# Lab No 01
# Objects and Classes

# Lab 1: Objects & Classes

## Objective

The objective of this lab is to define classes, create instances of these classes and perform operations using them.

## Scope:

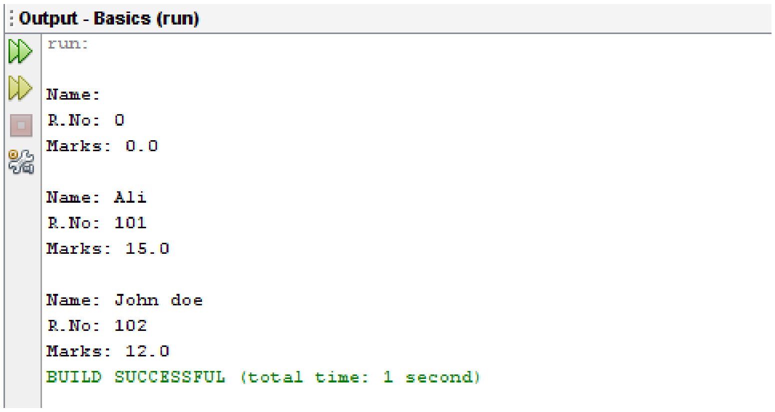
A simple java class is defined with some attributes and methods in it. Then some objects of this class are created in main function. In addition to that, a class containing an instance of another class has also been defined to familiarize the students with the concept of containership relation.

## Implementation:

### a) A Simple java class:

```
import java.util.Scanner;
public class Student {
    String name;
    int rno;
    float marks;
    public Student(){ // no arg constructor
      name=""; marks=0; rno=0;
      }

    public Student(String n, int r,float m){ //3-arg constructor
      name=n; rno=r; marks=m;
      }

    public void setname(String name) { //Method to set the name
      this.name = name;
      }
    public void setrno(int rno) {
      this.rno=rno;
      }
    public void setmarks(float marks) {
      this.marks=marks;
      }
    public String getname() {
      return name;
      }
    public int getrno() {
      return rno;
      }
    public float getmarks(){
      return marks;
      }
    public void setdata(){
        Scanner input=new Scanner(System.in);
        System.out.print("Enter name: ");
        name=input.nextLine();
        System.out.print("Enter Roll. No: ");
        rno=input.nextInt();
        System.out.print("Enter Marks: ");
        marks=input.nextFloat();
    }
    public void display() {
      System.out.print("\nName: " + name+ "\nR.No: " +rno+"\nMarks:
"+marks+"\n");
    }
}


public class Basics {

    public static void main(String[] args) {

        Student s1=new Student(); //creating object using no arg constructor
        Student s2=new Student("Ali", 101,15);
        s1.display();
        s2.display();
        s1.setname("John Doe"); //initializing object using setter methods;
        s1.setrno(102);
        s1.setmarks(12);
        s1.display();

    }
}
```

## Snapshot of Output:



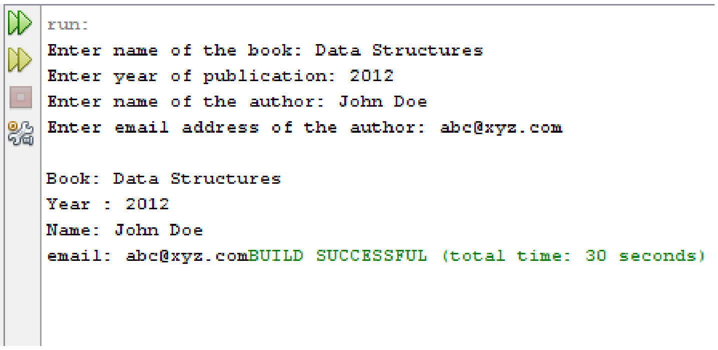
Output - Basics (run)

```
run:

Name:
R.No: 0
Marks: 0.0

Name: Ali
R.No: 101
Marks: 15.0

Name: John doe
R.No: 102
Marks: 12.0
BUILD SUCCESSFUL (total time: 1 second)
```

### b) Class Containing instance of another class as an attribute

```
class Author{
    private String name;
    private String email;

    public Author() {
      name=""; email="";
      }

    public Author(String name, String email) {
      this.name=name;
        this.email=email;
      }

    public void set_author() {
        Scanner input=new Scanner(System.in);
        System.out.print("Enter name of the author: ");
        name=input.nextLine();
        System.out.print("Enter email address of the author: ");
        email=input.next();
      }

    public void display_author(){
        System.out.print("\nName: " + name + "\nemail: "+email);
    }
      }// End of class author
    public class Book {
    String bname; //book name
    String year; //  year of publication
    Author author; // object of class Author

    public Book(){
       bname="";
       year="";
       author=new Author();
    }
    public Book(String name, String yr, String a_name, String email) {
      name=name;
      year=yr;
      author=new Author(a_name, email);
    }

    public void set_book() {
        Scanner input=new Scanner(System.in);
        System.out.print("Enter name of the book: ");
        bname=input.nextLine();
        System.out.print("Enter year of publication: ");
        year=input.next();
        author.set_author();
    }

public void display_book() {
        System.out.print("\nBook: " + bname + "\nYear : "+year);
        author.display_author();
    }
}


public class Basics {

    public static void main(String[] args) {
        Book b1=new Book();
        b1.set_book();
        b1.display_book();
    }
} // End of class Basics
```

**Snapshot of Output:**



```
run:
Enter name of the book: Data Structures
Enter year of publication: 2012
Enter name of the author: John Doe
Enter email address of the author: abc@xyz.com

Book: Data Structures
Year : 2012
Name: John Doe
email: abc@xyz.comBUILD SUCCESSFUL (total time: 30 seconds)
```

**Exercises:**

1. Create a class called Date. Its three attributes, all type int, should be called day, month, and year. Write a program that prompts the user to enter values in day, month, and year. The program then stores the Date in a variable of type Date, and finally prints it out.

2. Implement a class named Person having
    - attributes name, age and date-of-birth of types String, integer and Date respectively.
    - Methods to get and set these attributes.
    - A no-argument constructor.
    - A constructor that creates an instance of a person with specified values of attributes.
    - A method to display all the attributes of a person.

    Write a test program to create two persons and display them.

