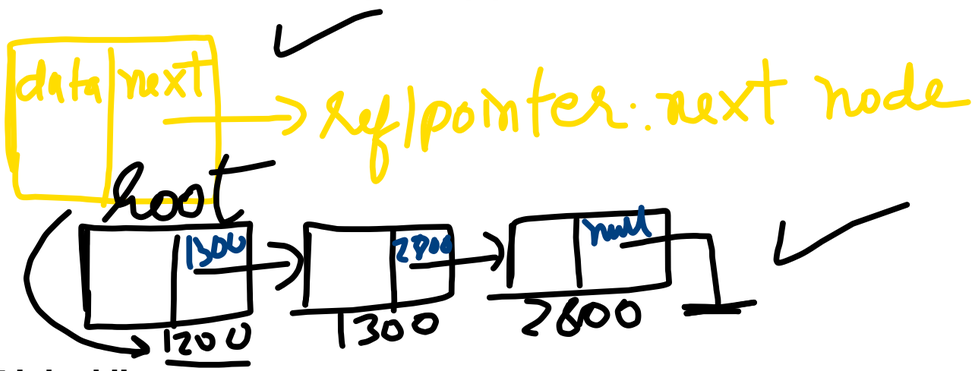
Linked List

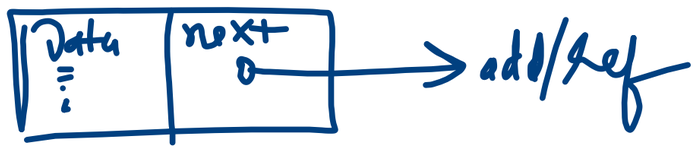
-collection of nodes in sequential manner.
-Node: strucutre that carry data(s) and link to next Node



Linked list:
-1st/left most is called root/head/start,it is the only node whos address will be remembered.
-all other nodes are ref using links to next.
-righmost/last has next as null

Linked List:
-dynamic linear strucutre
-most powerful as can implement all 5 DS using itself
linked list can create:stack,queue,tree,graph

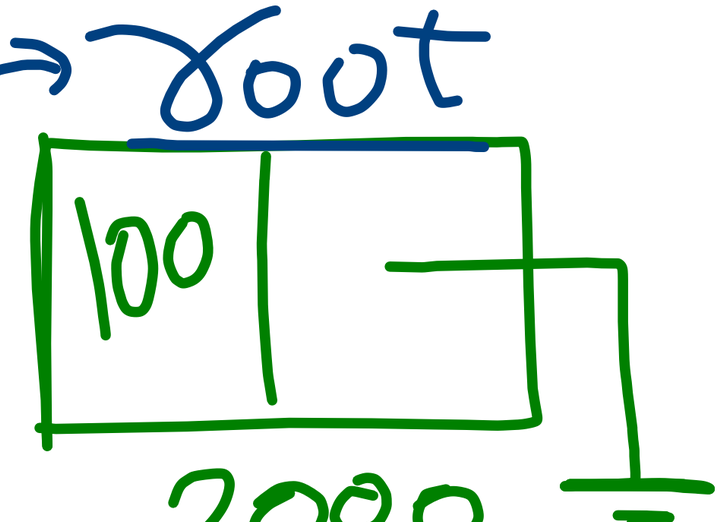
Node:

