CYDEO

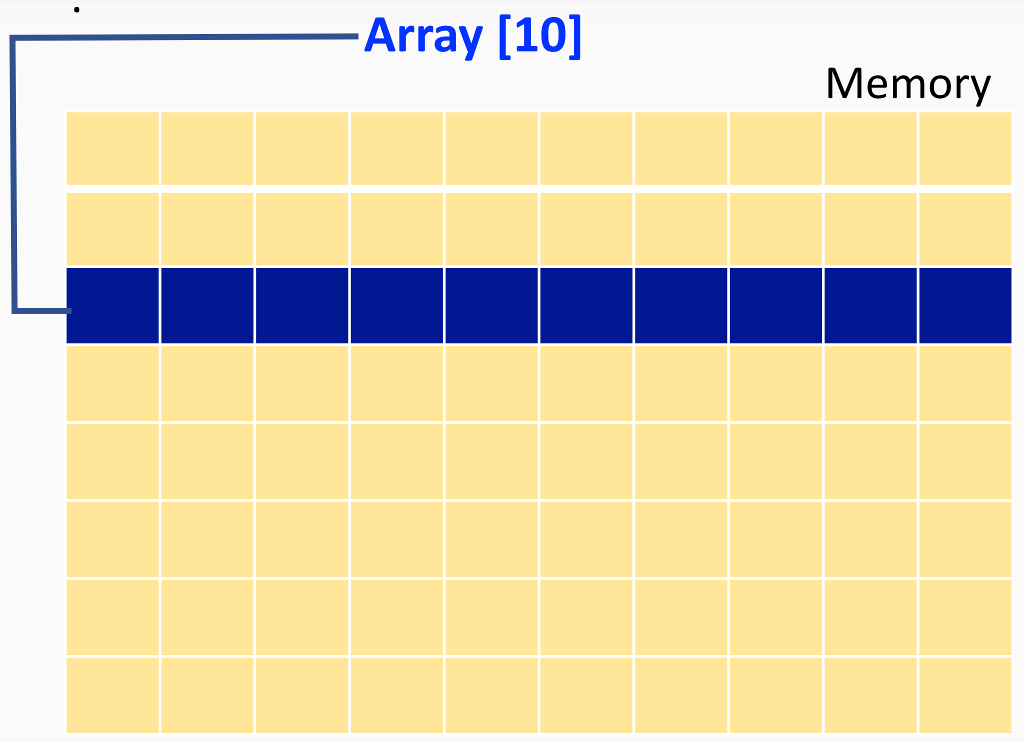
Data Structures and Algorithms Course

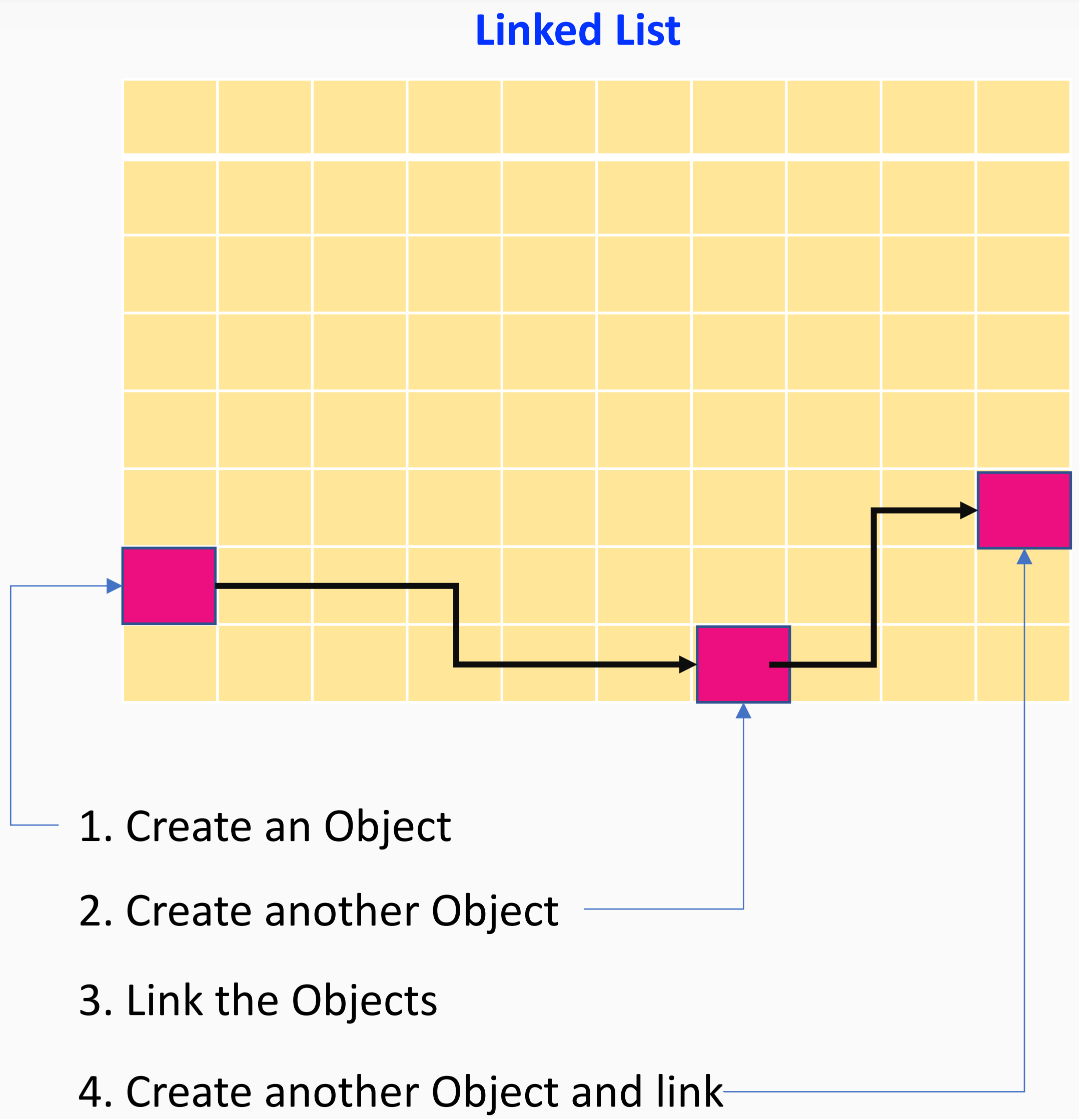
Linked Lists - Review

Agenda

- Recap of Linked Lists
- User List Example and Discussion of Linked Lists in Detail
    - Node Creation
    - LinkedList Class Creation
    - Insertion
    - Iteration
    - Delete
    - Search

