**Write a Java program to implement Circular Linked List Using Array And Class**

import java.util.Scanner;

class Node

{

protected int data;

protected Node link;

public Node()

{

link = null;

data = 0;

}

public Node(int d,Node n)

{

data = d;

link = n;

}

public void setLink(Node n)

{

link = n;

}

public void setData(int d)

{

data = d;

}

public Node getLink()

{

return link;

}

public int getData()

{

return data;

}

}

 class linkedList

{

protected Node start ;

protected Node end ;

public int size ;

  public linkedList()

{

start = null;

end = null;

size = 0;

}

public boolean isEmpty()

{

return start == null;

}

public int getSize()

{

return size;

}

/\* Function to insert element at the begining \*/

public void insertAtStart(int val)

{

Node nptr = new Node(val,null);

nptr.setLink(start);

if(start == null)

{

start = nptr;

nptr.setLink(start);

end = start;

}

else

{

end.setLink(nptr);

start = nptr;

}

size++ ;

}

/\* Function to insert element at end \*/

public void insertAtEnd(int val)

{

Node nptr = new Node(val,null);

nptr.setLink(start);

if(start == null)

{

start = nptr;

nptr.setLink(start);

end = start;

}

else

{

end.setLink(nptr);

end = nptr;

}

size++ ;

}

/\* Function to insert element at position \*/

public void insertAtPos(int val , int pos)

{

Node nptr = new Node(val,null);

Node ptr = start;

pos = pos - 1 ;

for (int i = 1; i < size - 1; i++)

{

if (i == pos)

{

Node tmp = ptr.getLink() ;

ptr.setLink( nptr );

nptr.setLink(tmp);

break;

}

ptr = ptr.getLink();

}

size++ ;

}

/\* Function to delete element at position \*/

public void deleteAtPos(int pos)

{

if (size == 1 && pos == 1)

{

start = null;

end = null;

size = 0;

return ;

}

if (pos == 1)

{

start = start.getLink();

end.setLink(start);

size--;

return ;

}

if (pos == size)

{

Node s = start;

Node t = start;

while (s != end)

{

t = s;

s = s.getLink();

}

end = t;

end.setLink(start);

size --;

return;

}

Node ptr = start;

pos = pos - 1 ;

for (int i = 1; i < size - 1; i++)

{

if (i == pos)

{

Node tmp = ptr.getLink();

tmp = tmp.getLink();

ptr.setLink(tmp);

break;

}

ptr = ptr.getLink();

}

size-- ;

}

/\* Function to display contents \*/

public void display()

{

System.out.print("\nCircular Singly Linked List = ");

Node ptr = start;

if (size == 0)

{

System.out.print("empty\n");

return;

}

if (start.getLink() == start)

{

System.out.print(start.getData()+ "->"+ptr.getData()+ "\n");

return;

}

System.out.print(start.getData()+ "->");

ptr = start.getLink();

while (ptr.getLink() != start)

{

System.out.print(ptr.getData()+ "->");

ptr = ptr.getLink();

}

System.out.print(ptr.getData()+ "->");

ptr = ptr.getLink();

System.out.print(ptr.getData()+ "\n");

}

}

/\* Class CircularSinglyLinkedList \*/

public class CircularSinglyLinkedList

{

public static void main(String[] args)

{

Scanner scan = new Scanner(System.in);

/\* Creating object of linkedList \*/

linkedList list = new linkedList();

System.out.println("Circular Singly Linked List Test\n");

char ch;

/\* Perform list operations \*/

do

{

System.out.println("\nCircular Singly Linked List Operations\n");

System.out.println("1. insert at begining");

System.out.println("2. insert at end");

System.out.println("3. insert at position");

System.out.println("4. delete at position");

System.out.println("5. check empty");

System.out.println("6. get size");

int choice = scan.nextInt();

switch (choice)

{

case 1 :

System.out.println("Enter integer element to insert");

list.insertAtStart( scan.nextInt() );

break;

case 2 :

System.out.println("Enter integer element to insert");

list.insertAtEnd( scan.nextInt() );

break;

case 3 :

System.out.println("Enter integer element to insert");

int num = scan.nextInt() ;

System.out.println("Enter position");

int pos = scan.nextInt() ;

if (pos <= 1 || pos > list.getSize() )

System.out.println("Invalid position\n");

else

list.insertAtPos(num, pos);

break;

case 4 :

System.out.println("Enter position");

int p = scan.nextInt() ;

if (p < 1 || p > list.getSize() )

System.out.println("Invalid position\n");

else

list.deleteAtPos(p);

break;

case 5 :

System.out.println("Empty status = "+ list.isEmpty());

break;

case 6 :

System.out.println("Size = "+ list.getSize() +" \n");

break;

default :

System.out.println("Wrong Entry \n ");

break;

}

/\* Display List \*/

list.display();

System.out.println("\nDo you want to continue (Type y or n) \n");

ch = scan.next().charAt(0);

} while (ch == 'Y'|| ch == 'y');

}

}

