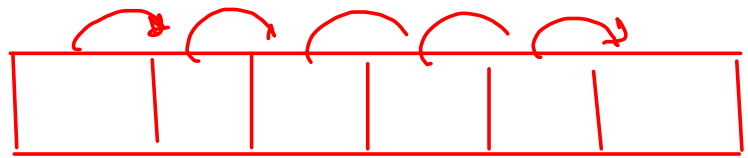
Data structures:

1. Get
2. Set
3. Remove

1. Arrays
2. Arraylist → Array → Dynamic Array.
3. Stack → LIFO
4. Queue → FIFO

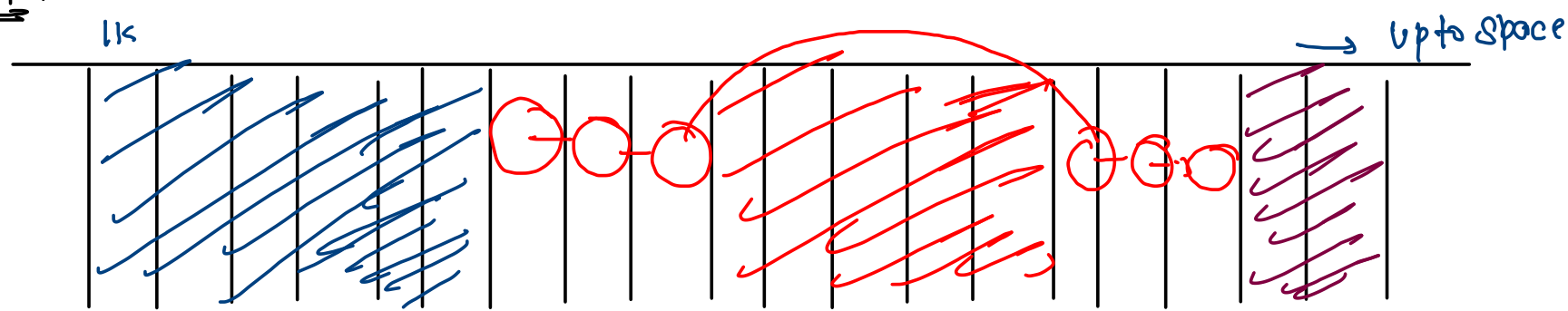
Special Linear DS

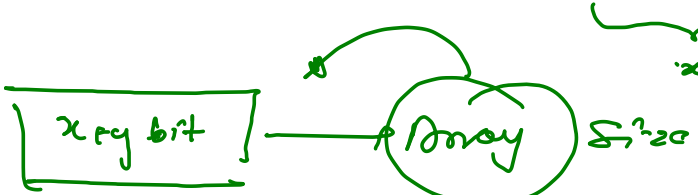
Linear type →



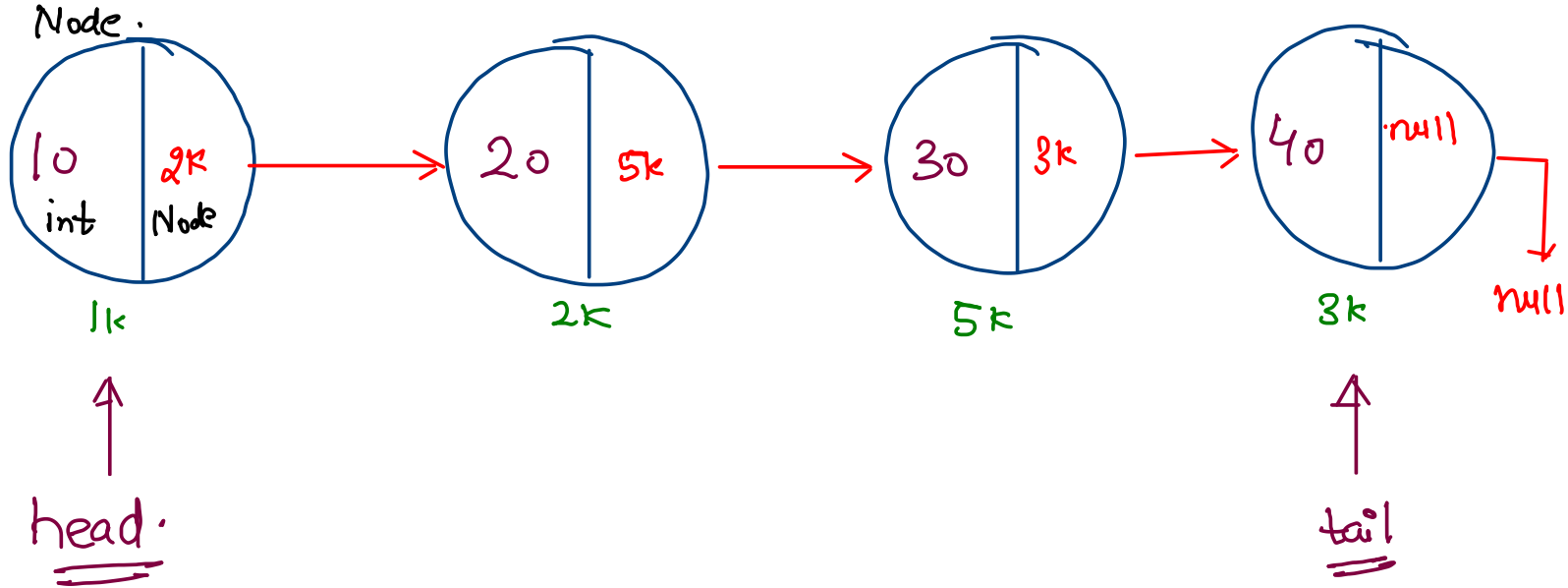
continuous Memory → 1K 1K+4 1K+8 1K+12 ----->