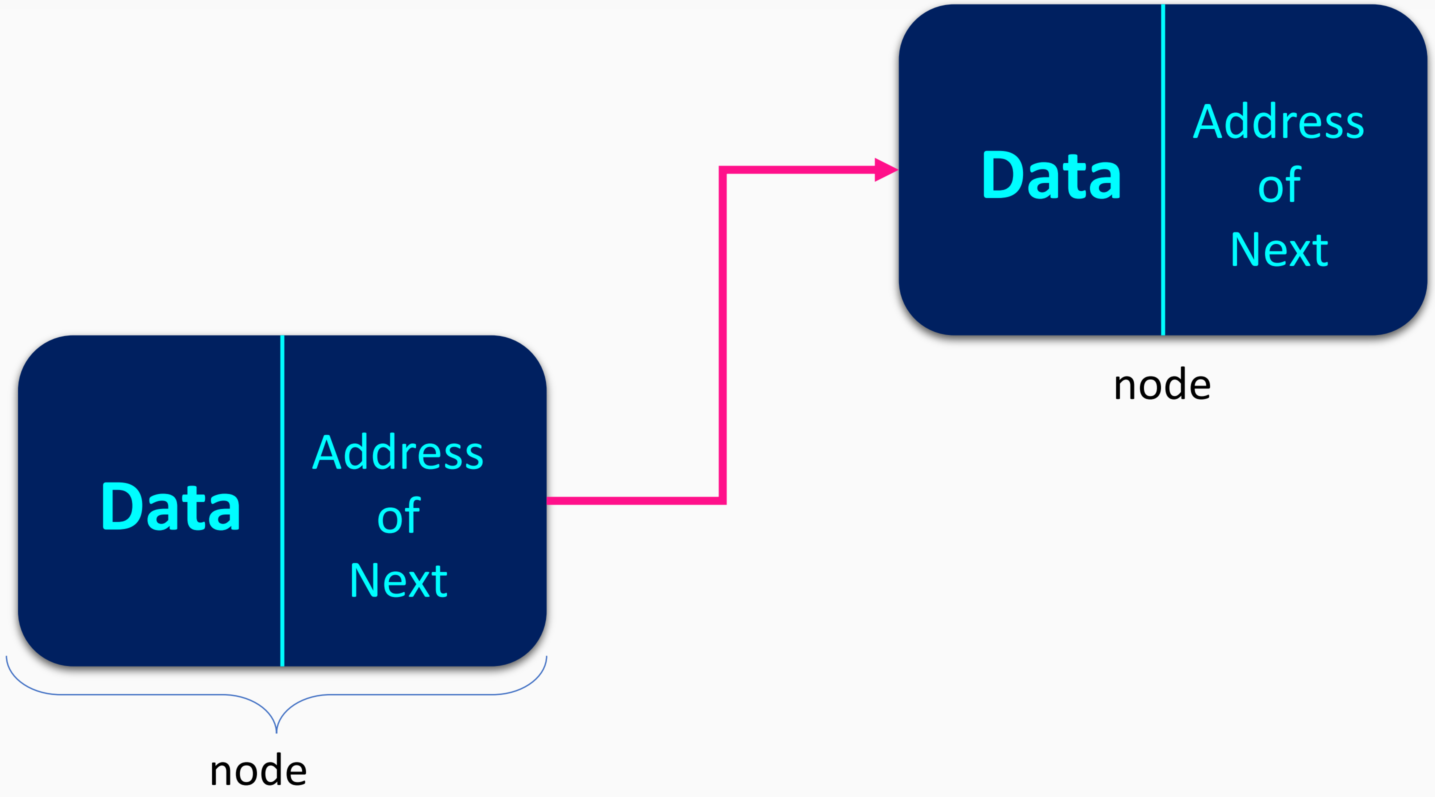
Linked Lists





1. Create an Object
2. Create another Object
3. Link the Objects
4. Create another Object and link

Node



Reference(*Address*) Types

Reference datatypes in java are those **which contains reference/address of dynamically created objects.** These are not predefined like primitive data types.

Following are the reference types in Java:

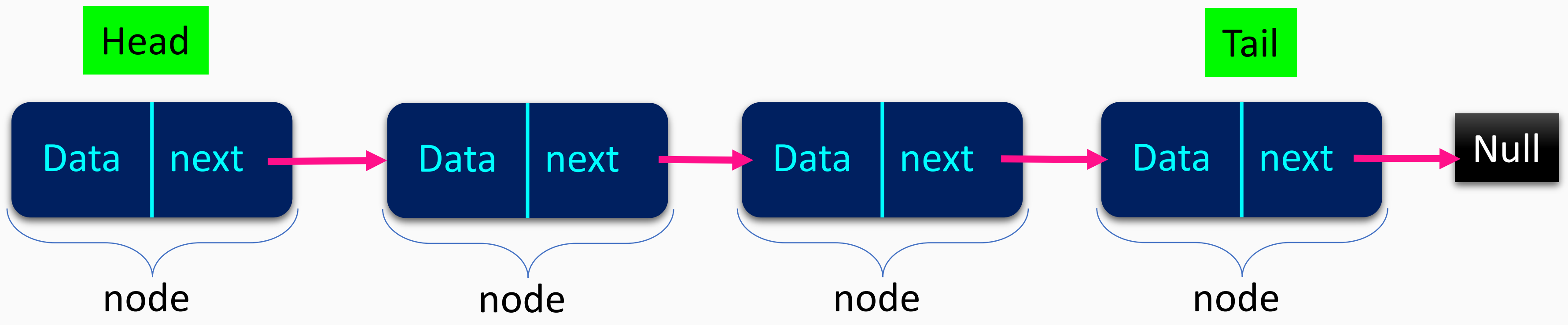
- **class types** − This reference type points to an object of a class.
- **array types** − This reference type points to an array.
- **interface types** − This reference type points to an object of a class which implements an interface.

Once we create a variable of these types (i.e. when we create an array or object, class or interface).

- These variables ***only store the address*** of these values.
- Default value of any reference variable is null.
- A reference variable can be used to refer any object of the declared type or any compatible type.

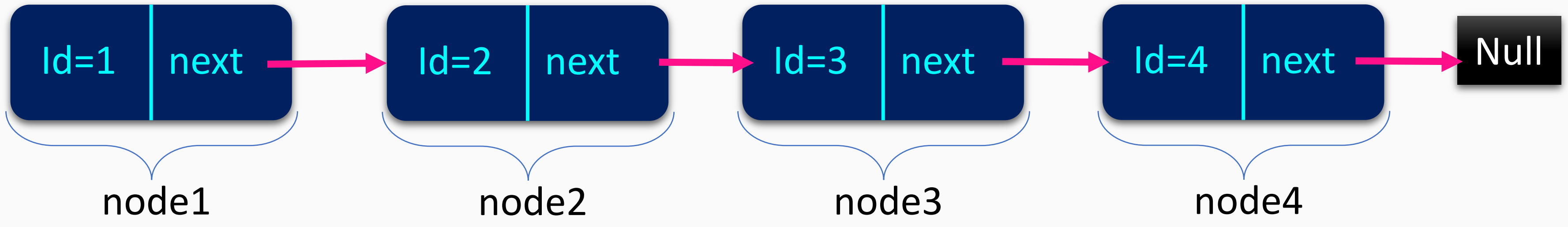
- Example *:* *User user= new User (“Roger”) ;* *What does new User() return?*

Singly Linked Lists

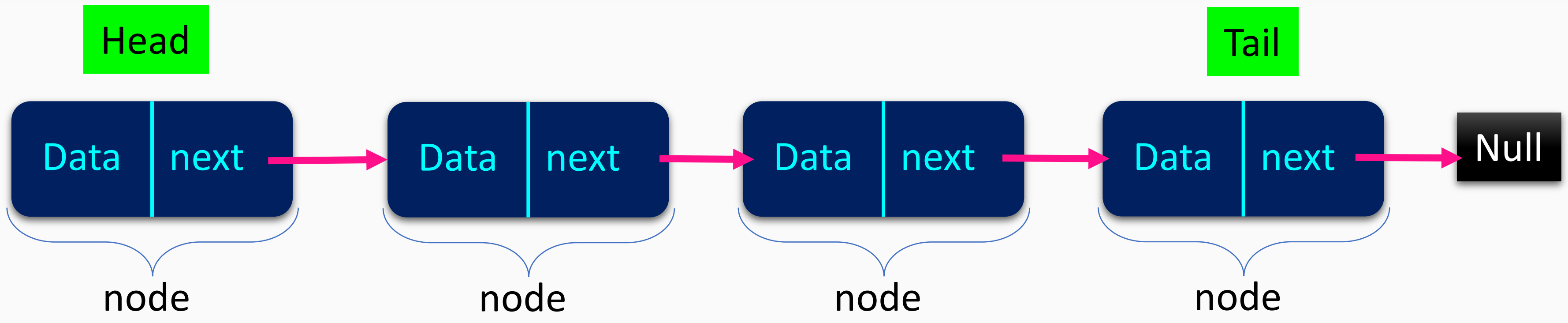


The addresses are not required to be in sequence !

