**Client**

/\*\*

\*

\* @author Marie Larson

\* @version 2/21/2018

\*/

public class Client {

/\*\*

\* @param args the command line arguments

\*/

public static void main(String[] args) {

System.out.println("Begin Lab105");

SinglyLinkedList ten;

ten = new SinglyLinkedList<>();

System.out.println("Making a SinglyLinkedList");

for(int i = 1; i<=10;i++){

ten.addLast(i);

}

System.out.println("Print out SinglyLinkedList: \n" + ten);

System.out.println("\nFirst Element is removed: " + ten.removeFirst());

System.out.println("\nPrint updated list: \n" + ten);

System.out.println("Begin Part 1/B");

//making doublylinkedlist

DoublyLinkedList ten\_double;

ten\_double = new DoublyLinkedList();

//populate list

for(int i = 1; i<=10; i++){

ten\_double.addLast(i);

}

System.out.println("Print doublylinkedlist: \n" + ten\_double);

System.out.println("\nRemoving first element which is: " + ten\_double.removeFirst());

System.out.println("\nRemoving last element which is: " + ten\_double.removeLast());

System.out.println("\nPrinting updated DoublyLinkedList: \n" + ten\_double);

//Starting part 2

//setting limit

long limit = 100000000;

SinglyLinkedList test\_s = new SinglyLinkedList();

long start = System.currentTimeMillis();

for(int i = 0;i<limit;i++){

test\_s.addFirst(i);

}

long currentTime = System.currentTimeMillis();

long timeElapsed = currentTime - start;

String n = String.format("N = %,d", limit);

String time = String.format("Time(msec) = %,6d", timeElapsed);

System.out.printf("\nSinglyLinked \t%s \t%s\n", n, time);

//testing doublylinkedlist

DoublyLinkedList test\_d = new DoublyLinkedList();

start = System.currentTimeMillis();

for(int i = 0;i<limit;i++){

test\_d.addFirst(i);

}

currentTime = System.currentTimeMillis();

timeElapsed = currentTime - start;

time = String.format("Time(msec) = %,6d", timeElapsed);

System.out.printf("DoublyLinked \t%s \t%s\n", n, time);

System.out.println("End Lab105");

}

}

**SinglyLinkedList**

/\*\*

\*

\* @author marie

\*/

public class SinglyLinkedList<E> {

//--------------------------nested Node class----------------------------

private static class Node<E>{

private E element;

private Node<E> next;

public Node(E e, Node<E> n){

element = e;

next = n;

}

public E getElement() {

return element;

}

public Node<E> getNext(){

return next;

}

public void setNext(Node<E> n){

next = n;

}

}

//---------------------End of nested Node Class-----------------------------

//instance variables of the SinglyLinkedList

private Node<E> head = null; //head node of the list (or null if empty)

private Node<E> tail = null; //last node of the list (or null if empty)

private int size = 0; //number of nodes in the list

public SinglyLinkedList(){

}

//access methods

public int size(){

return size;

}

public boolean isEmpty(){

return size == 0;

}

//returns but does not use remove the first element

public E first(){

if(isEmpty()) return null;

return head.getElement();

}

//returns but does not remove the last element.

public E last(){

if(isEmpty()) return null;

return tail.getElement();

}

//update methods

public void addFirst(E e){

head = new Node<>(e, head);

if(size==0){

tail = head;

}

size++;

}

public void addLast(E e){

Node<E> newest = new Node<>(e, null);

if(isEmpty())

head = newest;

else

tail.setNext(newest);

tail = newest;

size++;

}

public E removeFirst(){

if(isEmpty()) return null;

E answer = head.getElement();

head = head.getNext();

size--;

if(size==0)

tail = null;

return answer;

}

//string method

public String toString(){

String temp = getClass().getName()+ "@ size: " + size;

for(int i =0;i<size;i++){

E r = removeFirst();

temp += " : " + r;

addLast(r);

}

return temp;

}

}

**DoublyLinkedList**

/\*\*

\*

\* @author Marie Larson

\* @version 2/20/2018

\* @param <E>

\*/

public class DoublyLinkedList<E> {

//--------------------------nested node class----------------------------

public class Node<E>{

private Node<E> prev;

private E element;

private Node<E> next;

public Node(E e, Node<E> n, Node<E> p){

element = e;

next = n;

prev = p;

}

public Node<E> getPrev(){

return prev;

}

public Node<E> setPrev(Node<E> p){

return prev;

}

public E getElement(){

return element;

}

public Node<E> getNext(){

return next;

}

public void setNext(Node<E> n){

next = n;

}

}

//------------------------end of nested node class--------------------------------

private Node<E> head = null;

private Node<E> tail = null;

private int size = 0;

public DoublyLinkedList(){

head = new Node<>(null,null,null);

tail = new Node<>(null,head,null);

head.setNext(tail);

}

/\*\*

\* returns size of list

\* @return

\*/

//access methods

public int size(){

return size;

}

/\*\*

\* Tests list for emptiness

\* @return

\*/

public boolean isEmpty(){

return size == 0;

}

/\*\*

\* returns but does not use remove the first element

\* @return

\*/

public E first(){

if(isEmpty()) return null;

return head.getNext().getElement();

}

/\*\*

\* returns but does not remove the last element.

\* @return

\*/

public E last(){

if(isEmpty()) return null;

return tail.getPrev().getElement();

}

//update methods

/\*\*

\* adds element e to beginning of list

\* @param e

\*/

public void addFirst(E e){

addBetween(e, head, head.getNext());

size++;

}

public void addLast(E e){

addBetween(e, tail, tail.getPrev());

}

public E removeFirst(){

if(isEmpty()) return null;

return remove(head.getNext());

}

public E removeLast(){

if(isEmpty()) return null;

return remove(tail.getPrev());

}

//adding private methods

private void addBetween(E e, Node<E> predecessor, Node<E> successor){

Node<E> newest = new Node<>(e, predecessor, successor);

predecessor.setNext(newest);

successor.setPrev(newest);

size++;

}

private E remove(Node<E> node){

Node<E> predecessor = node.getPrev();

Node<E> successor = node.getNext();

predecessor.setNext(successor);

successor.setPrev(predecessor);

size--;

return node.getElement();

}

@Override

public String toString(){

String temp = getClass().getName() + "@ size: " + size;

for( int i = 0; i<size;i++){

E r = removeFirst();

temp += " : " + r;

addLast(r);

}

return temp;

}

}

**Output:**

Begin Lab105

Making a SinglyLinkedList

Print out SinglyLinkedList:

SinglyLinkedList@ size: 10 : 1 : 2 : 3 : 4 : 5 : 6 : 7 : 8 : 9 : 10

First Element is removed: 1

Print updated list:

SinglyLinkedList@ size: 9 : 2 : 3 : 4 : 5 : 6 : 7 : 8 : 9 : 10

Begin Part 1/B

Print doublylinkedlist:

DoublyLinkedList@ size: 10 : 1 : 2 : 3 : 4 : 5 : 6 : 7 : 8 : 9 : 10

Removing first element which is: 1

Removing last element which is: 10

Printing updated DoublyLinkedList:

DoublyLinkedList@ size: 8 : 2 : 3 : 4 : 5 : 6 : 7 : 8 : 9

BUILD SUCCESSFUL (total time: 7 seconds)