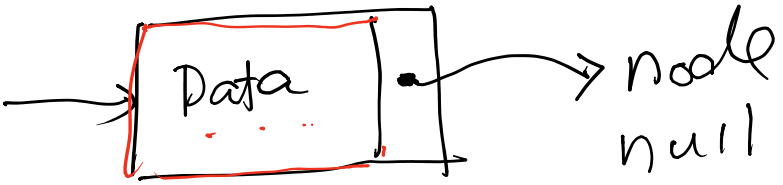
06/18/2018
CMSC132 Lecture 10
Linked Lists

(* review *)

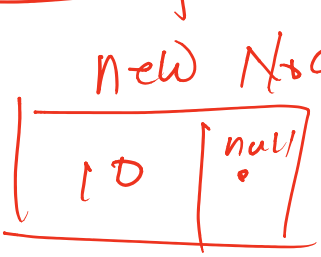
Linked list: a chain of nodes
Node: carries some data and has a reference to a next node

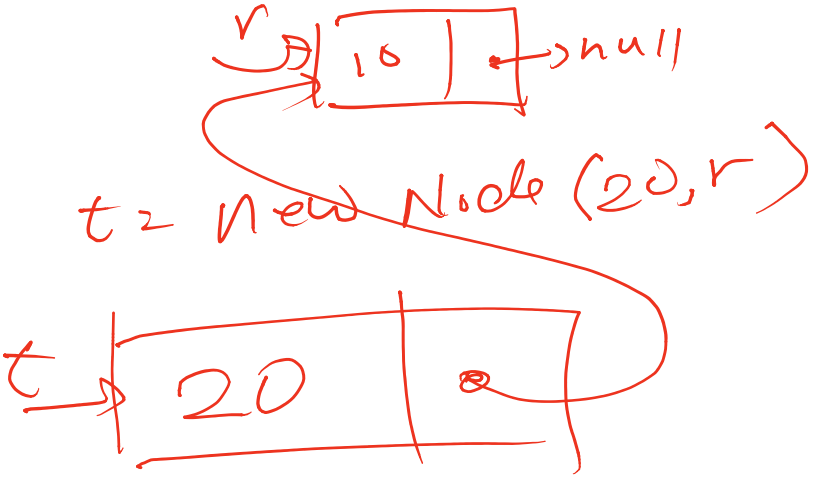



```
Class Node<E>{
    E data;
    Node<E> next;

    Public Node(){
        data = null;
        Next = null;
        }
    Public Node(E data){
        This.data = data;
        Next = null;
        }
    /*
        Take r as a tail
    */
    Public Node(E data, Node r){
        This.data = data;
        Next = r;
        }


}
```





Linked Lists:

Each node has one next reference: next node or null

Child cannot go back to parent. There is only one reference from parent to child. No reference from child to the parent

A--->B--->C--->D

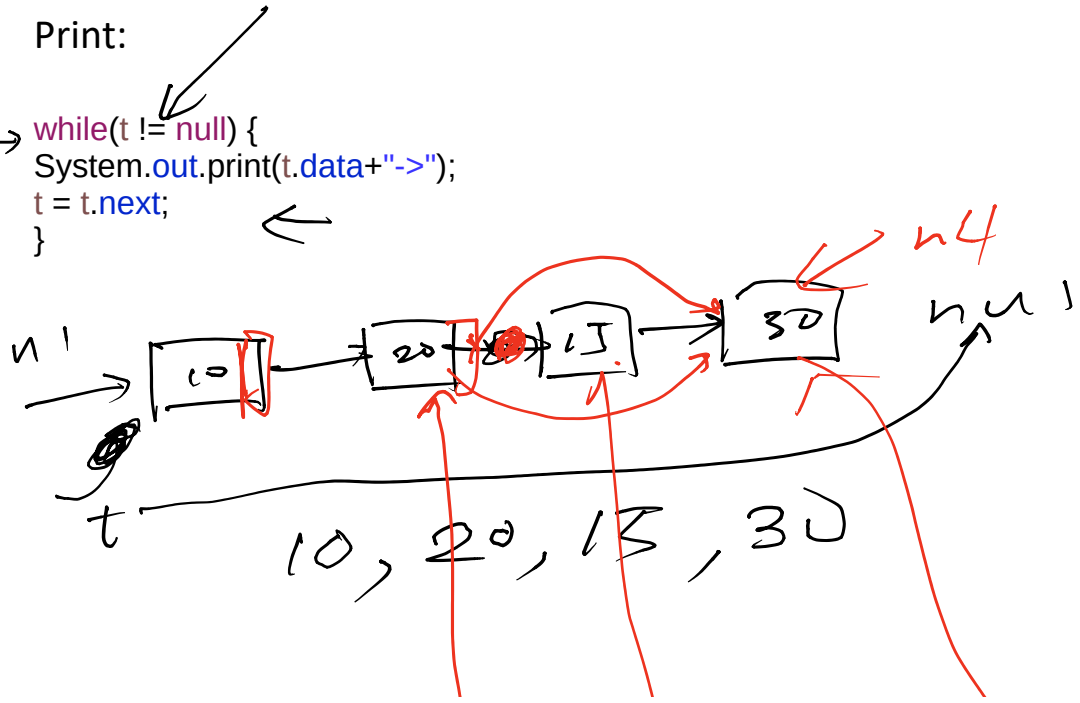
A does not know anything about C or D.
A only knows B.

