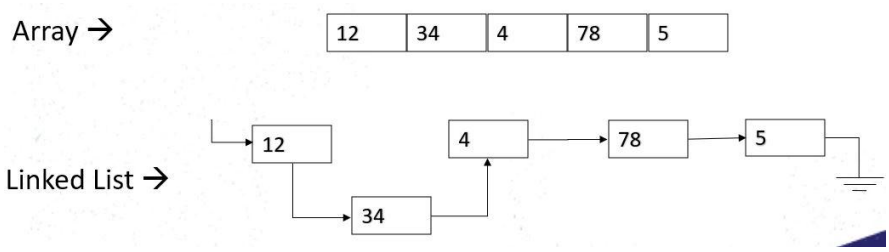
Array	LinkedList
each item occupies a particular position and can be directly accessed using an index number.	need to follow along the chain of element to find a particular element. A data item cannot be accessed directly.This can either be forward or backward



Real world application

•*Image viewer* – Previous and next images are linked, hence can be accessed by next and previous button.

•*Previous and next page in web browser* – We can access previous and next url searched in web browser by pressing back and next button since, they are linked as linked list.

•*Music Player* – Songs in music player are linked to previous and next song. you can play songs either from starting or ending of the list.

•Implementation of stacks and queues

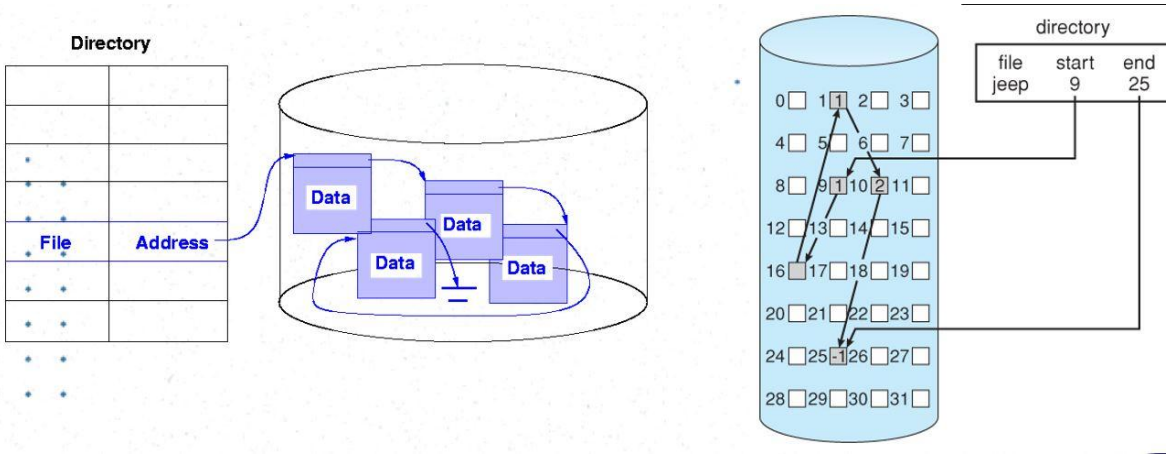
•Implementation of graphs : Adjacency list representation of graphs is most popular which is uses linked list to store adjacent vertices.

•Dynamic memory allocation : We use linked list of free blocks.

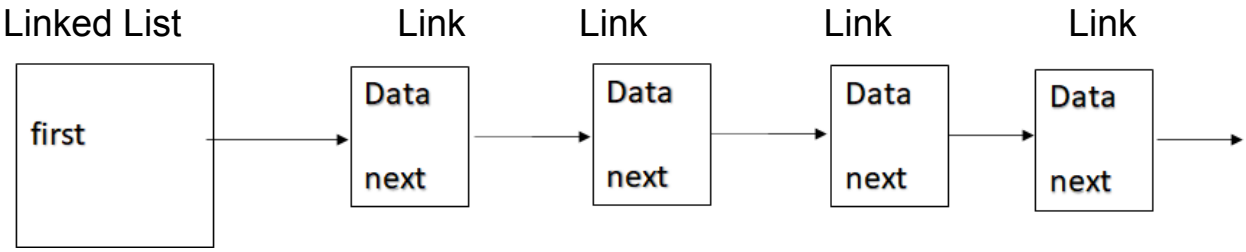
•Maintaining directory of names

Example to how it is implemented in the file system.