Singly Linked Lists-Example



Singly Linked Lists-Iteration

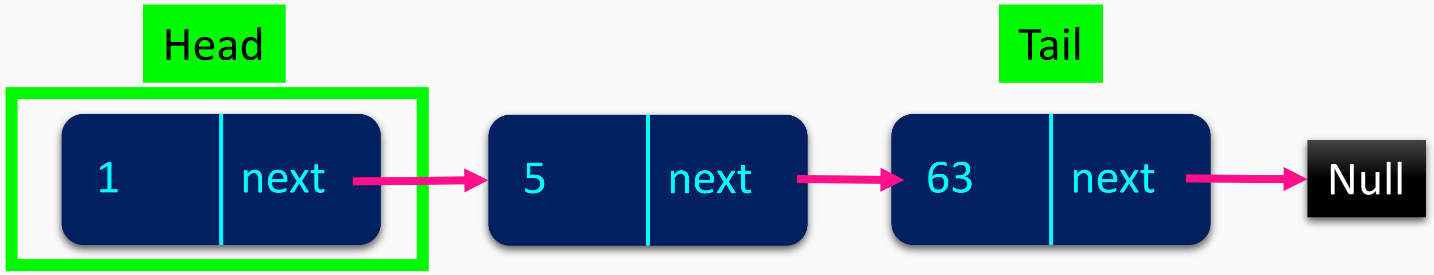


How can I iterate on a linked List

1. I need the starting point (Address of the Head Node)
2. I need a dummy node variable.(current)
3. Assign head address to current
4. While current is not "null" jump to next node by "current=current.next"

Singly Linked Lists-Iteration

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



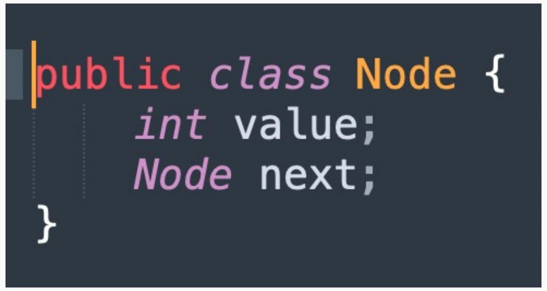
Output

1

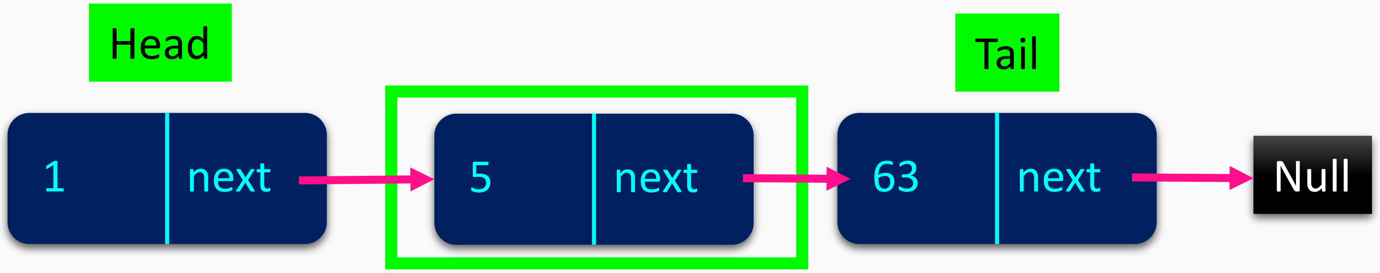
Iteration Code

```
current=Head;
 while (current!=Null) {
    System.out.println(current.value);
    current=current.next
 }
```

Singly Linked Lists-Iteration



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

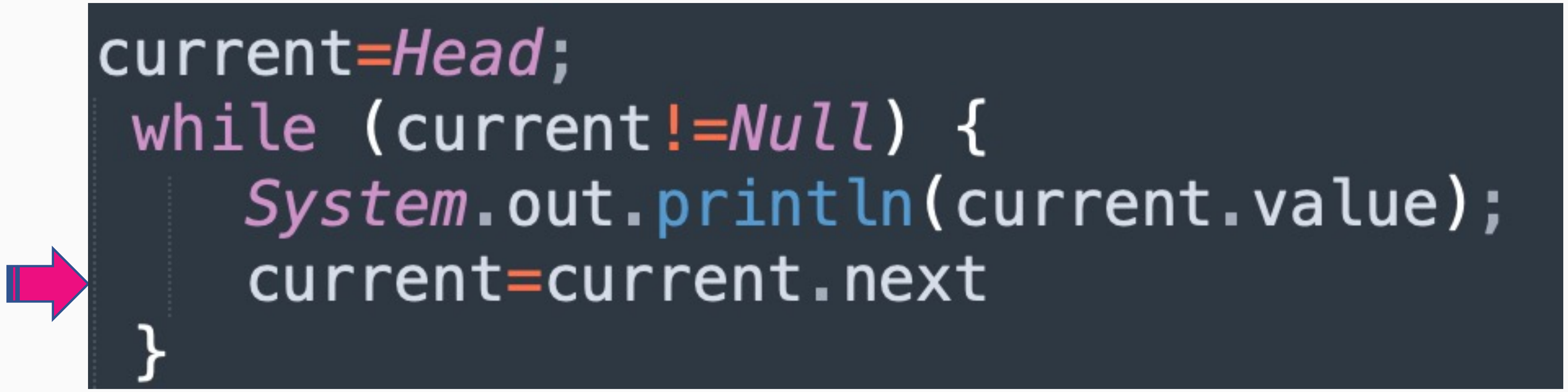


Output

1

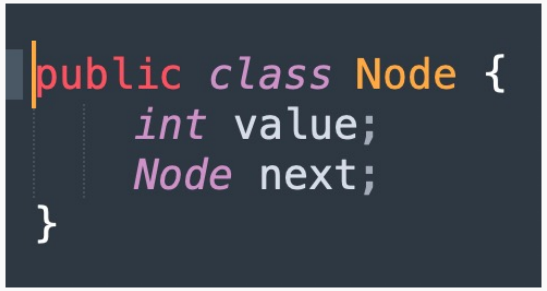
5

Iteration Code

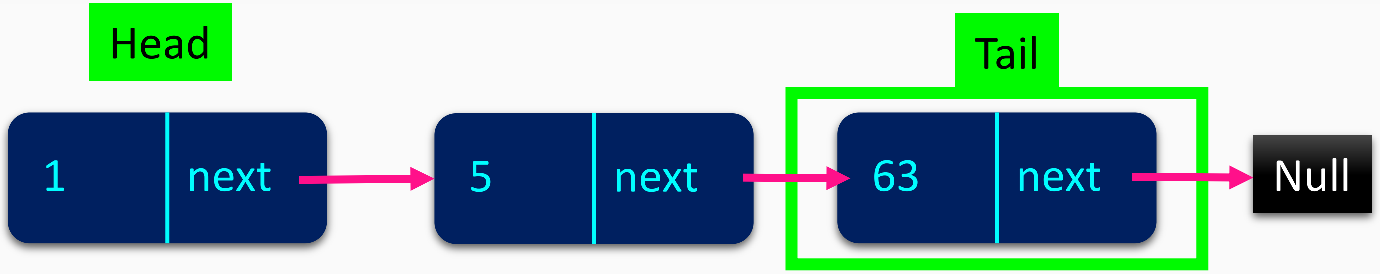


```
current=Head;
 while (current!=Null) {
     System.out.println(current.value);
     current=current.next
 }
```