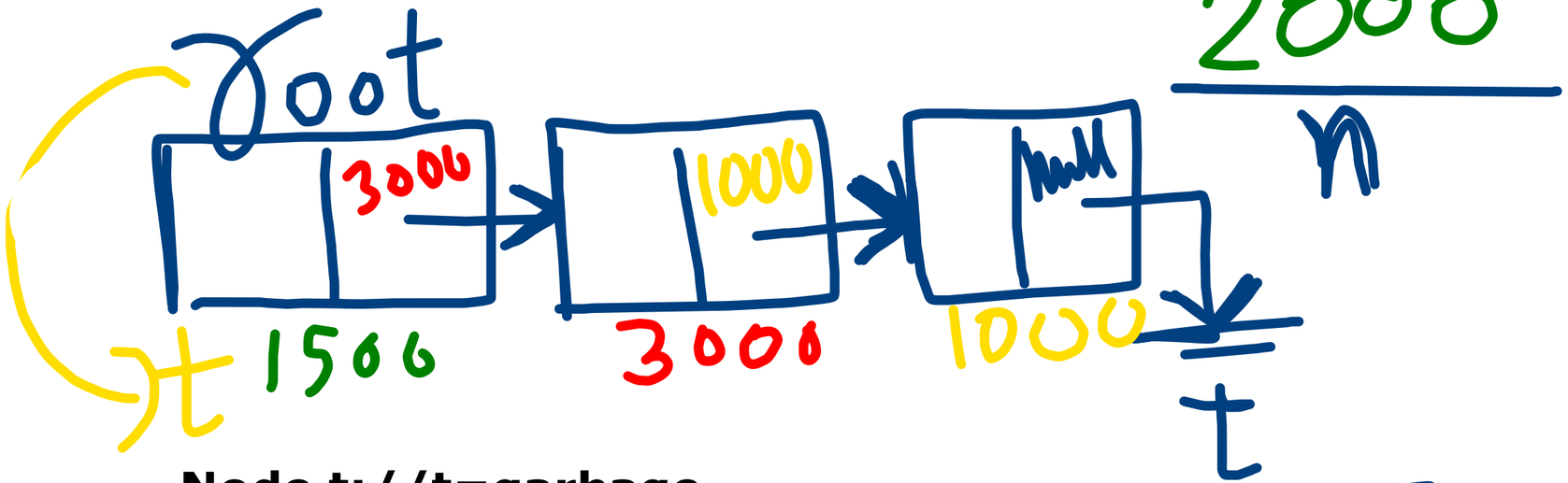



```
class Node
{
  int data;
  Node next;//self ref:ref refering to own type
  Node(int data)
  {
    this.data=data;
    this.next=null;
  }
  }
```

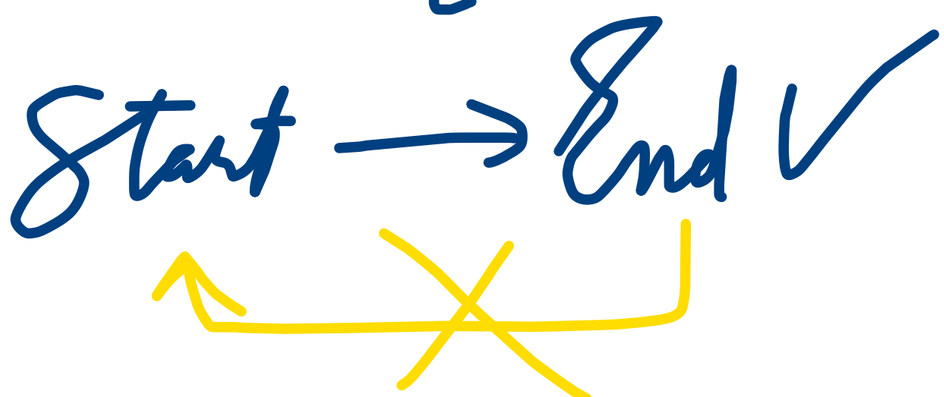
Node n=new Node(100);

