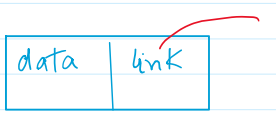
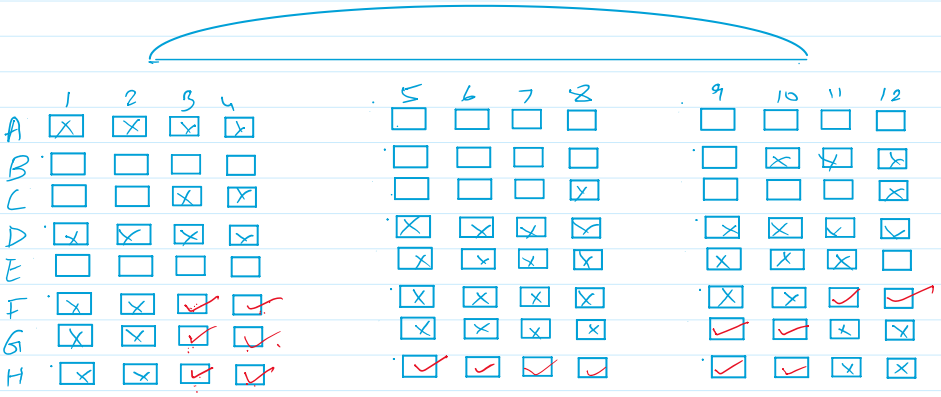
Linked List

13 November 2025 10:06

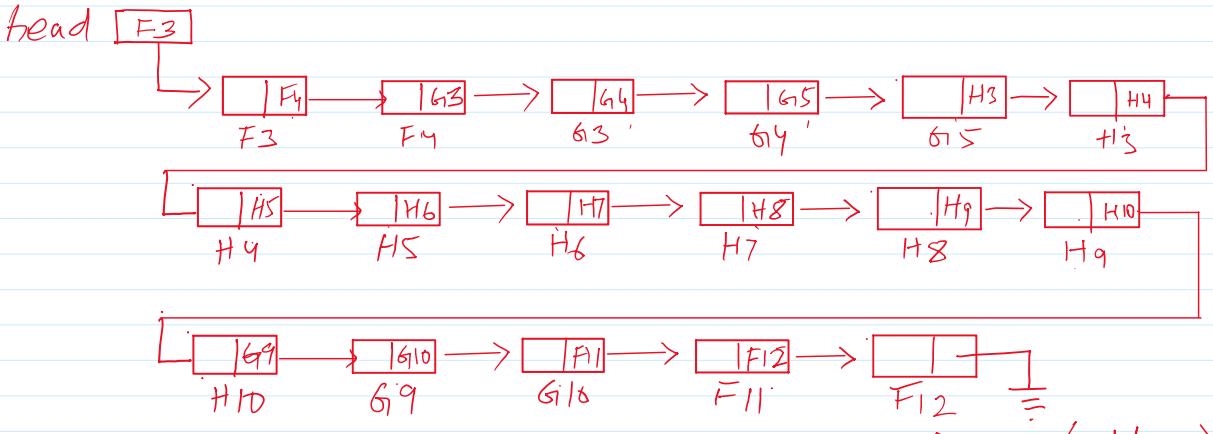
Linked List in Java is a linear data structure, where elements are stored in nodes, and each node contains data as well as a reference(link) to the next node in the sequence. Unlike arrays, linked list do not store elements in contiguous locations. Allowing easy insertion and deletion of elements without shifting other elements.



