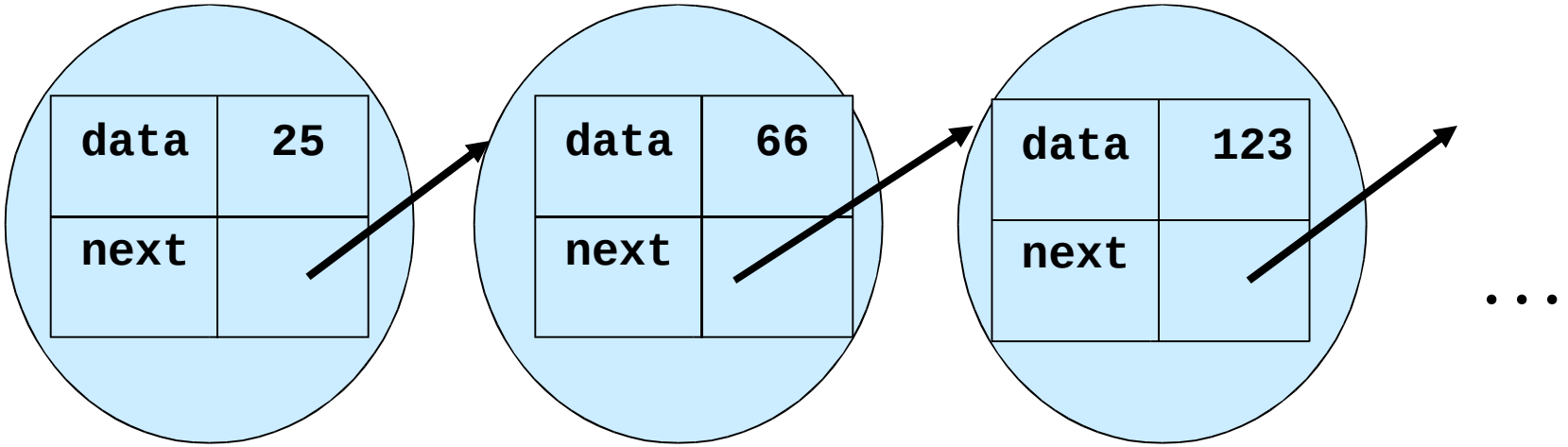
Linked Structures

- Self-referential classes can be used to create linked data structures:

```
class Node {
      private int  data;
      private Node next;
      public Node(int d, node n) {
            data = d;
            next = n;
      }
```

- **`next`** holds a reference to a Node object
- through the next reference, can link **`Node`** objects together:

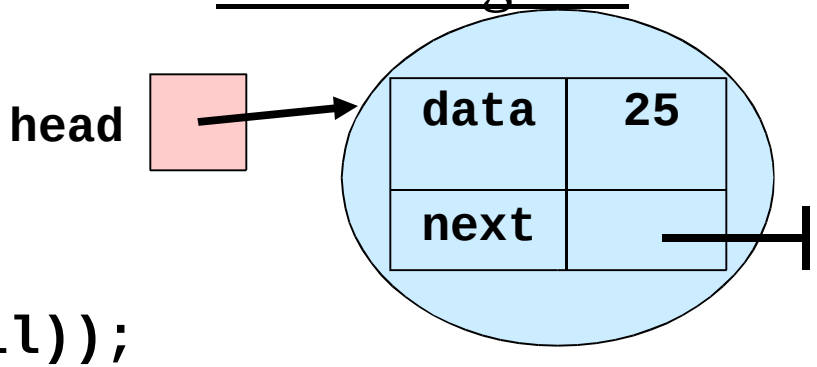


Linked List

- Ordered Collection of data
- Need a single variable which is pointer to 1st node on list
- Nodes are linked together in-order by following **`next`** references

