CENG 201 Veri Yapıları 3: Bağlı Listeler

Öğr.Gör. Şevket Umut ÇAKIR

Pamukkale Üniversitesi

Hafta 3

Anahat

Tek Bağlı Liste

- Ardışık olarak düğümleri(Node) barındıran bir veri yapısıdır
- Her düğüm bir değer(value) vede bir başka düğüme bağlantı(next) içerir
- Listedeki son düğümün bağlantısı boştur(null)
- Bağlı listenin bir başlangıç düğümü vardır

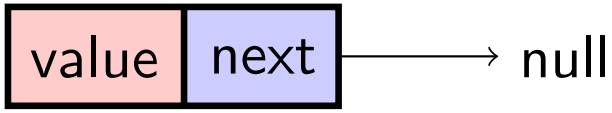


Figure: Tek bağlı liste düğümü(Node)

Düğüm Sınıfı

```
1  package sunum;
2  public class Node<T> {
3      public T value;//Saklanan değer
4      public Node<T> next;//Bir sonraki eleman
5
6      public Node(T value, Node<T> next) {
7          this.value = value;
8          this.next = next;
9      }
10
11     public Node() { }//Varsayılan yapıcı metod
12 }
```

Düğüm Sınıfı Kullanımı

```
1  package sunum;
2  public class Program {
3      public static void main(String[] args) {
4          Node<Integer> n2=new Node<>();
5          n2.value=5;
6          Node<Integer> n1=new Node<>(17, n2);
7          Node<Integer> n3=new Node<>();
8          n3.value=8;
9          n2.next=n3;
10     }
11 }
```

Örnek Kodla Oluşan Yapı

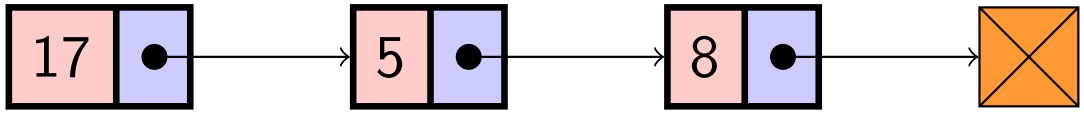
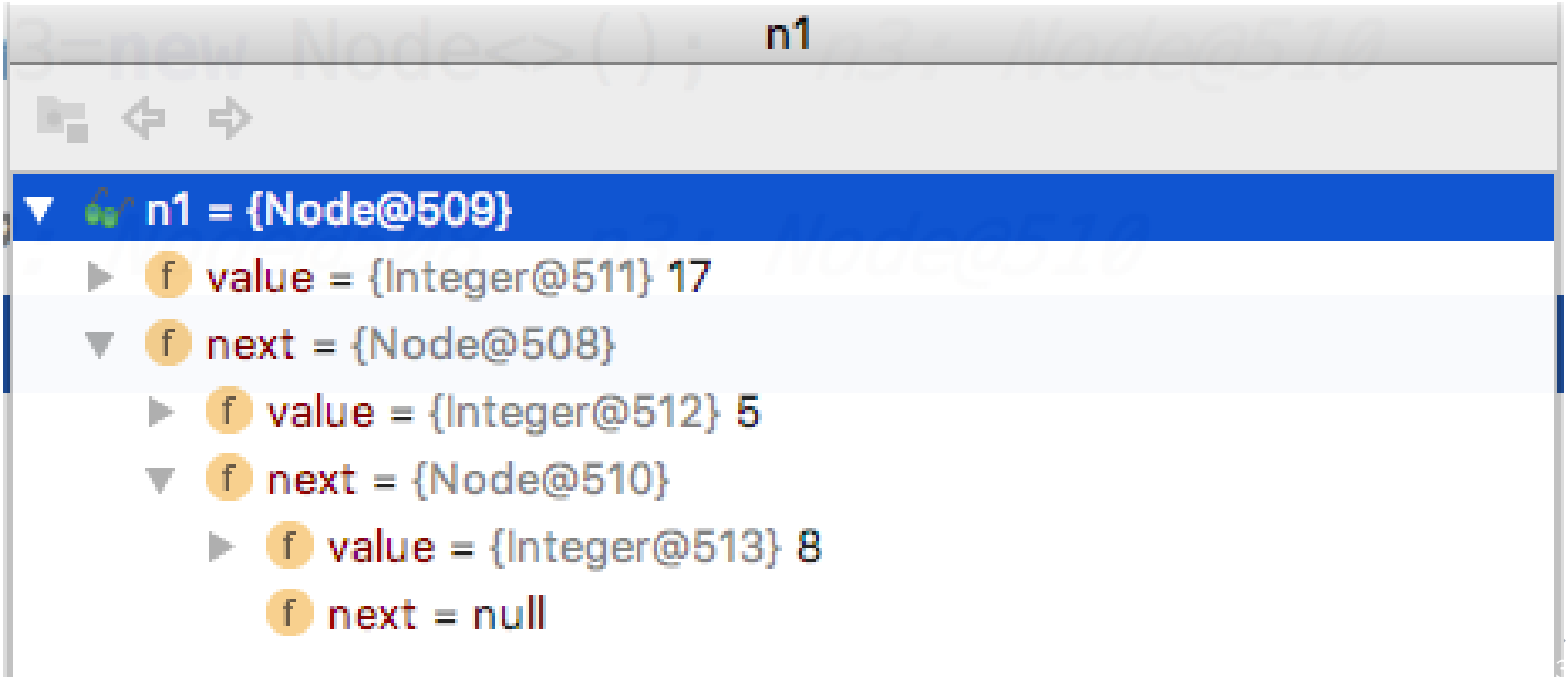
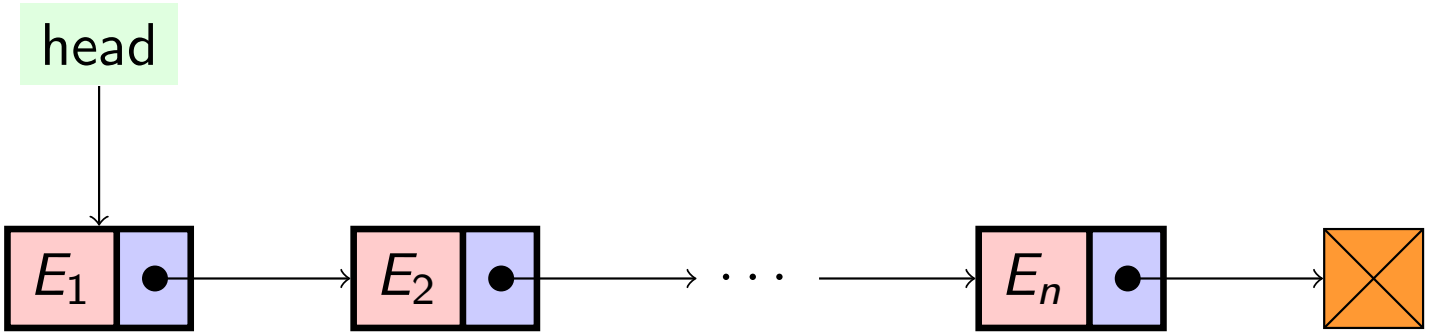


