what is it meant by

Collections Framework.

↳ collection of classes

& Interfaces, which helps

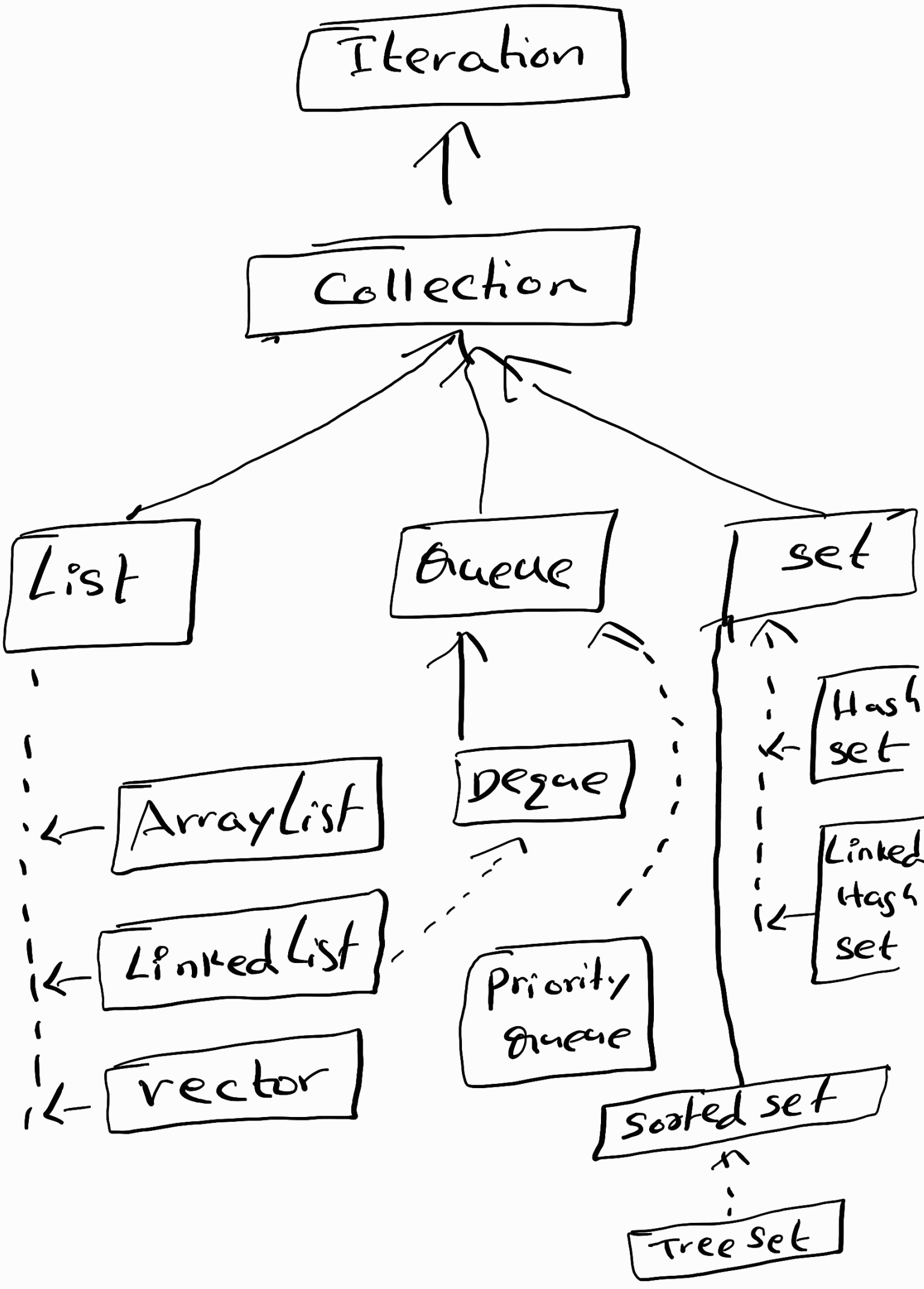
in Storing & processing the

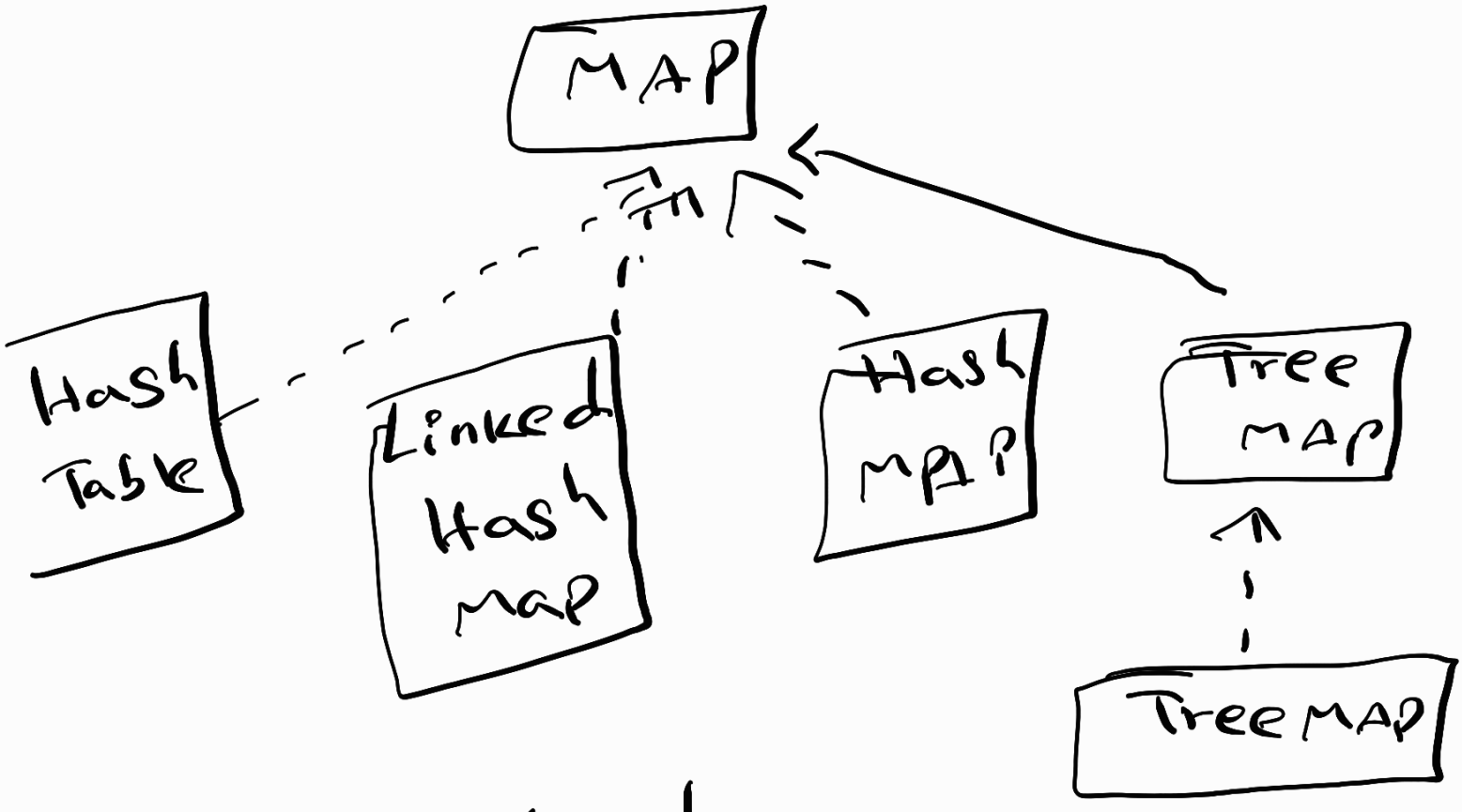
data.

see the following diagram

of Collections Framework

hierarchy


Iteration
Collection
List
Queue
set
ArrayList
LinkedList
rector
Deque
Priority Queue
Hash set
Linked Hash set
Sorted set
Tree Set



⟶ extends

- - -> implements

---

why we need Collections

to make the

programmers task Super Easy,

while dealing with large

amount of data.

