Axis assignment.2(Ashish Gupta)

Linked List :: Linked list is a part of Data structure in which data stored randomly but every node having 8 bit of memory 4 bits for data and 4 bits of address , every data node are capable to store sum data and address node capable to store address of another node.

Basically Link list are of three types::

1. Singly Linked List.
2. Doubly Linked List.
3. Circular Linked List.
4. Singly Linked list = Singly link is a part of data structure which is used to store collection of node .

* It contains sequence of node.
* A node having data and references to the next node.
* First node is an head node.
* Last node having data and points to null.

Representation of Singly linked list.

**package** list;

**public** **class** SingleLinkList {

**private** ListNode head;

**private** **static** **class** ListNode{

**private** **int** data;

**private** ListNode next;

**public** **void** ListNode(**int** data) {

**this**.data = data;

**this**.next = **null**;

}

}

}

* Insert data in singly Linked list

Code::

**package** list;

**public** **class** SingleLinkList {

**private** ListNode head;

**private** **static** **class** ListNode{

**private** **int** data;

**private** ListNode next;

**public** ListNode(**int** data) {

**this**.data = data;

**this**.next = **null**;

}

}

**public** **void** display() {

ListNode current = head;

**while**(current != **null**) {

System.***out***.print(current.data + "-->");

current = current.next;

}

System.***out***.println("null");

}

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

SingleLinkList obj = **new** SingleLinkList();

obj.head = **new** ListNode(20);

ListNode second = **new** ListNode(1);

ListNode third = **new** ListNode(5);

ListNode forth = **new** ListNode(6);

//connecting node

obj.head.next = second;

second.next = third;

third.next = forth;

obj.display();

}

}

Out put is:: 20-->1-->5-->6-->null

* Find the length of singly linked list.
* **package** list;
* **public** **class** SingleLinkList {
* **private** ListNode head;
* **private** **static** **class** ListNode{
* **private** **int** data;
* **private** ListNode next;


* **public** ListNode(**int** data) {
* **this**.data = data;
* **this**.next = **null**;
* }
* }
* **public** **void** display() {
* ListNode current = head;
* **while**(current != **null**) {
* System.***out***.print(current.data + "-->");
* current = current.next;

* }
* System.***out***.println("null");

* }
* **public** **int** length() {
* **if** (head == **null**) {
* **return** 0;
* }
* **int** count = 0;
* ListNode current = head;
* **while**(current !=**null**) {
* count++;
* current = current.next;
* }
* **return** count;
* }
* **public** **static** **void** main(String args[]) {
* SingleLinkList obj = **new** SingleLinkList();
* obj.head = **new** ListNode(20);
* ListNode second = **new** ListNode(1);
* ListNode third = **new** ListNode(5);
* ListNode forth = **new** ListNode(6);
* //connecting node
* obj.head.next = second;
* second.next = third;
* third.next = forth;
* obj.display();
* System.***out***.println("Length of the node is- " + obj.length());
* }
* }

Out put is:: 20-->1-->5-->6-->null

Length of the node is- 4

# Insert data in singly linked list is in head.

**package** list;

**public** **class** SingleLinkList {

**private** ListNode head;

**private** **static** **class** ListNode{

**private** **int** data;

**private** ListNode next;

**public** ListNode(**int** data) {

**this**.data = data;

**this**.next = **null**;

}

}

**public** **void** display() {

ListNode current = head;

**while**(current != **null**) {

System.***out***.print(current.data + "-->");

current = current.next;

}

System.***out***.println("null");

}

**public** **int** length() {

**if** (head == **null**) {

**return** 0;

}

**int** count = 0;

ListNode current = head;

**while**(current !=**null**) {

count++;

current = current.next;

}

**return** count;

}

**public** **void** insertFirst(**int** values) {

ListNode obj = **new** ListNode(values);

obj.next = head;

head = obj;

}

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

SingleLinkList obj = **new** SingleLinkList();

obj.insertFirst(7);

obj.insertFirst(5);;

obj.insertFirst(4);

obj.display();

}

}

Out put is:: 4-->5-->7-->null

Insert elements in last node::

**package** list;

**public** **class** SingleLinkList {

**private** ListNode head;

**private** **static** **class** ListNode{

**private** **int** data;

**private** ListNode next;

**public** ListNode(**int** data) {

**this**.data = data;

**this**.next = **null**;

}

}

**public** **void** display() {

ListNode current = head;

**while**(current != **null**) {

System.***out***.print(current.data + "-->");

current = current.next;

}

System.***out***.println("null");

}

**public** **int** length() {

**if** (head == **null**) {

**return** 0;

}

**int** count = 0;

ListNode current = head;

**while**(current !=**null**) {

count++;

current = current.next;

}

**return** count;

}

**public** **void** insertFirst(**int** values) {

ListNode obj = **new** ListNode(values);

obj.next = head;

head = obj;

}

**public** **void** insertLast(**int** value) {

ListNode obj = **new** ListNode(value);

**if**(head== **null**) {

head = obj;

**return**;

}

ListNode current =head;

**while**(**null** != current.next) {

current = current.next;

}

current.next = obj;

}

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

SingleLinkList obj = **new** SingleLinkList();

obj.insertLast(7);

obj.insertLast(5);;

obj.insertLast(4);

obj.display();

}

}

Out put is::

7-->5-->4-->null

# Bubble short

Bubble sort is **a simple sorting algorithm that compares adjacent elements of an array and swaps them if the element on the right is smaller than the one on the left**. It is an in-place sorting algorithm i.e. no extra space is needed for this sort, the array itself is modified.

1 Problem= -5,30,0,11,-2.

**package** shorting;

**public** **class** Boubel\_short {

**public** **static** **void** printArray(**int** arr[]) {

**for**(**int** i = 0 ;i<arr.length;i++) {

System.***out***.print(arr[i]+ " ");

}

}

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

**int** arr[] = {-5,30,0,11,-2};

**for**(**int** i = 0; i<arr.length -1; i++) {

**for**(**int** j = 0; j<arr.length-i-1;j++) {

**if**(arr[j]>arr[j+1]) {

**int** temp = arr[j];

arr[j] = arr[j+1];

arr[j+1]= temp;

}

}

}

*printArray*(arr);

}

}

Out put is::

-5 -2 0 11 30

2 problem= -11,0,20,11,-5

**package** shorting;

**public** **class** Boubel\_short {

**public** **static** **void** printArray(**int** arr[]) {

**for**(**int** i = 0 ;i<arr.length;i++) {

System.***out***.print(arr[i]+ " ");

}

}

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

**int** arr[] = {-11,0,20,11,-5};

**for**(**int** i = 0; i<arr.length -1; i++) {

**for**(**int** j = 0; j<arr.length-i-1;j++) {

**if**(arr[j]>arr[j+1]) {

**int** temp = arr[j];

arr[j] = arr[j+1];

arr[j+1]= temp;

}

}

}

*printArray*(arr);

}

}

Out put is:: -11 -5 0 11 20