Java - Numerical Problems (http://www.sanfoundry.com/java-programming-examples-numerical-problems-algorithms/)
Java - Combinatorial Problems (http://www.sanfoundry.com/java-programming-examples-combinatorial-problems-algorithms/)
Java - Graph Problems (http://www.sanfoundry.com/java-programming-examples-graph-problems-algorithms/)
Java - Hard Graph Problems (http://www.sanfoundry.com/java-programming-examples-hard-graph-problems-algorithms/)
Java - Computation Geometry (http://www.sanfoundry.com/java-programming-examples-computational-geometry-problems-algorithms/)
Java - Sets & Strings (http://www.sanfoundry.com/java-programming-examples-set-string-problems-algorithms/)
Java - Data-Structures (http://www.sanfoundry.com/java-programming-examples-data-structures/)
Java - Collection API Problems (http://www.sanfoundry.com/java-programming-examples-collection-api/)

C++ Algorithms

C++ - Numerical Problems (http://www.sanfoundry.com/cpp-programming-examples-numerical-problems-algorithms/)
C++ - Combinatorial Problems (http://www.sanfoundry.com/cpp-programming-examples-combinatorial-problems-algorithms/)
C++ - Graph Problems (http://www.sanfoundry.com/cpp-programming-examples-graph-problems-algorithms/)
C++ - Hard Graph Problems (http://www.sanfoundry.com/cpp-programming-examples-hard-graph-problems-algorithms/)
C++ - Computation Geometry (http://www.sanfoundry.com/cpp-programming-examples-computational-geometry-problems-algorithms/)
C++ - Sets & Strings (http://www.sanfoundry.com/cpp-programming-examples-set-string-problems-algorithms/)
C++ - Data-Structures (http://www.sanfoundry.com/cpp-programming-examples-data-structures/)
C++ - STL Library (http://www.sanfoundry.com/cpp-programming-examples-stl/)

C Algorithms

C - Numerical Problems (http://www.sanfoundry.com/c-programming-examples-numerical-problems-algorithms/)
C - Combinatorial Problems (http://www.sanfoundry.com/c-programming-examples-combinatorial-problems-algorithms/)
C - Graph Problems (http://www.sanfoundry.com/c-programming-examples-graph-problems-algorithms/)
C - Hard Graph Problems (http://www.sanfoundry.com/c-programming-examples-hard-graph-problems-algorithms/)
C - Computation Geometry (http://www.sanfoundry.com/c-programming-examples-computational-geometry-problems-algorithms/)
C - Sets & Strings (http://www.sanfoundry.com/c-programming-examples-set-string-problems-algorithms/)
C - Data-Structures (http://www.sanfoundry.com/c-programming-examples-data-structures/)

« Prev Page (http://www.sanfoundry.com/java-program-implement-doubly-linked-list/)
Next Page (http://www.sanfoundry.com/java-program-implement-d-ary-heap/) »

Java Program to Implement Circular Singly Linked List

This is a Java Program to implement a Circular Singly Linked List. A linked list is a data structure consisting of a group of nodes which together represent a sequence. Under the simplest form, each node is composed of a data and a reference (in other words, a link) to the next node in the sequence. This structure allows for efficient insertion or removal of elements from any position in the sequence. In a circular singly linked list each node has only one link which points to the next node in the list and the last node of the list points to the first element of the list.

Here is the source code of the Java program to implement Circular Singly Linked List. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

```java
/*
 *  Java Program to Implement Circular Singly Linked List
 */

import java.util.Scanner;

/*  Class Node  */
class Node
{
    protected int data;
    protected Node link;

    /*  Constructor  */
    public Node()
    {
        link = null;
        data = 0;
    }
    /*  Constructor  */
    public Node(int d,Node n)
    {
        data = d;
        link = n;
    }
    /*  Function to set link to next Node  */
    public void setLink(Node n)
    {
        link = n;
    }
    /*  Function to set data to current Node  */
    public void setData(int d)
    {
        data = d;
    }
    /*  Function to get link to next node  */
    public Node getLink()
    {
        return link;
    }
    /*  Function to get data from current Node  */
    public int getData()
    {
        return data;
    }
}

/* Class linkedList */
class linkedList
{
    protected Node start ;
    protected Node end ;
    public int size ;

    /* Constructor */
    public linkedList()
    {
        start = null;
        end = null;
        size = 0;
    }
    /* Function to check if list is empty */
    public boolean isEmpty()
    {
        return start == null;
    }
    /* Function to get size of the list */
    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
        {
```

```java
            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');
    }
}
```

```
Circular Singly Linked List Test


Circular Singly Linked List Operations

1. insert at begining
2. insert at end
3. insert at position
4. delete at position
5. check empty
6. get size
5
Empty status = true

Circular Singly Linked List = empty

Do you want to continue (Type y or n)

y

Circular Singly Linked List Operations

1. insert at begining
2. insert at end
3. insert at position
4. delete at position
5. check empty
6. get size
1
Enter integer element to insert
4

Circular Singly Linked List = 4->4

Do you want to continue (Type y or n)

y

Circular Singly Linked List Operations

1. insert at begining
2. insert at end
3. insert at position
4. delete at position
5. check empty
6. get size
1
Enter integer element to insert
1

Circular Singly Linked List = 1->4->1

Do you want to continue (Type y or n)

y

Circular Singly Linked List Operations

1. insert at begining
2. insert at end
3. insert at position
4. delete at position
5. check empty
6. get size
```