① List

⇒ what is List?

⇒ why the name kept it as List..?

⇒ what is the use of it,

⇒ what are the advantages & disadvantages of List.

⇒ In which scenario's we use List

⟶

it can be a list of names

it can be a list of car name

it can be a list of groceries

it can be a list of Notebooks

names :- {trinadh, shaida, Jakesh, Chandu, paran}

Carnames :- {Benz, Audi, Togota, TATA, TESLA, BMW, maruti}

Groceries :- {Tea, Coffee, milk, Toli powder}

etc....

List (Interface)

1) It is an ordered collection

2) List may contain duplicate elements

3) The classes that implemented List Interface are

   i) ArrayList
   ii) LinkedList
   iii) Vector
   iv) Stack.

1.1) ArrayList (class)

ArrayList in Java, is a resizable-array implementation of the List Interface.

ArrayList implements List Interface & it is based on the Array data structure.

Declaration of ArrayList:-

1) ArrayList<String> list = new ArrayList<>();

i) add() → to add elements

ii) set() → to change elements

iii) remove() → to delete element

Iterating ArrayList (for each loop)

```
for (String list : Arraylst)
    S.out (list);
```

ArrayList size :

```
ArrayList.size();
```

All methods of ArrayList

1. add(Object o)
2. add(int index, Object o)
3. remove(Object o)
4. remove(int index)
5. set(int index, Object o)
6. int indexOf(Object o)

=) int pos = obj.indexOf("Brad");

Gives the index of the object o. If value not found then this method returns (-1)

⑦ Object get(int index)

⇒ String str = obj.get(2)

It returns the object of List which is present at the specified index.

⑧ int size()

⇒ int noofItems = obj.size();

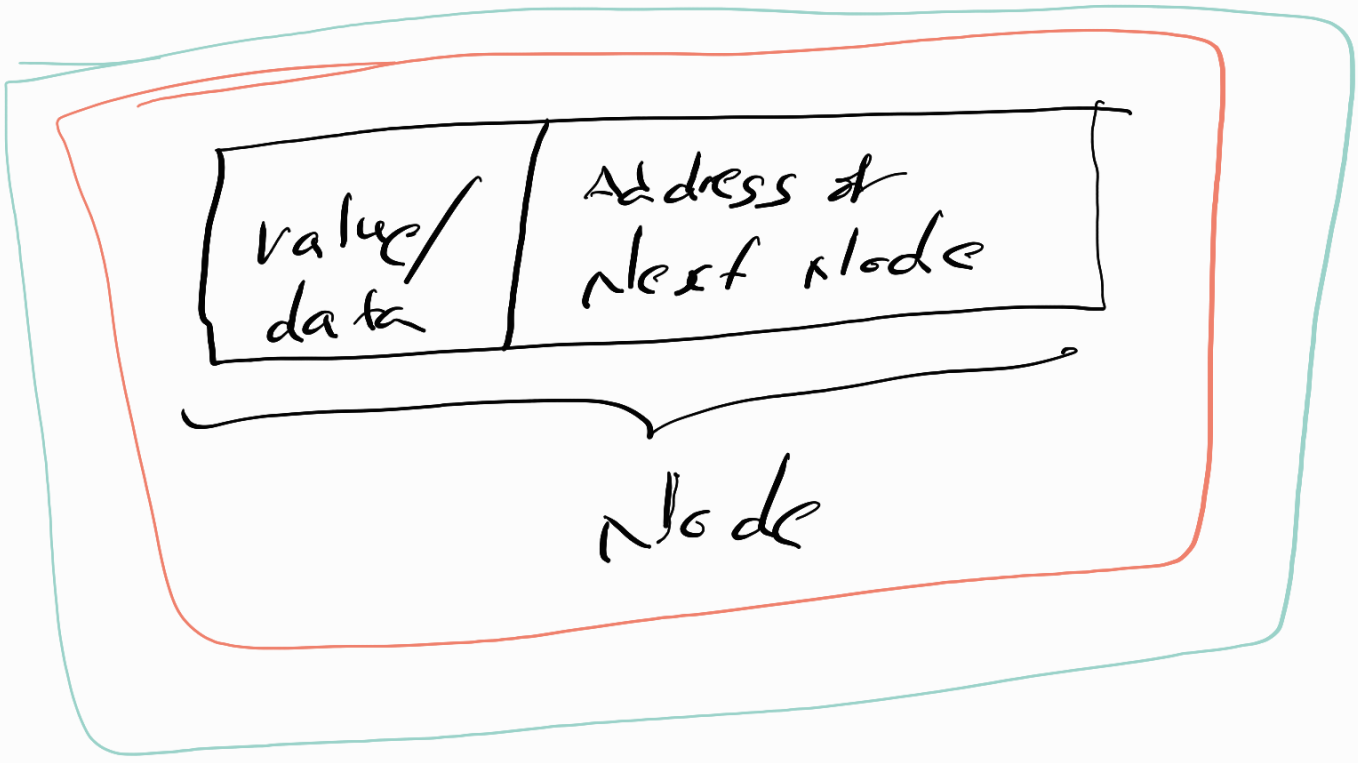
09/02/2023 Thursday

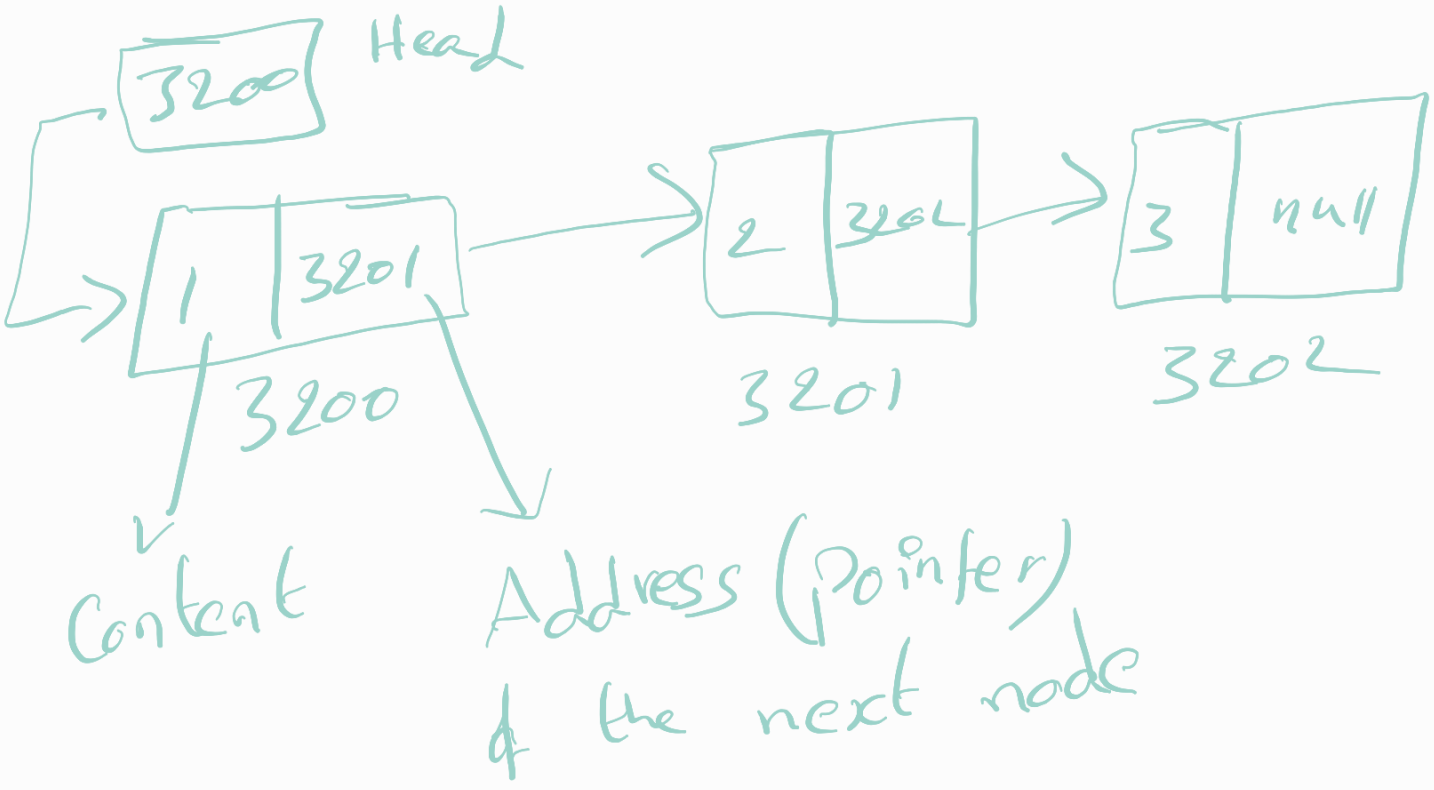
1.11) Linked List.