Singly Linked Lists-Iteration



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



Output

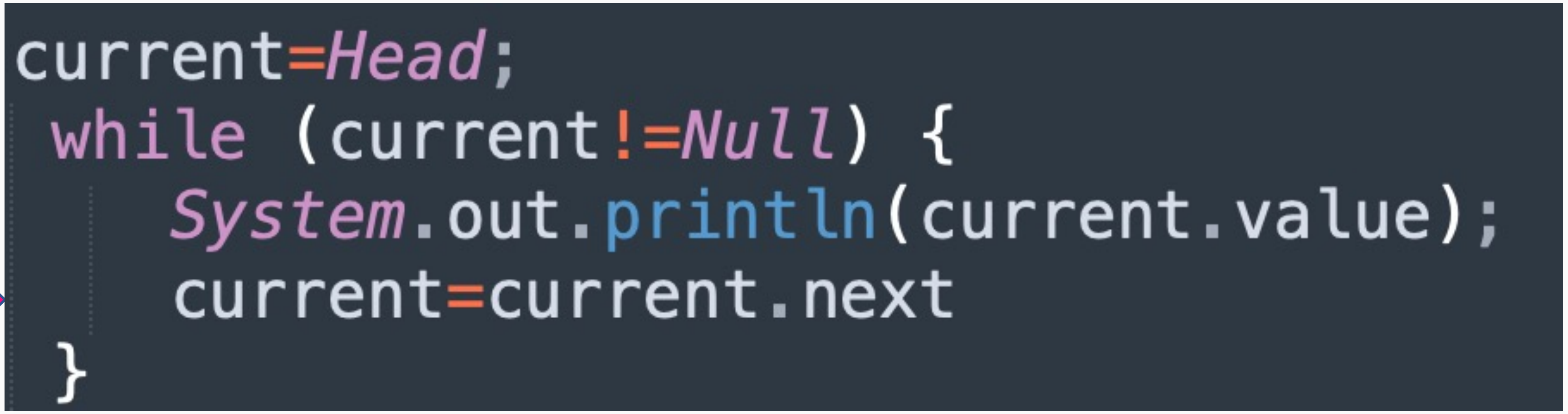
1

5

63

Exit while loop

Iteration Code

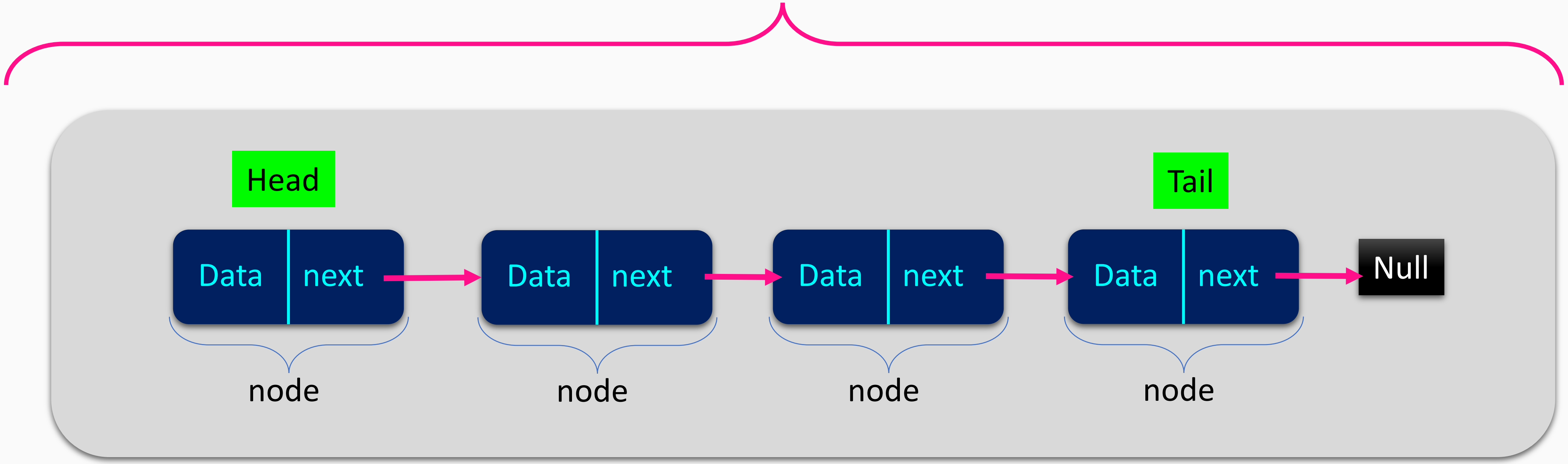


```
current=Head;
 while (current!=Null) {
    System.out.println(current.value);
    current=current.next
 }
```

How to build a custom Singly Linked List

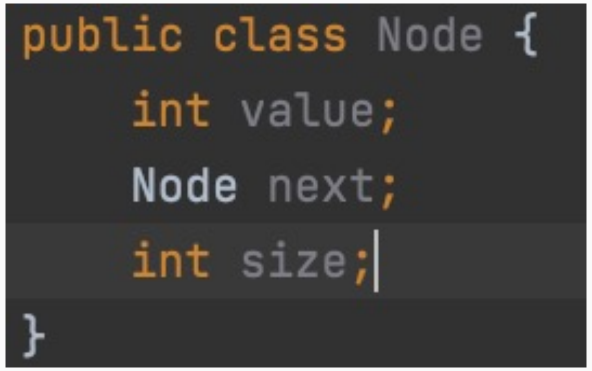
Class MySinglyList

{ Address of Head
Address of Tail
Size;
Methods like add, delete, indexOf}



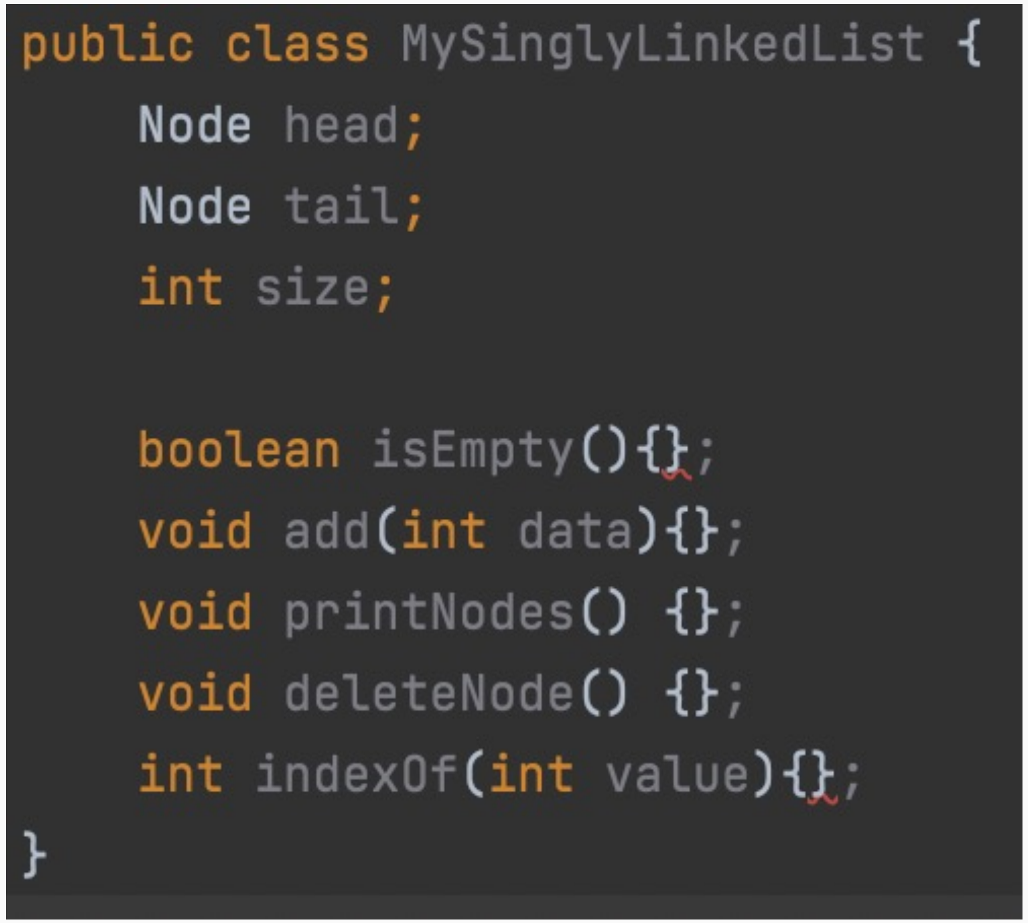
Implementation of Linked Lists

Node Class



```
public class Node {
    int value;
    Node next;
    int size;
}
```