Figure: Temsili gösterim



Bağlı Liste Gösterimi/Linked List



Bağlı Liste Sınıfı

```
2  public class LinkedList<T> {
3      private Node<T> head;//Başlangıç düğümü
4      public LinkedList(){}//Yapıcı metod
5      public void print(){}//Tüm listeyi yazdır
6      public Node<T> getHead(){}//Ilk düğümü verir
7      public void addFirst(T value){}//Listenin başına ekler
8      public void addLast(T value){}//Listenin sonuna ekler
9      public void add(int index, T value){}//Verilen konumdan sonra
       ↪  ekler
10     public void insertAfter(Node<T> node, T value){}//Verilen
       ↪  düğümden sonra ekler
11     public T removeAfter(Node<T> node){}//Verilen düğümden sonrakini
       ↪  siler
12     public T removeAt(int index){}//Verilen konumdaki düğümü siler
13     public T removeFirst(){}//Listenin başını siler
14     public T removeLast(){}//Listenin sonunu siler
15 }
```

Soru

Aşağıdaki işlemler ekran çıktıları nasıl olur:

```
5         LinkedList<Integer> ll=new
          ↪  LinkedList<>();
6         ll.addFirst(3);
7         ll.addFirst(5);
8         ll.addFirst(7);
9         ll.print();
10        ll.addLast(9);
11        ll.print();
12        ll.insertAfter(ll.getHead(),
          ↪  15);
13        ll.print();
14        ll.add(0, 76);
15        ll.add(5, 41);
16        ll.addFirst(12);
17        ll.print();
18        ll.removeAfter(ll.getHead());
19        ll.print();
20        ll.removeAt(3);
21        ll.print();
22        ll.removeFirst();
23        ll.print();
24        ll.removeLast();
25        ll.print();
```

