

dynamic data structure

LinkedList

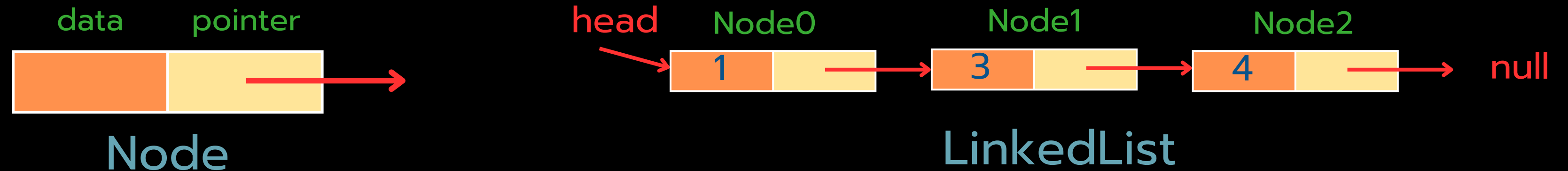
P' ណាត

AKA P'many / P'JadeS



LinkedList

data structure ประเภทหนึ่ง ที่มีการเก็บข้อมูลเรียงต่อกันเป็นเส้นตรง (linear data structure) คล้ายกับ Array ต่างกันตรงที่ LinkedList เก็บข้อมูลเป็นหน่วยย่อย เรียกว่า Node ต่อๆ กันเป็นลำดับ

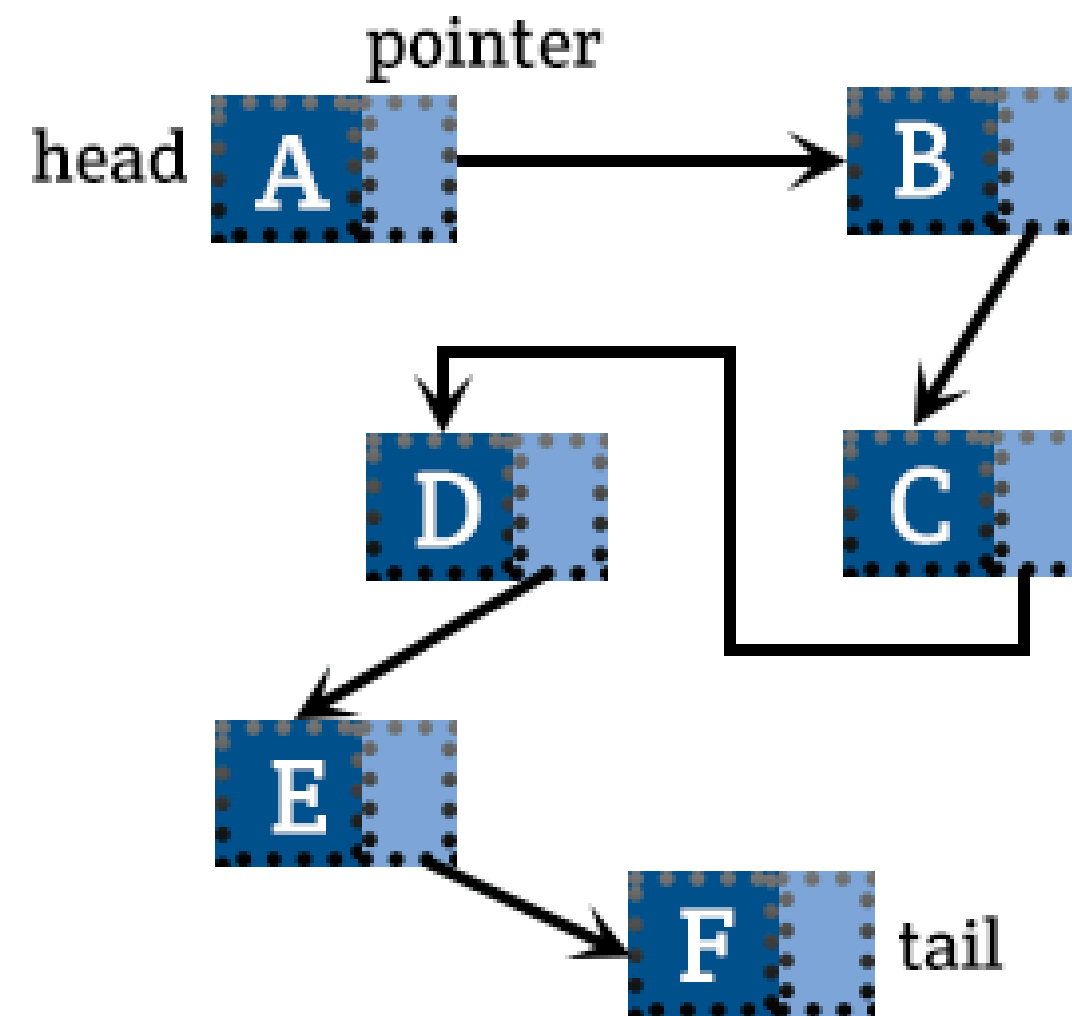


แต่ละ Node จะประกอบด้วย ข้อมูล(data) และ ตำแหน่ง Node ถัดไป(pointer)