**HomeWork:**

Implement a class named Point to represent a point with x and y coordinates. The class contains:

- Two attributes x and y that represent coordinates.
- Methods to get and set these attributes.
- A no-argument constructor that  creates a point (0,0).
- A constructor that creates a point with specified coordinates.
- A method named distance that returns the distance from the current point to another point whose x and y coordinates are sent as arguments.

# Lab No 02
# Arrays

# Lab 2: Arrays

## Objective

The objective of this lab is to familiarize the students to the arrays

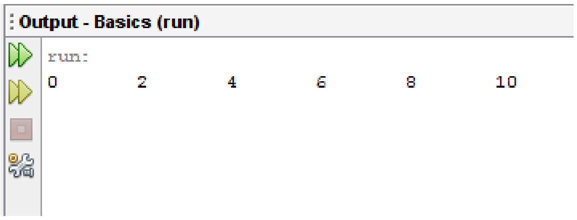
## Scope:

One and Two-dimensional Arrays of primitives as well as references types are defined and operations like copying, cloning, insertion and deletion are performed on them.

## Example 1: One Dimentional Arrays:

```
public class Basics {

    public static void main(String[] args) {

        int[] int_array= new int[6]; //creating an array of 6 elements

        for(int i=0;i<6;i++)
        {   int_array[i]=2*i;    } // initializing int_array

        for(int i=0;i<6;i++)
        {   System.out.print(int_array[i] + "\t"); //displaying int_array
          }
    }
}
```

## Snapshot of output

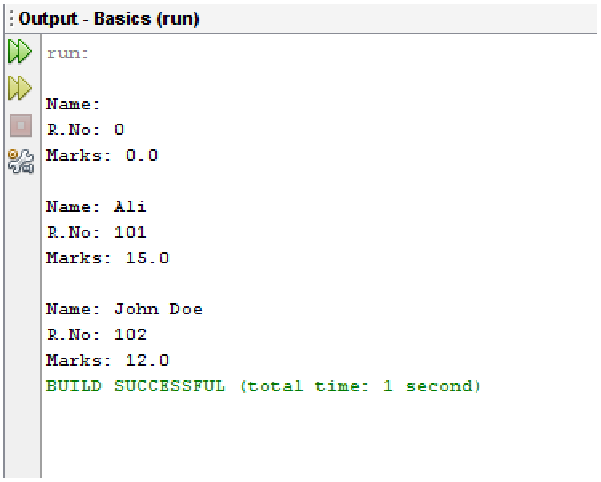


```
run:
0        2        4        6        8        10
```

## Example 2: Arrays of objects:

```
public class Basics {

    public static void main(String[] args) {

       Student s1=new Student(); //creating object using no arg constructor
       Student s2=new Student("Ali", 101,15);
       s1.display();
       s2.display();
       s1.setname("John Doe"); //initializing object using setter methods;
       s1.setrno(102);
```

```
        s1.setmarks(12);
        s1.display();
    }
}
```

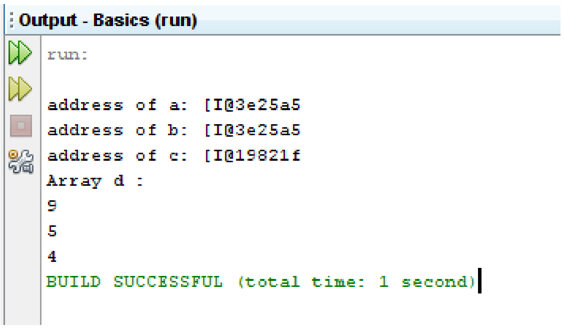


```
run:

Name: 
R.No: 0
Marks: 0.0

Name: Ali
R.No: 101
Marks: 15.0

Name: John Doe
R.No: 102
Marks: 12.0
BUILD SUCCESSFUL (total time: 1 second)
```

## Example 3: copying  and cloning arrays

```
public class Basics {

    public static void main(String[] args) {

        int[] a = {9, 5, 4};
        int[] b = a;// creates an alias to the object
        int[] c = (int[]) a.clone(); // creates  a new array of the same
                                     //size and copy elements of a in it.

        System.out.print("\naddress of a: " + a);
        System.out.print("\naddress of b: " + b); // prints same address as
                                                  //that of a

        System.out.println("\naddress of c: " + c);

        int[] d = new int[a.length];
        for(int i=0; i<a.length;i++) // copying array elements
            d[i]=a[i];

        System.out.println("Array d :");
        for(int i=0; i<a.length;i++) // copying array elements
            System.out.println(d[i]+"\t");
    }
}
```

## Snapshot of output

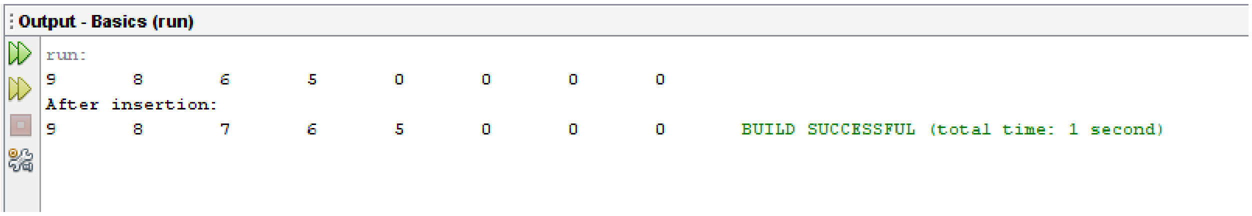


```
run:

address of a: [I@3e25a5
address of b: [I@3e25a5
address of c: [I@19821f
Array d :
9
5
4
BUILD SUCCESSFUL (total time: 1 second)
```

## Example 4: Insertion in arrays

```
     public static void main(String[] args) {

      int[] arr=new int[8]; //arr of 8 integers
      arr[0]=9;
      arr[1]=8;
      arr[2]=6;
      arr[3]=5;
      for(int i=0; i<arr.length;i++) // displaying array elements
          System.out.print(arr[i]+"\t");
      System.out.println();
      //inserting an element at an index
      int element=7;
      int index=2;
      for(int i=arr.length-1;i>index;i--)
      {
          arr[i]=arr[i-1]; // shifting array elements to the higher indexes
      }
      arr[index]=element;
      System.out.print("After insertion:\n");
      for(int i=0; i<arr.length;i++) // displaying array elements
          System.out.print(arr[i]+"\t");
  }

     }
```

