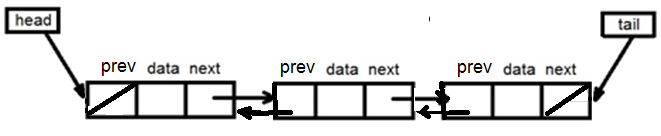
**CMSC204**

**Assignment #3**

**Assignment 3**

Double Linked Lists



**Assignment Description**

Your assignment is to write a generic double-linked list class with an iterator, and a generic sorted double-linked list class with an iterator that inherits from your generic double-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.

**Concepts tested by this assignment**

Exception handling

Generic Classes

Double Linked List

Ordered Double Linked List

Iterators

Comparators

**Classes**

**BasicDoubleLinkedList**

This generic double-linked implements the Java’s ***Iterable*** interface and relies on a head (reference to first element of the list) and tail (reference to the last 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. A node structure has three fields: data, next and the previous references.

*BasicDoubleLinkedList*class defines the following entities:

* A generic inner class *Node* - This class has the following attributes:
* *data* of type T
* *prev* of type Node
* *next* of type Node
* *A* generic inner class named *DoubleLinkedListIterator* that implements java’s *ListIterator* interface (for the iterator method).

This class only implements the *next*(), *hasNext*(), *previous*() and *hasPrevious*() methods of the *ListIterator* interface. (**Follow java API documentation of *ListIterator interface* for the implementing these methods.)**

The rest of the methods should throw the *UnsupportedOperationException*, such as:

public void remove() throws UnsupportedOperationException{

throw new UnsupportedOperationException();

}  
***Hint****:* You need at least one attribute for this class that can be initialized to the head of the *BasicDoubleLinkList* in order to implement the methods of this class.

* attributes *head* and *tail* of type *Node* class to reference the begin and end of the list and an integer attribute *size* representing the list size.

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.

**Hint**: define an attribute of type *Comparator* and use it to compare the data in the list.

It extends *BasicDoubleLinkedList* class. The ***addToFront*** and the ***addToEnd*** methods inherited from the*BasicDoubleLinkedList* will not be supported. There will be an *add* method to insert a node in the sorted linked list dependening 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 *DoubleLinkedListIterator*.
* NoSuchElementException – this exception is a Java library exception and will be returned by the *next*() method within the *DoubleLinkedListIterator* 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.

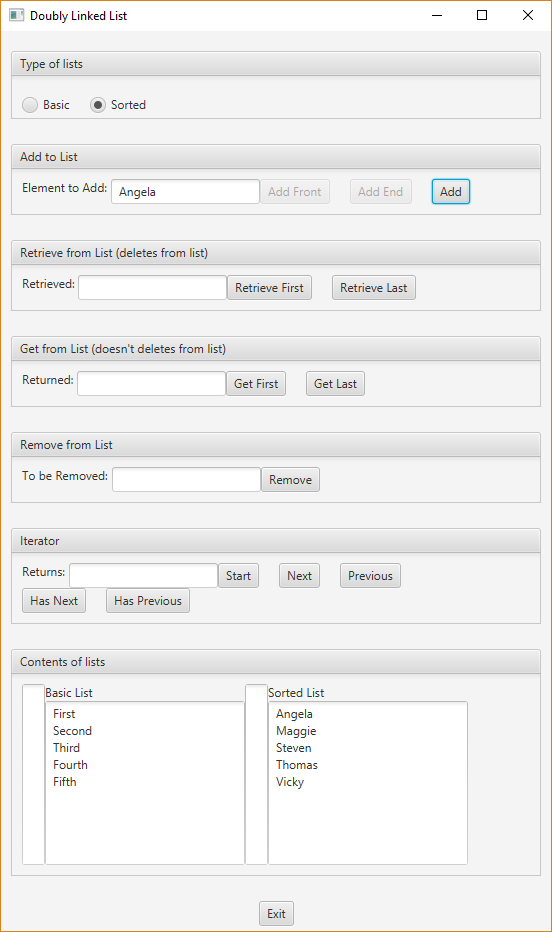
* All methods in your *BasicDoubleLinkedList* and *SortedDoubleLinkedList* must be stubbed.
* The ***addToFront*** or ***addToEnd*** method of the *BasicDoubleLinkedList* must be implemented to create a basic double doubly-linked list.
* The **add** method of the *SortedDoubleLinkedList* must be implemented to create a sorted double-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.