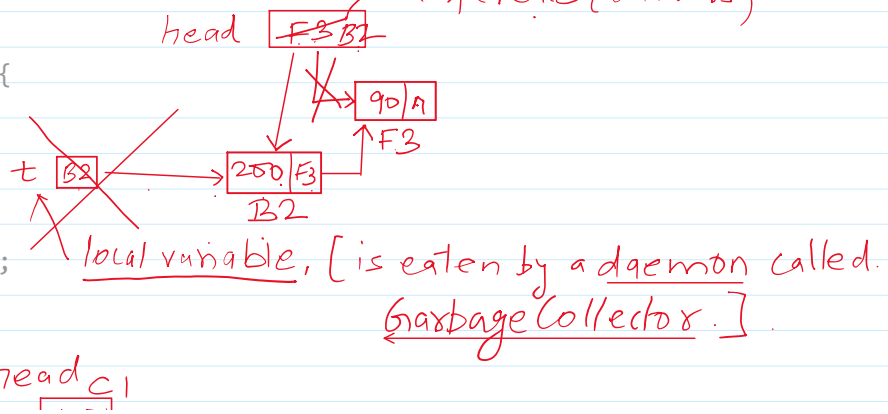
Node (used in singly linked list, which always points to the next node).



F3, F4, G3, G4, H3, H4, H5, H6, H7, H8, H9, H10, G9, G10, F11, F12

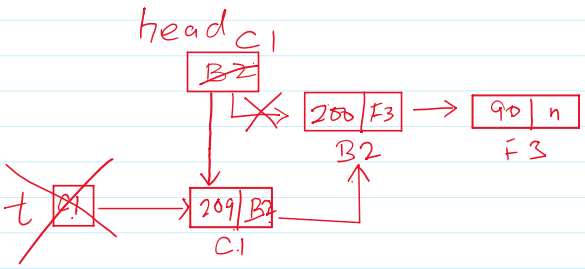



```java
// add the node at the head.  209
    public void addAtHead(int data){
        if(head == null){
            head = new Node(data);
        }
        else{
            //List contains a node
            Node t = new Node(data);
            t.next = head;
            head = t;
        }
    }
```



head c1

```
            head = t;
        }
    }
```




```
package MyLinkedList;
public class Node {
    int data;
    Node next;
    public Node(int data) {
        this.data = data;
        this.next = null;
    }
}
```

```
package MyLinkedList;
public class LinkedList {
    Node head;
    public LinkedList() {
        head = null;
    }
    //append the node at the end
    public void add(int data){
        //write the code here
    }
    // add the node at the head.
    public void addAtHead(int data){
        if(head == null){
            head = new Node(data);
        }
        else{
            //List contains a node
            Node t = new Node(data);
            t.next = head;
            head = t;
        }
    }
    public void printList(){
        Node t = head;
        if(t == null){
            System.out.println("List is empty!");
        }
        else{
            for(;t!=null; t = t.next){
                System.out.print("==>"+t.data);
            }
            System.out.println();
        }
    }
}
```

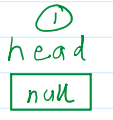
```
package MyLinkedList;
```

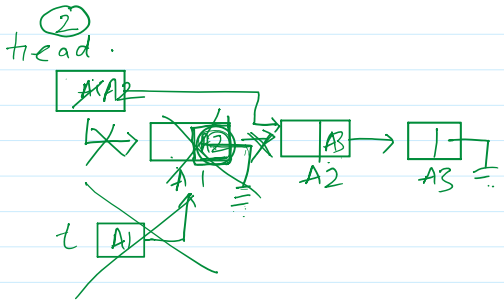
```java
import java.util.Scanner;
public class TestLinkedList {
    public static void main(String[] args) {
        LinkedList ll = new LinkedList();
        Scanner sc = new Scanner(System.in);

        while (true) {
            System.out.println("Press 1 to add data on head.");
            System.out.println("Press 2 to display the list");
            System.out.println("Press 0 to exit");
            System.out.println("Enter your choice(0-2): ");
            switch(sc.nextInt()){
                case 1:
                    System.out.println("Enter data to store on head of the list: ");
                    ll.addAtHead(sc.nextInt());
                    break;
                case 2:
                    ll.printList();
                    break;
                case 3:
                    System.out.println("Enter data to store at end of the list: ");
                    ll.addAtHead(sc.nextInt());
                    break;
                case 0:
                    sc.close();
                    System.out.println("Good Bye");
                    return; //exit from main.
                default:
                    System.out.println("Wrong option selected!");
            }
        }

    }
}
```