## Snapshot of output

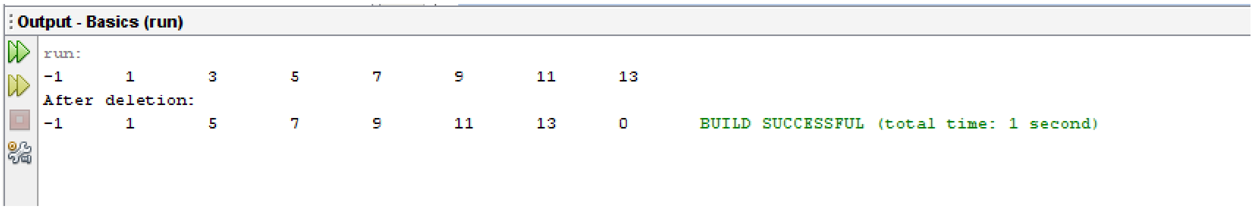


```
run:
9       8       6       5       0       0       0       0
After insertion:
9       8       7       6       5       0       0       0       BUILD SUCCESSFUL (total time: 1 second)
```

## Example 5: Deletion in arrays:

```
public class Basics {

    public static void main(String[] args) {

        int[] arr=new int[8]; //arr of 8 integers

        for(int i=0; i<arr.length;i++)
        { arr[i]=i*2;}
        for(int i=0; i<arr.length;i++) // displaying array elements
            System.out.print(arr[i]+"\t");
        System.out.println();
        //deleting an element from a given index
        int index=2;
        for(int i=index;i<arr.length-1;i++)
        {
            arr[i]=arr[i+1]; // shifting array elements to the lower indexes
        }
        arr[arr.length-1]=0; //just to show empty location
        System.out.print("After deletion:\n");
        for(int i=0; i<arr.length;i++) // displaying array elements
            System.out.print(arr[i]+"\t");
    }
}
```

## Snapshot of Output:



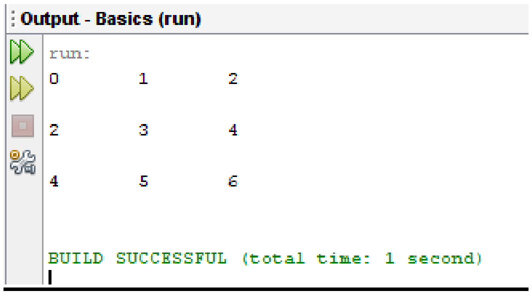
```
run:
-1      1       3       5       7       9       11      13
After deletion:
-1      1       5       7       9       11      13      0       BUILD SUCCESSFUL (total time: 1 second)
```

## Example 6: Two Dimensional Arrays:

```
public static void main(String[] args) {
         int row=3, col=3;
        int[][] arr=new int[row][col]; //2D-array declaration
```

```
       for(int i=0; i<row;i++)
           for(int j=0;j<col;j++) //2D-array initialization
                 arr[i][j]=i*2+j;

       for (int[] a : arr)
       { for (int i : a)
       { System.out.print(i + "\t"); }
       System.out.println("\n"); }
       System.out.println();

    }
   }
```

## Output:

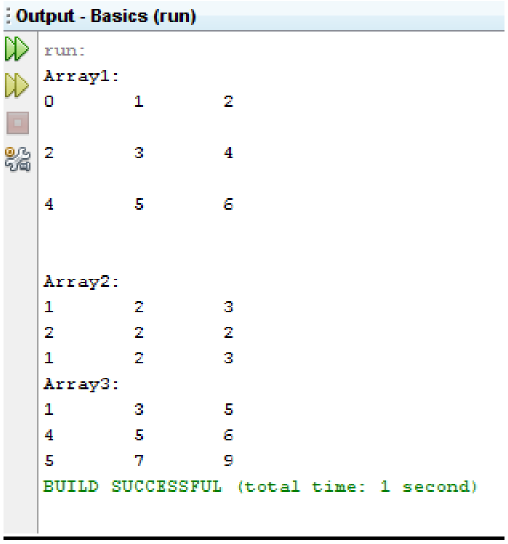


```
run:
0       1       2

2       3       4

4       5       6


BUILD SUCCESSFUL (total time: 1 second)
```

```
*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.


public class Basics {


    public static void main(String[] args) {
        int row=3, col=3;
       int[][] arr1=new int[row][col]; //2D-array declaration
       int[][] arr2= {{1,2,3},{2,2,2},{1,2,3}};
       int[][] arr3 = new int[3][3];
       for(int i=0; i<row;i++)
           for(int j=0;j<col;j++) //2D-array initialization
                 arr1[i][j]=i*2+j;


       System.out.print("Array1: \n"); // displaying arr1
       for (int[] a : arr1)
       { for (int i : a)
       { System.out.print(i + "\t"); }
       System.out.println("\n"); }
```

```
   System.out.println();

  System.out.print("Array2: \n");
   for(int i=0; i<row;i++)
   {
       for(int j=0;j<col;j++) //displaying 2D-array
            { System.out.print(arr2[i][j] + "\t");}
       System.out.print("\n");
    }
   System.out.print("Array3: \n");
   for(int i=0; i<row;i++)
   {
       for(int j=0;j<col;j++) //displaying 2D-array
            { arr3[i][j]=arr1[i][j] + arr2[i][j];
           System.out.print(arr3[i][j] + "\t");}
       System.out.print("\n");
    }

}
```

## Snapshot of output:



```
run:
Array1:
0       1       2

2       3       4

4       5       6


Array2:
1       2       3
2       2       2
1       2       3
Array3:
1       3       5
4       5       6
5       7       9
BUILD SUCCESSFUL (total time: 1 second)
```

## Exerecises:

1. Create a class called Student. Its three members should be studentName, registrationNumber and CGPA of appropriate data types. Write a function that accepts a list of students as input and returns student having maximum CGPA. (Size of the list can also be given as argument to the function).
2. Write a function to add the diagonal enteries of a square matrix.

## HomeWork:

Create two integer type arrays A1 and A2 such that A1 contains even numbers from 2 to 20 and A2 contains odd numbers from 1 to 19.
Write a function merge to merge the elements of both arrays into a single array that contains numbers from 1 to 20.