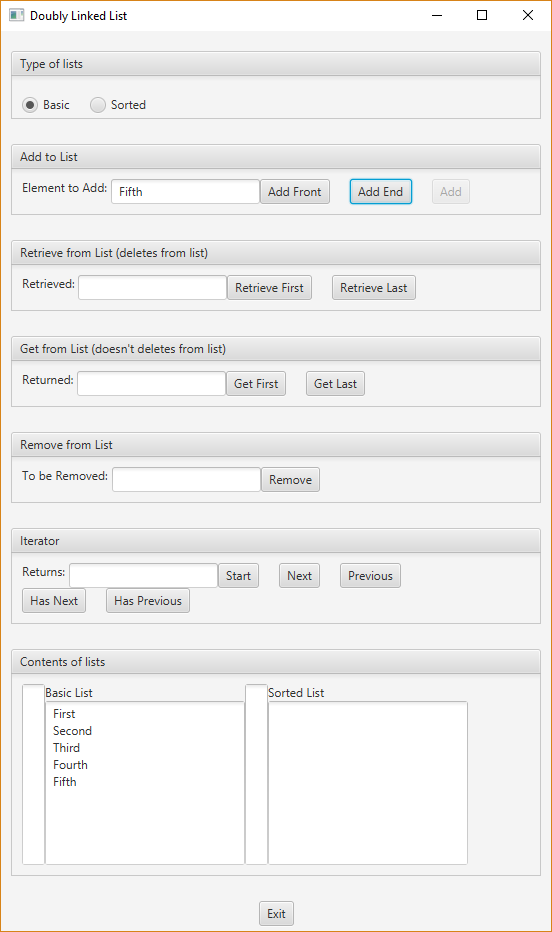
**Testing**

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.

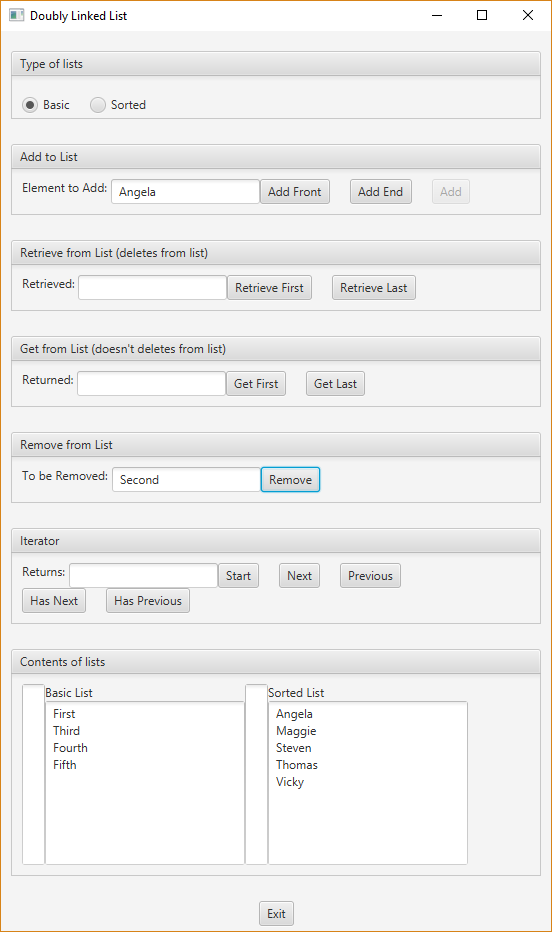
**Examples using GUI driver**

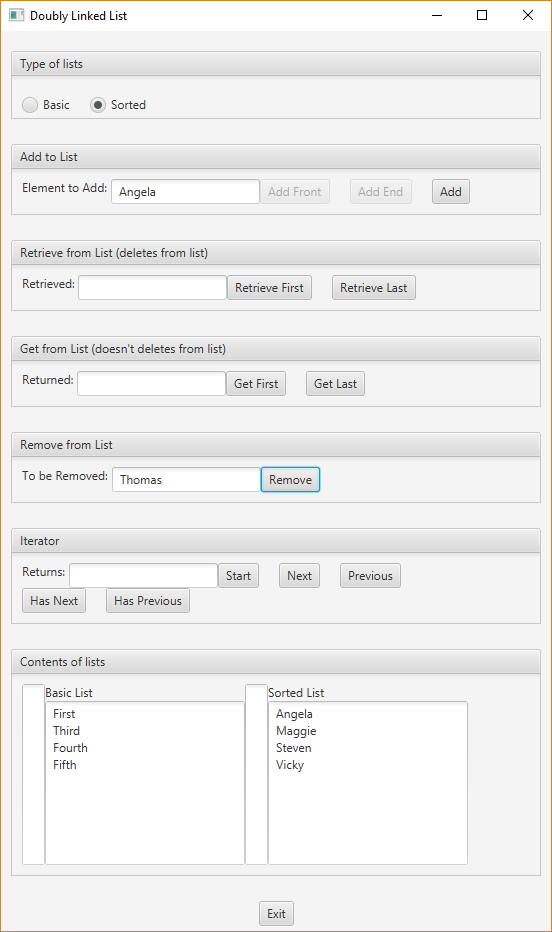
Adding to a Basic List Adding to a Sorted List