root=n;
root<---->2000



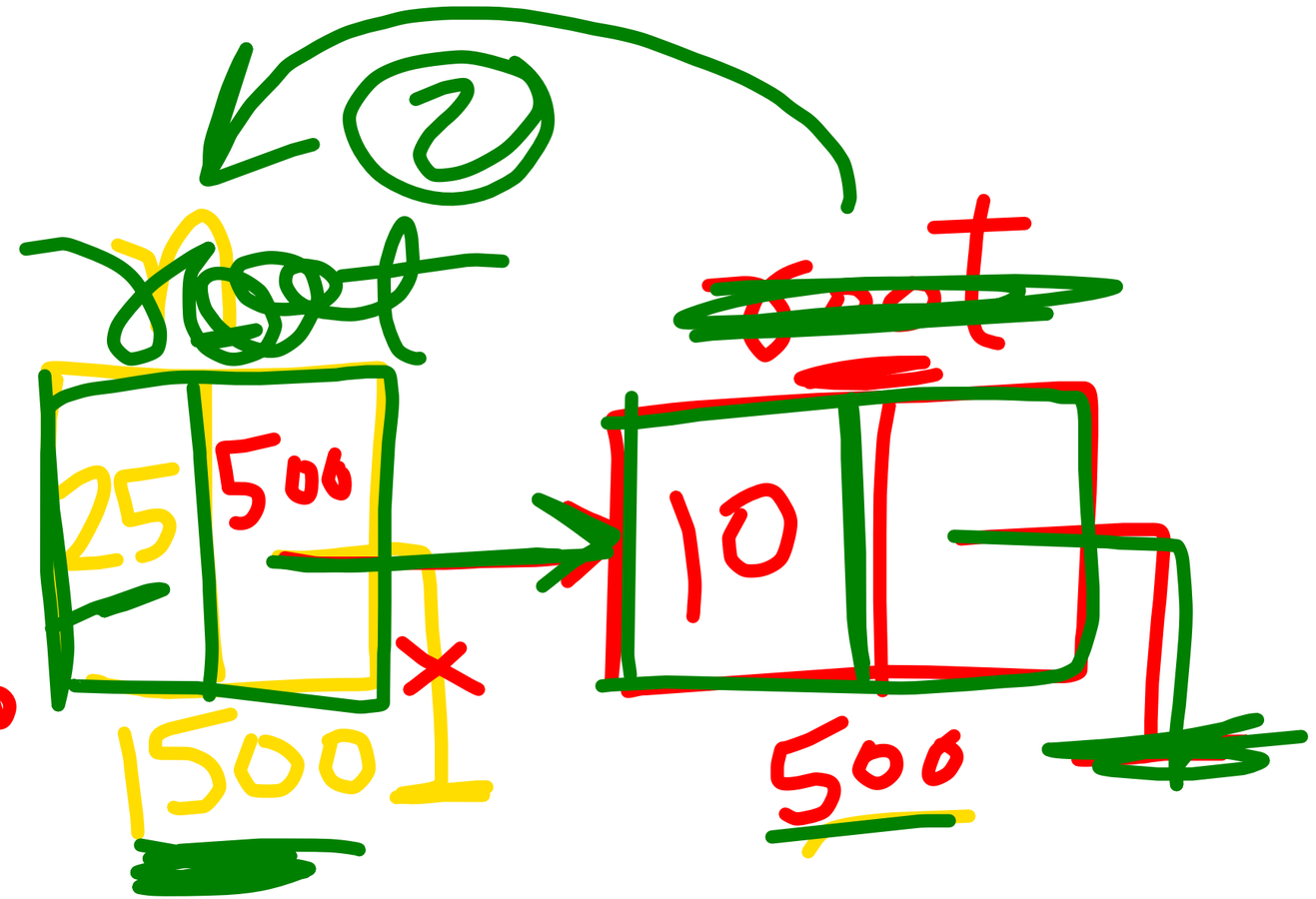


Node t;//t=garbage
t=root;//t=1500
t=t.next;//t=3000

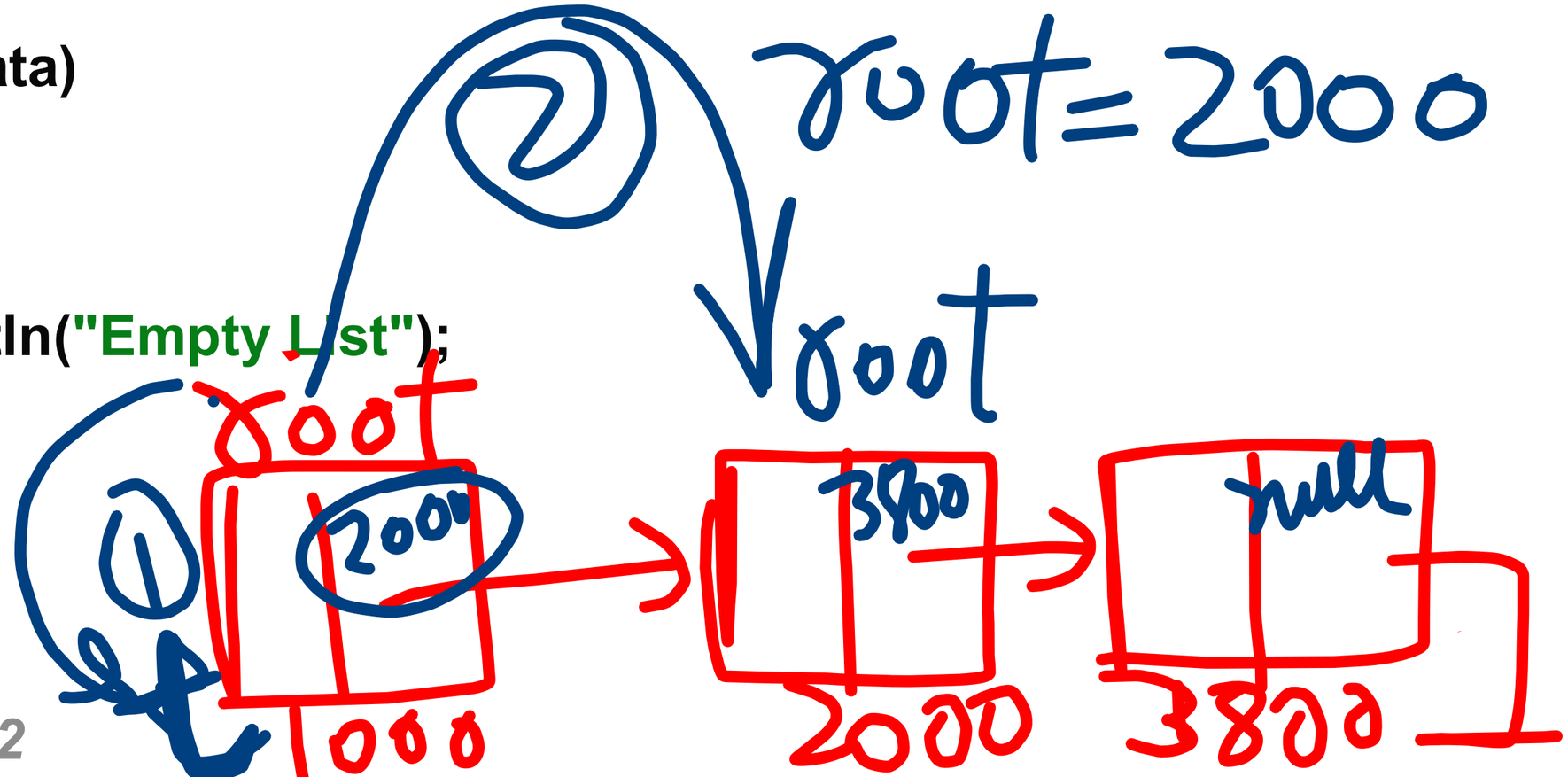


```
void inser_left(int data)
{
    Node n=new Node(data);
    if(root==null)
    {
        root=n;//n becomes 1st so root
    }
    else
    {
        n.next=root;//1
        root=n;//2
    }
    System.out.println(root.data+" inserted");

}
```