# Lab No 03
# List (Array based)

# Lab No. 3: List (Array Based)

**Objective:**

The objective of this lab is to implement List ADT using arrays.

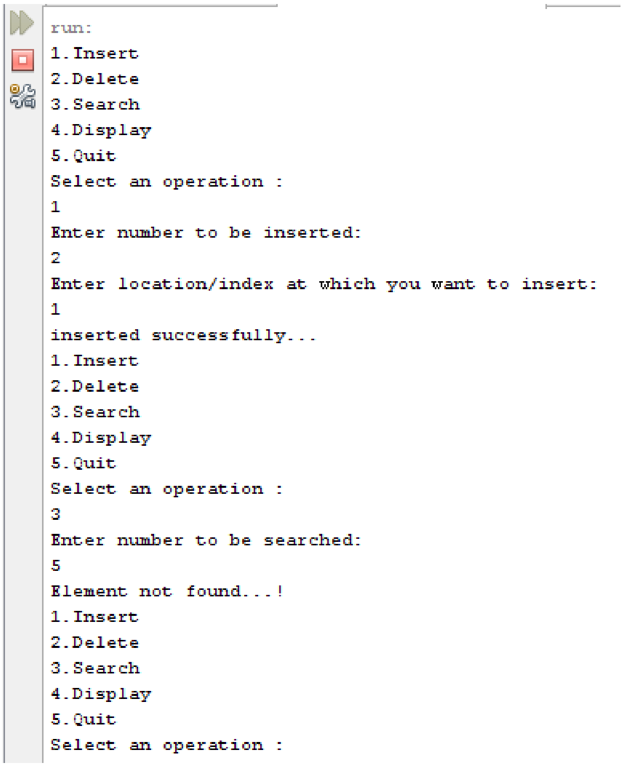
**Scope:**

An array of a given size will be declared and used as a list data structure. All the basic operations of a data structure such as insertion, deletion, searching, sorting and traversal will be discussed for this array based list.

**Implementation:** This program is an implementation of insertion and display of an array-based one-way linked list.

```
public class arrayList{
        int[] arr;
        int maxsize;
        int listsize;
        int curr_loc;

        arrayList() // no arg constructor
        {
            maxsize=10;
            arr=new int[10];
            curr_loc=0;
        }
        arrayList(int msize)// one arg constructor
        {
            maxsize=msize;
            arr=new int[maxsize];
            curr_loc=0;
        }

    public void insert(int loc, int val)
    {
        if(curr_loc>=arr.length)
        {System.out.println("Overflow...!!");}
        else
        {
            if(loc>=curr_loc)
            {arr[curr_loc]=val;}
            else
            { for(int i=curr_loc; i>loc;i--)
                {arr[i]=arr[i-1];}
              arr[loc]=val;
            }
            curr_loc++;
            System.out.println("inserted successfully...");
        }
    }

    public void delete(int val)
    {
        int loc=search(val);
        if(loc!=-1)
        {
            for(int i=loc;i<curr_loc-1;i++)
            {arr[i]=arr[i+1];}
            curr_loc--;
            System.out.println("Deleted successfully...!");
        }
        else
        {System.out.println("List already empty..! ");}
    }//end of delete

    public void display()
    {
        for(int i=0;i< curr_loc;i++)
        {
        System.out.println(arr[i]);
        }
    }

    public int search(int val)
    {   int loc=0;
       boolean found=false;
        for (int i=0;i<curr_loc;i++)
        {
          if(val==arr[i])
          {
              loc=i;
              found=true;
              break;
          }
        }
       if(found)
            { System.out.println("Found at location " + loc);
            return loc;
            }
        else
            { System.out.println("Element not found...!");
            return -1;
            }
        }// end of search
}

public class ArrayList_Demo {

    public static void main(String[] args) {
        arrayList a1=new arrayList(10);
        int opt;
        Scanner input = new Scanner(System.in);
        do {
            //System.out.println("0.Construct List\n");
            System.out.println("1.Insert");
            System.out.println("2.Delete");
            System.out.println("3.Search");
            System.out.println("4.Display");
            System.out.println("5.Quit");
            System.out.println("Select an operation : ");
            opt = input.nextInt();

            switch (opt) {
                case 1:
                    System.out.println("Enter number to be inserted: ");
                    int val = input.nextInt();
                    System.out.println("Enter location/index at which you 
want to insert: ");
                    int loc = input.nextInt();
                    a1.insert(loc, val);
                    break;
                case 2:
                    System.out.println("Enter number to be deleted: ");
                    val = input.nextInt();
                    a1.delete(val);
                    break;
                case 3:
                    System.out.println("Enter number to be searched: ");
                    val = input.nextInt();
                    a1.search(val);
                    break;
                case 4:
                    a1.display();
                    break;
                case 5:
                    System.exit(0);
                default:
                    System.out.println("Wrong choice\n");
            }/*End of switch*/
        }//end do
                while(opt!=5);
    }//end of main
}
```

## Snapshot of Output:



```
run:
1.Insert
2.Delete
3.Search
4.Display
5.Quit
Select an operation :
1
Enter number to be inserted:
2
Enter location/index at which you want to insert:
1
inserted successfully...
1.Insert
2.Delete
3.Search
4.Display
5.Quit
Select an operation :
3
Enter number to be searched:
5
Element not found...!
1.Insert
2.Delete
3.Search
4.Display
5.Quit
Select an operation :
```

## Exercises:

Write a program to delete all occurrences of a certain value from a given array list.

## Homework:

Create an array of Students(name, roll number, cgpa) and sort its elements according to the cpga from highest to lowest.

# Lab No 04
# Singly Linked List

# Lab No 04: Singly Linked List

## Objective:

The objective of this lab is to implement singly linked list using pointers.

## Scope:

A class of nodes will be specified. Using this class, a linked list will be created dynamically. All typical operations of a data structure that is insertion, deletion, searching, and appending will be dicussed and implemented for this dynamic list.