You cannot jump from A to C.

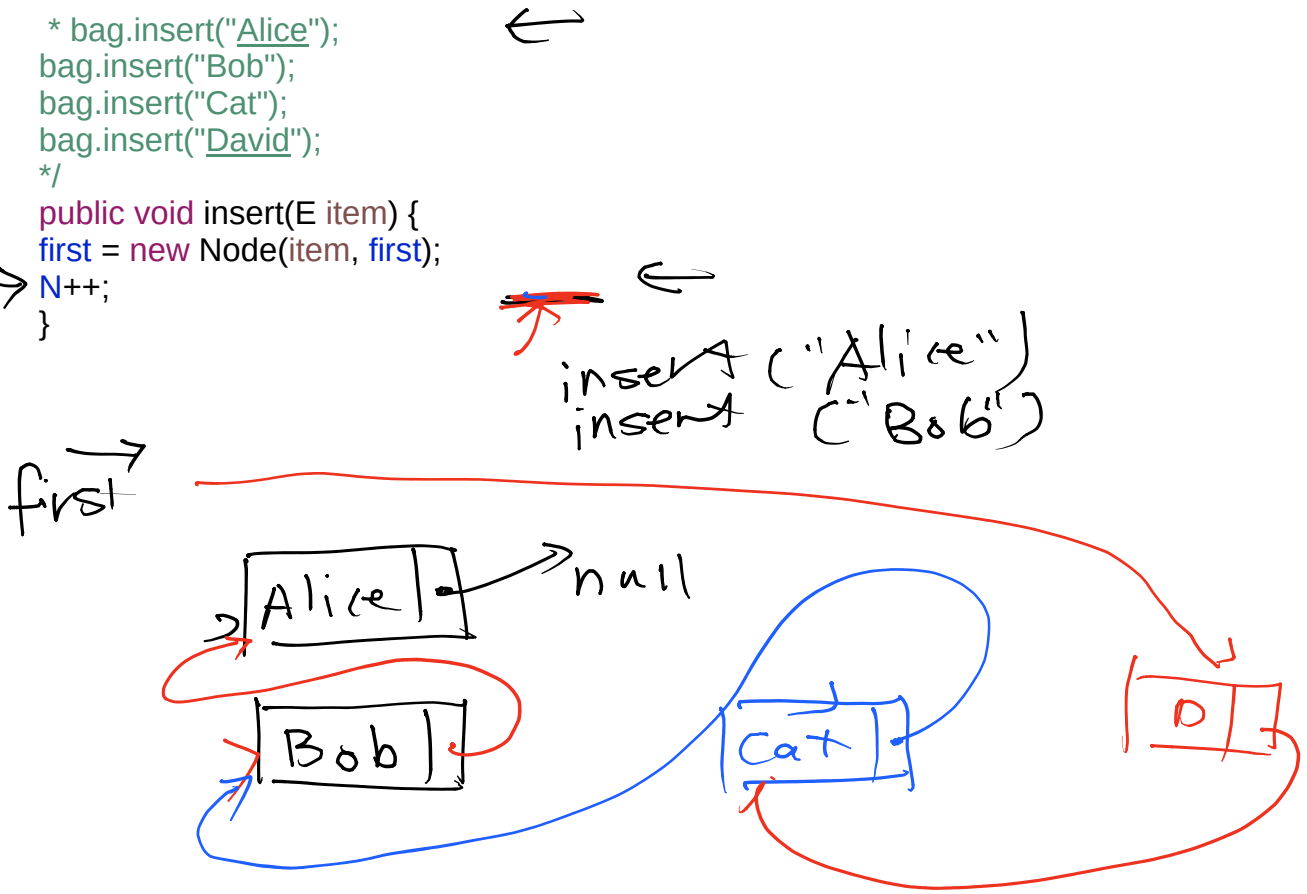
To access D, you have to walk through the list nodes one at a time
A--->B--->C--->D
ARRAY: |A|B|C|D| direct random access D with the index, a[3]

Print:

```
while(t != null) {
System.out.print(t.data+"->");
t = t.next;
}
```




```
 * bag.insert("Alice");
bag.insert("Bob");
bag.insert("Cat");
bag.insert("David");
*/
public void insert(E item) {
first = new Node(item, first);
N++;
}
```




```
LinkedBag<Integer> bag = new LinkedBag<>();
bag.insertIntoTail(10);
bag.insertIntoTail(20);
bag.insert(30);
bag.insertIntoTail(40);
```

A. 10,20,30,40
B. 20,10,30,40
C. 30,10,20,40
D. 40,30,20,10

Remove
A--->B--->C--->E-->D--->F

To delete D, walk through the list, stop at E, the parent of D
Cur.next = cur.next.next;

```
if(first.data.equals(item)) {
first = first.next;
}
Node parent = first;
Node cur = parent.next;
while(cur != null) {
if(cur.equals(item)) {
parent.next = cur.next;
return true;
}
parent = cur;
cur = cur.next;
}
return false;
```