
Fifth	Vicky

Example of “Next” for basic and “Previous” for Sorted. Think of iterators being “in between” nodes.

Deliverables

Type of lists

☐ Basic ☒ Sorted

Add to List

Element to Add:

Retrieve from List (deletes from list)

Retrieved:

Get from List (doesn't delete from list)

Returned:

Remove from List

To be Removed:

Iterator

Returns:

Contents of lists

Basic List

First
Third
Fourth
Fifth

Sorted List

Angela
Maggie
Steven
Vicky

Deliverables / Submissions:

Design: UML class diagram with algorithm (pseudo-code) for methods

Implementation: Submit a compressed file containing the follow (see below): The Java application (it must compile and run correctly); Javadoc files **in a directory**; a write-up as specified below. Be sure to review the provided project rubric to understand project expectations. The write-up will include:

- Final design: UML diagram with pseudo-code
- In three or more paragraphs, highlights of your learning experience

Deliverable format: The above deliverables will be packaged as follows. Two compressed files in the following formats:

- LastNameFirstName_Assignment3_Complete.zip, a compressed file in the zip format, with the following:
 - Write up (Word document) - reflection paragraphs
 - UML Diagram - latest version (Word or jpg document)
 - doc (directory) - Javadoc
 - File1.html (example)
 - File2.html (example)
 - Sub-directory (example)
 - src (directory)
 - File1.java (example)
 - File2.java (example)
- LastNameFirstName_Assignment3_Moss.zip, a compressed file containing one or more Java files:
 - File1.java (example)
 - File2.java (example)

This folder should contain Java source files only