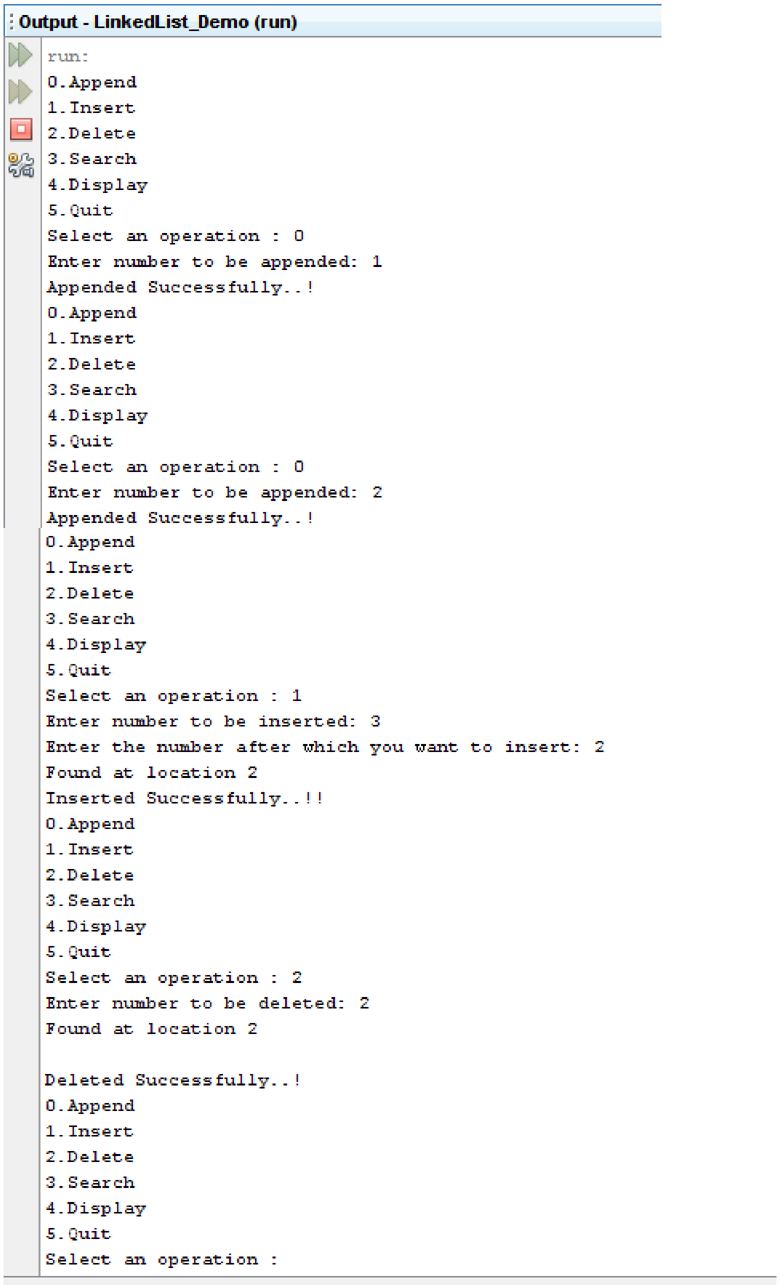
## Implementation:

```
public class Node {
    private int data;
    private Node next;

    public Node() // no arg constructor
    {
        data=-1;
        next=null;
    }
    public Node(int d) //one arg constructor
    {
        data=d;
        next=null;
    }

    public void setdata(int d)
    {   data=d;}

    int getdata()
    {   return data;    }

    public void setnext(Node n)
    {   next=n; }

    public Node getnext()
    {   return next;    }
}


public class Singly{
    private Node head;
    private Node tail;
    private Node x;
    private Node pre_x;

    public Singly()
    {   head=new Node();
        x=tail=head;
        pre_x=head;
```

```
   }

   public void move()
   {   pre_x=x;
       x=x.getnext();
   }
   public void append(int val) // appends a node at the end of the list
   {
       Node n=new Node(val);
       tail.setnext(n);
       tail=n;
       n.setnext(head);
       System.out.print("Appended Successfully..!\n");
   }//end of append

   public boolean search(int val)
   {   x=head.getnext(); pre_x=head; int loc=0;
       while(x!=head)
       {   if(x.getdata()==val)
           { System.out.println("Found at location "+ loc);
               return true;}
           else  { loc++; move();}
       }
       System.out.println("Number not found");
       return false;
    }// end of search


   public void delete(int val)
   {
       if(search(val))
       {
           pre_x.setnext(x.getnext());
           x=x.getnext();
           System.out.println("\nDeleted Successfully..!");
       }
   } // end of delete

   public void display()
   {   pre_x=head;
       x=pre_x.getnext();
       System.out.println("LinkedList: ");
       while(x!=head)
       {  System.out.println(x.getdata());
           move();
       }
   } // end of display

public void insert_after(int d1, int d2)// inserts d1 after d2
{
    if (search(d2))
    {
        Node nn=new Node(d1);
        nn.setnext(x.getnext());
        x.setnext(nn);
```

```
         System.out.println("Inserted Successfully..!!");
     }
 }// end of insert_after
}// End of class singly

import java.util.Scanner;

public class LinkedList_Demo {

    public static void main(String[] args) {

        Singly LList=new Singly();
        int opt; int val;
        Scanner input = new Scanner(System.in);
        do {
            System.out.println("0.Append ");
            System.out.println("1.Insert");
            System.out.println("2.Delete");
            System.out.println("3.Search");
            System.out.println("4.Display");
            System.out.println("5.Quit");
            System.out.print("Select an operation : ");
            opt = input.nextInt();

            switch (opt) {
                case 0:
                    System.out.print("Enter number to be appended: ");
                    val = input.nextInt();
                    LList.append(val);
                    break;
                case 1:
                    System.out.print("Enter number to be inserted: ");
                      val = input.nextInt();
                    System.out.print("Enter the number after which you want 
to insert: ");
                    int loc = input.nextInt();
                    LList.insert_after(val, loc);
                    break;
                case 2:
                    System.out.print("Enter number to be deleted: ");
                    val = input.nextInt();
                    LList.delete(val);
                    break;
                case 3:
                    System.out.print("Enter number to be searched: ");
                    val = input.nextInt();
                    LList.search(val);
                    break;
                case 4:
                    LList.display();
                    break;
                case 5:
                    System.exit(0);
                default:
                    System.out.println("Wrong choice\n");
            }//End of switch
        }//end do
                while(opt!=5);
    }//end of main
}
```

## Snapshot of output:



```
run:
0.Append
1.Insert
2.Delete
3.Search
4.Display
5.Quit
Select an operation : 0
Enter number to be appended: 1
Appended Successfully..!
0.Append
1.Insert
2.Delete
3.Search
4.Display
5.Quit
Select an operation : 0
Enter number to be appended: 2
Appended Successfully..!
0.Append
1.Insert
2.Delete
3.Search
4.Display
5.Quit
Select an operation : 1
Enter number to be inserted: 3
Enter the number after which you want to insert: 2
Found at location 2
Inserted Successfully..!!
0.Append
1.Insert
2.Delete
3.Search
4.Display
5.Quit
Select an operation : 2
Enter number to be deleted: 2
Found at location 2

Deleted Successfully..!
0.Append
1.Insert
2.Delete
3.Search
4.Display
5.Quit
Select an operation :
```

## Exercises:

Write a function to sort a pointer based one-way linked list of integers.