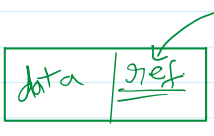
deleteNodeAtHead() {

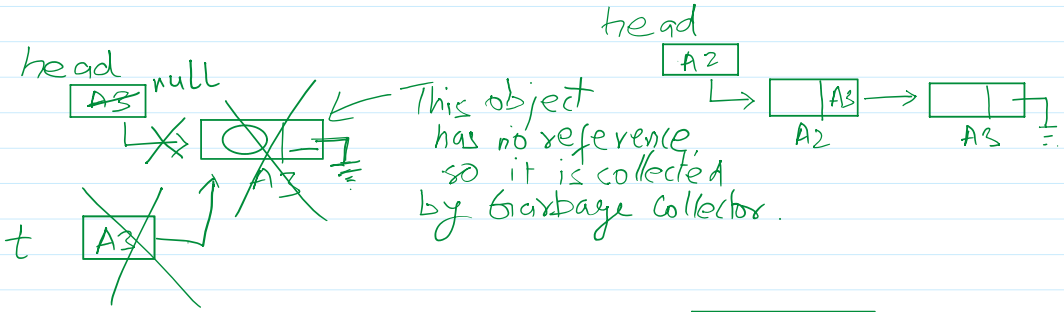
if head == null
    print "list is empty", exit.
else
→ Node t = head.
→ head = head.next.
→ t.next = null
    print "\n Deleted element" + t.data.

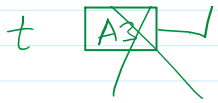
}

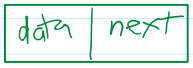




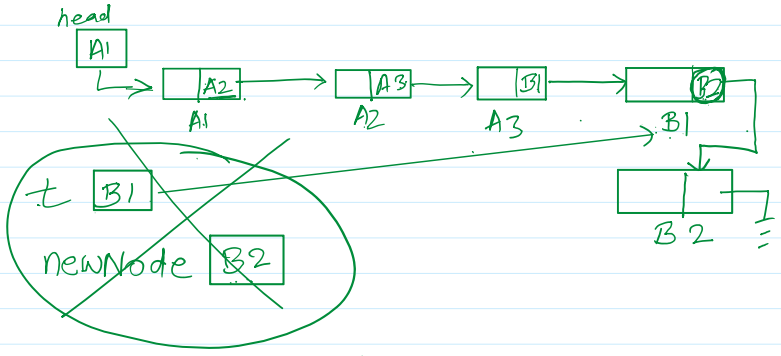




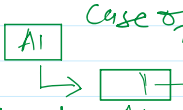




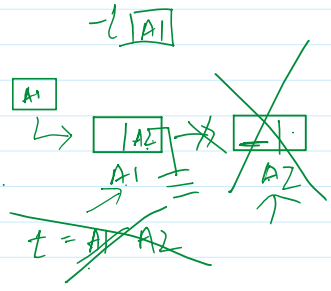

```java
public void addNodeAtEnd(int data){
    if(head == null){
        head = new Node(data);
    }
    else{
        Node newNode = new Node(data);
        Node t = head;
        while(t.next != null){
            t = t.next;
        }
        t.next = newNode;
    }
}
```

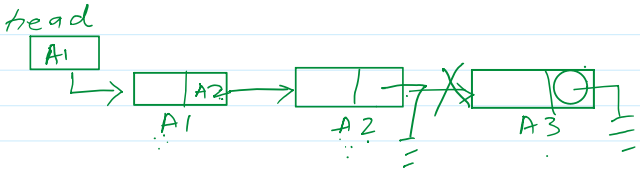


Case of 1 node.




```
Node t = head.
if (t.next == null){ //case of 1 node
    head = null;
}
else { // case of more than 1 node
while (t.next.next != null){
    t = t.next;
}
print "deleted node :-" + t.next.data.
t.next = null;
}
```