Array

index	
0	A
1	B
2	C
3	D
4	E
5	F

Linked List



สร้าง array ความยาว 10 หน่วย เก็บข้อมูลตัวอักษร A,B,C,D,E

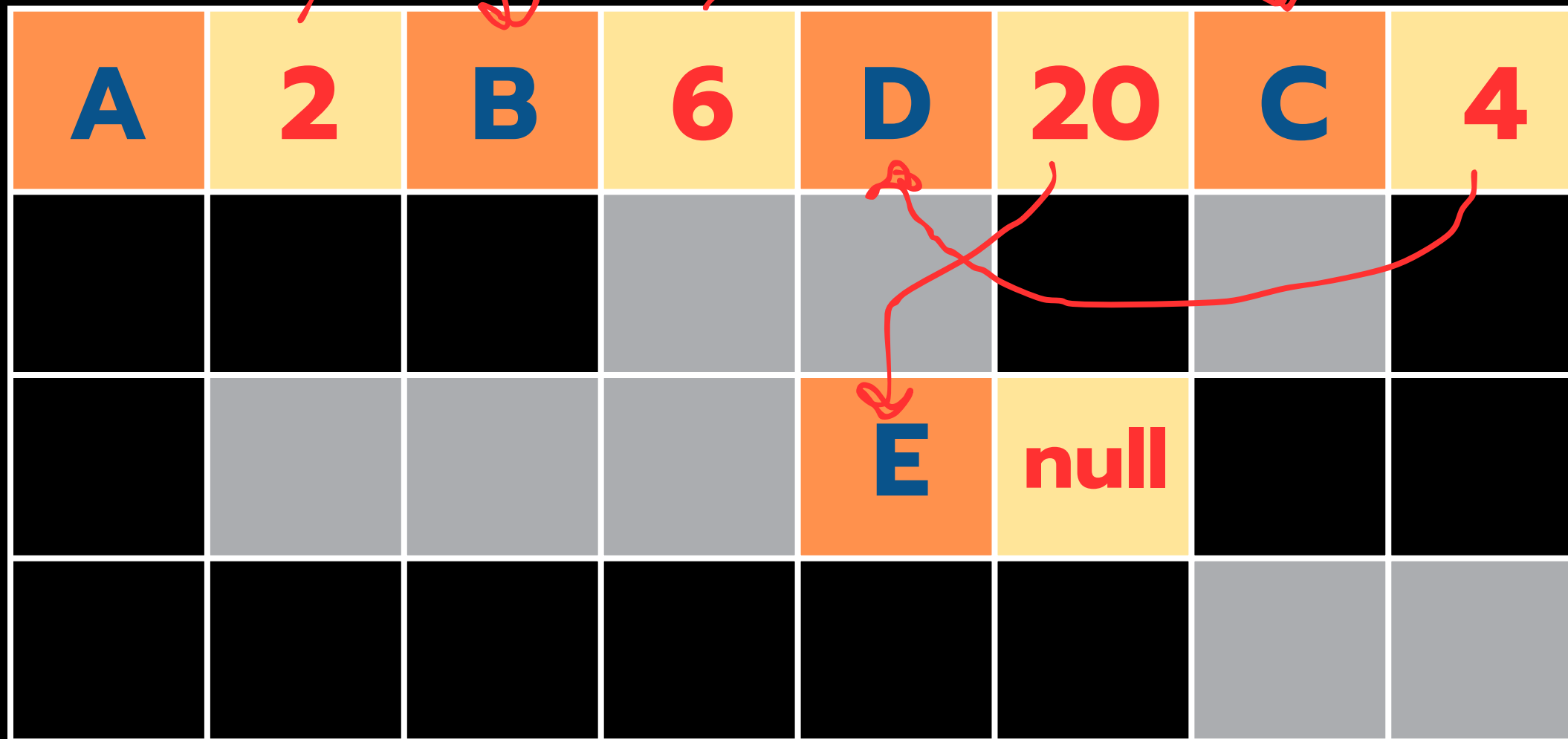
A	B	C	D	E			

- พื้นที่ใน memory ถูกจองทิ้งไว้
- ตำแหน่งที่ว่าง memory ไม่ต่อเนื่องกัน

เมื่อเราสร้าง array พื้นที่ใน memory จะถูกจองไว้ตามความยาวของ array ที่เราได้ประกาศตอนสร้าง

LinkedList concept

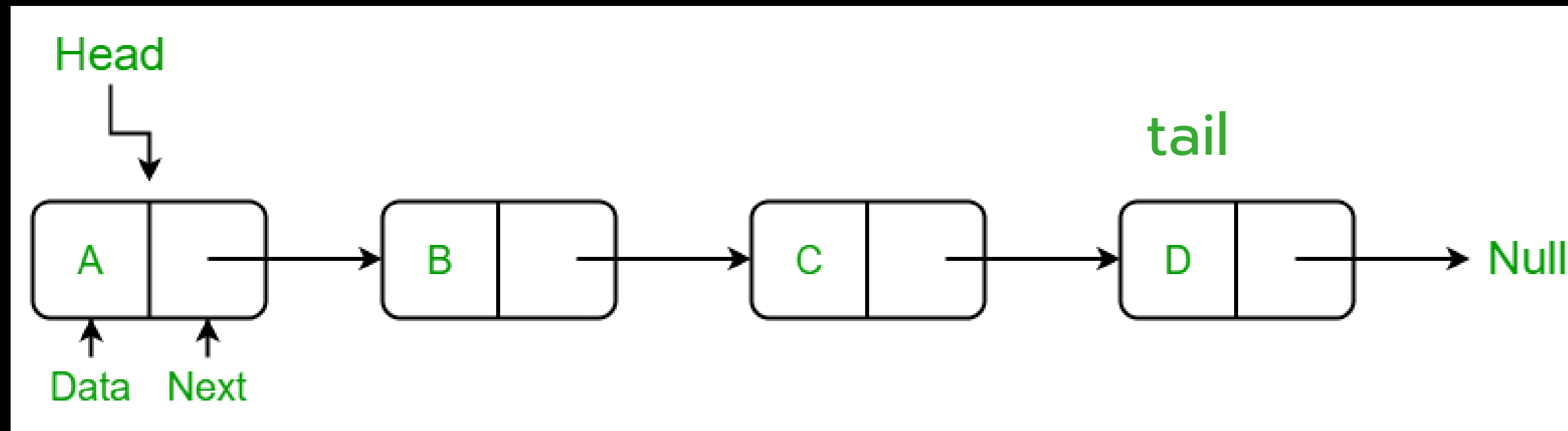
head



- ยืดหยุ่นในการเพิ่มลดขนาด
- ใช้งาน memory ได้อย่างเต็มประสิทธิภาพ

การบันทึกค่าของ LinkedList ไม่จำเป็นต้องเรียงต่อกันใน memory
เพราะมี pointer คอยชี้บอกว่า Node ถัดไปถูกเก็บอยู่ที่ตำแหน่งไหน

Node



<https://www.geeksforgeeks.org/introduction-to-singly-linked-list/>

head : จุดเริ่มต้นของข้อมูล
data : ข้อมูลที่เก็บใน Node นั้นๆ

tail : จุดสิ้นสุดของข้อมูล ชี้ไปที่ null
next : ชี้ไปที่ตำแหน่งของ Node ถัดไป

โครงสร้าง Class Node

```
1 public class Node {  
2     int data;  
3     Node next;  
4  
5     public Node(int d){  
6         data = d;  
7     }  
8 }
```



ในแง่ของการเขียนโปรแกรมภาษา java เราไม่ต้องระบุตำแหน่งบน memory เอง
สามารถระบุเป็น Object ของ Node ถัดไปได้เลย

create LinkedList

```
1 public class myLinkedList {  
2     private int size;  
3     Node head;  
4     myLinkedList(){  
5         this.head = null;  
6         size = 0;  
7     }  
8
```

กำหนดค่าเริ่มต้นของ
LinkedList เมื่อถูกสร้าง

```
9     public class Node {  
10         int data;  
11         Node next;  
12  
13         public Node(int d){  
14             data = d;  
15         }  
16     }
```

สร้าง Node เป็น inner Class ของ
LinkedList เพื่อให้ง่ายต่อการเรียกใช้งาน

```
1     public boolean isEmpty(){  
2         return size == 0;  
3     }  
4  
5     public int getSize(){  
6         return size;  
7     }
```

method แสดงขนาดของ LinkedList กับ เช็คว่าเป็นค่าว่าง
หรือไม่

create LinkedList

```
1 public class testLinkedList {  
2  
3     public static void main(String[] args) {  
4         myLinkedList l1 = new myLinkedList();  
5  
6         System.out.println("List size : "+ l1.getSize());  
7         System.out.println(l1.isEmpty());  
8  
9     }  
10 }
```

LinkedList ชื่อ = new LinkedList();