The directory has the pointer to the first block in memory(block is usually 512mb).the first block contains the pointer to the second block.the second to the third and so on.



Linked List



The final element points to a null value

•In a linked list each data item is embedded in a link.

•There are many similar links.

•Each link object contains a reference to the next link in the list.
In a typical application there would be many more data items in a link

Operations

•Mainly the following operations can be performed on a linked list.

- Find

  Find a link with a specified key value.

- Insert

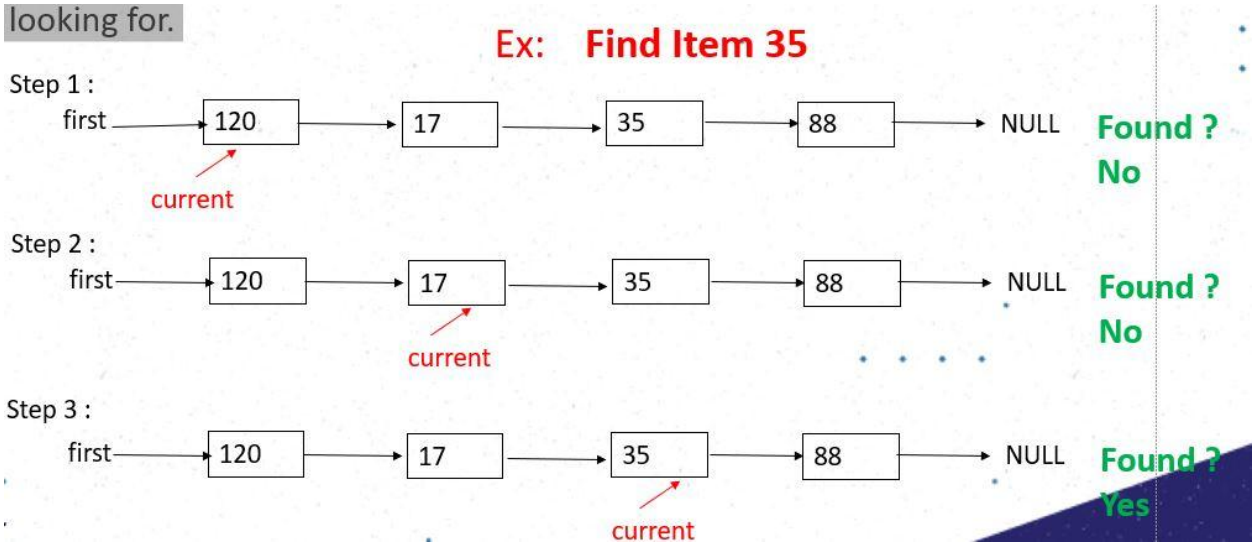
  Insert links anywhere in the list.

- Delete

  Delete a link with the specified value.

Find

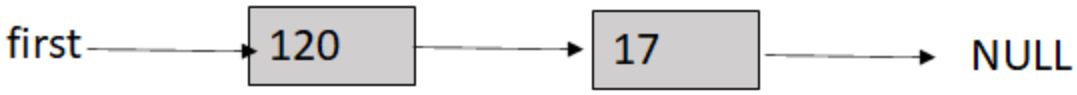
Start with the first item, go to the second link, then the third, until you find what you are looking for.