Cevap

7 5 3

7 5 3 9

7 15 5 3 9

12 7 76 15 5 3 9 41

12 76 15 5 3 9 41

12 76 15 3 9 41

76 15 3 9 41

76 15 3 9

print metodu

```
4   public void print(){ //Tüm listeyi yazdır
5       Node<T> current=head;
6       while (current!=null){
7           System.out.println(current.value+" ");
8       }   current = current.next;
9       System.out.println();
10  }
```

addFirst

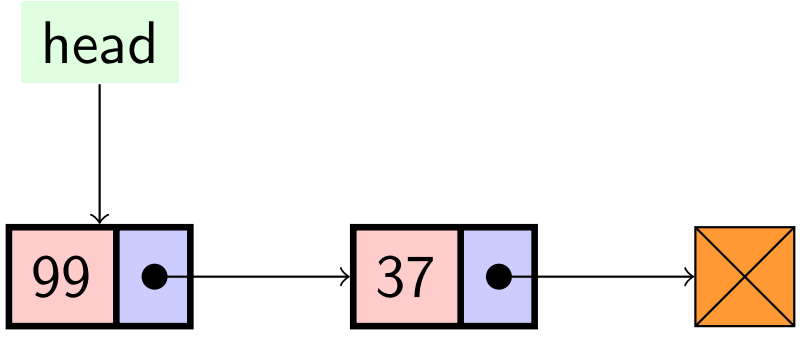


Figure: addFirst(12) öncesi

addFirst

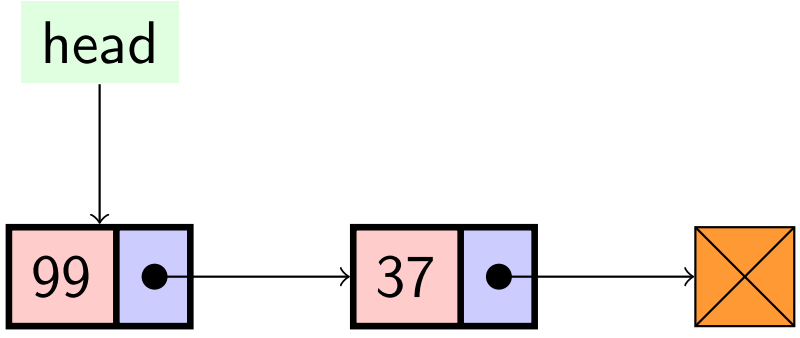


Figure: addFirst(12) öncesi

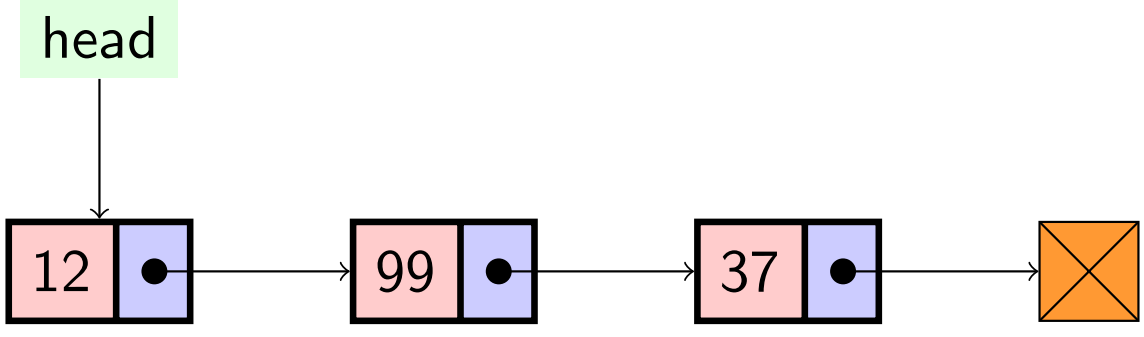


Figure: addFirst(12) sonrası

addFirst metodu

```
12      public void addFirst(T value){ //Listenin başına
        ↪  ekler
13          Node<T> newNode=new Node<T>(value, head);
14          head=newNode;
15      }
```