สร้าง LinkedList เปล่าชื่อ l1

head → null

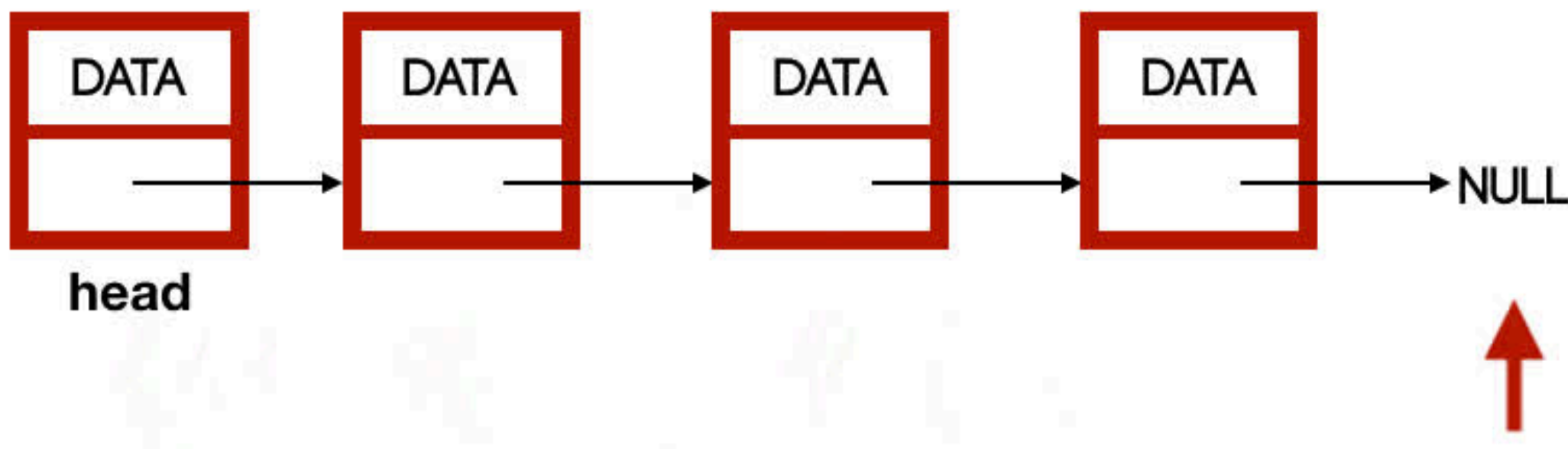
OUT PUT

```
List size : 0  
true
```

Read

```
1 public void display(){
2     Node p = head;
3     System.out.print("head");
4     while (p != null) {
5         System.out.print(" -> " + p.data );
6         p = p.next;
7     }
8     System.out.print(" -> null");
9 }
```

Big-O (n)



การเข้าถึงข้อมูลใน linkedList จำเป็นจะต้อง เริ่มไล่จาก head ก่อนเสมอ ไม่สามารถเข้าถึงโดย index ตรงๆ ได้เหมือนกับ array

ใช้ next หรือ pointer ของ Node ปัจจุบัน เพื่อเข้าถึง Node ตัวถัดไป

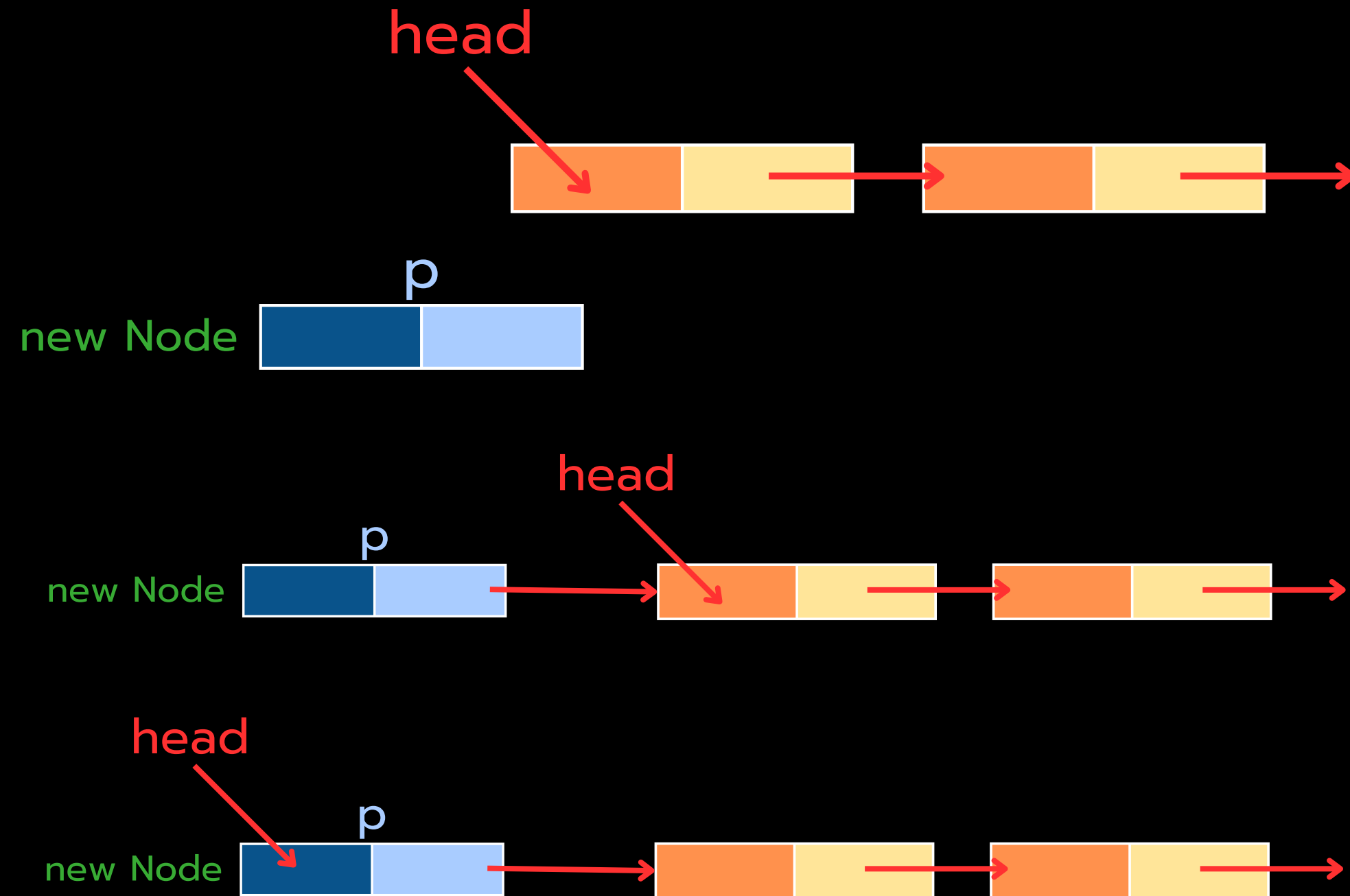
Add (from HEAD)