**Homework:**

Write a function to reverse a pointer based one-way linked list.

# Lab No 05
# Doubly Linked List

# Lab No. 5: Doubly Linked List

## Objective:

The objective of this lab is to implement doubly linked list.

## Scope:

A class of integer nodes containg two pointers will be specified. Using this structure, a two-way linked list will be created dynamically. All typical operations of a data structure that is insertion, deletion, searching etc. will be dicussed for this dynamic doubly list.

## Implementation:

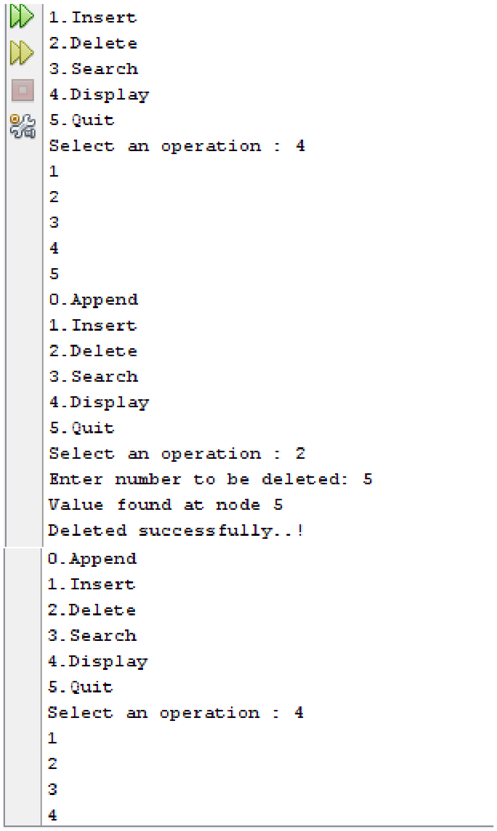
```
public class Node {
    private int  data;
    private Node next;
    private Node prev;

    public Node()//no arg constructor
    {
        data=-1;
        next=null;
        prev=null;
    }

    public Node(int d)//one arg constructor
    {
        data=d;
        next=null;
        prev=null;
    }

    public void setdata(int d)
    {   data=d;}

    int getdata()
    {   return data;    }

    public void setnext(Node n)
    {   next=n; }

    public void setprev(Node n)
    {   prev=n; }

    public Node getnext()
    {   return next;    }

    public Node getprev()
    {   return prev;    }
}
```

```
public class Doubly {
```

```
private Node head;
private Node x;
private Node tail;

public Doubly()//constructor
{   head=new Node();
    x=head;
    tail=head;
}
public void append(int val)
{
    Node n=new Node(val);
    x=tail;
    x.setnext(n);
    n.setprev(x);
    x=x.getnext();
    tail=n;
}//end of append

public boolean search(int val)
{
    x=head;  int loc=0;
    while(x!=null)
    {   if(x.getdata()==val)
        { System.out.print("Value found at node " + loc);
            return true;}
        else
        { x=x.getnext();
            loc++;}
    }
    System.out.print("Value not found...!!");
    return false;
 }//end of search

public void delete(int val)
{
    if(search(val))
    {
        Node p= x.getprev();
        p.setnext(x.getnext());
        if(x.getnext()!=null)
        {   x=x.getnext();
            x.setprev(p);
        }
        else {x=p;tail=x;}
        System.out.println("\nDeleted successfully..!");
    }
}//end of delete

 public void insert(int d1, int d2)//inserts d1 after d2
 {
     if(search(d2))
     {
         Node n1=new Node(d1);
         Node temp=x.getnext();
```

```
             n1.setnext(x.getnext());
             n1.setprev(temp.getprev());
             temp.setprev(n1);
             x.setnext(n1);
         }
     }//insert ends

      public void display()
    {
        x=head.getnext();
        while(x!=null)
        {  System.out.println(x.getdata());
            x=x.getnext();
        }
    }//display ends

}//class ends


public class Doubly_Demo {
public static void main(String[] args) {
        Doubly LList=new Doubly();
        int opt; int val;
        Scanner input = new Scanner(System.in);
        do {
            System.out.println("0.Append ");
            System.out.println("1.Insert");
            System.out.println("2.Delete");
            System.out.println("3.Search");
            System.out.println("4.Display");
            System.out.println("5.Quit");
            System.out.print("Select an operation : ");
            opt = input.nextInt();

            switch (opt) {
                case 0:
                    System.out.print("Enter number to be appended: ");
                    val = input.nextInt();
                    LList.append(val);
                    break;
                case 1:
                    System.out.print("Enter number to be inserted: ");
                      val = input.nextInt();
                    System.out.print("Enter the number after which you want 
to insert: ");
                    int loc = input.nextInt();
                    LList.insert(val, loc);
                    break;
                case 2:
                    System.out.print("Enter number to be deleted: ");
                    val = input.nextInt();
                    LList.delete(val);
                    break;
                case 3:
                    System.out.print("Enter number to be searched: ");
                    val = input.nextInt();
                    LList.search(val);
                    break;
                case 4:
                    LList.display();
                    break;
                case 5:
                    System.exit(0);
                default:
                    System.out.println("Wrong choice\n");
            }//End of switch
        }//end do
                while(opt!=5);
    }//main ends
}//class ends
```

Snapshot of output:



```
1.Insert
2.Delete
3.Search
4.Display
5.Quit
Select an operation : 4
1
2
3
4
5
0.Append
1.Insert
2.Delete
3.Search
4.Display
5.Quit
Select an operation : 2
Enter number to be deleted: 5
Value found at node 5
Deleted successfully..!
0.Append
1.Insert
2.Delete
3.Search
4.Display
5.Quit
Select an operation : 4
1
2
3
4
```

## Exercises:

1. Write a function to delete all the nodes having certain value from a doubly linked list.
2. Write a function to sort a doubly linked list of integers using bubble sort.