InsertFirst

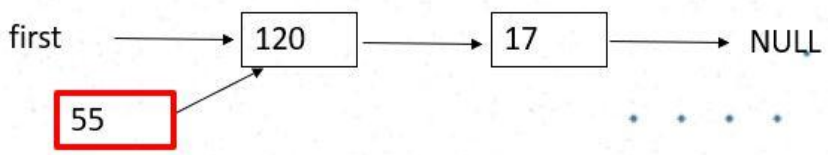
Inserting is done at the beginning of list

Before inserting

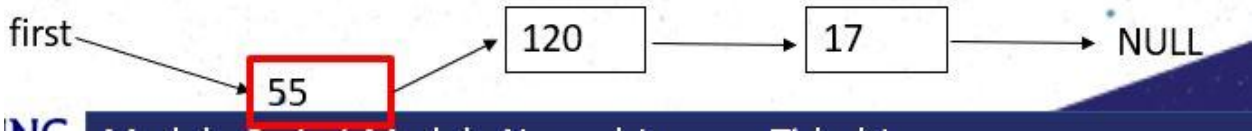


Inserting 55
Step 1: create a new link

Step 2: bind the next field of the new link to point to the old first link

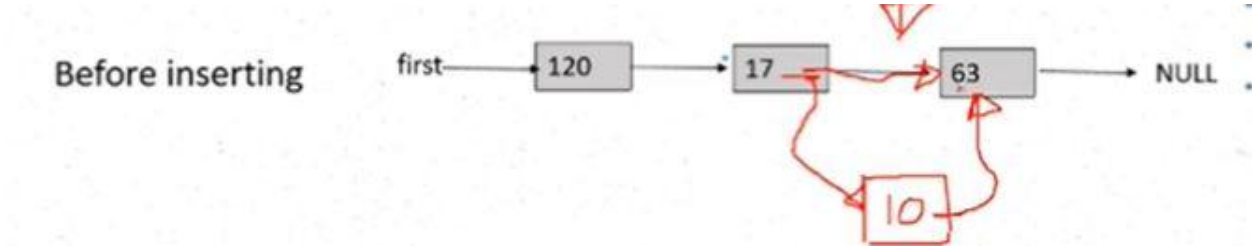


Step 3: bind the ‘first’ pointer to the newly created link



Insert

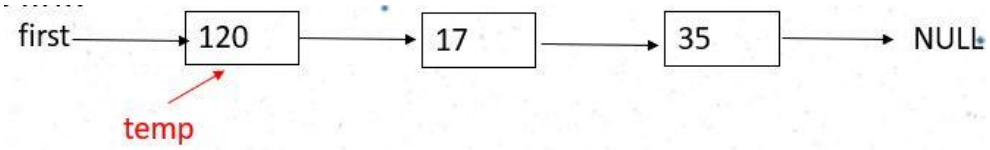
Inserting 55
Step 1: create a new link
Step 2: retrieve the address from the previous block to the next block and add it to the link as next
Step 3:Add the new links address to the previous blocks pointer.



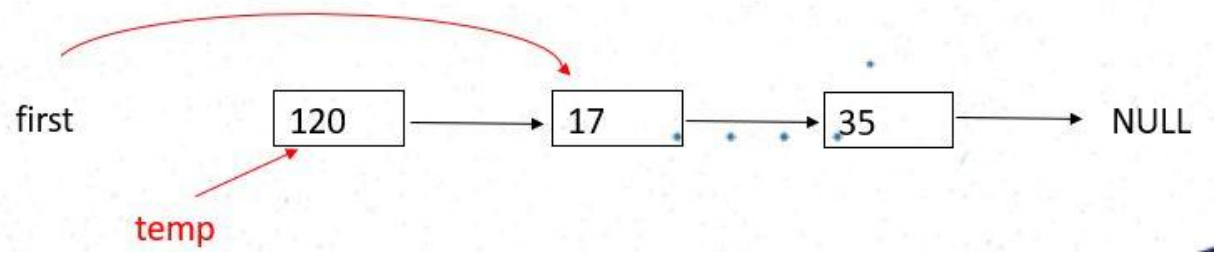
DeleteFirst

Deleting an item from the beginning of the list

Step 1: Save the link that needs to be deleted in a temp variable



Step 2:Disconnect the first link by routing the first pointer to the link that is next to the one to be deleted. Using the value of the ‘next pointer in the link that is to be deleted’