เพิ่มข้อมูลใน LinkesList จะเป็นการสร้าง Node ขึ้นมาใหม่ และนำไปเชื่อมกับ Node เดิม

```
1 public void addFirst(int d){  
2     Node p = new Node(d);  
3     p.next = head;  
4     head = p;  
5     size++;  
6 }
```

Big-O (1)

การเพิ่มที่จุดเริ่มต้น (add from head) เป็นวิธีที่เร็วที่สุดในกรณีที่เราไม่สนใจเรื่องลำดับก่อนหลังของข้อมูล

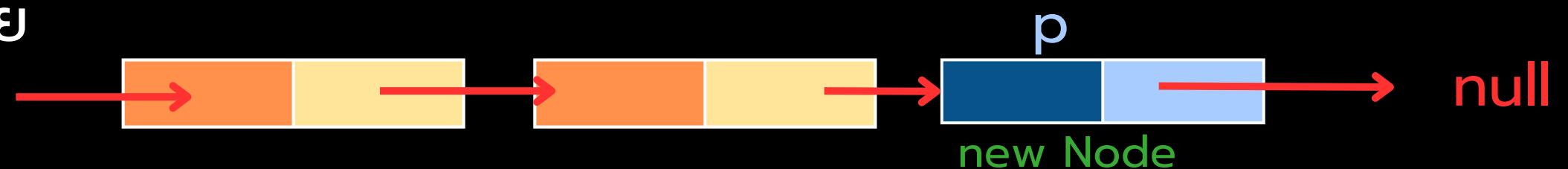


Add (from TAIL)

```
1 public void addLast(int d){
2     Node p = new Node(d);
3     Node current = head;
4     if(current == null){
5         head = p;
6     }else{
7         while(current.next != null){
8             current = current.next;
9         }
10        current.next = p;
11        p.next = null;
12    }
13    size++;
14 }
```

Big-O (n)

การเพิ่มที่ท้ายสุด (add from tail) จะต้องไล่
ตั้งแต่ Node เริ่มต้นจนถึง Node สุดท้าย

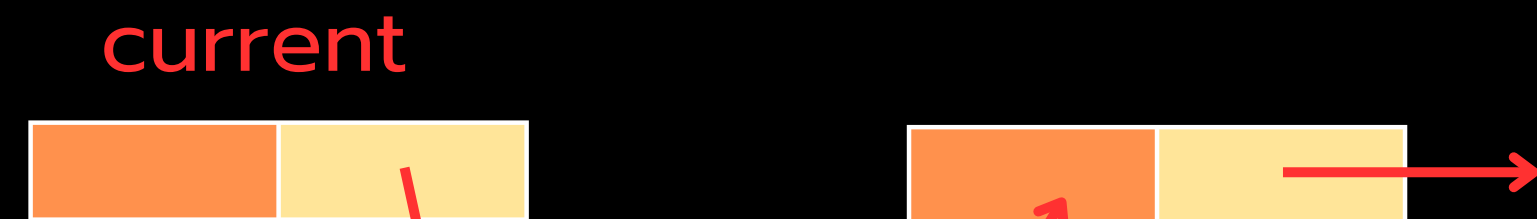
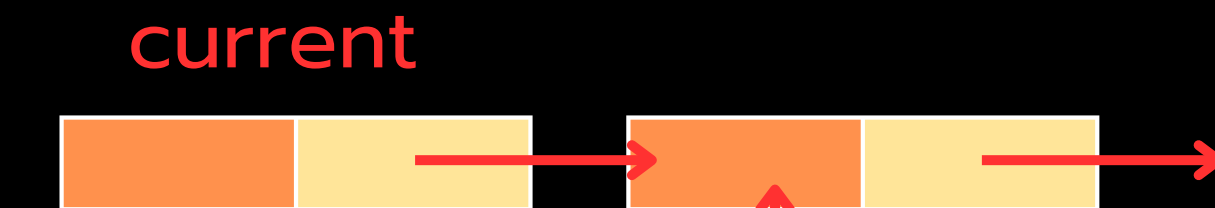
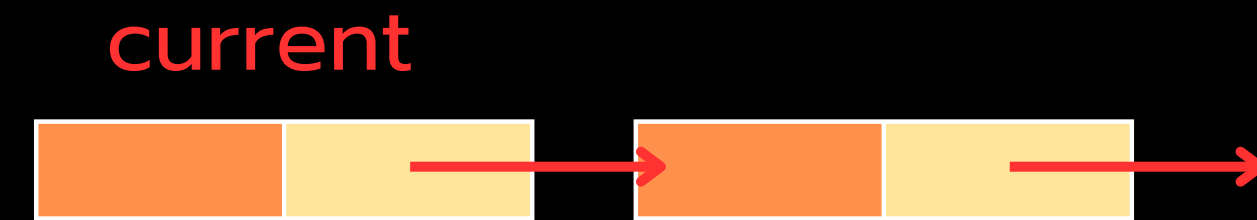


insert

การเพิ่มโดยระบุตำแหน่ง idx

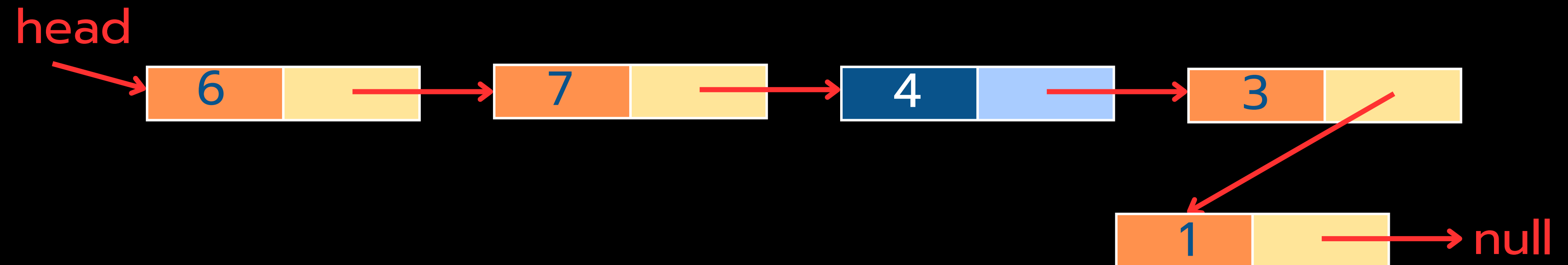
```
1 public void insert(int idx,int d){
2     if(idx == 0) {
3         add(d);
4     } else {
5         Node current = head;
6         for (int i = 1; i < idx; i++) {
7             current = current.next;
8         }
9
10        Node p = new Node(d);
11        p.next = current.next;
12        current.next = p;
13        size++;
14    }
15 }
```