RAM





LinkedList



add

1. AddFirst
2. addLast
3. addAt

Remove

1. removeFirst
2. removeLast
3. removeA

Get

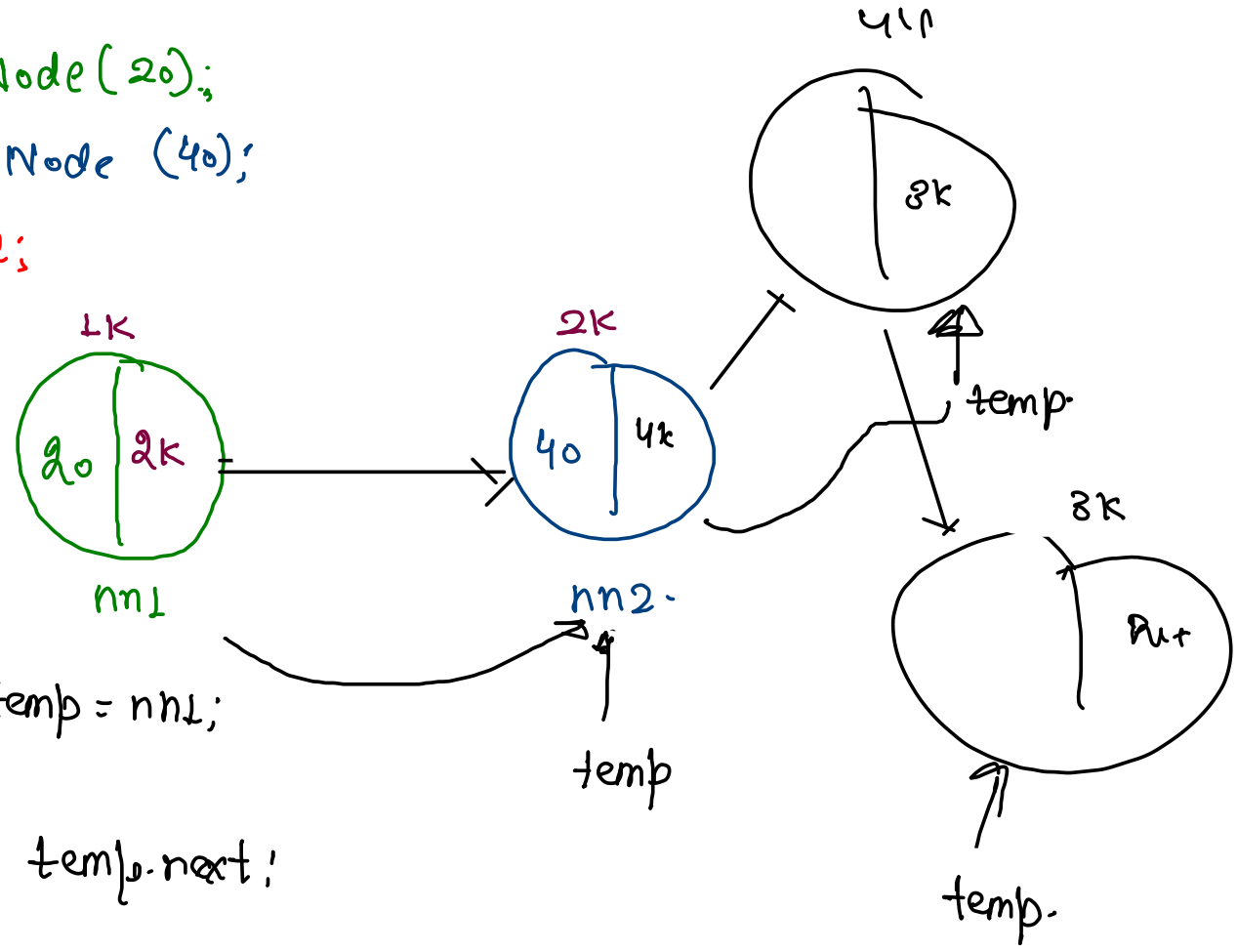
1. GetFirst
2. GetLast
3. GetAt

* Size
* Display

Structure of Single Node →

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

```
Node nn1 = new Node(20);
Node nn2 = new Node(40);
nn1.next = nn2;
```



Node temp.

