/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Class: CMSC204 CRN 22378

Program: Assignment # 3

Instructor: Professor Alexander

Description: Double Linked Lists Pseudocode/UML

Due: 10/14/2020

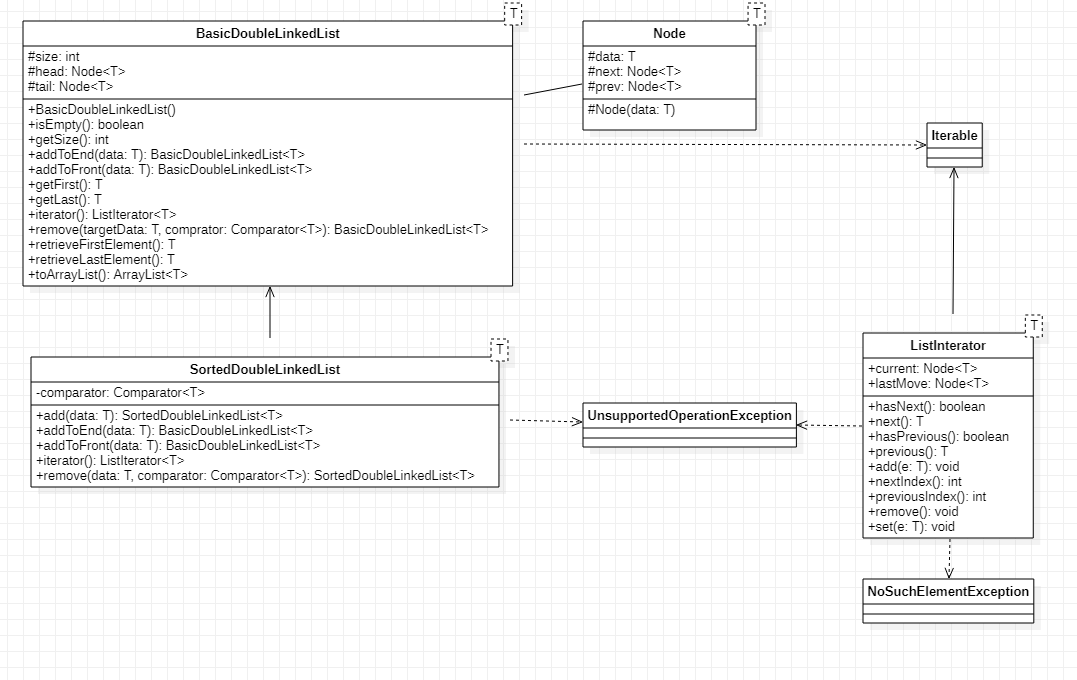
I pledge that I have completed the programming assignment independently.

I have not copied the code from a student or any source.

Anthony Liu

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

UML Diagram



**Pseudocode**

**BasicDoubleLinkedList**

**addToEnd**

DECLARE new node with parameter

IF list is empty

SET head to new node

SET tail to new node

ELSE

SET tail next node to new node

SET new node previous node to tail

SET tail to the new node

INCREASE size by one

RETURN this

**addToFront**

DECLARE new node with parameter

IF list is empty

SET head to new node

SET tail to new node

ELSE

SET head previous node to new node

SET new nodes next node to head

SET head to new node

INCREASE size by one

RETURN this

**getFirst**

IF list is empty

Return null

Return head data

**getLast**

IF list is empty

Return null

Return tail.data

**interator**

return ListerIterator item

**hasNext**

if current is not null return true

**next**

IF the list does not have next item

THROW no such element exception

DECLARE currentData to head data

SET last moved node to current node

RETURN currentData

**hasPrevious**

IF last move is not null return true

**previous**

IF previous value is null

THROW no such element exception

SET previous data to last move data

SET current to last move

SET last move to last moves previous

RETURN previousData

**remove**

SET current node to head

WHILE current is not null

IF current data is equal to target data

IF current is the head

SET head to head next

DECREASE list size by one

BREAK

ELSE IF current is the tail

SET Tail to Tail previous

SET tail next to null

DECREASE size by one

BREAK

ELSE

SET current previous next node to current next node

SET current next node previous node to current node previous node

DECREASE size by one

BREAK

SET current to current next

RETURN this

**retrieveFirstElement**

IF list is empty

Return null

SET first data to head data

SET head to head next

SET head previous to null

DECRESE size by one

RETURN first data

**retrieveLastElement**

IF list is empty

RETURN null

SET last data to tail data

SET tail to tail previous

SET tail next to null

DECREASE size by one

RETURN last data

**toArrayList**

CREATE new ArrayList of size size

SET current node to head node

WHILE current node is not null

ADD current node to list

SET current node to current next node

RETURN ArrayList

**SortedDoubleLinkedList**

**add**

DECLARE new node with data

IF list is empty

SET head and tail node to new node

ELSE IF data is less than head node data

SET new node next node to head

SET head previous node to new node

SET head to new node

ELSE IF data is greater than tail node data

SET new node previous node to tail node

SET tail node next node to new node

SET tail node to new node

ELSE

DECLARE current node to head next

WHILE current next node is not null and current node data is less than new node data

SET current node to current next

SET new node next node to current node

SET new node previous to current previous node

SET current previous next node to new node

SET current previous node to new node

INCREASE size by one

RETURN this