Implementation of Circular Linked List

import java.util.Scanner; class Node

{

int data;//to store data

Node link;//to store the address of next node Node(int data)

{

this.data=data; this.link=null;

}

}

class CircularLL

{

Node head; int lenght; CircularLL()

{

this.head=null; this.lenght=0;

}

public void insertBeg(Node newNode)

{

if(head==null)

{

head=newNode; newNode.link=head;

++lenght;

}

else

{

Node cp=head; while(cp.link!=head)

{

cp=cp.link;

}

newNode.link=head; head=newNode; cp.link=newNode;

++lenght;

}

display();

}

public void insertEnd(Node newNode)

{

Node cp=head; if(head==null)

{

head=newNode; newNode.link=head;

++lenght;

}

else

{

while(cp.link!=head)

{

cp=cp.link;

}

cp.link=newNode; newNode.link=head;

++lenght;

}

display();

}

public void insertPos(Node newNode, int pos)

{

int index=0; Node cp=head; Node pp=null;

boolean found=false;

while(cp.link!=head)

{

if(index==pos)

{

found=true; break;

}

index++; pp=cp; cp=cp.link;

}

if(found)

{

pp.link=newNode; newNode.link=cp;

++lenght;

}

else

{

System.out.println("Invalid Position");

}

display();

}

public void deleteBeg()

{

Node cp=head; if(head==null)

{

System.out.println("Empty LL");

}

else if(cp.link==head)

{

head=null; lenght=0;

}

else

{

Node firstNode=head; while(cp.link!=head)

{

cp=cp.link;

}

head=firstNode.link; cp.link=firstNode.link;

--lenght;

}

display();

}

public void deleteEnd()

{

Node cp=head; if(head==null)

{

System.out.println("Empty LL");

}

else if(cp.link==head)

{

head=null; lenght=0;

}

else

{

Node firstNode=head; Node pp=null; while(cp.link!=head)

{

pp=cp; cp=cp.link;

}

pp.link=firstNode;

--lenght;

}

display();

}

public void deletePos(int pos)

{

if(pos==0)

{

deleteBeg();

}

else if(pos==lenght-1)

{

deleteEnd();

}

else

{

int index=0; Node cp=head; Node pp=null;

boolean found=false;

while(cp.link!=head)

{

if(index==pos)

{

found=true; break;

}

index++; pp=cp; cp=cp.link;

}

if(found)

{

pp.link=cp.link;

--lenght;

}

else

{

System.out.println("Invalid Position");

}

}

display();

}

public void display()

{

if(head==null)

{

System.out.println("No Nodes");

}

else

{

System.out.print("Head"); Node cp=head; while(cp.link!=head)

{

System.out.print("|"+cp.data+"|->"); cp=cp.link;

}

System.out.print("|"+cp.data+"|->"); System.out.print("Head");

}

}

public void search(int data)

{

boolean found=false;

Node cp=head; while(cp.link!=head)

{

if(cp.data==data)

{

found=true; break;

}

cp=cp.link;

}

if(cp.data==data)

{

found=true;

}

if(found)

{

System.out.println("Data found ");

}

else

{

System.out.println("Not Found ");

}

}

}

public class CLLOperations {

public static void main(String[] args) { int data;

Scanner scanner=new Scanner(System.in); int choice=0;

CircularLL mylist=new CircularLL(); Node newNode;

int pos=0; while(choice<9)

{

System.out.println("1.Insert-Begin 2.Insert-End 3.Insert-Pos 4.Delete-Begin 5.Delete-End 6.

Delete-Pos 7.Display 8.Search 9.Exit "); choice=scanner.nextInt(); switch(choice)

{

case 1:System.out.println("Data:"); data=scanner.nextInt(); newNode=new Node(data); mylist.insertBeg(newNode); break;

case 2:System.out.println("Data:"); data=scanner.nextInt(); newNode=new Node(data); mylist.insertEnd(newNode); break;

case 3:System.out.println("Data:"); data=scanner.nextInt(); System.out.println("Position:"); pos=scanner.nextInt(); newNode=new Node(data); mylist.insertPos(newNode,pos); break;

case 4:mylist.deleteBeg();break; case 5:mylist.deleteEnd();break;

case 6:System.out.println("Position:"); pos=scanner.nextInt();

mylist.deletePos(pos); case 7:mylist.display();break;

case 8:System.out.println("Data:"); data=scanner.nextInt(); mylist.search(data);

}

}

}

}