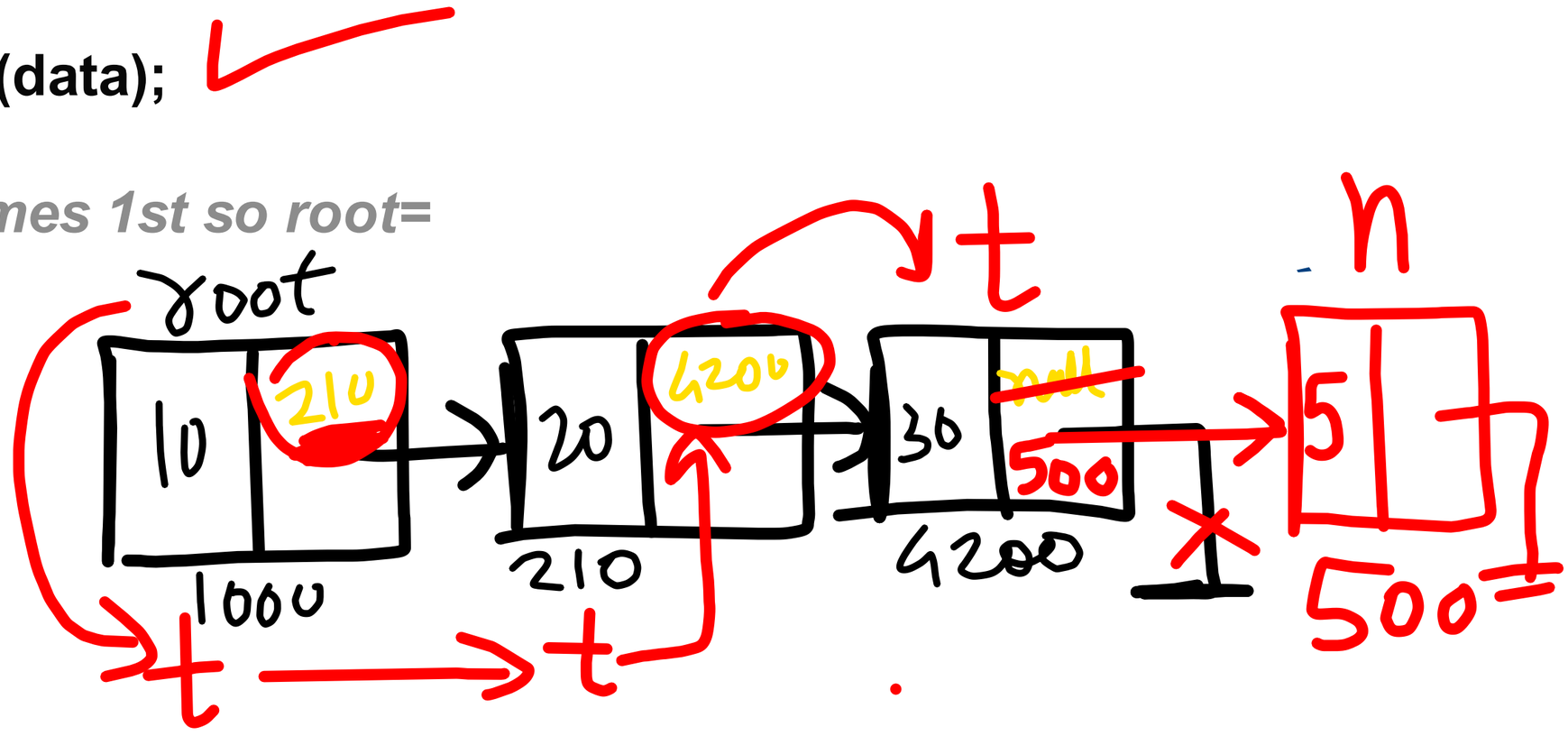


```
void delete_left(int data)
{
    if(root==null)
    {
        System.out.println("Empty List");
    }
    else
    {
        Node t;
        t=root;//1
        root=root.next;//2
        System.out.println(t.data+" deleted");
    }
}
```



```
void insert_right(int data)
{
    Node n=new Node(data);
    if(root==null)
        root=n;//n becomes 1st so root=
    else
    {
     Node t=root;
    while(t.next!=null)
     {    t=t.next;}
     t.next=n;
    }
    System.out.println(root.data+" inserted");

}
```



```
void delete_right()
{
    if(root==null)
        System.out.println("Empty List");
    else
    {
        Node t,t2;
        t=t2=root; ///
        while(t.next!=null)
        {
            t2=t;
            t=t.next;
        }
        if(t==root) //single node
            root=null; //reset root as only node left
        else
            t2.next = null;
        System.out.println(t.data + " deleted");
    }
}
```