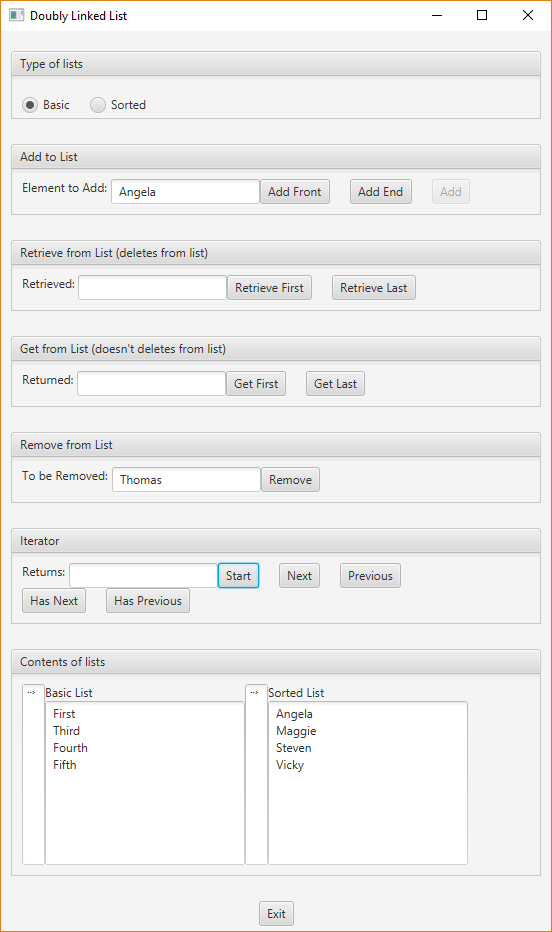


Removing Second from basic Removing Thomas from sorted

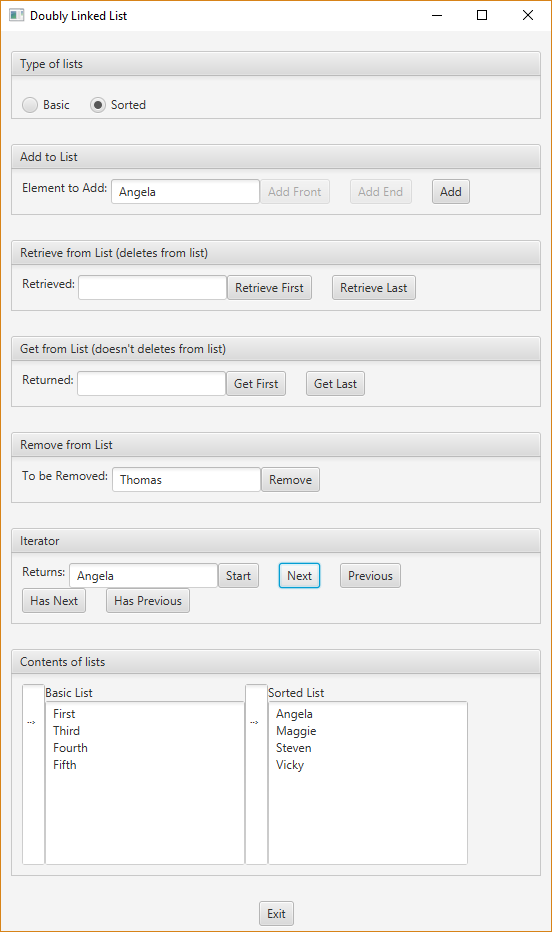




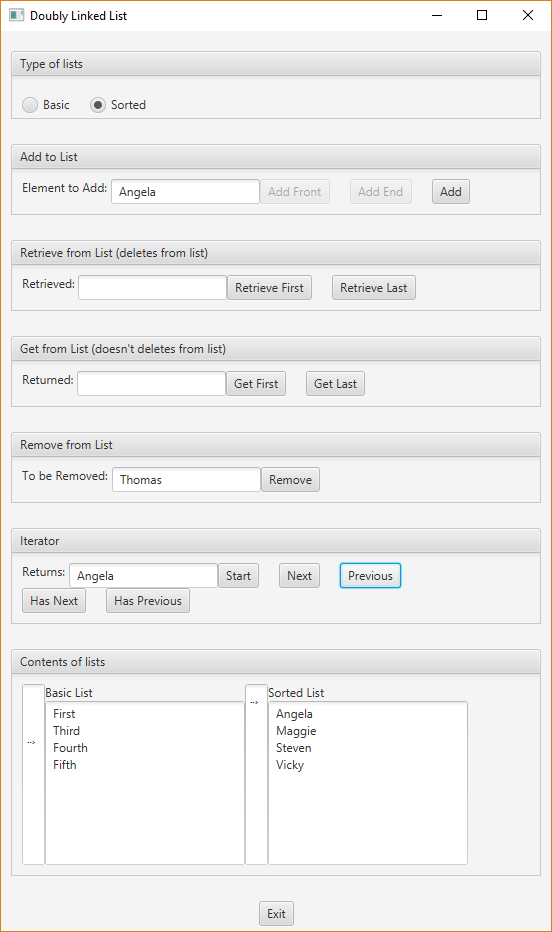
Start the iterators for Basic and Sorted. Think of iterators being “in between” nodes.



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



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



**Deliverables**

**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