addLast

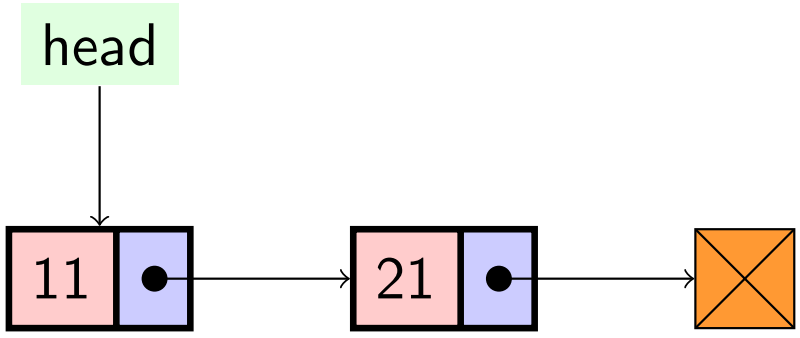


Figure: addLast(24) öncesi

addLast

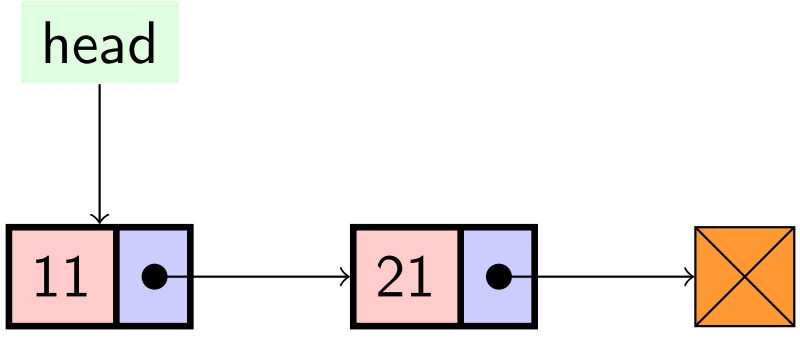


Figure: addLast(24) öncesi

اذا كانت اللينكد ليست فاضية وحطينا اضافة للآخر فبكون خطأ لان الليست الفارغة آخر عنصر فيها هو الهيد لهيك لازم نحط اضافة للاول وليس للاخير

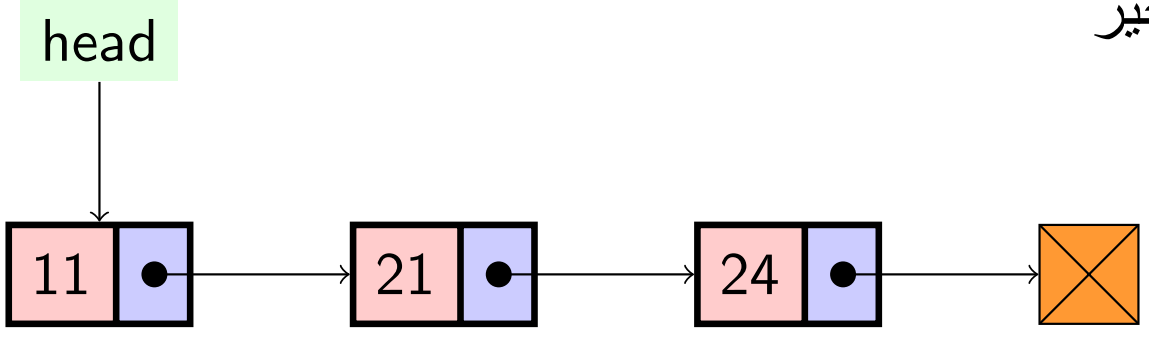


Figure: addLast(24) sonrası

addLast metodu

```
16      public void addLast(T value){ //Listenin sonuna ekler
17          Node<T> current=head;
18          while (current.next != null)
19              current=current.next;
20          current.next=new Node<>(value, null);
21      }
```