Custom Linked List Class



```
public class MySinglyLinkedList {
    Node head;
    Node tail;
    int size;

    boolean isEmpty(){};
    void add(int data){};
    void printNodes() {};
    void deleteNode() {};
    int indexOf(int value){};
}
```

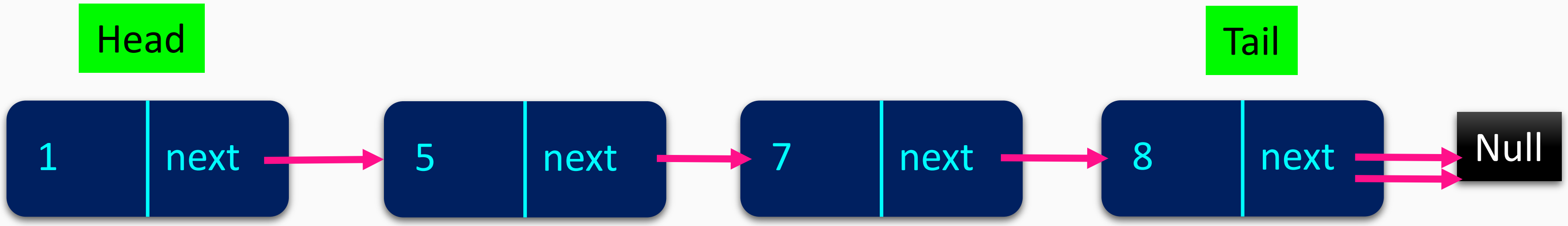
You have to implement :

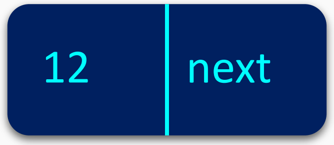
- Add() or Insert()
- Delete()

Nice to have :

- IndexOf()
- isEmpty()
- getsize();

How to insert a new node into a Singly Linked List



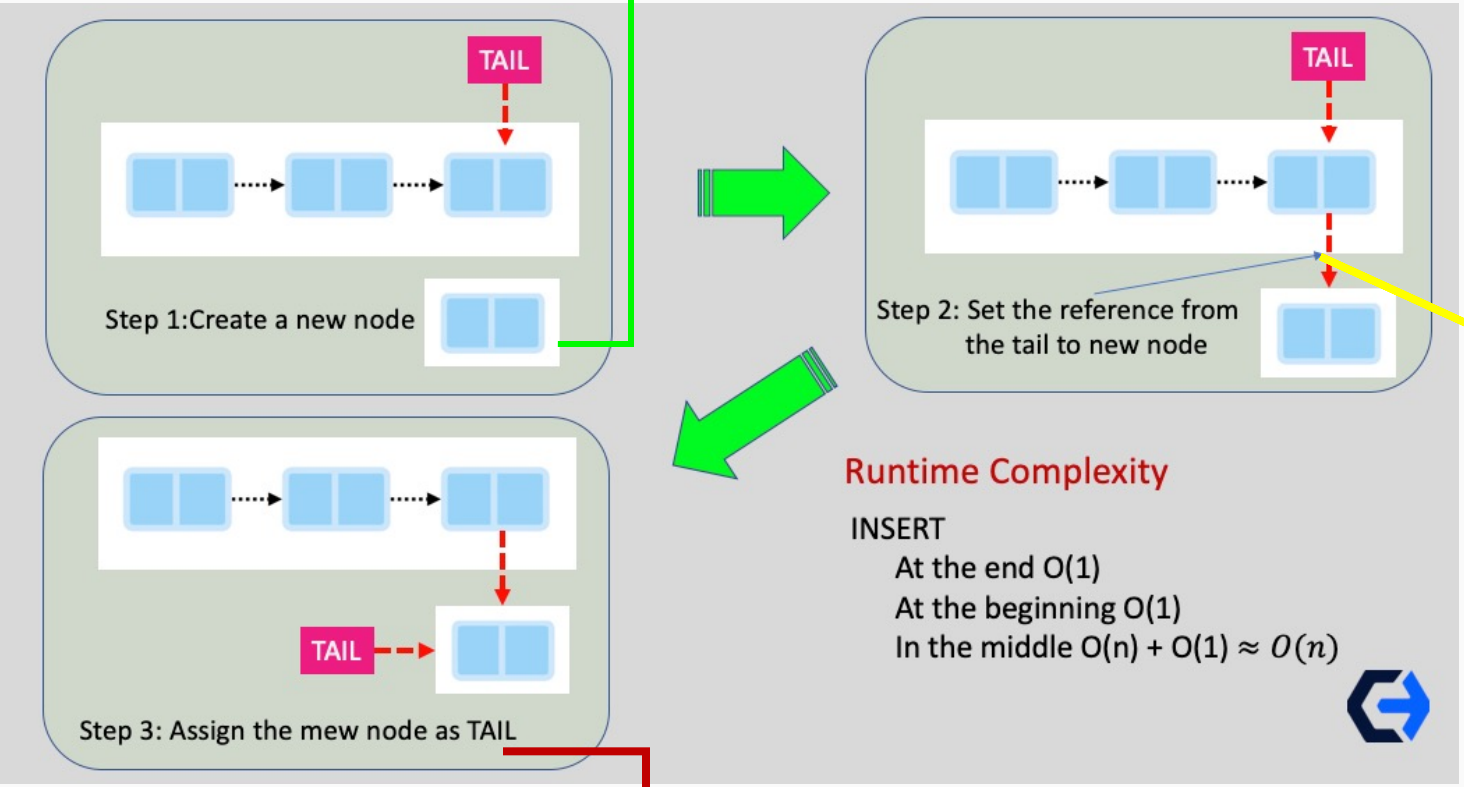


Node12

1. Tail.next= Node12;
2. Tail = Node12;

Insertion into Singly Linked Lists

Insertion


TAIL
Step 1:Create a new node
TAIL
Step 2: Set the reference from the tail to new node
TAIL
Step 3: Assign the mew node as TAIL

Runtime Complexity

INSERT

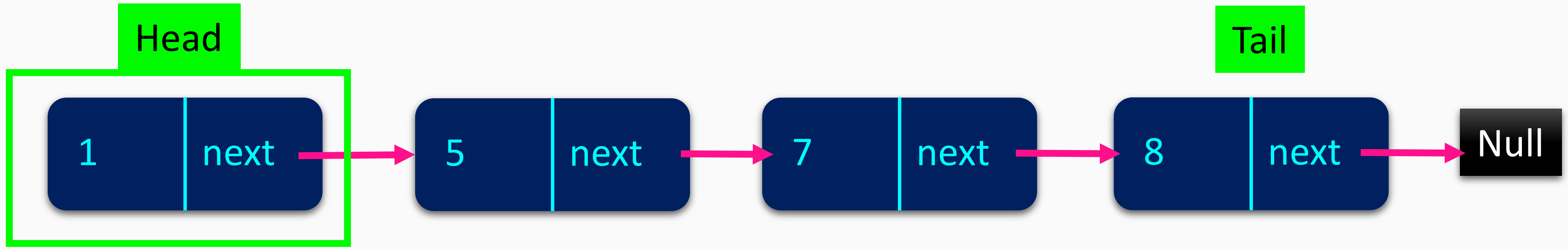
- At the end O(1)
- At the beginning O(1)
- In the middle O(n) + O(1) ≈ $O(n)$