Big-O (1)



```
1 public static void main(String[] args) {  
2     myLinkedList l1 = new myLinkedList();  
3  
4     System.out.println("List size Before add : "+ l1.getSize());  
5     l1.addLast(1);  
6     l1.addFirst(3);  
7     l1.addFirst(7);  
8     l1.addFirst(6);  
9     l1.insert(2,4);  
10  
11     System.out.println("List size after add: "+ l1.getSize());  
12     l1.display();  
}
```

```
List size Before add : 0  
List size after add: 5  
head -> 6 -> 7 -> 4 -> 3 -> 1 -> null
```



head → null

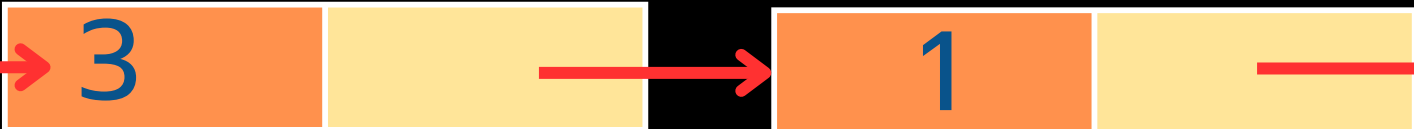
```
5      l1.addLast(1);
```

head →  → null


```
6      l1.addFirst(3);
```

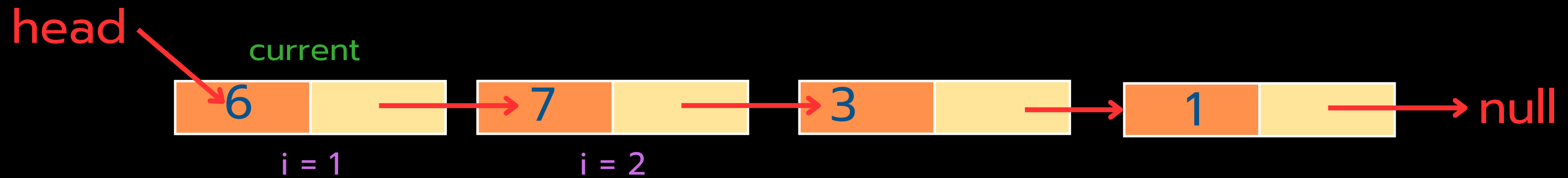
```
7      l1.addFirst(7);
```

```
8      l1.addFirst(6);
```

head →  → null

head →  → null

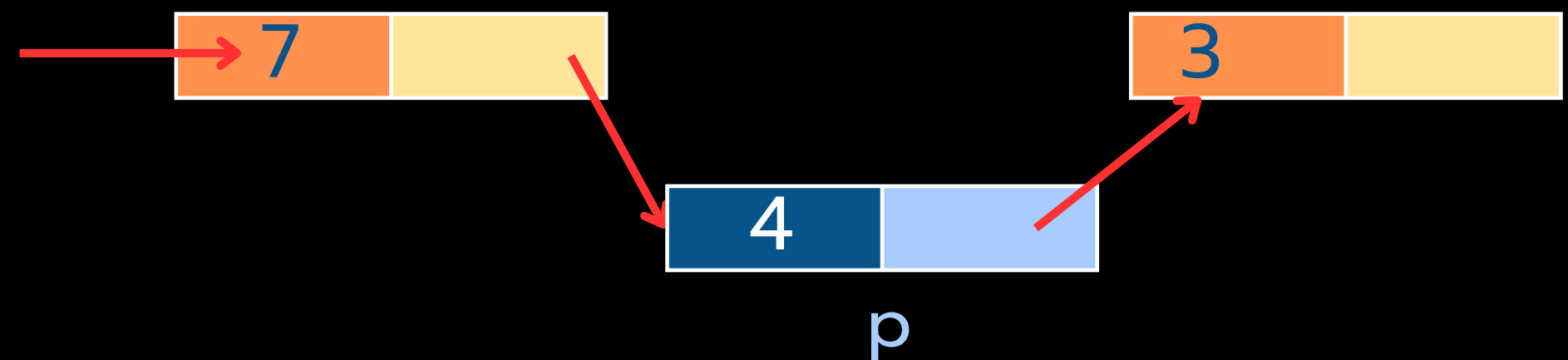
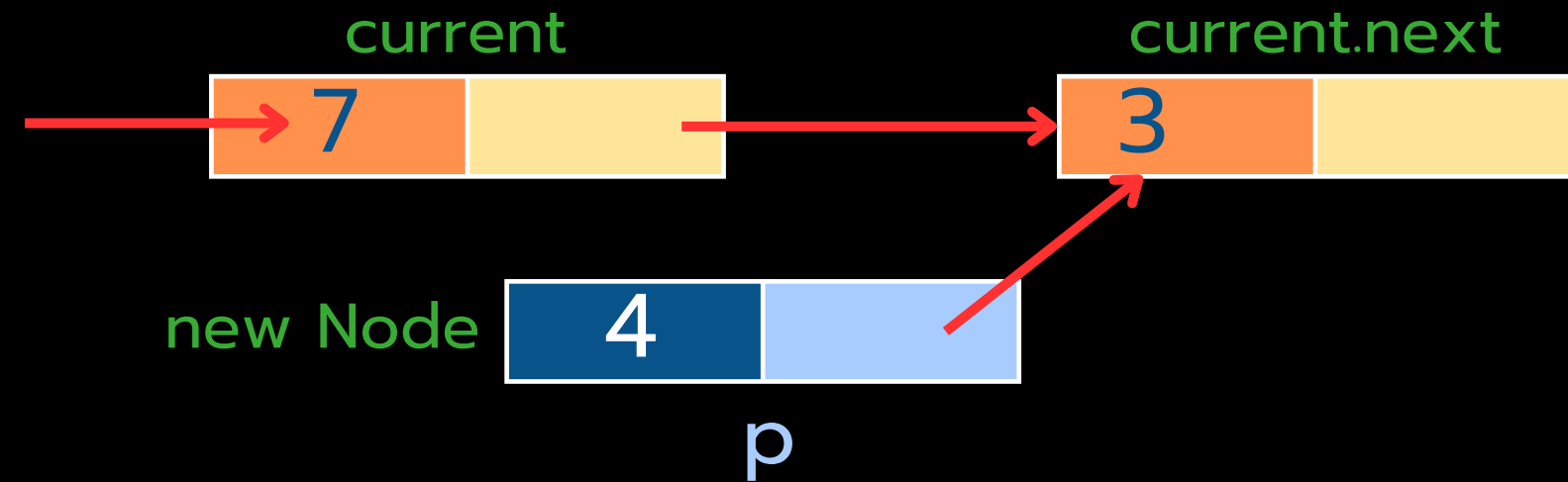
head →  → null



```
9      ll.insert(2,4);
10
```