**Homework:**

Create two sorted linked lists L1 and L2 and merge them to form a sorted list L3.

# Lab No 06
# One Way Circular Linked List

# Lab No 06: One Way Circular Linked List

## Objective:

The objective of this lab is to implement one-way circular linked list.

## Scope:

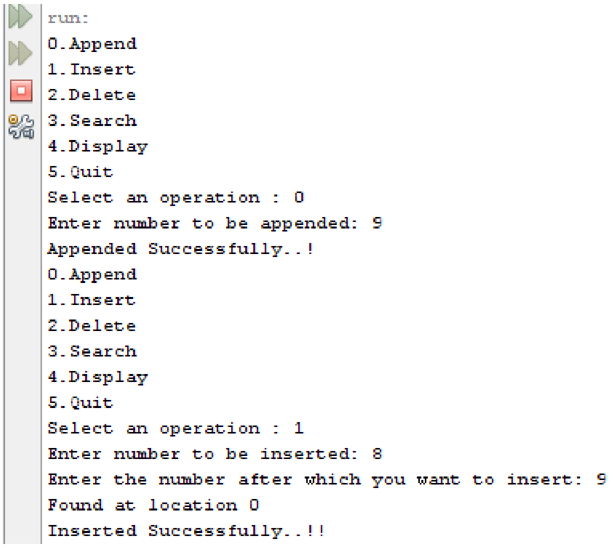
A circular linked list will be created dynamically. All typical operations of a data structure that is insertion, deletion, searching  will be dicussed and implemented for this circular pointer-based list.

## Implementation:

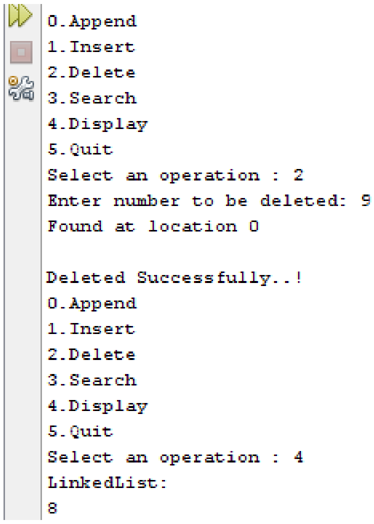
```
public class oneway_circular {
    private Node head;
    private Node tail;
    private Node x;
    private Node pre_x;

    public oneway_circular()
    {   head=new Node();
        x=tail=head;
        pre_x=head;
    }

    public void move()
    {   pre_x=x;
        x=x.getnext();
    }
    public void append(int val) // appends a node at the end of the list
    {
        Node n=new Node(val);
        tail.setnext(n);
        tail=n;
        n.setnext(head);
        System.out.print("Appended Successfully..!\n");
    }//end of append

    public boolean search(int val)
    {   x=head.getnext(); pre_x=head; int loc=0;
        while(x!=head)
        {   if(x.getdata()==val)
            { System.out.println("Found at location "+ loc);
                return true;}
            else  { loc++; move();}
        }
        System.out.println("Number not found");
        return false;
     }// end of search

    public void delete(int val)
    {
        if(search(val))
        {
            pre_x.setnext(x.getnext());
            x=x.getnext();
            System.out.println("\nDeleted Successfully..!");
        }
    }// end of delete

    public void display()
    {   pre_x=head;
        x=pre_x.getnext();
        System.out.println("LinkedList: ");
        while(x!=head)
        {  System.out.println(x.getdata());
            move();
        }
    } // end of display

    public void insert_after(int d1, int d2)// inserts d1 after d2
      {
     if (search(d2))
     {
         Node nn=new Node(d1);
         nn.setnext(x.getnext());
         x.setnext(nn);
         System.out.println("Inserted Successfully..!!");
     }
       }// end of insert_after
      }


import java.util.Scanner;

public class LinkedList_Demo {

    public static void main(String[] args) {

        oneway_circular LList=new oneway_circular();
        int opt; int val;
        Scanner input = new Scanner(System.in);
        do {
            System.out.println("0.Append ");
            System.out.println("1.Insert");
            System.out.println("2.Delete");
            System.out.println("3.Search");
            System.out.println("4.Display");
            System.out.println("5.Quit");
            System.out.print("Select an operation : ");
            opt = input.nextInt();

            switch (opt) {
                case 0:
                    System.out.print("Enter number to be appended: ");
                    val = input.nextInt();
                    LList.append(val);
                    break;
                case 1:
                    System.out.print("Enter number to be inserted: ");
```

```
           val = input.nextInt();
          System.out.print("Enter the number after which you want
to insert: ");
          int loc = input.nextInt();
          LList.insert_after(val, loc);
          break;
        case 2:
          System.out.print("Enter number to be deleted: ");
          val = input.nextInt();
          LList.delete(val);
          break;
        case 3:
          System.out.print("Enter number to be searched: ");
          val = input.nextInt();
          LList.search(val);
          break;
        case 4:
          LList.display();
          break;
        case 5:
          System.exit(0);
        default:
          System.out.println("Wrong choice\n");
      }//End of switch
    }//end do
        while(opt!=5);
  }//end of main
}
```

**Snapshot of Output:**



```
run:
0.Append
1.Insert
2.Delete
3.Search
4.Display
5.Quit
Select an operation : 0
Enter number to be appended: 9
Appended Successfully..!
0.Append
1.Insert
2.Delete
3.Search
4.Display
5.Quit
Select an operation : 1
Enter number to be inserted: 8
Enter the number after which you want to insert: 9
Found at location 0
Inserted Successfully..!!
```



```
0.Append
1.Insert
2.Delete
3.Search
4.Display
5.Quit
Select an operation : 2
Enter number to be deleted: 9
Found at location 0

Deleted Successfully..!
0.Append
1.Insert
2.Delete
3.Search
4.Display
5.Quit
Select an operation : 4
LinkedList:
8
```

**<u>Exercises:</u>**

Write a function to find if a linked list is circular or not.
Write a function to remove odd enteries from a circular linked list of integers.