Java Implementation

```java
public void Add(int item) {
   var node = new Node(item);

   if (isEmpty())
      head = tail = node;
   else {
      tail.next = node;
      tail = node;
   }
   size++;
}
```

Node node is equivalent to var node

Singly Linked Lists- Deletion (Case1 : Head)

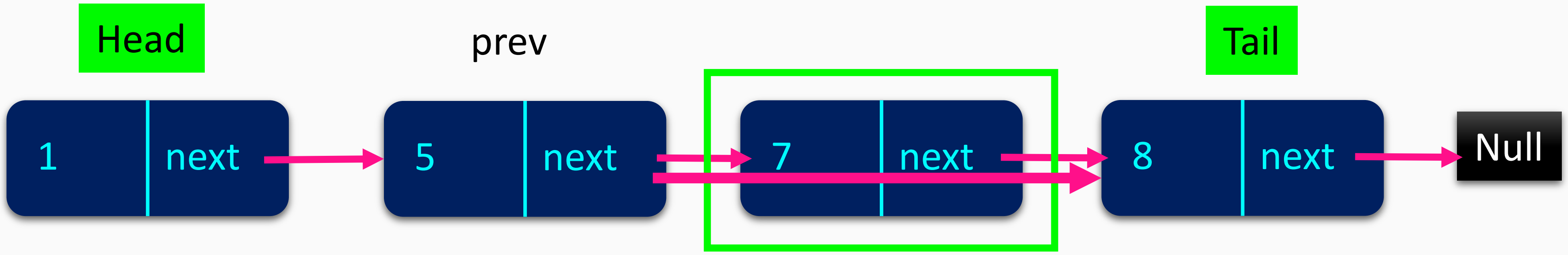


```
Head=current.next;
Current.next=null;
Size--;
```

Runtime Complexity

DELETE
At the beginning O(1)

Singly Linked Lists- Deletion (Case2 : Middle)



- Iterate to the Node you will delete
- Before jumping assign the current to a temp called prev
    - Prev=current;
    - Current=current.next;
- Assign prev.next with the Node.next

Runtime Complexity

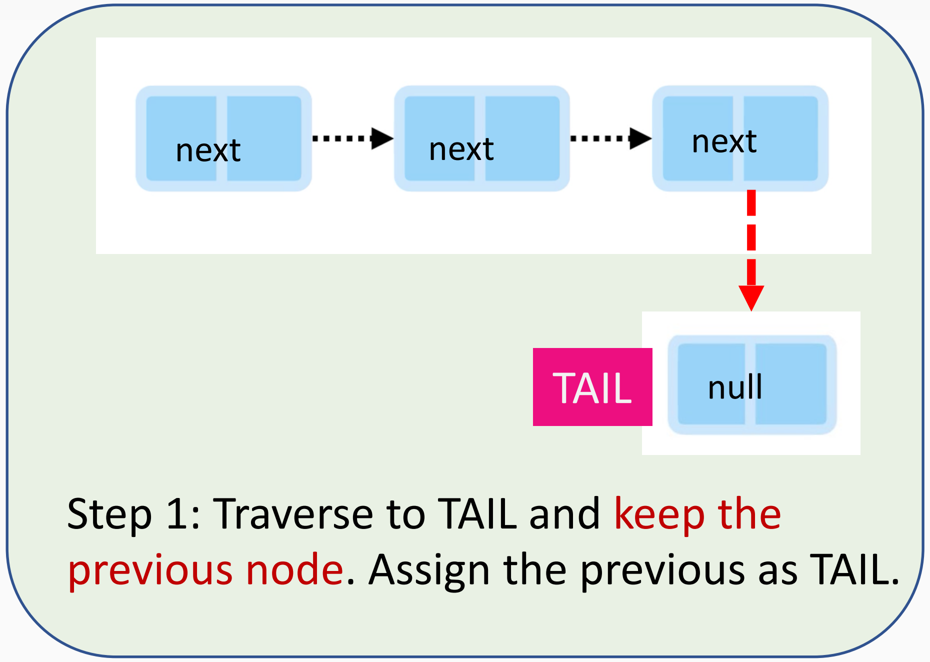
DELETE

At the end O(n) -> you have to traverse to node before the last one

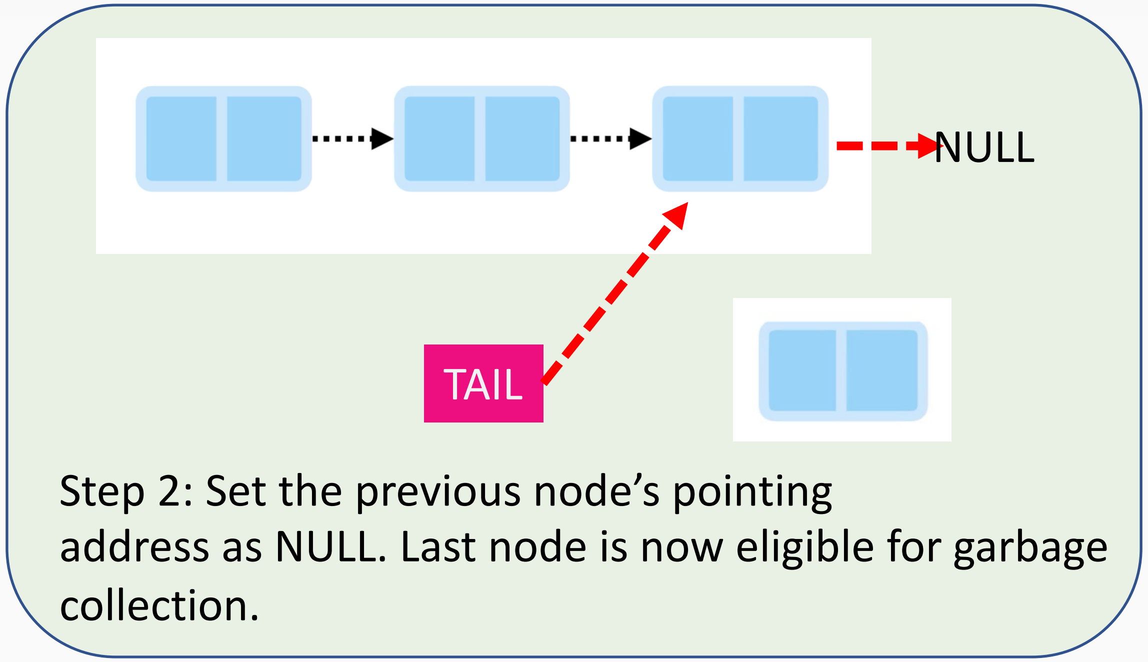
At the beginning O(1)

In the middle O(n)

Singly Linked Lists- Deletion (Case3 : At the End)



Step 1: Traverse to TAIL and keep the previous node. Assign the previous as TAIL.



Step 2: Set the previous node's pointing address as NULL. Last node is now eligible for garbage collection.

