|  |
| --- |
| 0hnQGAuNPo4wojnr77c2x8cWiaT70FGHBZ2h-3__FYQ.png |
| Linked List |
| Data Structures Made Easy |
|  |

dublin city university

# 1. *Singly LinkedList*

import java.util.\*;

class singly\_LinkedList<T>{

private static class Node<T>{

private T item;

private Node<T> next = null;

Node(T item0, Node<T> next0){

item = item0;

next = next0;

}

}

private Node <T> head = null;

private Node <T> tail = null;

private int numItems = 0;

public int size(){

return numItems;

}

public T get(int i){

if(i < 0 || i >= numItems)

throw new IndexOutOfBoundsException();

Node<T> tNode = head;

int index = 0;

while(index != i){

tNode = tNode.next;

index++;

}

T temporary = tNode.item;

return temporary;

}

public T set(int i, T t){

if(i < 0 || i >= numItems)

throw new IndexOutOfBoundsException();

Node<T> tNode = head;

int index = 0;

while(index != i){

tNode = tNode.next;

index++;

}

T temp = tNode.item;

tNode.item = t;

return temp;

}

public boolean add(T t){

tail = new Node<T>(t, tail);

t = tail.item;

add(numItems, t);

return true;

}

public boolean addFirst(T t){

head = new Node<T>(t, head);

t = head.item;

add(numItems, t);

return true;

}

public void add(int i, T t){

if(i < 0 || i > numItems)

throw new IndexOutOfBoundsException();

if(i == 0){

head = new Node<T>(t, head);

if(tail == null)

tail = head;

}

else{

Node<T> tNode = head;

int index = 0;

while(index != i -1){

tNode = tNode.next;

index++;

}

tNode.next = new Node<T>(t, tNode.next);

if(tail == tNode)

tail = tail.next;

}

numItems++;

}

public int indexOf(T t){

if(tail == null)

return -1;

else{

Node<T> tNode = head;

int index = 0;

while(tNode.item != t){

index++;

tNode = tNode.next;

}

return index;

}

}

public T removeLast(){

if(numItems == 0)

throw new NoSuchElementException();

T t = tail.item;

Node<T> tNode = head;

int index = 0;

while(index != numItems - 1){

tNode = tNode.next;

index++;

}

tail = tNode.next;

numItems--;

return t;

}

public static void main(String [] args){

singly\_LinkedList<String> s\_List = new singly\_LinkedList<String>();

System.out.print("Enter words: ");

while(!Console.endOfFile()){

String word = Console.readToken();

s\_List.add(word);

}

System.out.println('\n' + "LINKEDLIST");

System.out.println("==========");

for(int index\_1 = 0; index\_1 < s\_List.size(); index\_1++)

System.out.println("Index " + index\_1 + ": " + s\_List.get(index\_1));

System.out.println('\n' + "Insert to front: dog");

s\_List.addFirst("dog");

System.out.println('\n' + "Remove last word: " + s\_List.removeLast());

System.out.println('\n' + "LINKEDLIST");

System.out.println("==========");

for(int index\_2 = 0; index\_2 < s\_List.size(); index\_2++)

System.out.println("Index " + index\_2 + ": " + s\_List.get(index\_2));

}

}

# 2. *Doubly LinkedList*

import java.util.\*;

class doubly\_LinkedList<T>{

private static class Node<T>{

private T item;

private Node<T> pred = null;

private Node<T> next = null;

Node(T item0, Node<T> pred0, Node<T> next0){

item = item0;

pred = pred0;

next = next0;

}

}

private Node<T> head = null;

private Node<T> tail = null;

private int numItems = 0;

public int size(){

return numItems;

}

public T get(int i){

if(i < 0 || i >= numItems)

throw new IndexOutOfBoundsException();

Node<T> tNode = head;

int index = 0;

while(index != i){

tNode = tNode.next;

index++;

}

T temp = tNode.item;

return temp;

}

public T set(int i, T t){

if(i < 0 || i >= numItems)

throw new IndexOutOfBoundsException();

Node<T> tNode = head;

int index = 0;

while(index != i){

tNode = tNode.next;

index++;

}

T temp = tNode.item;

tNode.item = t;

return temp;

}

public void add(int i, T t){

if(i < 0 || i > numItems)

throw new IndexOutOfBoundsException();

if(head == null){

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

if(tail == null)

tail = head;

else

head.next.pred = head;

}

else{

Node<T> tNode = head;

int index = 0;

while(index != i -1){

tNode = tNode.next;

index++;

}

tNode.next = new Node<T>(t, tNode, tNode.next);

if(tail == tNode)

tail = tNode.next;

else

tNode.next.pred = tNode.next;

}

numItems++;

}

public T removeLast(){

if(numItems == 0)

throw new NoSuchElementException();

T t = tail.item;

tail = tail.pred;

if(tail!=null)

tail.next = null;

else

head = null;

numItems--;

return t;

}

public static void main(String [] args){

doubly\_LinkedList<String> d\_List = new doubly\_LinkedList<String>();

System.out.print("Enter words: ");

int count = 0;

while(!Console.endOfFile()){

String word = Console.readToken();

d\_List.add(count, word);

count++;

}

System.out.println('\n' + "LINKEDLIST");

System.out.println("==========");

for(int index\_1 = 0; index\_1 < d\_List.size(); index\_1++)

System.out.println("Index " + index\_1 + ": " + d\_List.get(index\_1));

System.out.println('\n' + "Insert at index 2: dog");

d\_List.add(2, "dog");

System.out.println('\n' + "Remove last word: " + d\_List.removeLast());

System.out.println('\n' + "LINKEDLIST");

System.out.println("==========");

for(int index\_2 = 0; index\_2 < d\_List.size(); index\_2++)

System.out.println("Index " + index\_2 + ": " + d\_List.get(index\_2));

}

}