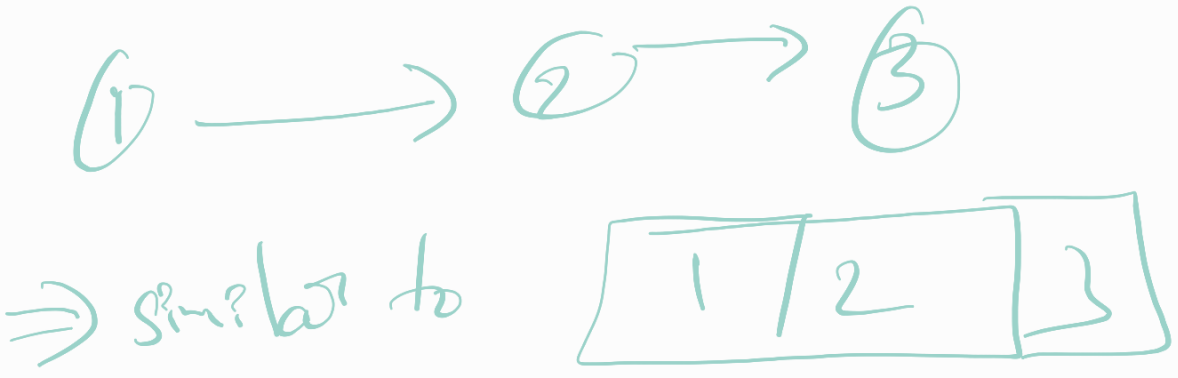
Linked List - (class)

Linked List is a linear data structure.

Linked List elements are not stored in Contiguous locations like arrays.

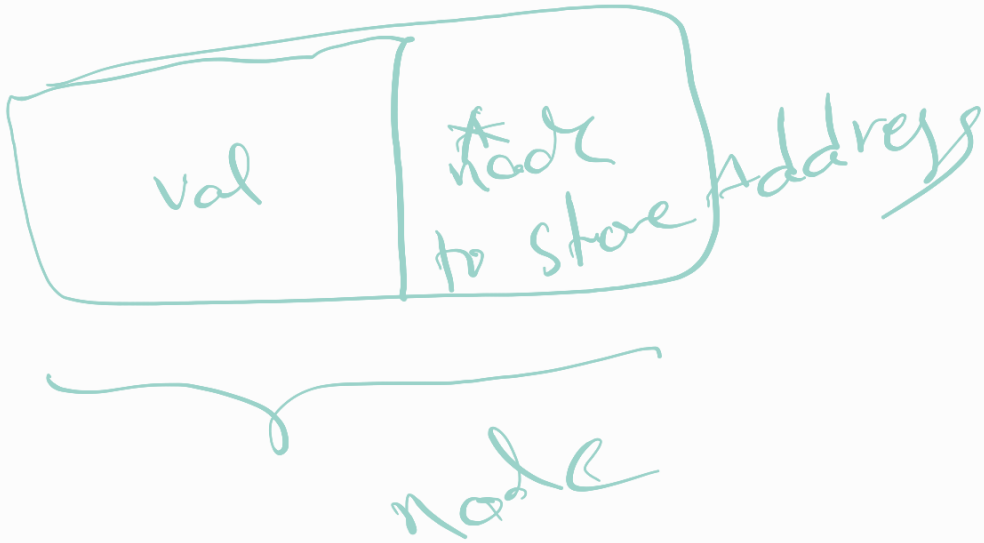


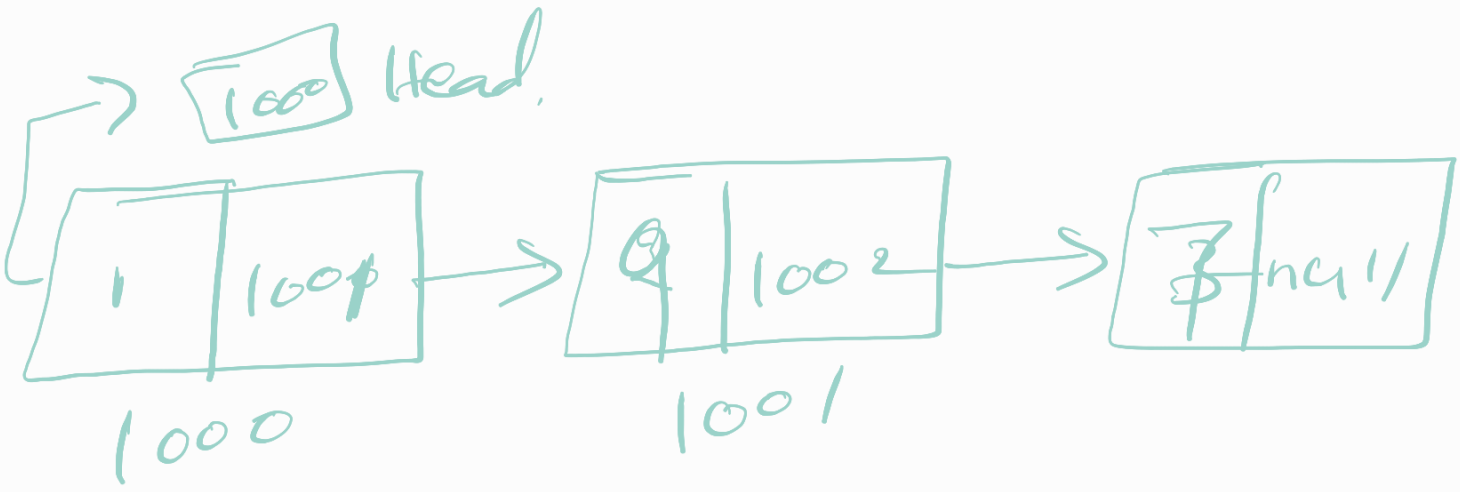


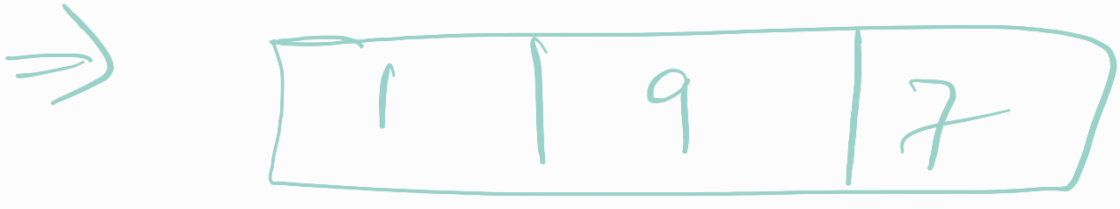



```
class Node {
    int data;
    Node next;
    Node() { };
```

```
Node (int val) {
    this.val = val
}
Node (int val, Node next) {
    this.val = val;
    this.node = node;
}
```