Runtime Complexity

DELETE

At the end O(n) -> you have to traverse to node before the last one

At the beginning O(1)

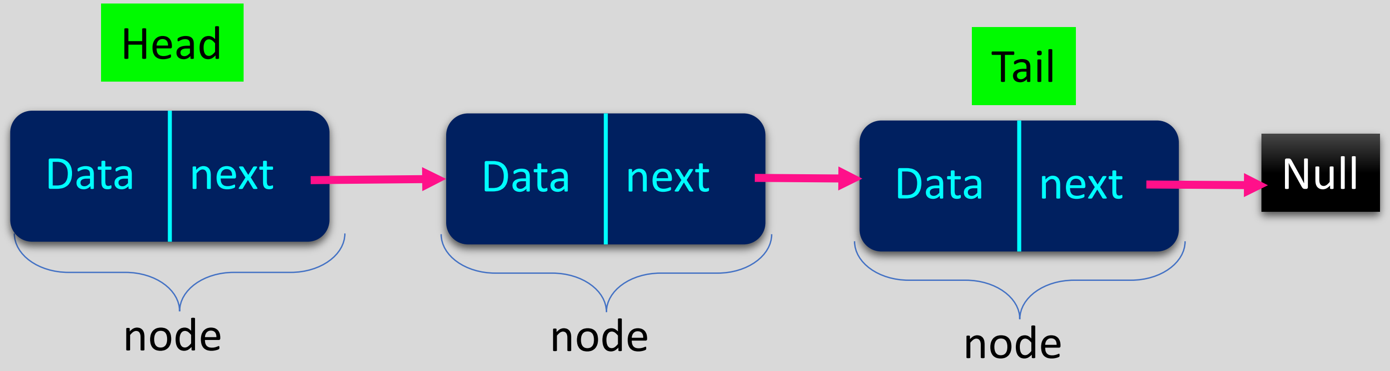
In the middle O(n)

Algorithm for Deletion (for 3 cases)

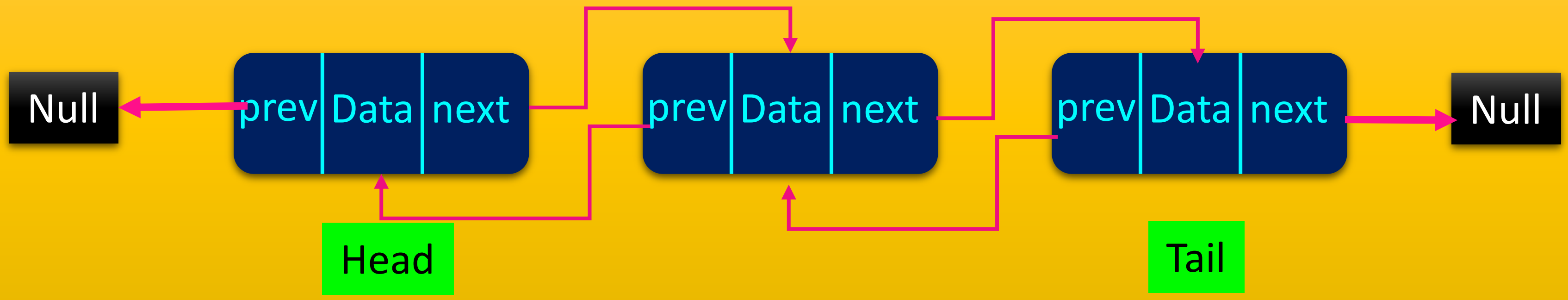
```
void deleteNode(int value) {
    if (isEmpty())  print message("List is empty!!!!!")

    Node current = head;
    Node prev = head;

    while (current != null) {
        if (current.value == value) {// if you find a match
                        // Case 1: current is head
                        // Case 2: current is tail
                        // Case 3: current is middle

                        // after deletion
                                size--;
            }
        //  move to other nodes
            prev = current;
            current = current.next;
        }
    }
```