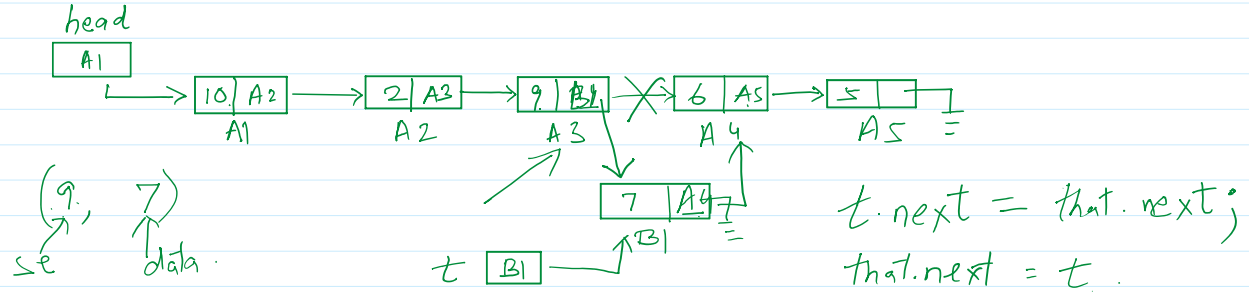


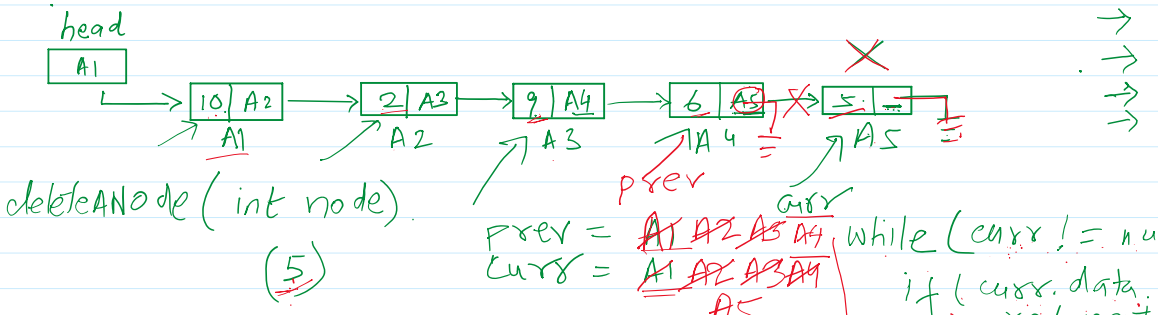


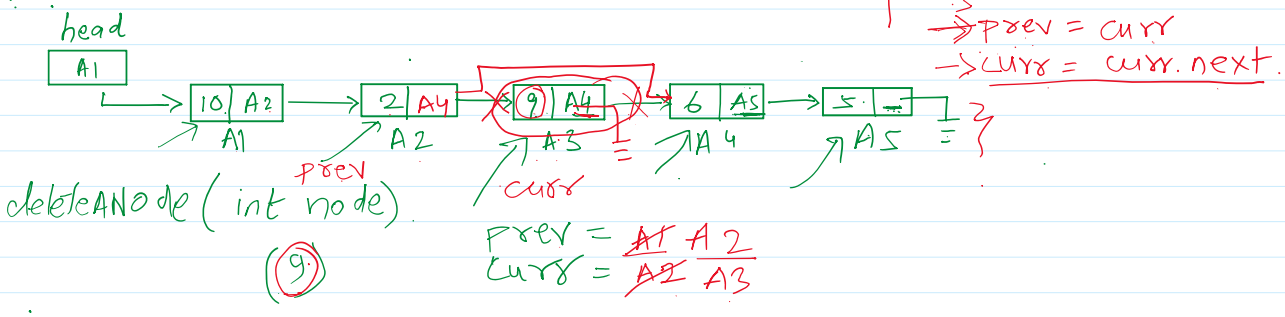
t = ~~A1~~ A2

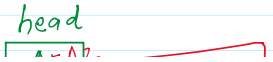
```java
public void deleteAtStart(){
        if(head == null){
            System.out.println("List is empty");
        }
        else{
            Node t = head;
            head = head.next;
            t.next = null; //to disconnect the node from the list.
            System.out.println("Deleted node: " + t.data);
        }


public void addNodeAtEnd(int data){
        if(head == null){
            head = new Node(data);
        }
        else{
```

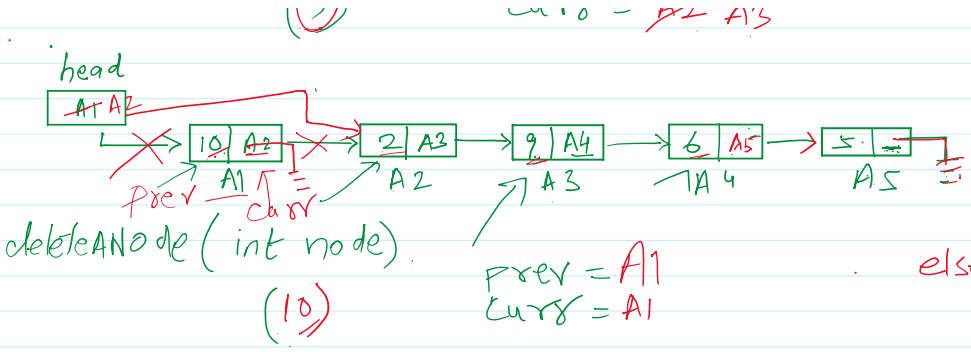
```
            Node newNode = new Node(data);
            Node t = head;
            while(t.next != null){
                t = t.next;
            }
            t.next = newNode;
        }
    }
    public void deleteAtEnd(){
        if(head == null){
            System.out.println("List is empty!");
        }
        else{
            Node t = head;
            if(t.next == null){
                head = null;
                System.out.println("Deleted node: " + t.data);
                return;
            }
            while(t.next.next != null){
                t = t.next;
            }
            System.out.println("Deleted node: "+ t.next.data);
            t.next = null;
        }
    }
```












```
if(head.data == data){
    head = head.next
    head.next = null;
}
else{
    while(curr != null){
        if(curr.data == data){
            prev.next = curr.next;
            curr.next = null;
            return; //exit the method.
        }
        prev = curr
        curr = curr.next.
    }
}
```

```java
public void insertNodeAfterANode(int searchNode, int data){
        if(head == null){
            System.out.println("List is empty!");
        }
        else{
            Node t = head;
            while(t != null){
                if(t.data == searchNode){
                    //we need to perform insertion
                    Node nn = new Node(data);
                    nn.next = t.next;
                    t.next = nn;
                    return; //exit the method
                }
                t = t.next;
            }
        }
    }
    /**
     * This method will delete a node passed as a parameter
     * @param node is a search node
     */
    public void deleteANode(int node){
        if(head == null){
            System.out.println("List is empty.");
        }
        else if(head.data == node){
            //case of first node
            System.out.println("Node deleted: " + node);
            Node t = head;
            head = head.next;
            t.next = null; //detaching the node from the list
        }
        else{
            //case after the first node
            Node curr = head.next, prev = head;
```

```
            while(curr!= null){
                if(curr.data == node){
                    System.out.println("Node deleted: " + node);
                    prev.next = curr.next;
                    curr.next = null;
                    return;
                }
                prev = curr;
                curr = curr.next;
            }
        }
    }
```