Elizabeth Perez

Professor Monshi

CMSC 204

10/19/2020

**Assignment 3 Design Document**

|  |
| --- |
| BasicDoubleLinkedList implements Iterable |
| Node head  Node tail  Int size |
| |  | | --- | | Node | | T data  Node next  Node previous | | Node(T dataEntry, Node nextNode, Node previousNode)  Set data to dataEntry  Set next to nextNode  Set previous to previousNode  changeNext(Node n)  Set next to n  changePrevious(Node n)  Set previous to n |  |  | | --- | | basicLLIterator implements ListIterator | | Int cursor  ArrayList<T> list | | basicLLIterator()  Set cursor to 0  Set list to toArrayList()  hasNext()  If cursor is less than list size  Return true  Else  Return false  next()  If hasNext() is true  Declare and initialize T element to list entry at cursor index  Add 1 to cursor  Return element  Else  Throw a NoSuchElementException  hasPrevious()  If cursor is greater than 0  Return true  Else  Return false  previous()  If hasPrevious() is true  Subtract 1 from cursor  Declare and initialize T element to list entry at cursor index  Return element  Else  Throw a NoSuchElementException  add(), nextIndex(), previousIndex, remove(), set(T e)  Throw an UnsupportedOperationException |   addToEnd(T data)  If tail is null  Set tail and head to a Node(data, null, null)  Else  Declare a Node old and set to tail  Set tail to a Node(data, null, old)  Change old’s next to tail  Add 1 to size  Return the list  addToFront(T data)  If head is null  Set tail and head to a Node(data, null, null)  Else  Declare a Node old and set to tail  Set head to a Node(data, old, null)  Change old’s previous to head  Add 1 to size  Return the list  getFirst()  Return head  getLast()  Return tail  getSize()  Return size  iterator()  Return a new basicLLLIterator()  remove(T targetData, Comparator comparator)  If compare(targetData, head’s data) equals 0  Set head to Node(head’s next’s data, head’s next’s next, null)  Subtract 1 from size  Return the list  Declare a Node current and set to head  While compare(targetData, current’s data) is not equal to 0  If current’s next is null  Return null  Else  Set current to current’s next  If current’s next equals null  Change current’s previous’s next to null  Set tail to current’s previous  Subtract 1 from size  Return the list  Change current’s previous’s next to current’s next  Change current’s next’s previous to current’s previous  Set current to null  Subtract 1 from size  Return the list  retrieveFirstElement()  Declare T firstElement and set to head’s data  Set head to Node(head’s next’s data, head’s next’s next, null)  Subtract 1 from size  Return firstElement  retrieveLastElement()  Declare T lastElement and set to tail’s data  Set tail to Node(tail’s previous’s data, null, tail’s previous’s previous)  Subtract 1 from size  Return lastElement  toArrayList()  Create an ArrayList called list  Declare a Node n and set to head  While n’s next isn’t null  Add n’s data to list  Set n to n’s next  Add tail’s data to list  Return list |

|  |
| --- |
| SortedDoubleLinkedList extends BasicDoubleLinkedList |
| Comparator<T> comp |
| add(T data)  If head is equal to null  Set head and tail to Node(data, null, null)  Add 1 to size  Return the list  If compare(data, head’s data) is less than or equal to 0  Declare Node n and set to Node(data, head, null)  Change head’s previous to n  Set head to n  Add 1 to size  Return the list  Declare Node current and set to head  While compare(data, current’s data) is greater than 0  If current’s next is equal to null  Declare Node n and set to Node(data, null, tail)  Change tail’s next to n  Set tail to n  Add 1 to size  Return the list  Set current to current’s next  Declare Node n and set to Node(data, current, current’s previous)  Change current’s previous’s next to n  Change current’s previous to n  Add 1 to size  Return the list  addToEnd(T data)  Throws an UnsupportedOperationException  addToFront(T data)  Throws an UnsupportedOperationException  iterator()  Call super’s iterator()  remove(T data, Comparator comparator)  Call super’s remove(data, comparator) |