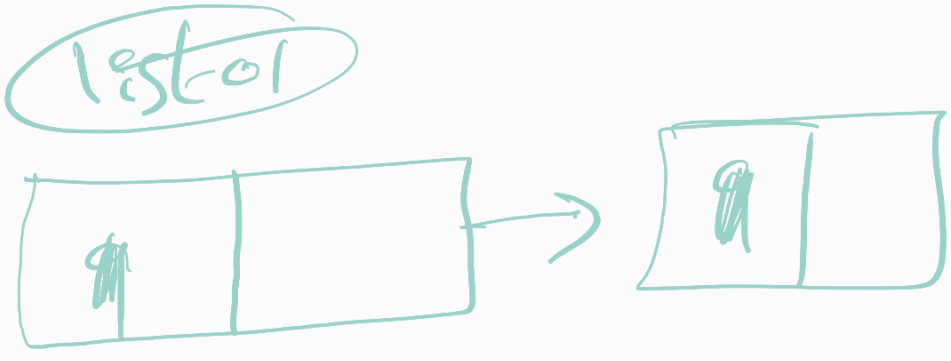




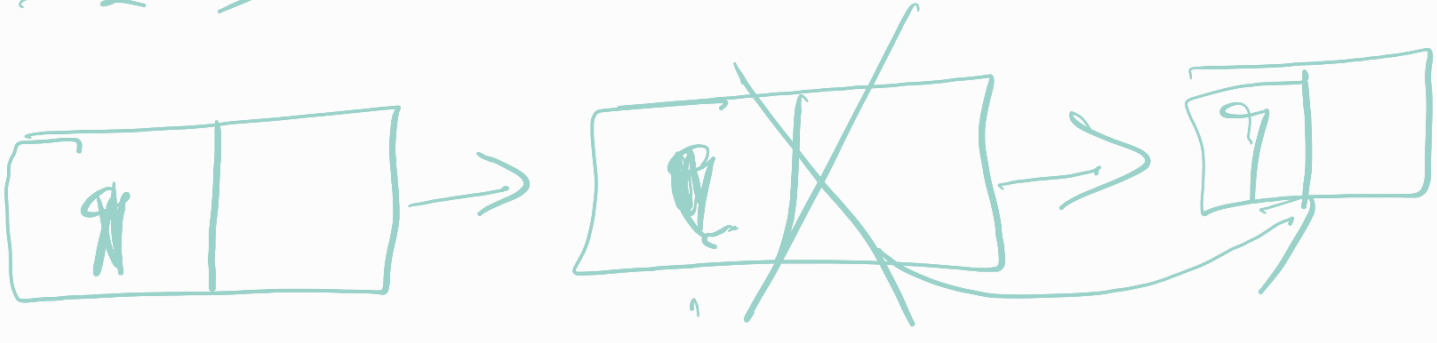

```
Node listol = null;
Node tailol = null;
for(int i=1; i<=3; i++){

if (listol == null){
    listol = newList(1);
    tailol = listol;
```

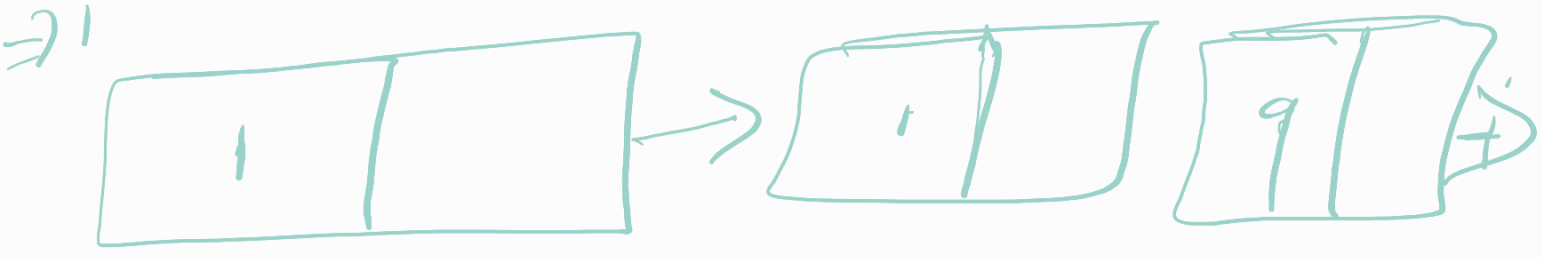
```
        }
else {  tail.next = new List(9);
        tail = listol.next;
     }
```