Types Linked Lists

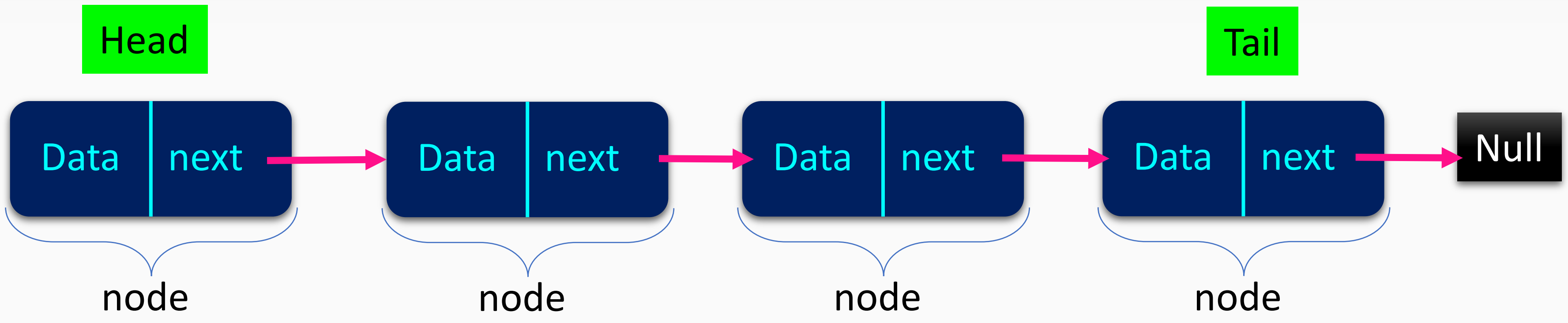
Singly Linked List



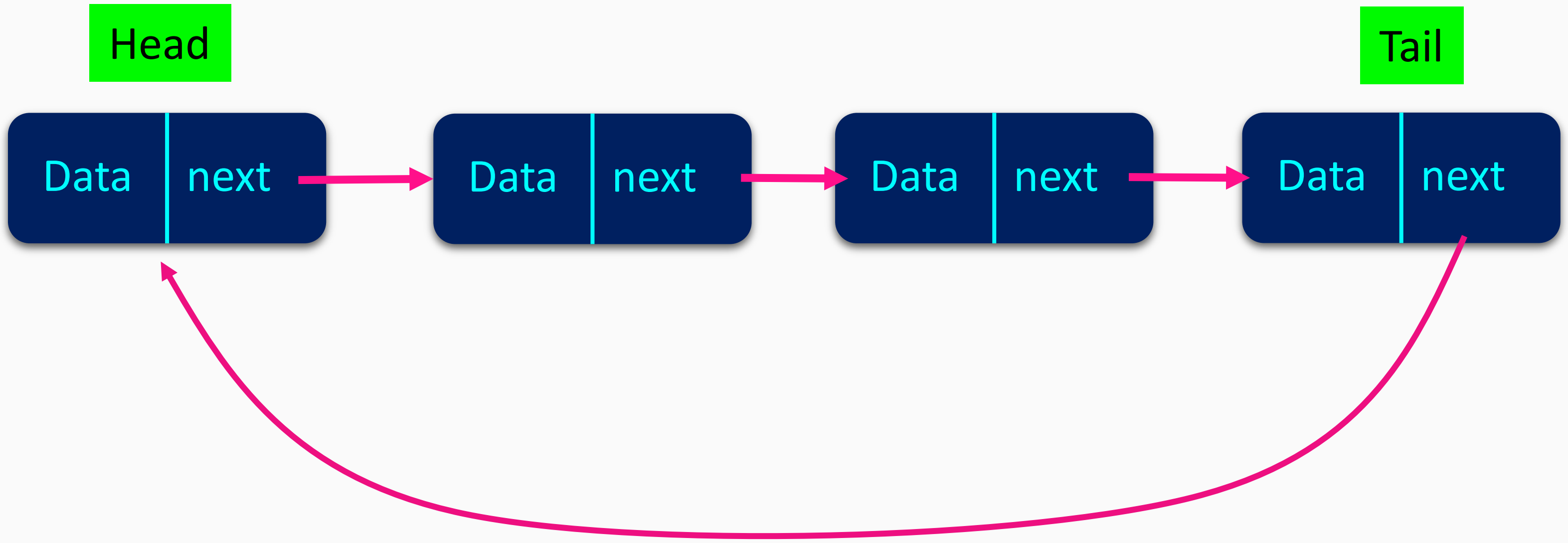
Doubly Linked List



Types Linked Lists

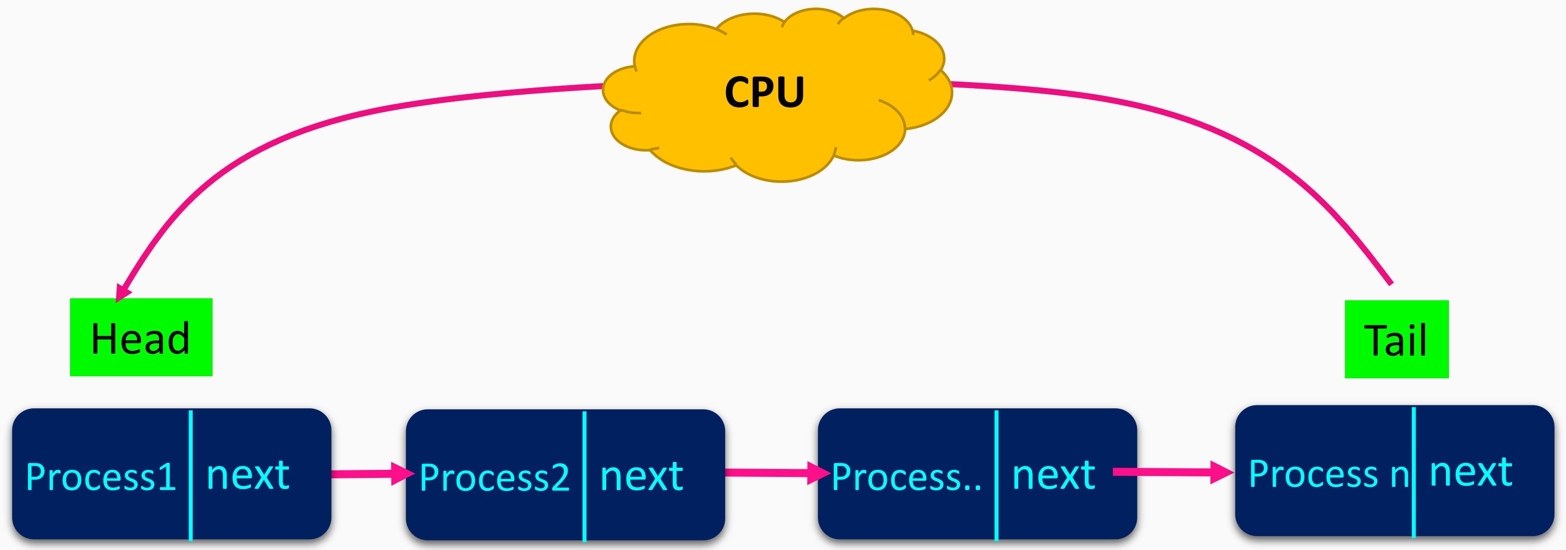


Circular Linked List



Circular Linked List

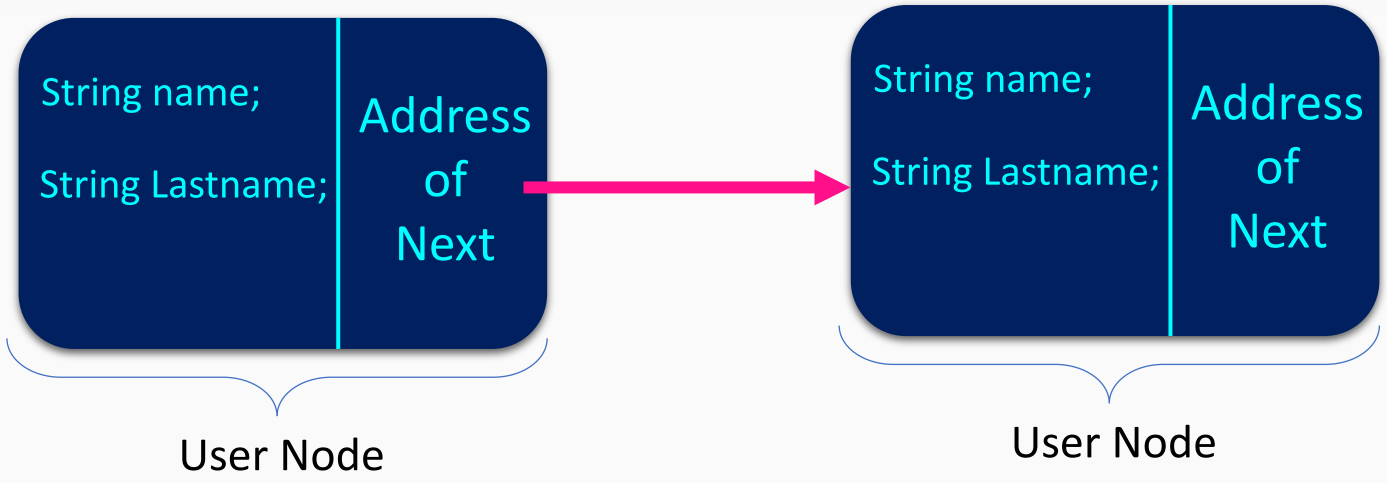
Round Robin Scheduling



Linked Lists Performance Analysis

Operation	Time Complexity
Add item to last	O(1)
Add item to first	O(1)
Add item in a position	O(n)
Delete last item	O(1)
Delete an item with value	O(n)
Access an index (indexOf())	O(n)
Look up (Search)	O(n)

Example : User Linked List



1. Create a User Node

2. Create a User Node List Class

3. Implement isEmpty() method

4. Implement insertLast method

5. Implement printNames method

6. Implement deleteByName method

1. Create a User Node

```
Class UserNode {
    String name;
    String lastName;
    UserNode next;
}
```

2. Create a User Node List Class

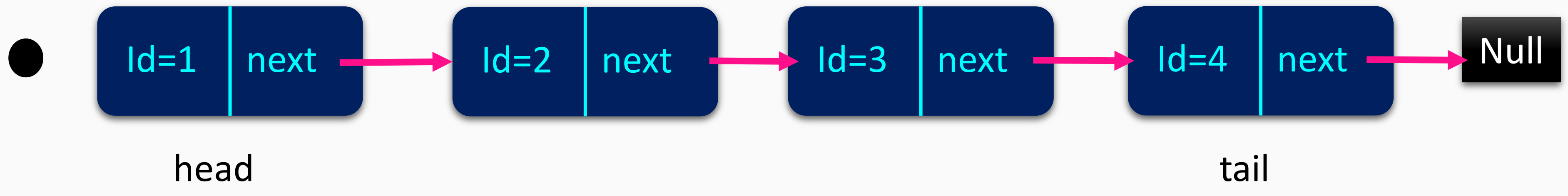
3. Implement isEmpty() method

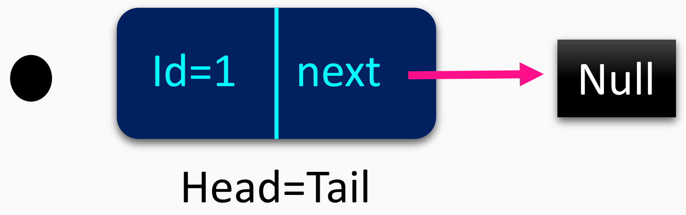
4. Implement insert method

5. Implement Print method - Iteration

6. Implement DeleteByName method

- Case 1: Deleting head node





- Case 2: Deleting tail node
- Case 3: Deleting middle