temp: ~~1K~~ ~~2K~~ ~~4K~~ 8K

temp = nn1;

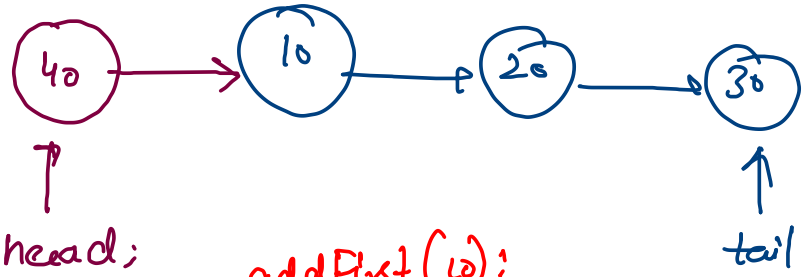
temp = temp.next;

Add First →

Head: 1k

tail = 1k

Size = 0

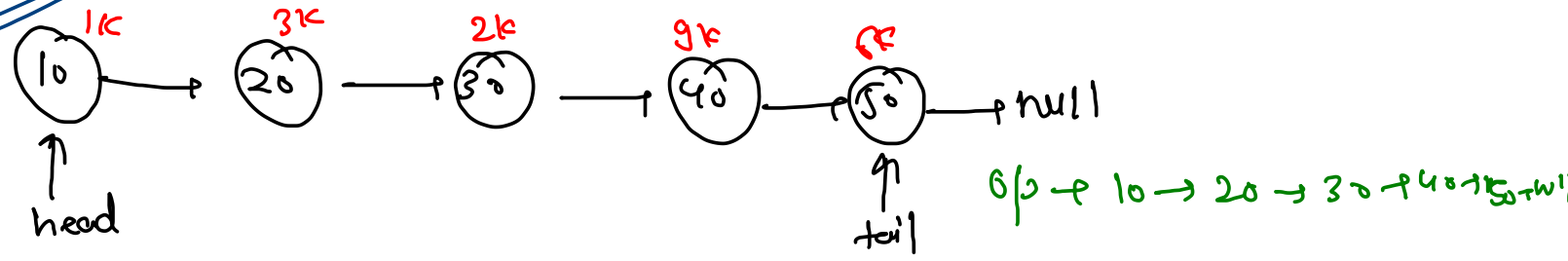


addFirst(10);
addFirst(40);

// memory allocation
Node nn = new Node(data);
// connection.
nn.next = head;
head = nn;
size ++;

Random Linked List

Display -



o/p → 10 → 20 → 30 → 40 → 50 → null

temp.

temp = head;