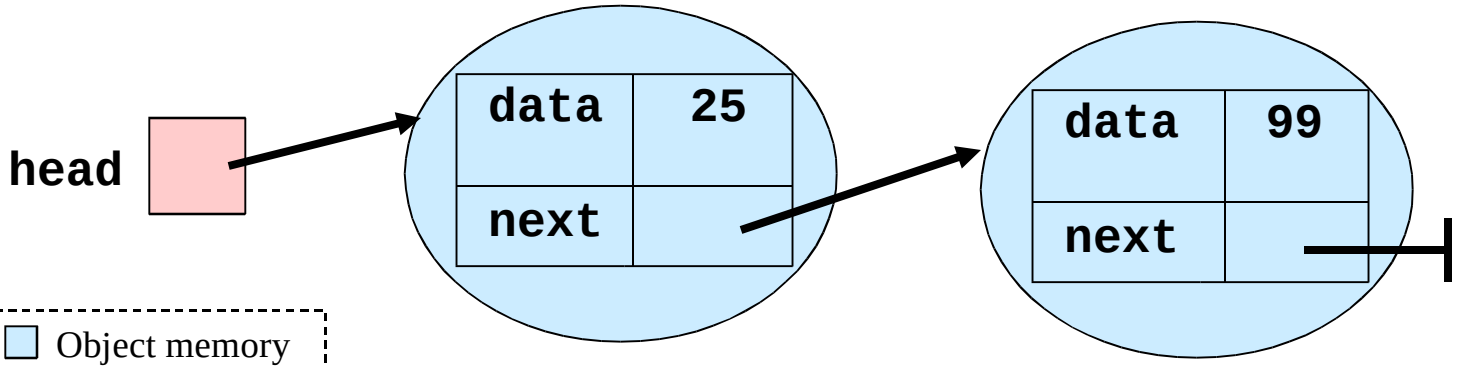
```
Node head;
head = new Node(25, null);
```

List of length 1:




```
head.setNext(new Node(99, null));
```

List of length 2:



