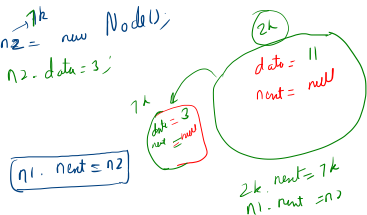


Memory Mapping

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

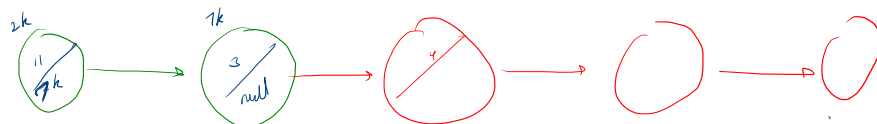
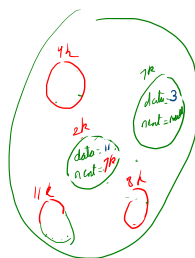
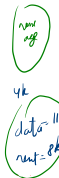
- 1) Node n1 = new Node();
n1.data = 11
- 2) Node n2 = new Node();
n2.data = 3;
- 3) n1.next = n2;

2k data = 11



Student &
str name
int age

s.name
s.age



linked list

```

class Node {
    int data;
    Node next;

    public Node() {
    }

    public Node(int data) {
        this.data = data;
    }
}

public class Main {
    public static void main(String[] args) {
        Node n1 = new Node();
        n1.data = 11;

        Node n2 = new Node();
        n2.data = 3;

        n1.next = n2; // connecting first and second

        Node n3 = new Node(data: 5);
        n2.next = n3;

        System.out.println(n1.data);
        System.out.println(n1.next.data);
        System.out.println(n1.next.next.data);
    }
}
  
```

3k next data
7k data =

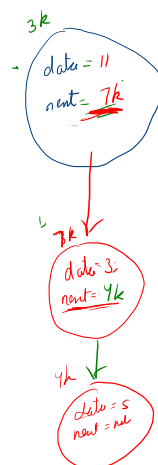
7k next data
4k data

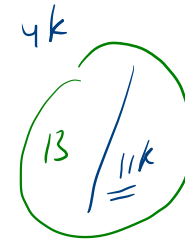
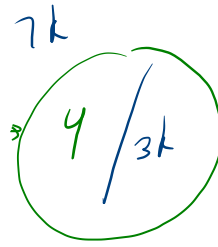
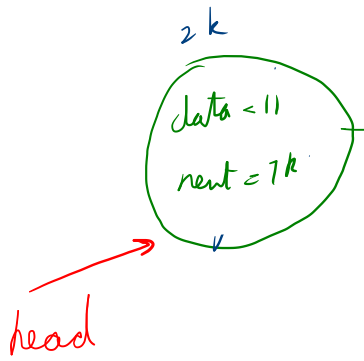
Stack

3k data

n1 = 3k
n2 = 7k
n3 = 4k

Heap

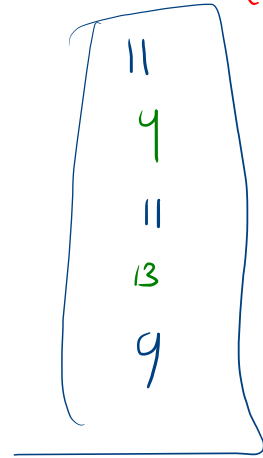




curr = 2k 7k 3k 4k 11k null

nextNode = 11k, next
→ null

curr = null

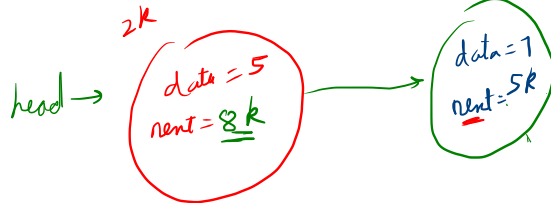


```
public static void printLinkedList(Node n1){
    Node curr = n1;

    while(curr!=null){
        1) System.out.println(curr.data);
        2) Node nextNode = curr.next;
        3) curr = nextNode;
    }
}
```

head = null 2k
 tail = null 2k 8k 5k

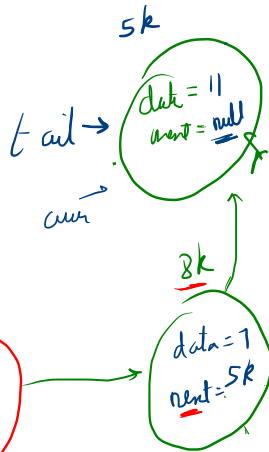
removeFirst();



