Data Models

This Project Contains 2 Classes:

NodeOneSided: A singly sided Node class which contains attributes below:

- **Data:** Integer type data of the Node
- **Next:** Pointer to the next Node

This particular type has two constructores:

1. **NodeOneSided(int data):** which creates the Node with a given integer.
2. **NodeOneSided(int data, NodeOneSided next):** which creates the Node with a given integer and a pointer to the next Node

NodeTwoSided: A double sided Node class which contains attributes below:

- **Data:** Integer type data of the Node
- **Next:** Pointer to the next Node
- **Previous:** pointer to the previous Node

This type can only be instantiated with an integer.

Data Structures

This Project Contains 2 Data Structures below with the same methods

1. **LinkedList:** Is a singly way linked list
2. **TwoSidedLinkedList:** Is a double way linked list

These Data Structures have been given a Boolean flag named “AUTO_SORT” which can be given once through the constructor. Once the flag is active, these Data Structures will automatically use a sorted insert method which will cause an always sorted linked list.

This flag has been declared as a private final attribute and can’t be changed again.

Data Structures Parameters are explained below:

- **LinkedList(boolean AUTO_SORT):** Default constructor which only has the auto sort flag.
- **LinkedList(boolean AUTO_SORT, int head):** Secondary constructor which can accept an integer as the head value.
- **insertFirst(int data):** Receives an integer and inserts it in the first of the list. This method will cause an unsorted list if used directly.
- **insertAfter(int data, NodeOneSided node):** Receives an integer and a specific Node. The integer will be inserted after the Node. This method will cause an unsorted list if used directly.
- **insert(int data):** Main method of insertion. It receives an integer and checks for the auto sort flag and calls the proper insertion methods below.
- **Sortedinsert(int data):** This method uses a sorted insert algorithm. It receives an integer and inserts it in the list.
- **insertLast(int data):** Unsorted method of insertion which receives an integer and puts it in the last of the list.
- **deleteNode(NodeOneSided key):** Receives a key Node and deletes it if exists.
- **deleteFirst():** Deletes the head Node if exists.
- **deleteIndexOf(int index):** Deletes a specific index of the list passed as an argument. Only works if the given index can be reached. Indexes start from zero.
- **deleteLast():** Deletes the tail of the list if not empty.
- **insertRec(int data):** A recursive insert method which receives an integer and inserts it in the list using sorted insert algorithm.
- **insertRecHelper(int data, NodeOneSided ptr):** Helper method for the above method.
- **sortRec(LinkedList linkedList):** A recursive sort method which takes a LinkedList object and sorts it recursively.
- **search(int key):** Takes an integer key and searches for it in the List. returns the Node containing the key. Returns null if not found or the list is empty.
- **indexOf(int index):** Returns the value located in the given index. Only works if the given index can be reached. Indexes start from zero.

- **isPresent(int key):** Returns true of the given key is present in the list.
- **printList():** Helper method to automatically print the list.
- **getHead():** Returns the head of the List.
- **getTail():** Returns the Tail of the list.
- **isEmpty():** Checks if the list is empty.
- **getSize():** Returns the size of the list.

Helper Functions

This project contains a helper Class named Functions which contains static methods. Methods are explained below:

- **singlySidedSeedData(int n, int bound):** Seeds the data of a singly way Linked List with random numbers. It Receives an integer as the size of the list and a bound integer for the generated random numbers.
- **DoubleSidedSeedData(int n, int bound):** Seeds the data of a double way Linked List with random numbers. It Receives an integer as the size of the list and a bound integer for the generated random numbers.
- **commonNodes(TwoSidedLinkedList first, TwoSidedLinkedList second):** A normal method which takes two linked lists and returns the common data between them.
- **commonNodesRec(TwoSidedLinkedList first, TwoSidedLinkedList second):** Recursive algorithmof above method.
- **commonNodesRecOuter(NodeTwoSided firstPTR, NodeTwoSided secondPTR, TwoSidedLinkedList finalList):** Helper method for the recursive algorithm of finding common data between two lists.
- **commonNodesRecInner(NodeTwoSided firstPTR, NodeTwoSided secondPTR, TwoSidedLinkedList finalList):** Helper method for the recursive algorithm of finding common data between two lists.