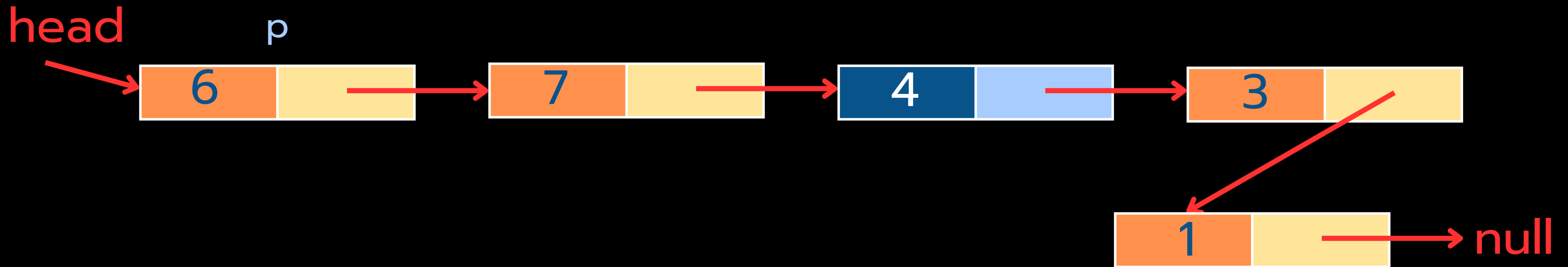
index = 2 data = 4

```
1  public void insert(int idx,int d){
2      if(idx == 0) {
3          add(d);
4      } else {
5          Node current = head;
6          for (int i = 1; i < idx; i++) {
7              current = current.next;
8          }
9
10         Node p = new Node(d);
11         p.next = current.next;
12         current.next = p;
13         size++;
14     }
15 }
```




```
System.out.println("List size after add: "+ l1.getSize());  
l1.display();
```

```
1 public void display(){  
2     Node p = head;  
3     System.out.print("head");  
4     while (p != null) {  
5         System.out.print(" -> " + p.data );  
6         p = p.next;  
7     }  
8     System.out.print(" -> null");  
9 }
```



```
List size Before add : 0  
List size after add: 5  
head -> 6 -> 7 -> 4 -> 3 -> 1 -> null
```

Find

```
1 public Node find(int d){
2     Node p = head;
3     while (p != null) {
4         if(p.data == d){
5             return p;
6         }
7         p = p.next;
8     }
9     return null;
10 }
```

Big-O (n)

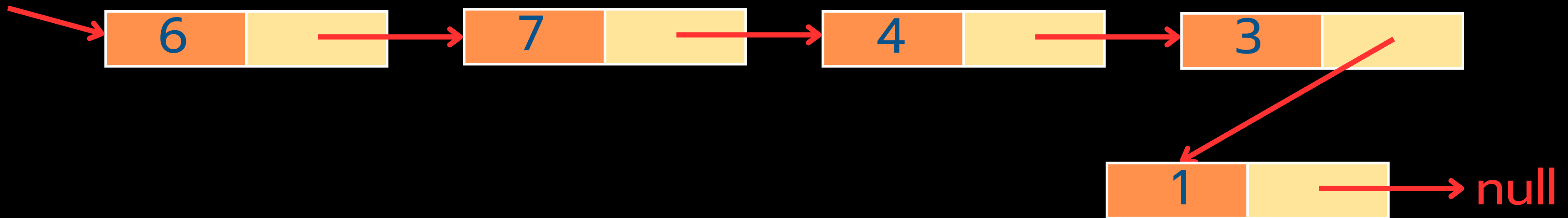
หา Node ที่เก็บค่าตามที่กำหนดแล้วให้
return ออกมาเป็น Node นั้น
ใช้ .data เพื่อเข้าถึงข้อมูลใน Node

OUT PUT

```
System.out.println(l1.find(d:4));
```

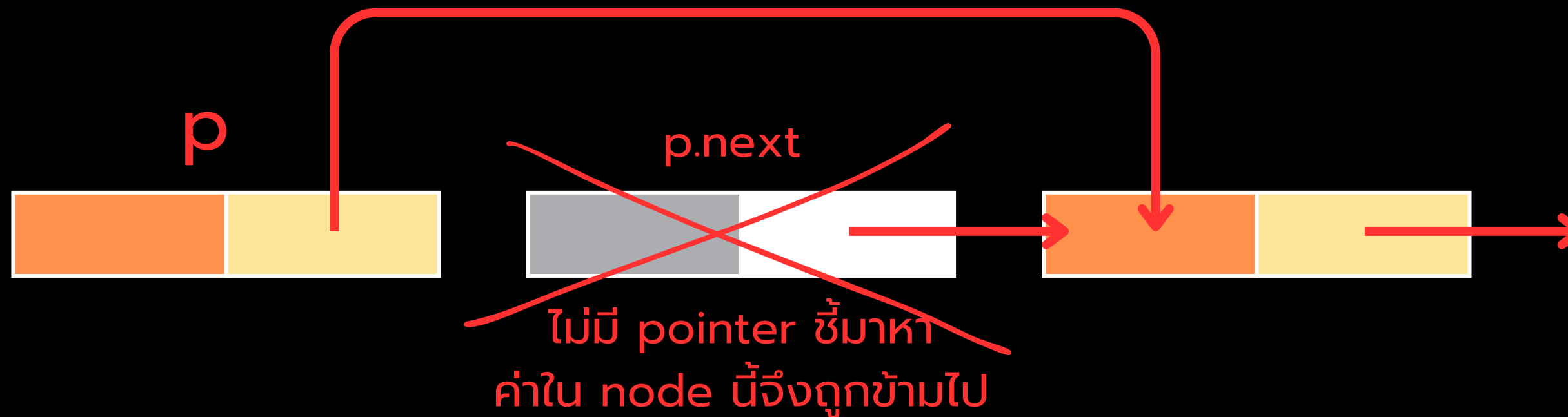
```
myLinkedList$Node@33c7353a
```

head



Delete

การลบค่าออกจาก LinkedList สามารถทำได้โดยการเปลี่ยน Pointer
ที่ชี้ไปยัง Node นั้นๆ ให้ชี้ไปหา Node อื่นแทน



Delete

delete โดยระบุ Node ที่ต้องการ

```
1 public void delete(Node p){
2     p.next = p.next.next;
3     size--;
4 }
5
```

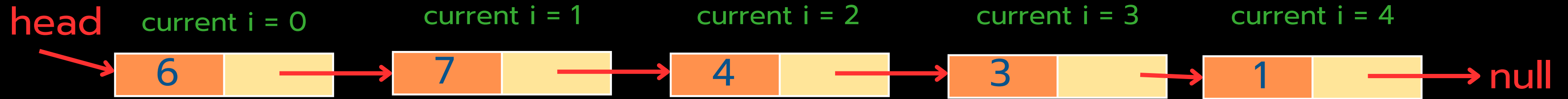
Big-O (1)

ลบ Node ที่อยู่ถัดจาก p

delete โดยระบุ index ของ Node

```
1 public void deleteAt(int idx){
2     Node current = head;
3     if(idx > size-1){
4         System.out.println("Index out of bound");
5         return;
6     }
7     if(idx == 0){
8         head = head.next;
9         size--;
10        return;
11    }
12    for (int i = 0; i < idx - 1; i++) {
13        current = current.next;
14    }
15    current.next = current.next.next;
16    size--;
17 }
```