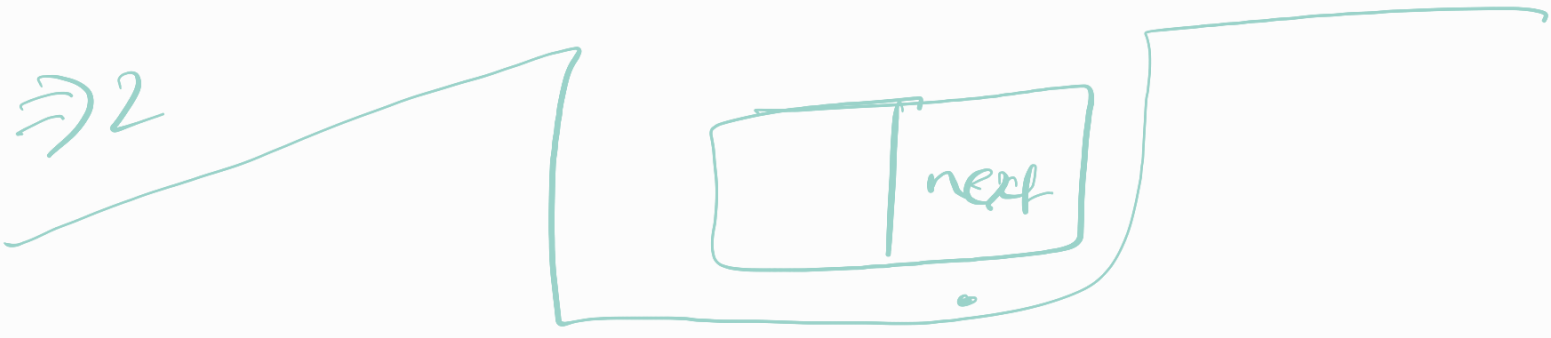
i = 1 ⇒

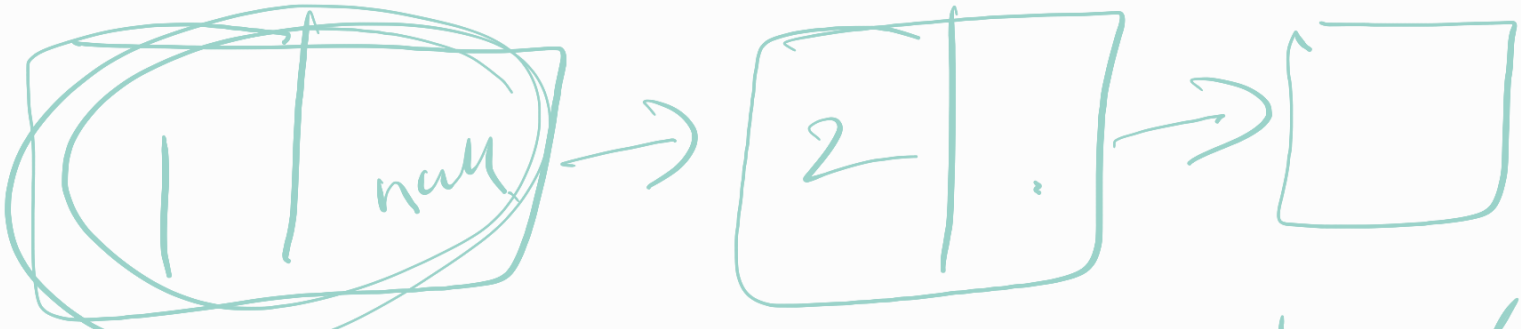


i = 2 ⇒

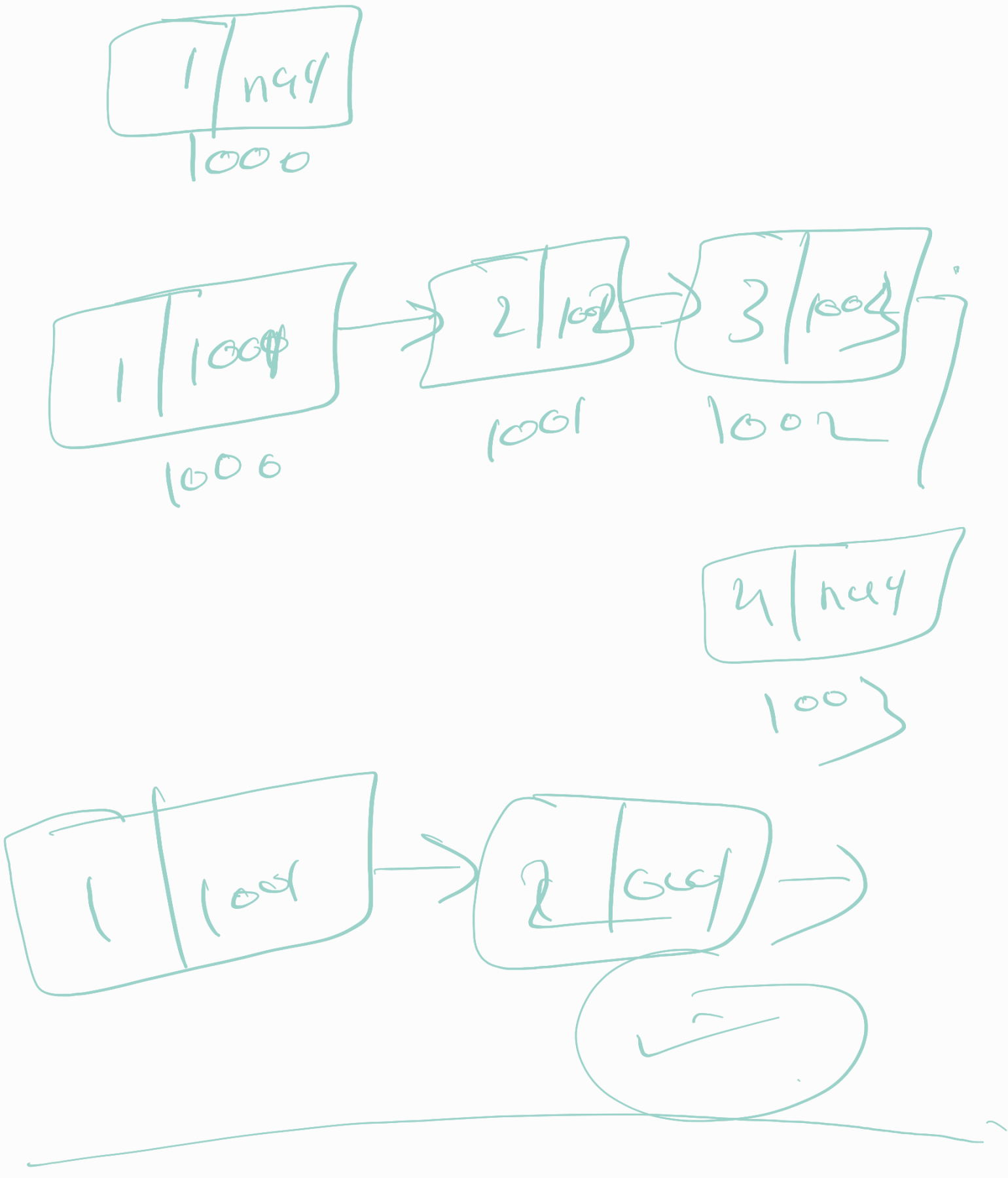


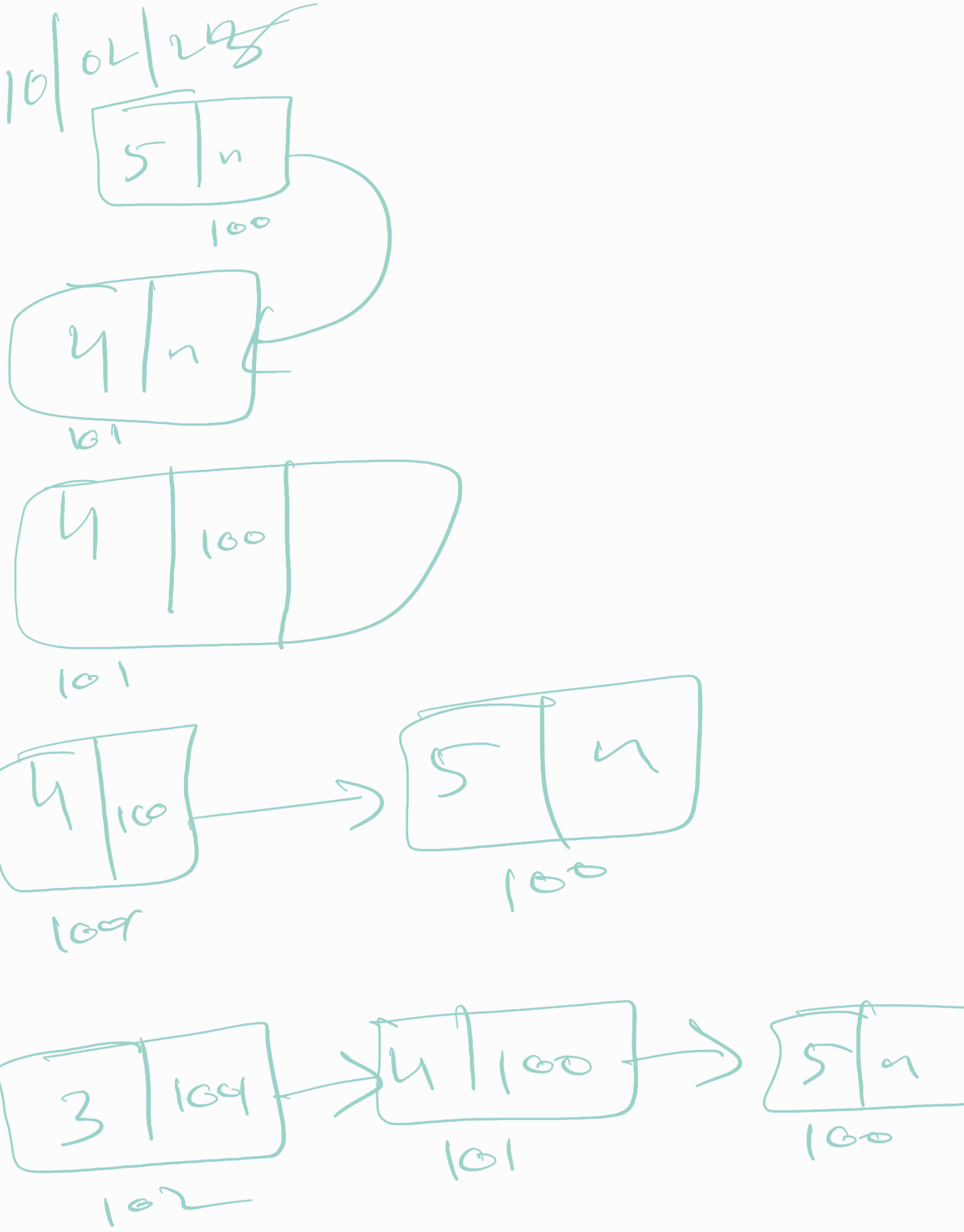


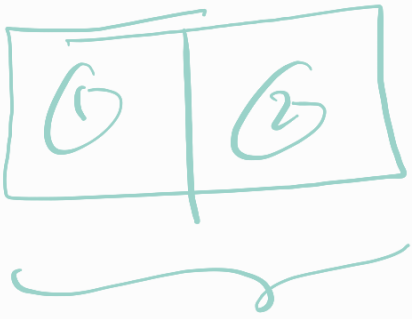




tail = list.next


1 | null
1000
1 | 1001
2 | 1002
3 | 1003
1000
1001
1002
4 | null
100
1 | 1004
2 | 0000