Step 3:return temp

Delete

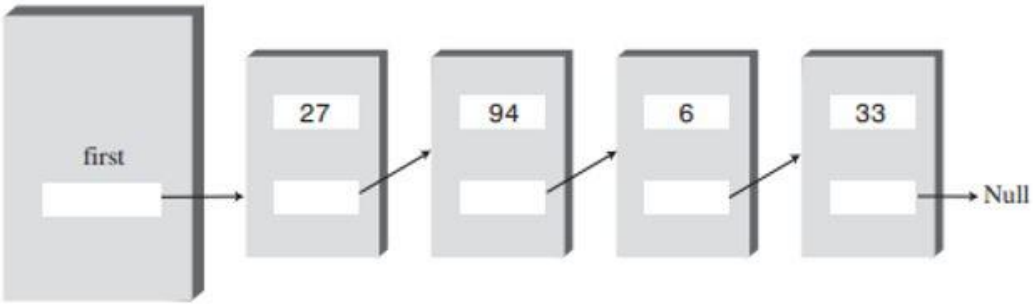
Deleting 17
Step 1: add the link that is needed to be deleted to a temp variable
Step 2:using the pointer value in the link that is to be deleted that points to the next element from the link that is to be deleted, update the pointer of the previous links ‘next’ pointer .
Step 3: return the temp.

LinkedList implementation

Link class

In a linked list, a link is an object of a class called something like “**Link”**.
There are many similar links in a linked list.
Each link contains Data Items and a reference to the next link in the list.

```
class Link {
    public int iData;     // data item
    public Link next;     // reference to the next link

    public Link(int id) { // constructor

                 iData = id;
                next   = null;
     }
     public void displayLink() {  // display data item

            System.out.println(iData);
     }
}
```



27,94,6 and 33 are contained in the data part while the next refers to the next link.
The link list class which is denoted by the large tab in the picture contains the address to the first data block.

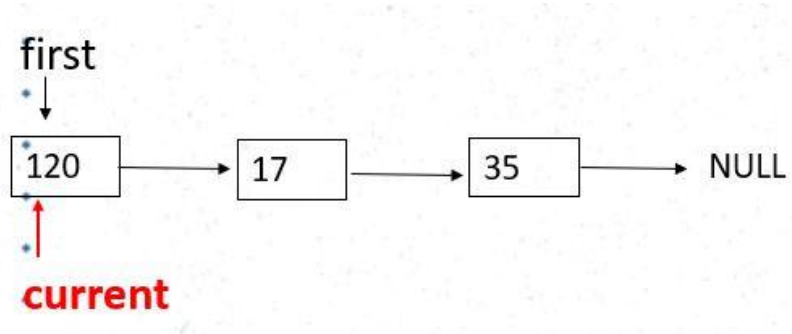
Linked list class

-The LinkList class contains only one data item, a reference to the first link on the list called **'first'**.

-It is possible to find the other links by following the chain of references from '**first**', using each link's next field.

```
class LinkList {
    private Link first;

     public LinkList() {       //constructor

            first = null;
     }
     public boolean isEmpty() {  // true if list is empty

             return (first == null);
     }
     // ……….          other methods
}
```