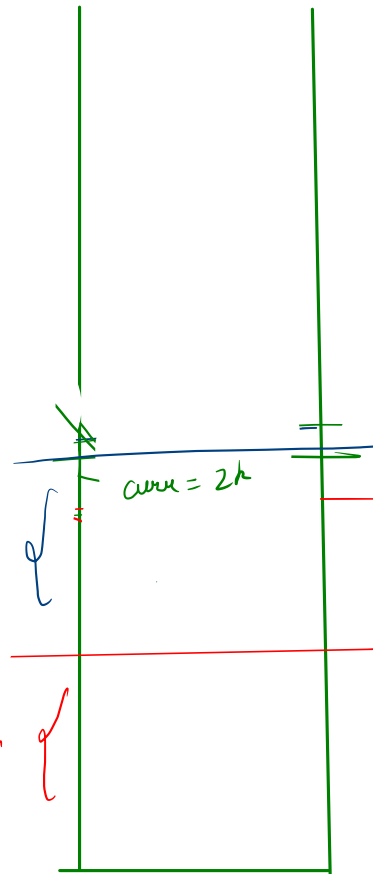
curr = 2k 8k
 next Node = curr.next
 → 5k, next
 → null;

curr = null

5
7
11



main



```

public class Main {
    static Node head;
    static Node tail;

    public static void addLast(int value){
        1) Node nn = new Node(value);

        2) if(head == null){
            head = nn;
            tail = nn;
        } else {
            3) tail.next = nn; (adding node at last)
            tail = nn; (moving tail)
        }
    }

    public static void printLinkedList(Node n1){
        Node curr = n1;

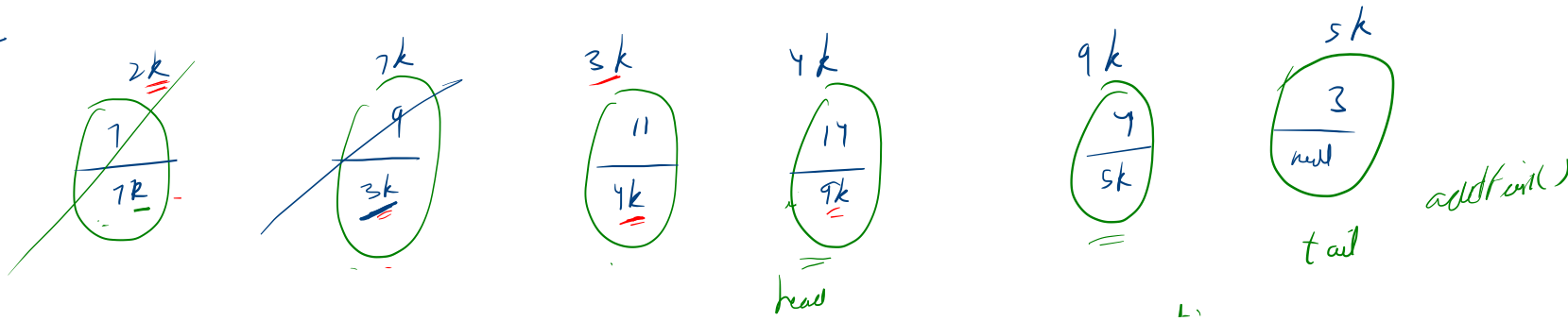
        while(curr != null){
            1) System.out.println(curr.data);
            2) Node nextNode = curr.next;
            3) curr = nextNode;
        }
    }

    Run | Debug
    public static void main(String[] args) {
        1) head = null;
        2) tail = null;

        3) addLast(value: 5);
        4) addLast(value: 7);
    }
  
```

add last (11) print (head)

removeFirst()



now head = 2k

```
public static void removeFirst(){
    if(head==null){
        System.out.println(x: "LinkedList is empty, not possible to remove first");
    } else {
        Node nextOfHead = head.next;
        head = nextOfHead;
    }
}
```

② addFirst() → add in front

① create linked list with user input