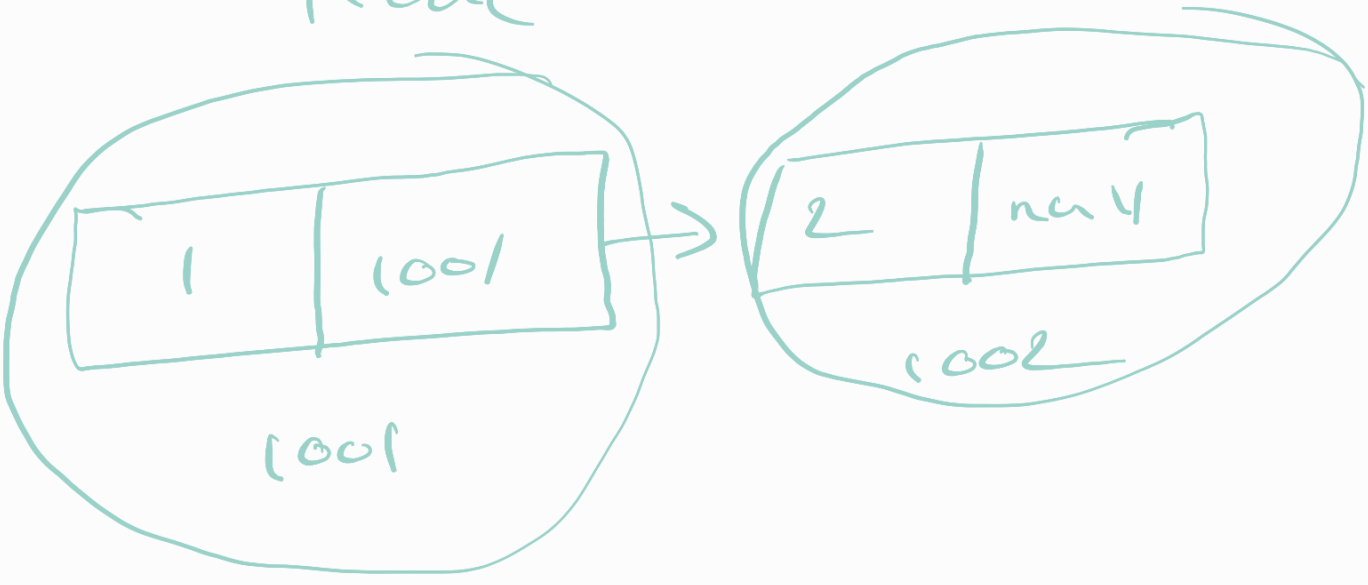

5
n
100
4
n
101
4
100
101
4
100
104
5
n
100
3
104
102
4
100
101
5
n
100

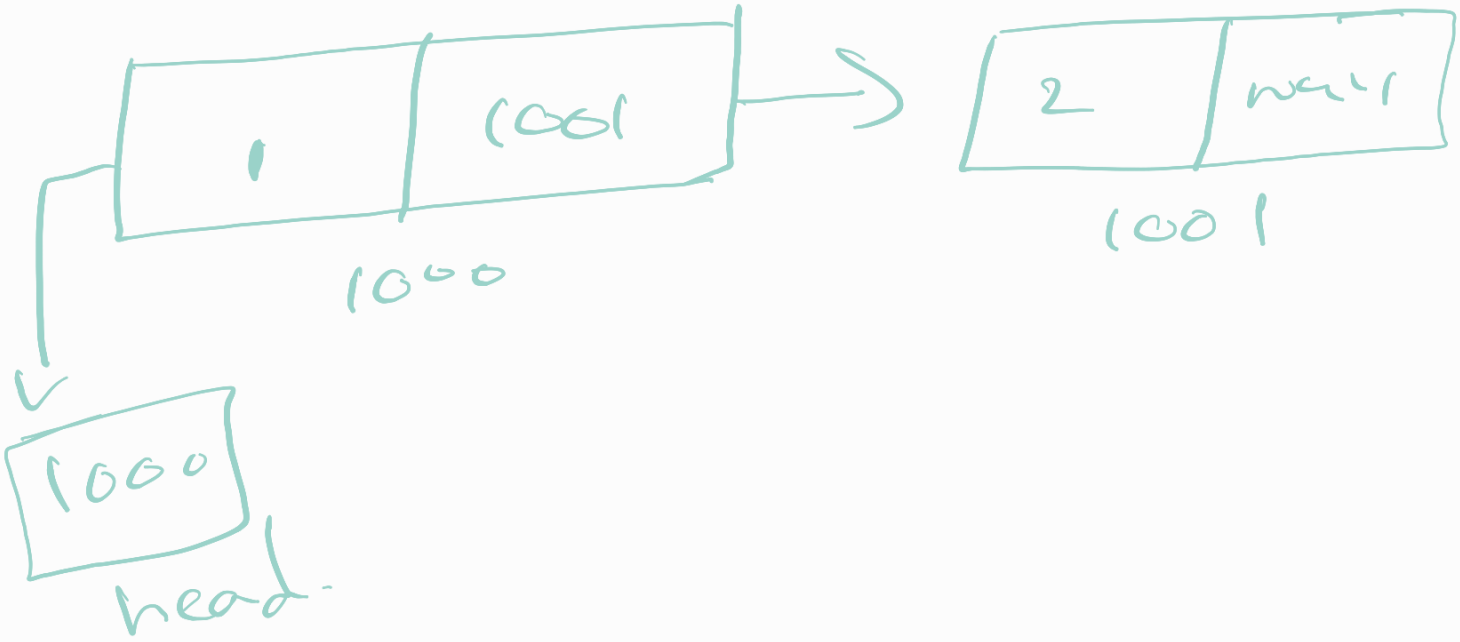


① → data

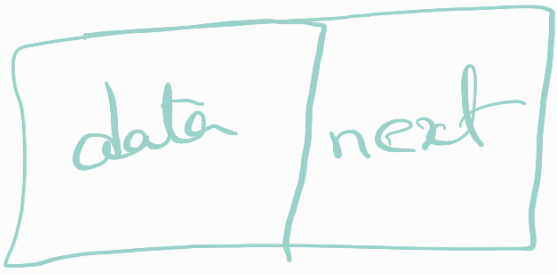
② → store to save Address

Node

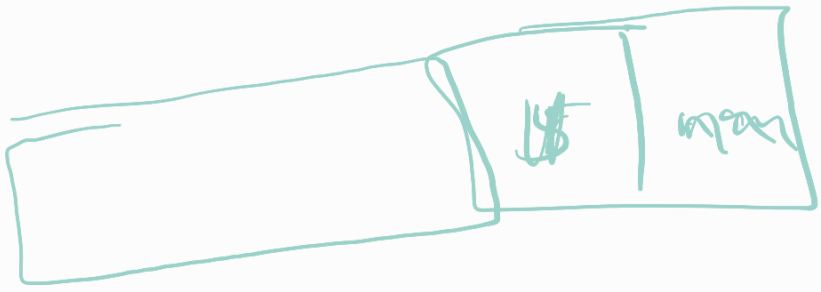




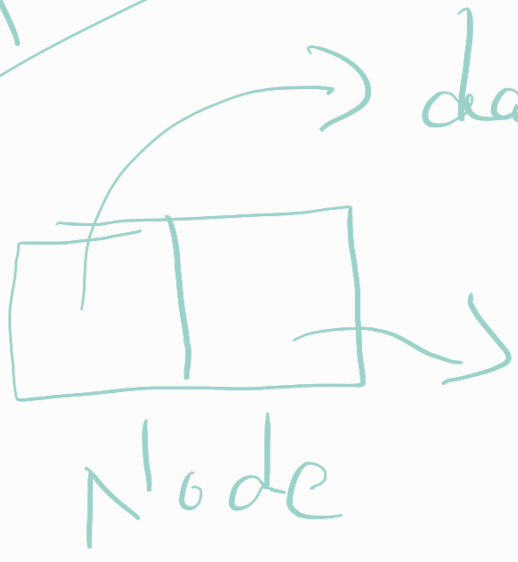
