Khalid Mahmood

LINKED DATA STRUCTURE

Data structure in which the memory occupied only at the time when the data is remains into the memory. As well as the data is remove then the memory is also free. It means that the size of data structure increase as the data comes and the size reduce as the as the data is deleted to the memory. It is totally unlike the linear data structure in which the size of the memory occupied remain fix and not changeable.

SINGLY LINKED LIST

A singly or simple linked list is a data structure, each node of which has one or more data item fields (Data) but only a single link field (LINK).

BENEFITS:

1. Easy to insert & delete at any time anywhere.
2. Efficient memory use.
3. Fast operations.

Whenever we create a single link list it must have the following characteristics:

1. Method to create node.
2. Method to find which node is full and which is free.
3. Find the proper location of node.
4. A method to delete node.

NODE:

Node is basically a chunk of memory, caries and address. When a set of data elements to be used by and application are represented using a linked list, each data element is represented by a node. A singly linked list only a single link field is use to point to node which represents its neighboring element in the list.

The first node in the linked list is called start or head node and the last node of the linked list is called null node.

```
// Program to Implement Singly Linked List
importjava.util.*;
/*  Class Node  */
class Node
{  int data;   Node link;
   /*  Constructor  */
Node(intdd)
   {link = null;       data = dd;    }
   /*  Constructor  */
Node(intdd, Node next)
 {       link = next;       data = dd;  }          }
 /*  ClasslinkedList */
classlinkedList
{    Node start;     Node end ;int size ;
    /*  Constructor  */
publicLinkedList()
    {        start = null;    end = null;    size = 0;
}
/*  Function to check if list is empty  */
publicbooleanisEmpty()
    {        return start == null;    }
   /*  Function to insert an element at
begining  */
public void insertAtStart(intval)
    {       Node nptr = new Node(val);
size++;
if(isEmpty())
        {          start = nptr;          end = start;
}


else
        {  nptr.link= start;          start = nptr;
}   }
/*  Function to insert an element at end  */
public void insertAtEnd(intval)
    {       Node nptr = new Node(val);
size++ ;
if(isEmpty())
{  start = nptr;        end = start;       }
else
        {          end.link=nptr;          end = nptr;
}   }
/*  Function to insert an element at position
*/
public void insertAtPos(intval , int x)
{  Nodenptr = start;
while(nptr.link!=null)
{ if (nptr.data==x )
break;
nptr=nptr.link;}
```

```
nptr.link= new Node(val,nptr.link);
size++;   }

/*  Function to delete an element at position
*/

public void deleteAtPos(int x)

  {    int done=0;       if(start==null)

          System.out.println("empty");

elseif(start.data==x)

start=start.link;

else




for (Node p=start; p.link!=null; p=p.link){

if (p.link.data==x)

{ done++;

System.out.println("data inserted");

p.link=p.link.link;

break;}}}

 /*  Function to display elements  */

public void display()

{  System.out.print("\nSingly Linked List
display = ");

for(Node p=start; p!=null; p=p.link)

System.out.print("\t"+p.data);

System.out.print("\t total  nodes are "+size);

}}

*  AppClass SinglyLinkedList  */

public class SinglyLinkedList

{   public static void main(String[] args)

{ Scanner in = new Scanner(System.in);

    /* Creating new node for linkedList */

linkedList list = new linkedList();

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

int choice;

      /*  Perform list operations  */

do

{ System.out.println("\nSingly 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. Display list");

System.out.println("0. To exit");

choice = in.nextInt();

          switch (choice)

          {case 0 :

System.out.println(" porgram is ending ");
break;

case 1 :

System.out.println("Enter element
toinsert");

list.insertAtStart(in.nextInt());

break;

case 2 :

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

list.insertAtEnd(in.nextInt() );

break;

case 3 :

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

intnum = in.nextInt();

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

intpos = in.nextInt() ;

list.insertAtPos(num, pos);

break;

case 4 :

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

int p = in.nextInt() ;

list deleteAtPos(p);

break;

case 5 :

list.display();

break;

          case 6 :

System.out.println("end program ");

          break;


default :

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

break;

} }while(choice!=0);

                        }

}
```