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

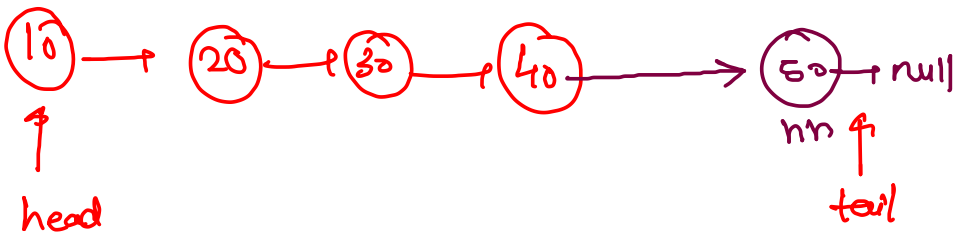
10 → 20 → 30 → 40 → 50 → null

AddLast

case I → Size == 0

head = tail = null

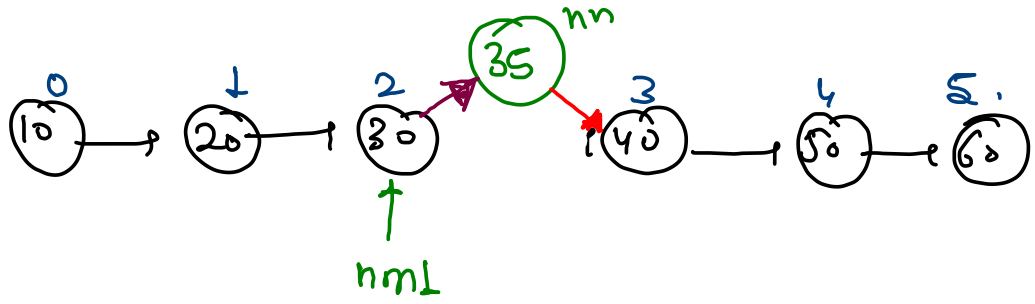
Size > 0



data = 50

```
// memory allocation.
Node nn = new Node(50)
// connection.
tail.next = nn;
tail = nn;
size++;
```

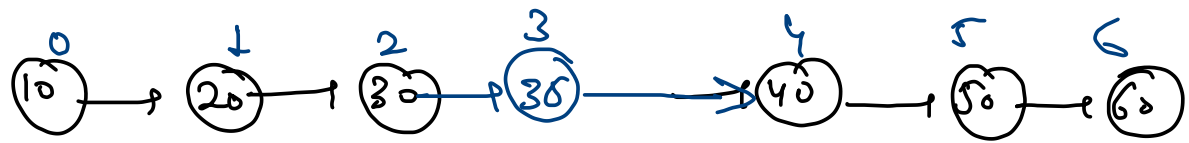
Add →




```
nm1 = getNodeAt(n-1);
nn = newNode(data)
nn.next = nm1.next;
nm1.next = nn;
```

add → 35, 3

o/p →

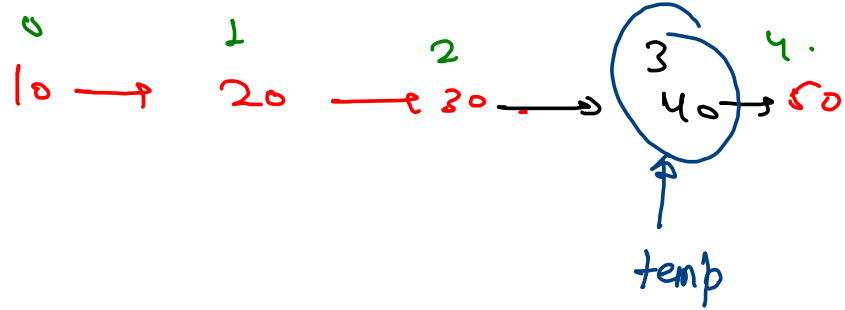


idx = 0] addFirst

idx = size] addLast

Valid Inputs

getNodeAt →



pos = 3

```
temp = head;
while(pos > 0) {
    temp = temp.next;
    pos--;
}
return temp;
```