```
class Node {
    int data;
    Node next;
}
```

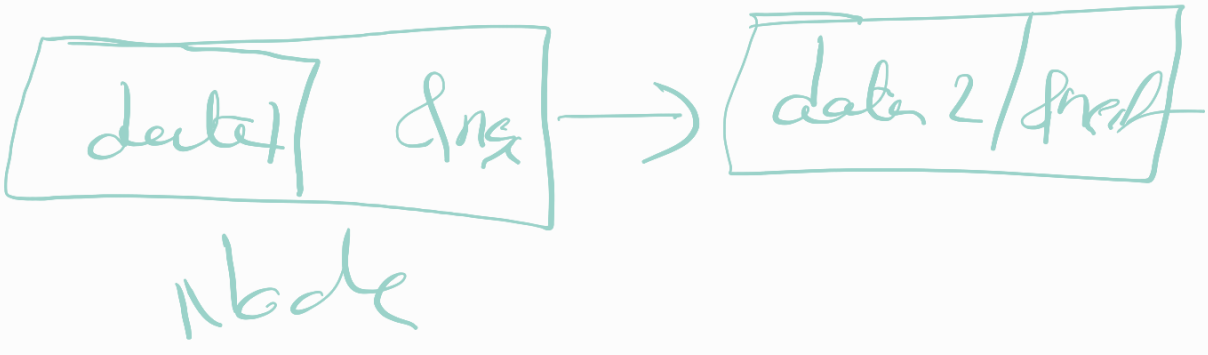


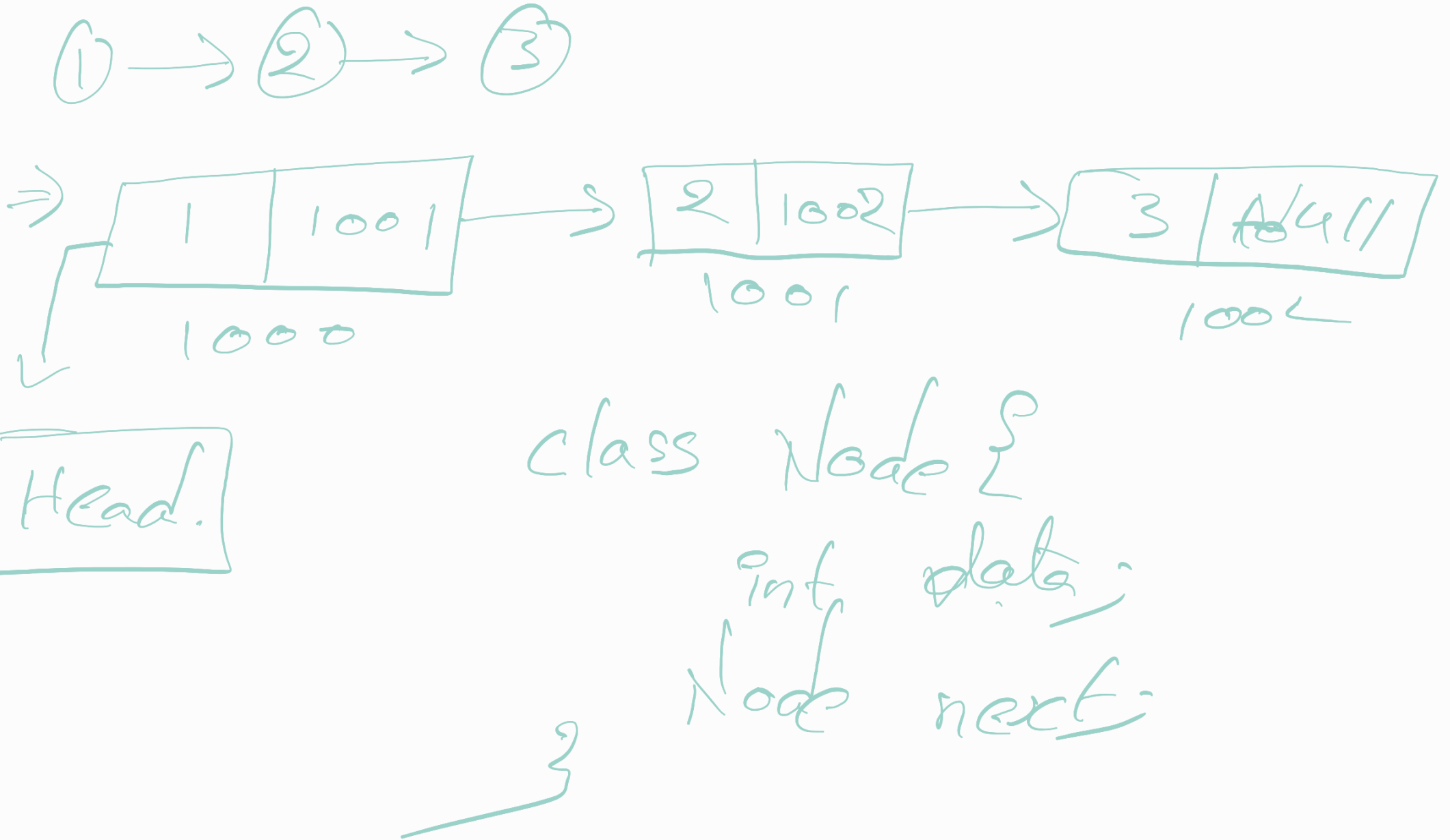

```
class CreateListNode {
    Node LL = null;
    Node head = null;

    while (-) {
        LL = new Node (data);
        LL.next = head;
        head = LL.next;
    }
}
```



