# 

# Worksheet #3

# CS 249 – Data Structures

30 September 2014

by

Brandon Horner

A circularly linked list has no start and no end. Instead the “last” node points back to the “first” node. The only reference to nodes within the list is through the node current.

Below is part of the implementation of circularly linked list. You will need to complete it. Notice the CircList class uses the Node class provided below. Your job is to implement the following methods in the CircList class:

* insert(long id) inserts a new node (before the current if other nodes exist). After insertion current should point to new node. Be sure to consider the following cases: the list is empty, the list has one element, the list has more than one element
* delete() removes the current node – be sure to consider the same cases considered for insertion. After deletion current should now point to whatever the node you deleted was pointing to.
* find(long key) starting at current find and return the first node that matches the key – remember to only search through the list once. If there is no such node, return an empty node.
* delete(long key) – starting at current delete the first node that matches the key – remember, you must find it first. If there is no match, return an empty node.

Public class Node{

public long iData; // data item (key)

public Node next; // next node in the list

// -------------------------------------------------------------

public Node(long id) { // constructor

iData = id; // (next automatically null)

}

// -------------------------------------------------------------

public void displayNode() { // display yourself

System.out.print(iData + " ");

}

} // end class Node

Public class CircList{

private Node current; // ref to current node

private int count; // # of nodes on list

public CircList() { // constructor

count = 0; // no nodes in the list yet

current = null;

}

public void insert(long id) { // insert node before current

Node newNode = new Node(id);

newNode.next = current;

current = newNode;

}

public Node delete() { // delete current node

Node temp = current;

current = current.next;

temp.next = null; //make sure no access to temp (optional)

return temp;

}

public Node find(long key) { // find first node starting at current with given key

Node temp = current;

int getHome = count;

while(getHome>0){

if(temp.iData == key)

return temp;

else{

temp = temp.next;

getHome--

}

}//end while

return null;

}

public Node delete(long key){ // delete node with given key

Node temp = current;

int getHome = count;

while(getHome>0){

if(temp.iData == key)

current = current.next;

temp.next = null;

return temp;

else{

temp = temp.next;

getHome--

}

}//end while

return null;

}

} // end class CircList