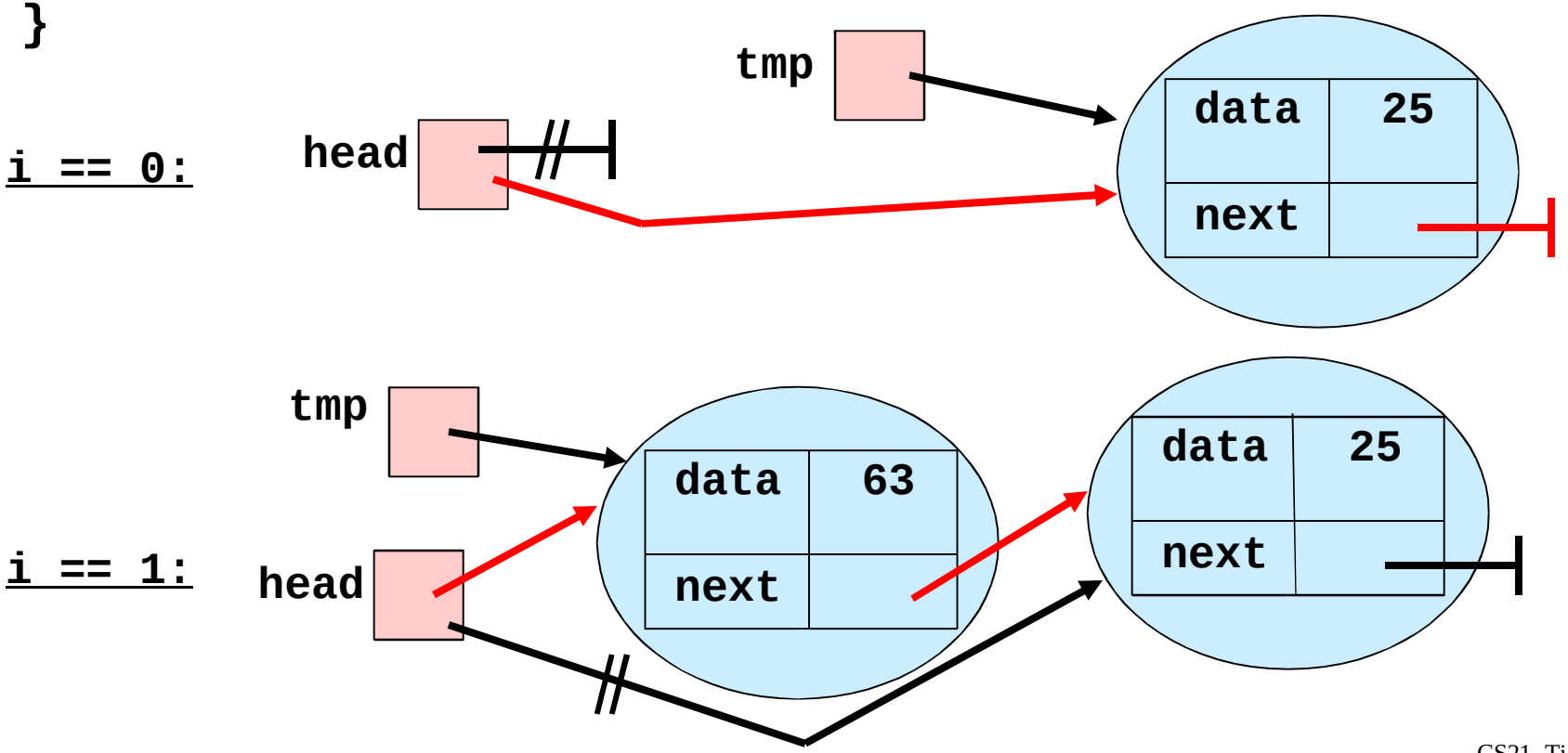
Stack memory Object memory

Operations on a List

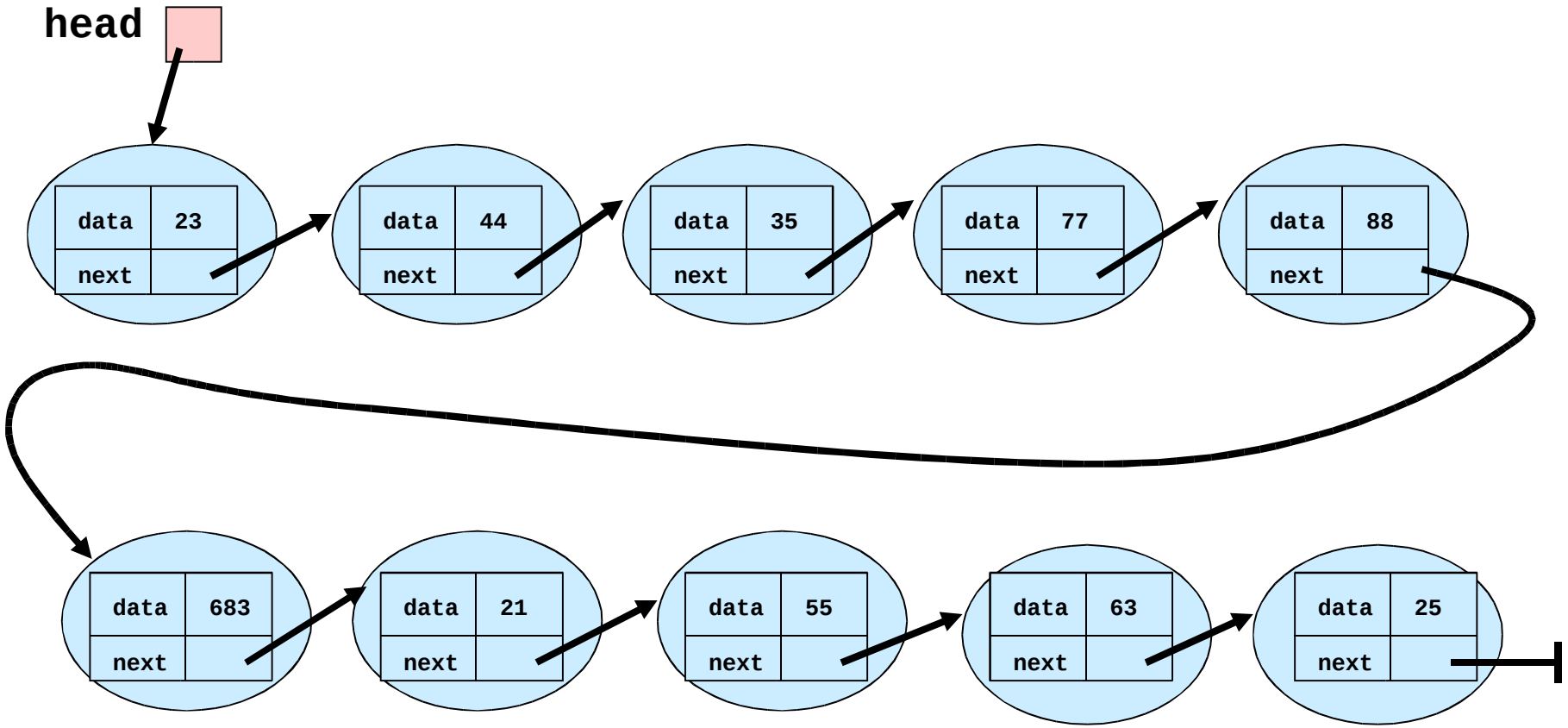
- All start at Node referred to by head reference, and traverse next references to access other nodes in the list
- Accessing the ith node is O(n):
  - first access head Node, follow its next reference to access the 2nd Node, follow its next reference to access the 3rd Node, and so on