Displaying the linkedList



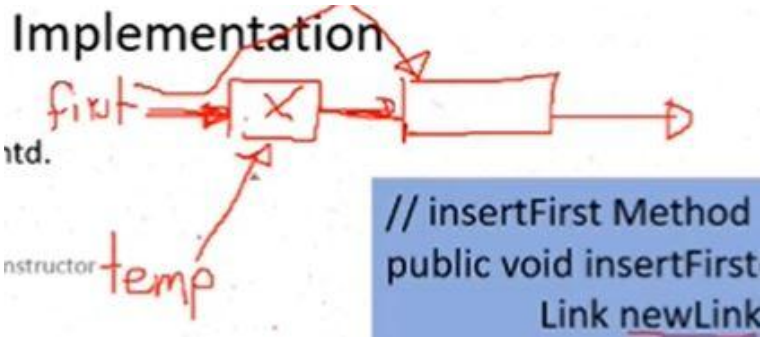
```
class LinkList {
    // private Link first;
    //  public  LinkList() { //constructor
    //      first = null;
    //  }
    //  public boolean isEmpty() {  // true if list is empty
    //      return (first == null);
    //  }
     public void displayList()  {
//current is a variable of type Link
               Link current = first;
//this loops through the list upto the null value
//displayLink() is a method in the link class
//next is an attribute in the link class
               while (current != null) {
                       current.displayLink();
                       current = current.next;
               }
               System.out.println("Empty linked list");
   }
}
```

Insert first method

```
// insertFirst Method
public void insertFirst(int id) {
//creating a new Link type object.
    Link newLink = new Link(id);
//update the next attribute of the newLink object with the first val off
the linked list class
    newLink.next = first;
//updating the fist variable of the linked list class with the pointer to
the newLink
    first = newLink;
}
```

Delete first method

```
// deleteFirst Method
public Link deleteFirst() {
// create a temporary variable of type link and add the pointer from first
to it.
    Link temp = first;
//update the pointer variable of first using the next variable in the
object that the first points to.
    first = first.next;
//return the temp
    return temp;
}
```



Q1)

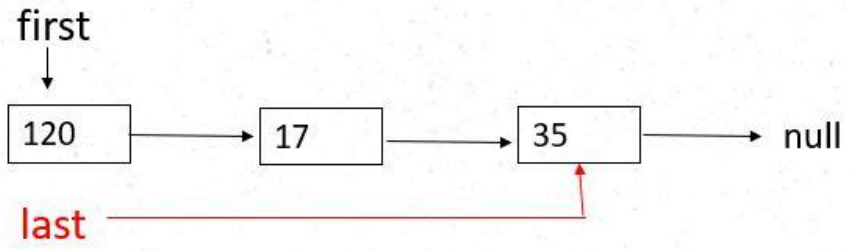
Write a program to
i)Create a new linked list and insert four new links.
ii)Display the list.
iii)Remove the items one by one until the list is empty.
(Use the LinkList class created)

```
class  myList {
public static void main(String[] args)      {
        LinkList theList  = new LinkList();    // create a new list

        theList.insertFirst(23);    // insert four items
        theList.insertFirst(89);
```

```
        theList.insertFirst(12);
        theList.insertFirst(55);

        theList.displayList();            //display the list

        while( !theList.isEmpty() ) {    // delete item one by one

            Link aLink = theList.deleteFirst();
            System.out.print("Deleted ");
            aLink.displayLink();
}
```

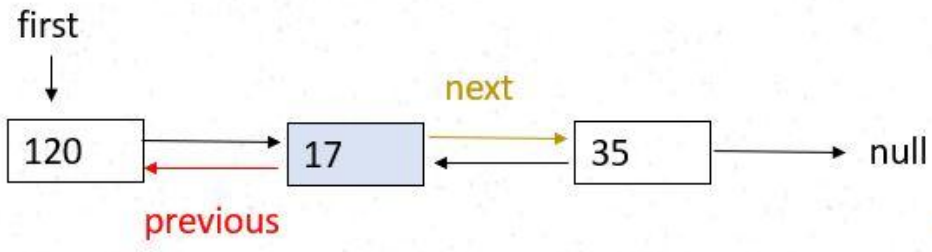
Double ended List

A double-ended list is similar to an ordinary linked list with an additional reference to the last link.



Doubly Linked List

A doubly linked list allows to traverse backwards as well as forward through the list.
Each link has two referen



ces.