Big-O (n)



```

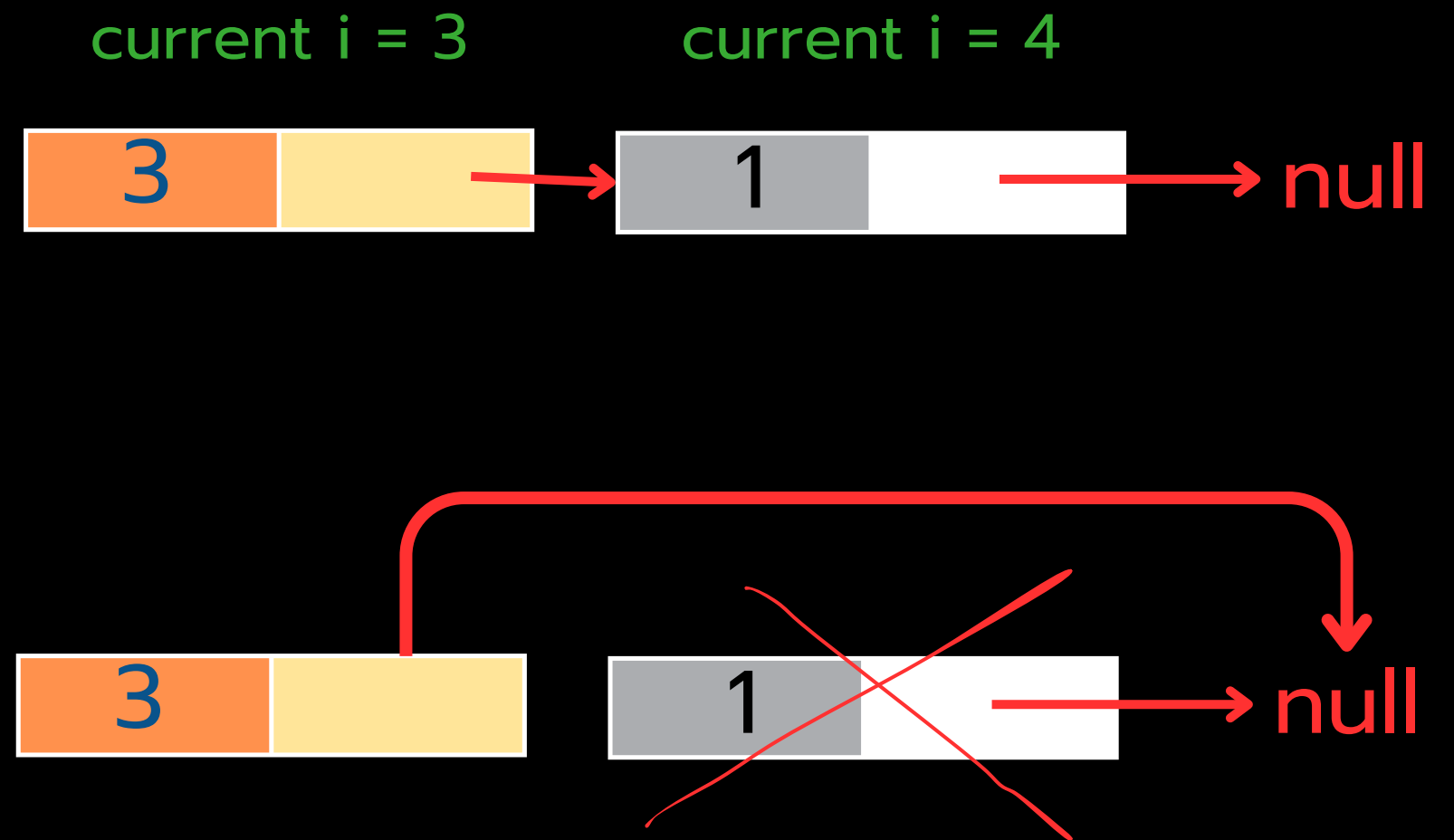
1  l1.deleteAt(5);
2  l1.deleteAt(4);
3  l1.display();

```

```

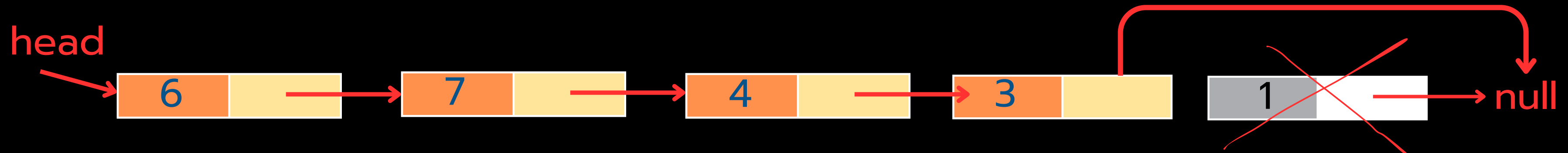
1  public void deleteAt(int idx){
2      Node current = head;
3      if(idx > size-1){
4          System.out.println("Index out of bound");
5          return;
6      }
7      if(idx == 0){
8          head = head.next;
9          size--;
10         return;
11     }
12     for (int i = 0; i < idx - 1; i++) {
13         current = current.next;
14     }
15     current.next = current.next.next;
16     size--;
17 }

```



```
3  l1.display();
```

```
1  public void display(){  
2      Node p = head;  
3      System.out.print("head");  
4      while (p != null) {  
5          System.out.print(" -> " + p.data );  
6          p = p.next;  
7      }  
8      System.out.print(" -> null");  
9  }
```



```
head -> 6 -> 7 -> 4 -> 3 -> null
```