Insert at Head of List

```
head = null;
for(i=0; i < 10; i++) {
      int val = reader.nextInt();
      tmp = new Node(val, null);
      tmp.setNext(head);
      head = tmp;
}
```

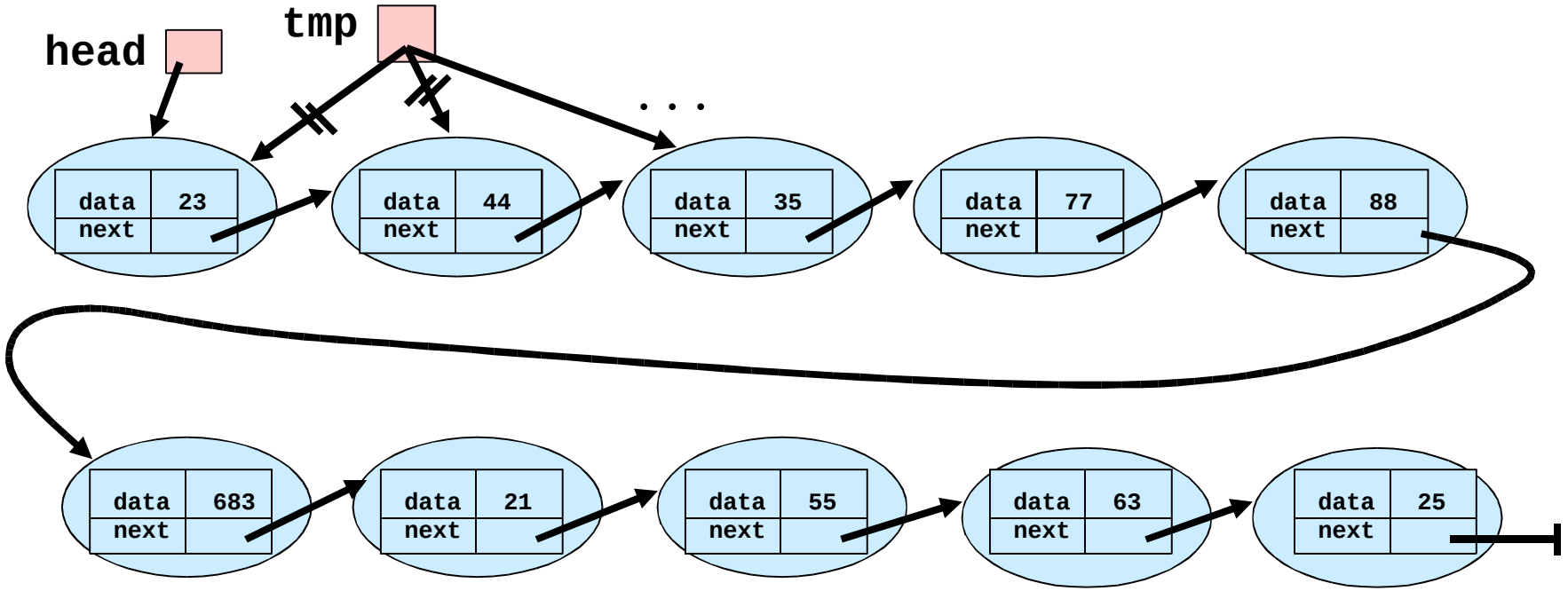


Resulting List of 10 nodes:



Traverse the List

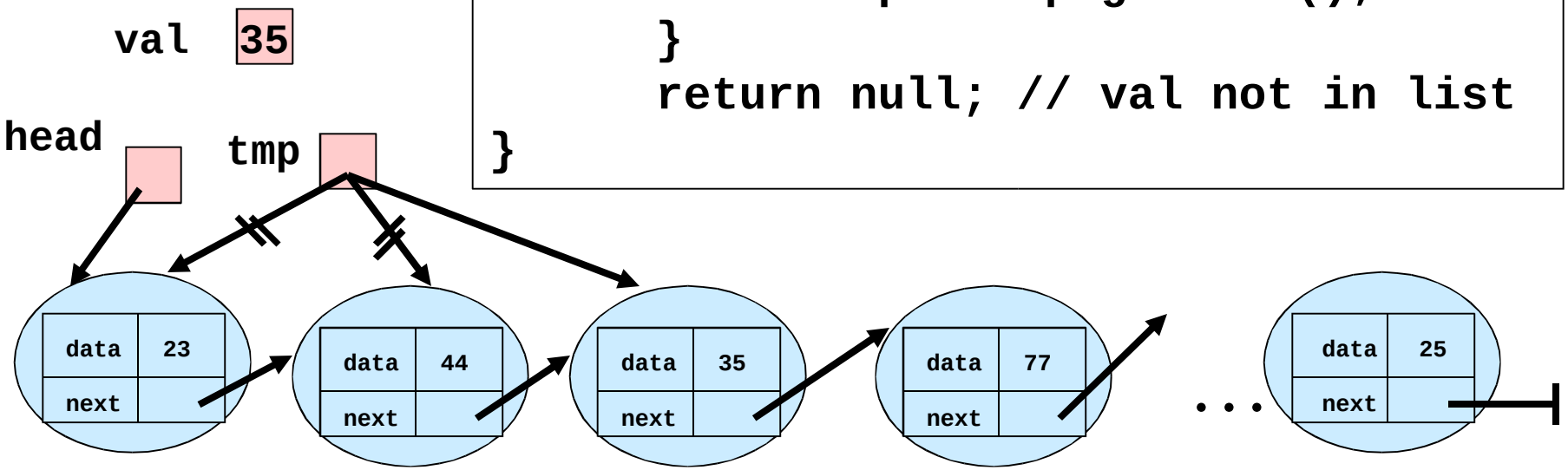
```
tmp = head;    // start at the 1st node
while(tmp != null) {
      System.out.print(tmp.getData() + " ");
      tmp = tmp.getNext(); // make tmp ref to next node
}
// output:  23 44 35 77 88 683 21 55 63 25
```



Find Element In List

- Start at head Node, compare search value to data field
- traverse next refs until matching data field is found, or until no more list

```
Public node Find(Node head, int val) {
        Node tmp;
        tmp = head;
        while(tmp != null) {
                if(tmp.getData() == val)
                        return tmp;
                tmp = tmp.getNext();
        }
        return null; // val not in list
}
```



Insert in the middle

```
Node new_node, tmp;

new_node = new Node(20, null);
tmp = head.getNext();  // lets just make tmp point
                       // to some Node after head

// insert new_node after tmp
new_node.setNext(tmp.getNext());
tmp.setNext(new_node);
```