(40)

pos = 3 2 1 0

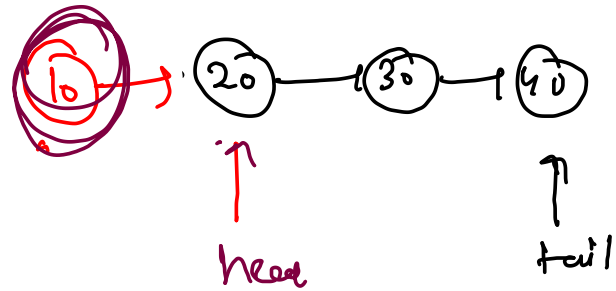
getFirst → head.data

getLast → tail.data

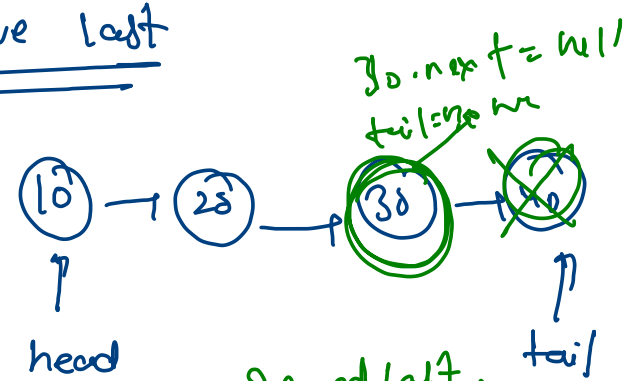
getAt(5) → validity → valid

else → -1

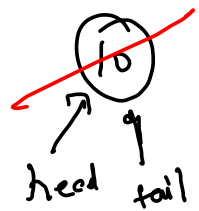
Remove First



Remove Last

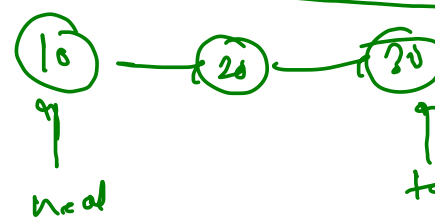


size == 1



head = tail = null;

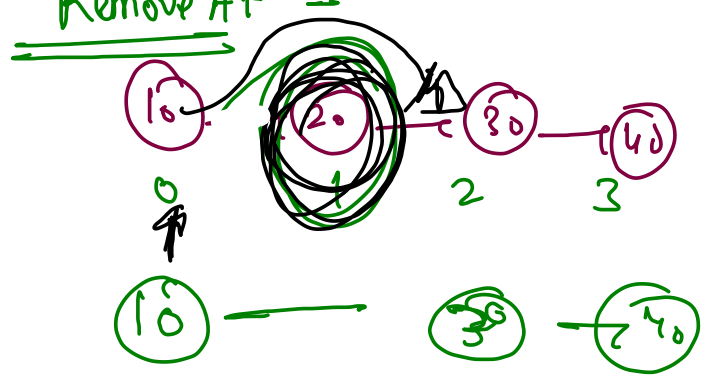
Second Last node



[size] available

size = 0

Remove At



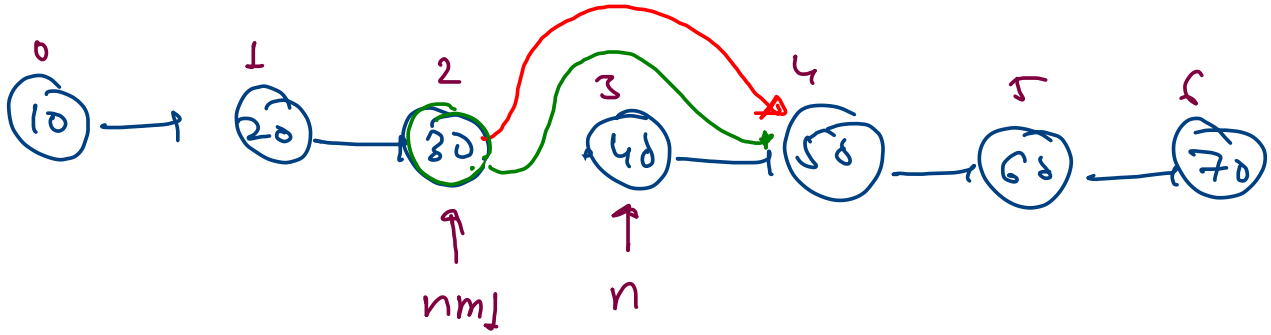
Size

Indx valid

obj.indx = 0 → remove first

indx == size - 1

remove last



removeAt(3)

```
Node nm1 = getNodeAt(indx-1);
Node n = nm1.next;
data = n.data;        // 40
// connection
nm1.next = n.next;
return data;
```

```
Node nm1 = getNodeAt(indx-1);
int data = nm1.next.data; // 40
nm1.next = nm1.next.next;
return n data;
```