ArrayList

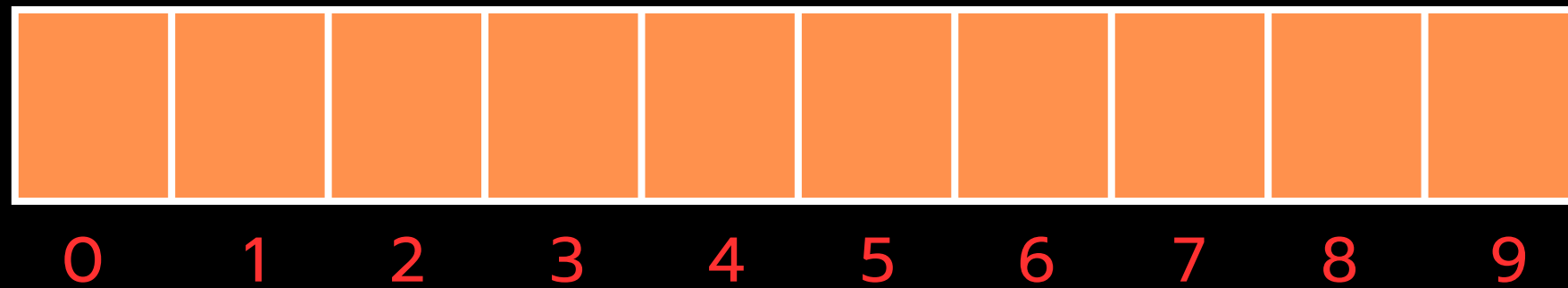
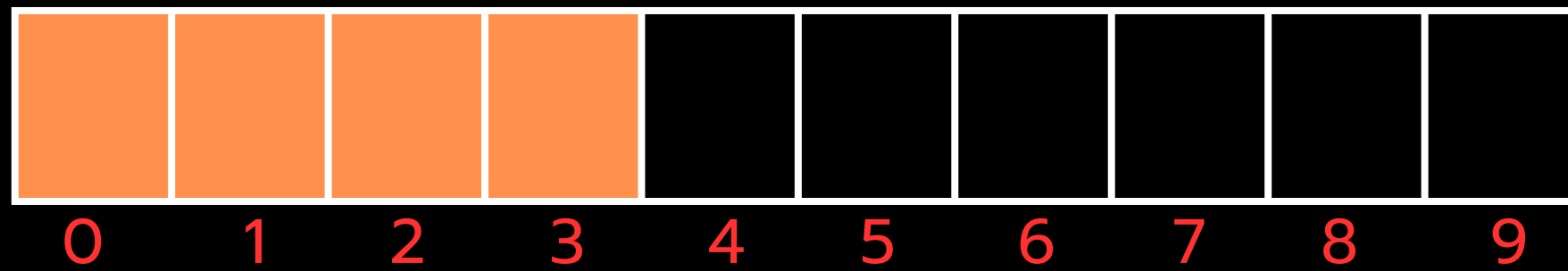
ArrayList

collection ที่เก็บข้อมูลในรูปแบบเดียวกับ Array คือ เก็บเป็นลำดับต่อกัน มี index ระบุตำแหน่งของข้อมูล ต่างกันตรงที่ ArrayList ไม่ได้มีการจำกัดขนาดที่ตายตัว แบบ Array เพราะสามารถย่อ และขยายขนาดได้เอง
อัตโนมัติตามจำนวนข้อมูลที่มี

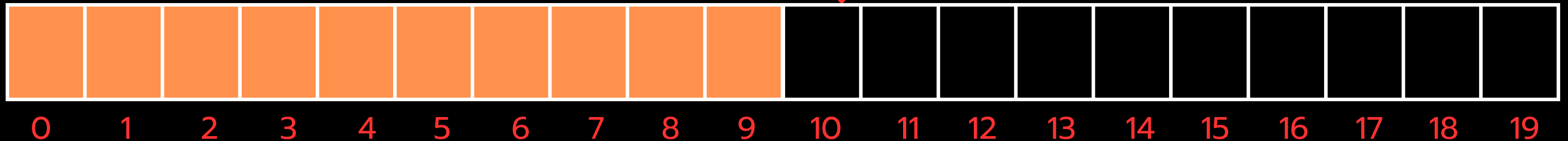
2	5	12	1	79	11
0	1	2	3	4	5

<https://www.geeksforgeeks.org/arraylist-in-java/>

ArrayList เริ่มต้นขนาด 10 หน่วย



สร้าง ArrayList อันใหม่ที่ขนาดใหญ่กว่าเดิม
แล้ว copy ค่าไปเก็บใน ArrayList ใหม่



Create ArrayList

`ArrayList<ชนิดตัวแปรที่เก็บ> ชื่อ = new ArrayList<ชนิดตัวแปรที่เก็บ>(ขนาดเริ่มต้น);`

```
1 public static void main(String[] args) {  
2     ArrayList arr = new ArrayList();  
3     ArrayList arr2 = new ArrayList(5);  
4     ArrayList<Integer> intArr = new ArrayList<Integer>();  
5     ArrayList<String> strArr = new ArrayList<String>();  
6  
7 }
```

ชนิดตัวแปรที่เก็บเป็นตัวกำหนดว่า ค่าที่อยู่ใน ArrayList เป็นค่าประเภทใด
ถ้าไม่ได้กำหนดชนิดตัวแปรที่จะเก็บใน ArrayList จะถือว่าสามารถเก็บตัวแปรต่างชนิดรวมกันได้

Add

`arr.add(data)`

เพิ่มไว้ตัวท้ายสุด

`arr.add(index,data)`

เพิ่มข้อมูลโดยระบุตำแหน่ง

ArrayList ที่ไม่กำหนดชนิดของข้อมูล

```
1 arr.add(1);  
2 arr.add("ABC");  
3 arr.add('A');  
4 arr.add(true);
```

`[1, ABC, A, true]`

ArrayList ที่กำหนดว่าเก็บแค่ Integer

```
1 intArr.add(1);  
2 intArr.add(2);  
3 intArr.add(3);  
4 intArr.add(1,4);
```

`[1, 4, 2, 3]`

Read

`arr.get(index)` อ่านค่าในตำแหน่ง index ของ ArrayList

```
1 for (int i = 0; i < arr.size(); i++) {  
2     System.out.print(arr.get(i) + " ");  
3 }
```

1 ABC A true

```
1 for (Integer num : intArr) {  
2     System.out.print(num + " ");  
3 }
```

1 4 2 3

`arr.toString()` แสดงค่าทั้งหมดของ ArrayList ในรูปของ String

```
1 System.out.println(arr.toString());  
2 System.out.println(intArr.toString());
```

[1, ABC, A, true]

[1, 4, 2, 3]

```
1 System.out.println(arr);
```

[1, ABC, A, true]

Update

```
arr.set(index,data)
```

แทนที่ค่าในตำแหน่ง index
ของ ArrayList ด้วยค่าใหม่

** ต้องเป็นตัวแปรประเภทเดียวกันกับที่ประกาศไว้ตอนสร้าง ArrayList

```
1 System.out.print("Before : ");
2 System.out.println(arr.toString());
3
4 arr.set(1,2024);
5 arr.set(3,"CAMPER");
6
7 System.out.print("After : ");
8 System.out.println(arr.toString());
```

```
Before : [1, ABC, A, true]
After : [1, 2024, A, CAMPER]
```

Delete

```
1 System.out.print("Before : ");
2 System.out.println(arr.toString());
3
4 arr.remove(1);
5 arr.remove(true);
6
7 System.out.print("After : ");
8 System.out.println(arr.toString());
```

```
Before : [1, ABC, A, true]
After : [1, A]
```

`arr.remove(index)`
`arr.remove(data)`

ลบค่าออกจาก ArrayList โดยระบุด้วย
ตำแหน่ง `index` หรือ ค่าที่ต้องการลบ

** ถ้าหากมีค่าเหมือนกันมากกว่า 1 ตัวใน ArrayList จะทำการลบแค่ตัวแรก

```
Before : [1, ABC, A, true, true, true]
After : [1, A, true, true]
```

Big-O

Data Structure	Access	Search	Insert	delete	Space
Array	$O(1)$	$O(n)$	$O(n)$	$O(n)$	$O(n)$
LinkedList	$O(n)$	$O(n)$	$O(1)$	$O(1)$	$O(n)$
ArrayList	$O(1)$	$O(n)$	$O(n)$	$O(n)$	$O(n)$

Thank you