11/02/23



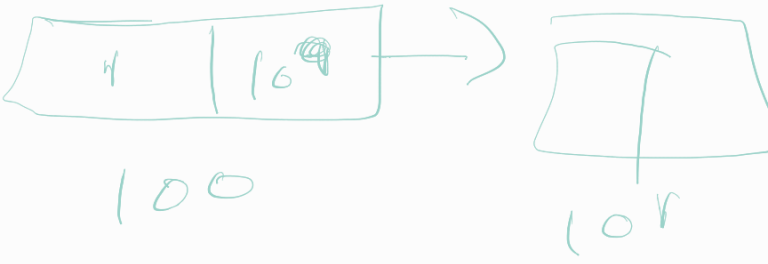



1 → 2 → 3
1
1001
1000
2
1002
1001
3
Null
1002
Head.
class Node {
int data;
Node next;
}

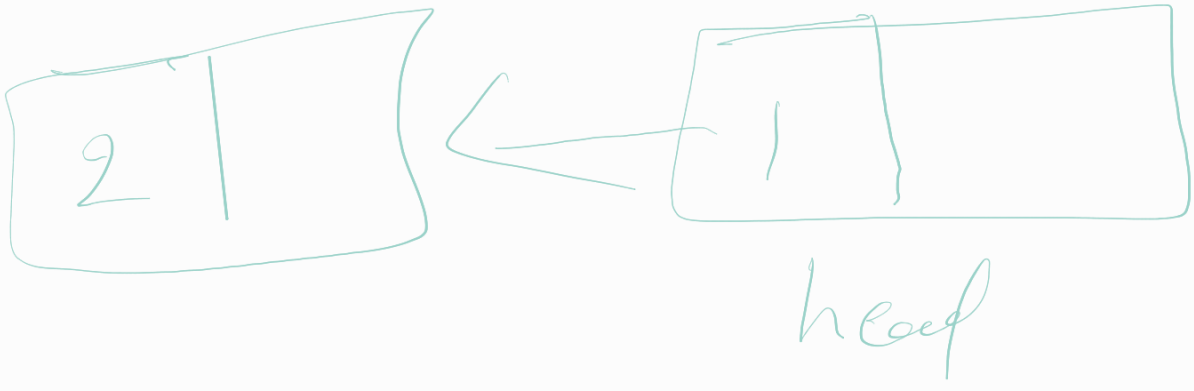
```
class Node{
    int data;
    Node next;
}
class LinkedList{
    Node head,temp;
    public void LinkedList(){
        head = null;
    }
    public void add(int data)
{
    for(int i=0; i<=3;i++){
        if(head==null){
            temp = new Node(1);
            head = temp;
```

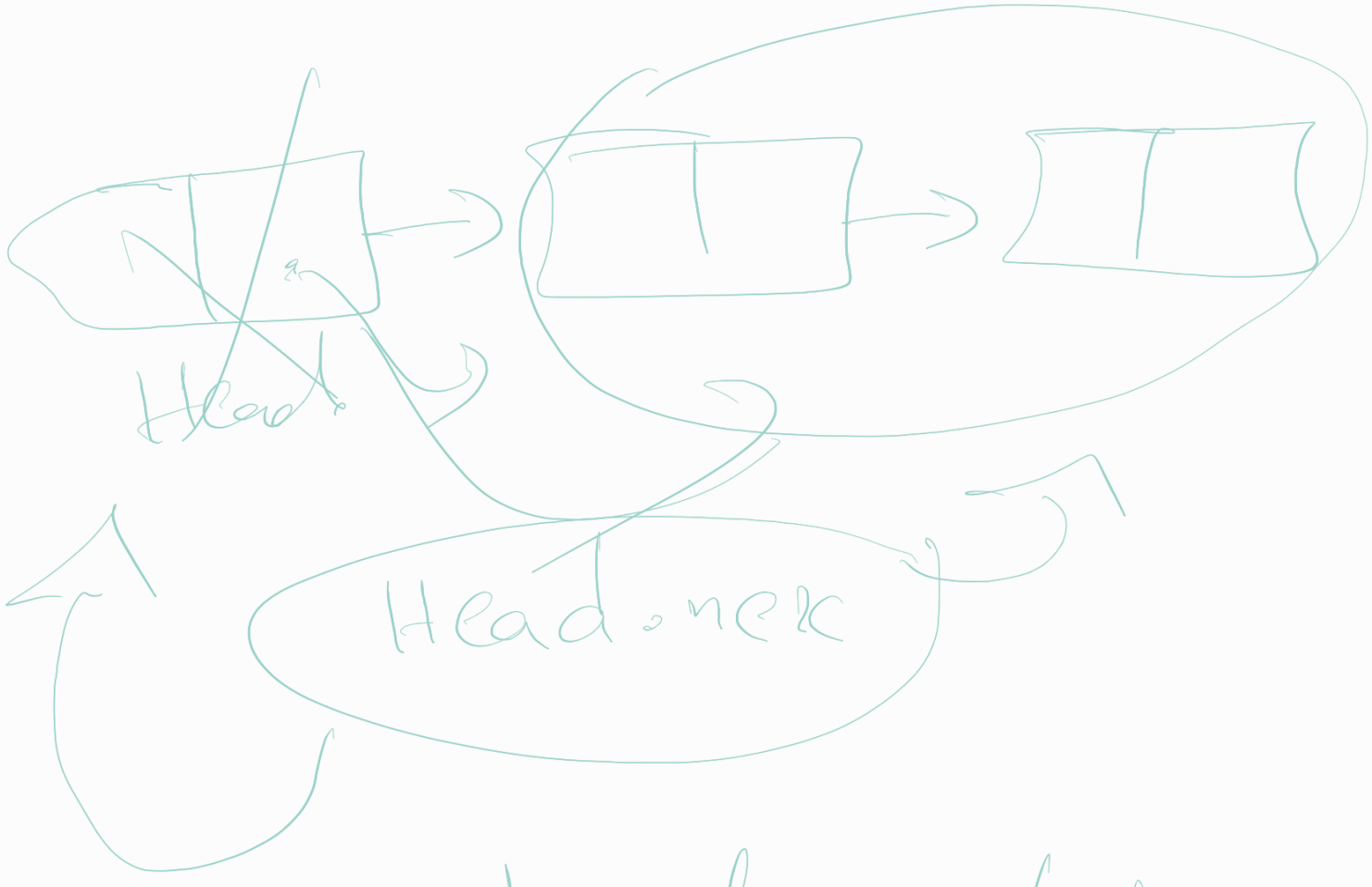
temp = head.next;

} else {



temp.next = new Node(2);





head = head.next

will delete first node

ie first head node is replaced from the second node to last.

Trinadh Thought process after tried different ways to Insert new Temp node at the End of the Linked List.

Just creak one more temp variable (i.e another temp) and then traverse to the End of the node i.e where

node.next == null

once traverse then all temp node blocks will delete except last block because that blocknext = = null

so here is the place we need to add new node.

Completed