- Abstract class can't be made without child class
- We want to restrict object to be created without inheritance.

① What is the way?
- Make constructor private
  Declaring a private constructor can help to do so (stop creation of object)

② Why to do this?
  We want to restrict number of objects (It doesn't mean we don't want to make object. It means we want to control the number of objects)

- If constructor is used private then no obj can't be made. In this situation we made static functions.
- Static function will call constructor.
- Take count of count < range create new object. otherwise don't create object.

lekin copy constructor bhi bnau sakta new obj.

```
int main() {
*obj = Check :: get object();    or
Check *obj = Obj1 → getobject();
    Obj1 → show();
```

```
Check *obj2 = Check(*obj
Check obj2(*obj1)
        ↓
  copy constructor
```

```
class check {
    check() { }  → count++
public    work without object
    static check* getobject() { }
    void show() {
};
```

Static member function can't call simple member functions

• Copy constructor must be private
    check (check &) { } count++

Pointer means to return address of object or return NULL. or ptr
    static check* getobject() { }

→ Static int count;

OOAD software Engineering II
object oriented Analysis & Design

- Design Patterns
- Singleton Pattern → say there must be only 1 object
- MVC (Model View Controller

How to?

```
get Object() {
    if (ptr == NULL)
        ptr = new check;
    return ptr.
}
```

Example: principal must be 1
This pattern enforce there must be 1 object

• MVC pattern is used most of the time
eg webapps . mobile apps

$\boxed{\text{View}}$ The thing that we will view to the user.
show on screan, browser, mobile screen

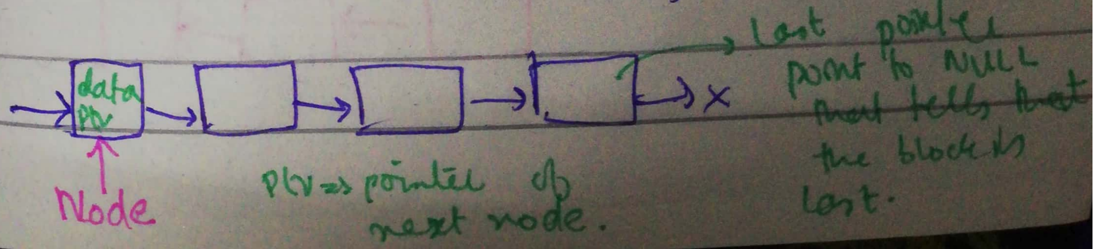
$\boxed{\text{Model View}}$ It is basically your data

$$\begin{tikzcd}
 & \text{Student Member} \\
 & \text{Teacher Member} \\
\boxed{\text{data}} \arrow[ur] \arrow[uur] \arrow[r] \arrow[dr] & \text{Books} \\
 & \text{Reference books}
\end{tikzcd}$$

$\boxed{\text{controller}}$ It is business logic
↓
Add courses
drop courses

Linked list

• When you make array it is contiguous
• compaction separates space and reserved memory. but it's difficult task
• But speed is improved.
• Sometimes you want to store data contiguous so you used linked list
• store elements by linking


data
Ptr
Node
Ptr => pointer of next node.
Last pointer point to NULL ~~that tells that~~ the block is last.
x

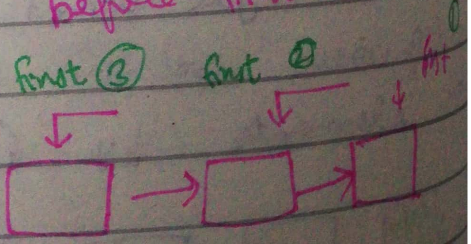
• conceptually it looks like train
• but practically its not.

```
class Node{
        int data;
        Node *next;

    Node(int d){
            data=d;
          next=NULL;
    }
};

class LinkedList{
        Node *first;
```
↗ means not a single node in class

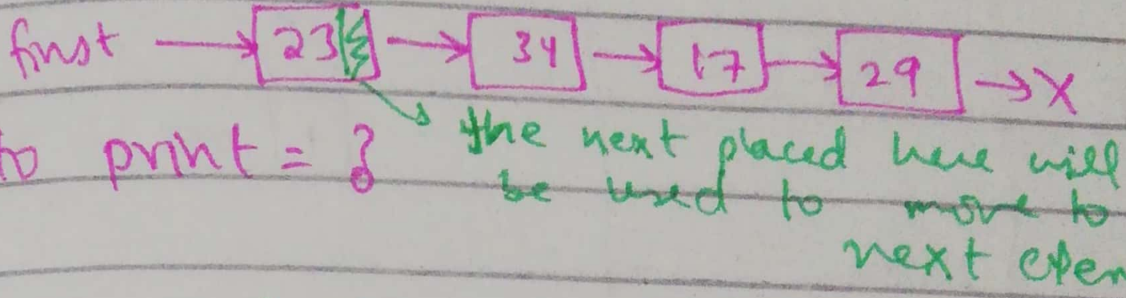
```
Public:
    LinkedList( ){ first==NULL;}
    void addElement (int e){
```
• First create a node

```
        Node n(d);
```

• 2nd step how to link
There are 2 possibilities
• if it is first node then
  first = &n;
• If first is pointing any node
• Then make a node before first and
call it first



• Save address & starting & last if you want to go to reverse order.


first
23
34
17
29
X
the next placed here will be used to ~~move to~~ next element

How to print = ?

```
for(Node *t=first; t!=NULL, t=t->next)
        cout << x[i] << ' ';
```

```
If (first = = NULL)  first = n;
else {
  ~~first = first~~ n -> next = first;
                first = n;
  }
};
```

Scanned with CamScanner