**<u>HomeWork:</u>**

Split a circular linked list into two equal sized non-circular linked lists.

# Lab No 07
# Two Way circular Linked List

# Lab No 07: Two Way Circular Linked List

## Objective:

The objective of this lab is to implement two-way circular linked list.

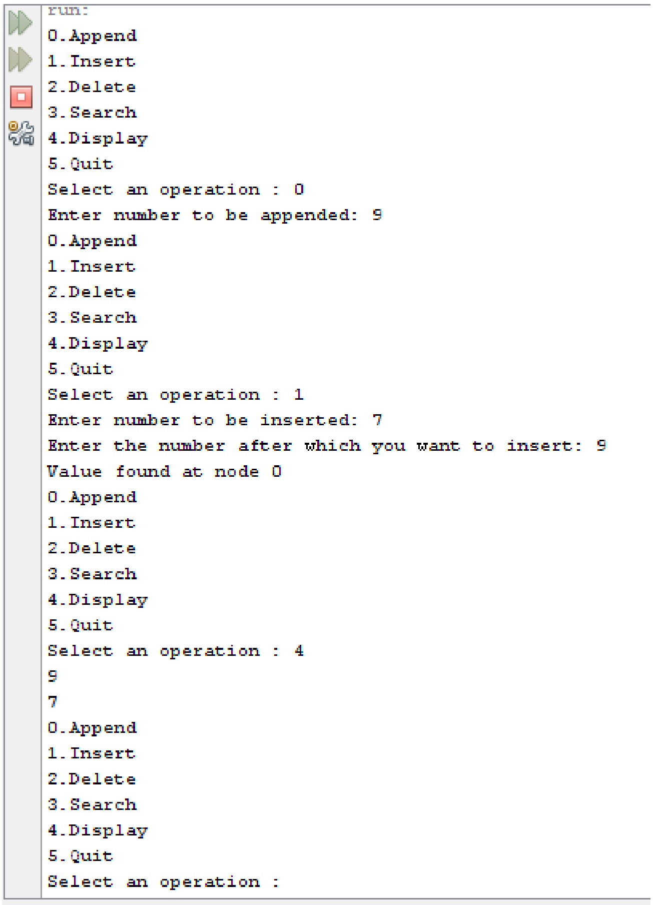
## Scope:

A two-way circular linked list will be created dynamically. All typical operations of a data structure that is insertion, deletion, searching and appending will be dicussed for this circular doubly list.

## Implementation:

```
public class twoway_circular {
    private Node head;
    private Node x;

    public twoway_circular()//constructor
    {   head=new Node();
        x=head;
        head.setnext(head);
        head.setprev(head);
    }
    public void append(int val)
    {
        Node n=new Node(val);
        x=head.getprev();
        x.setnext(n);
        n.setprev(x);
        n.setnext(head);
        head.setprev(n);
    }//end of append

    public boolean search(int val)
    {
        x=head.getnext();  int loc=0;
        while(x!=head)
        {   if(x.getdata()==val)
            { System.out.println("Value found at node " + loc);
                return true;}
            else
            { x=x.getnext();
                loc++;}
        }
        System.out.print("Value not found...!!");
        return false;
     }//end of search

    public void delete(int val)
    {
        if(search(val))
        {
            Node p= x.getprev();
            p.setnext(x.getnext());
            x=x.getnext();
            x.setprev(p);
            System.out.println("\nDeleted successfully..!");
        }
    }//end of delete

     public void insert(int d1, int d2)//inserts d1 after d2
     {
         if(search(d2))
         {
             Node n1=new Node(d1);
             Node temp=x.getnext();
             n1.setnext(x.getnext());
             n1.setprev(temp.getprev());
             temp.setprev(n1);
             x.setnext(n1);
         }
     }//insert ends

      public void display()
    {
        x=head.getnext();
        while(x!=head)
        {  System.out.println(x.getdata());
            x=x.getnext();
        }
    }//display ends
}

public class LinkedList_Demo {

   public static void main(String[] args) {
        twoway_circular LList=new twoway_circular();
        int opt; int val;
        Scanner input = new Scanner(System.in);
        do {
            System.out.println("0.Append ");
            System.out.println("1.Insert");
            System.out.println("2.Delete");
            System.out.println("3.Search");
            System.out.println("4.Display");
            System.out.println("5.Quit");
            System.out.print("Select an operation : ");
            opt = input.nextInt();

            switch (opt) {
                case 0:
                    System.out.print("Enter number to be appended: ");
                    val = input.nextInt();
                    LList.append(val);
                    break;
                case 1:
                    System.out.print("Enter number to be inserted: ");
                      val = input.nextInt();
                    System.out.print("Enter the number after which you want
to insert: ");
                    int loc = input.nextInt();
                    LList.insert(val, loc);
                    break;
                case 2:
                    System.out.print("Enter number to be deleted: ");
                    val = input.nextInt();
                    LList.delete(val);
                    break;
                case 3:
                    System.out.print("Enter number to be searched: ");
                    val = input.nextInt();
                    LList.search(val);
                    break;
                case 4:
                    LList.display();
                    break;
                case 5:
                    System.exit(0);
                default:
                    System.out.println("Wrong choice\n");
            }//End of switch
        }//end do
                while(opt!=5);
    }//main ends
}//class ends
```

**Snapshot of output:**



```
run:
0.Append
1.Insert
2.Delete
3.Search
4.Display
5.Quit
Select an operation : 0
Enter number to be appended: 9
0.Append
1.Insert
2.Delete
3.Search
4.Display
5.Quit
Select an operation : 1
Enter number to be inserted: 7
Enter the number after which you want to insert: 9
Value found at node 0
0.Append
1.Insert
2.Delete
3.Search
4.Display
5.Quit
Select an operation : 4
9
7
0.Append
1.Insert
2.Delete
3.Search
4.Display
5.Quit
Select an operation :
```

**Exercises:**

1. Write a function to remove even enteries from a circular doubly linked list of integers.
2. Create two linked lists L1 and L2 and merge them to form a single linked list.

**Homework:**

Write a function to delete a specified number of consecutive nodes from a linked list.