addAfter metodu

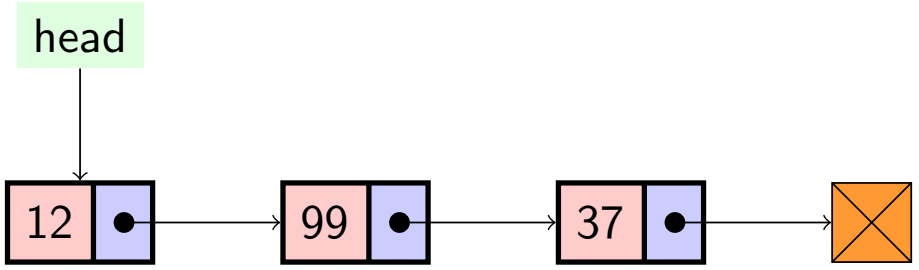


Figure: insertAfter(0,26) öncesi

addAfter metodu

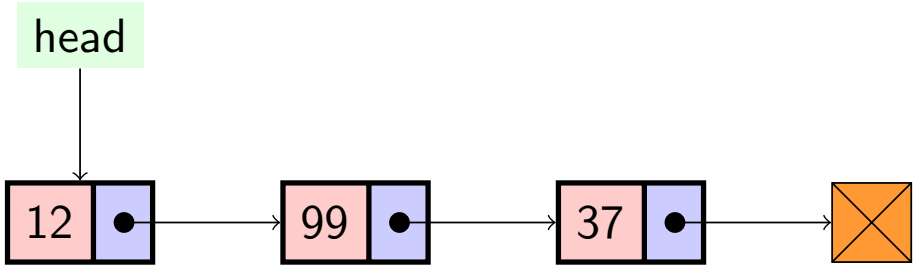


Figure: insertAfter(0,26) öncesi

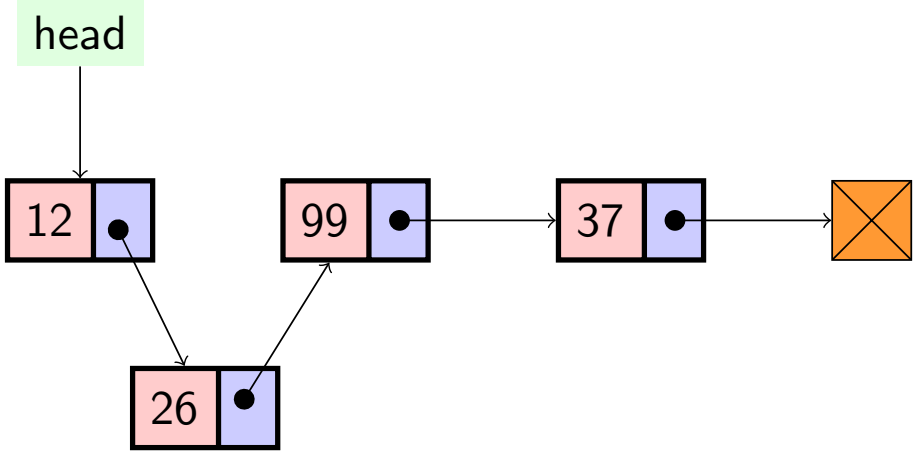


Figure: insertAfter(0,26) sonrası

insertAfter metodu ve add metodu

```
22      public void insertAfter(Node<T> node, T value){ //Verilen düğümden sonra ekler
23          Node<T> newNode=new Node<>(value, node.next);
24          node.next=newNode;
25      }
```

insertAfter metodu ve add metodu

```
22      public void insertAfter(Node<T> node, T value){ //Verilen düğümden sonra ekler
23          Node<T> newNode=new Node<>(value, node.next);
24          node.next=newNode;
25      }
```

```
26      public void add(int index, T value){ //Verilen konumdan sonra ekler
27          Node<T> current=head;
28          int currentLocation=0;
29          while (current != null && currentLocation < index) {
30              current=current.next;
31              currentLocation++;
32          }
33          if (currentLocation != index)
34              throw new IndexOutOfBoundsException("Listede yeterli eleman yok!");
35          insertAfter(current, value);
36      }
```

removeFirst metodu

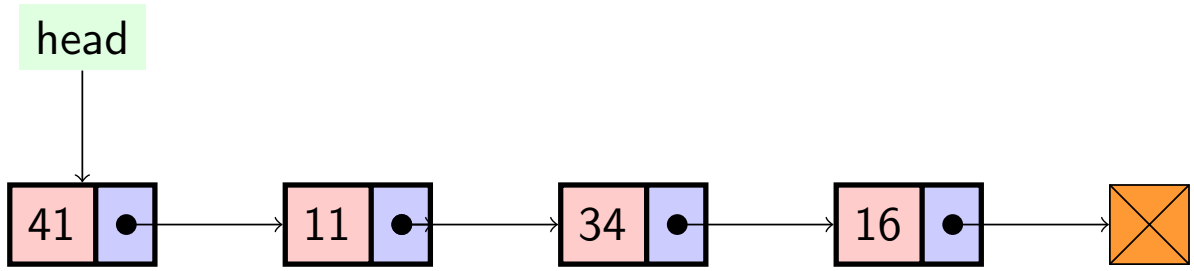


Figure: removeFirst() öncesi

removeFirst metodu

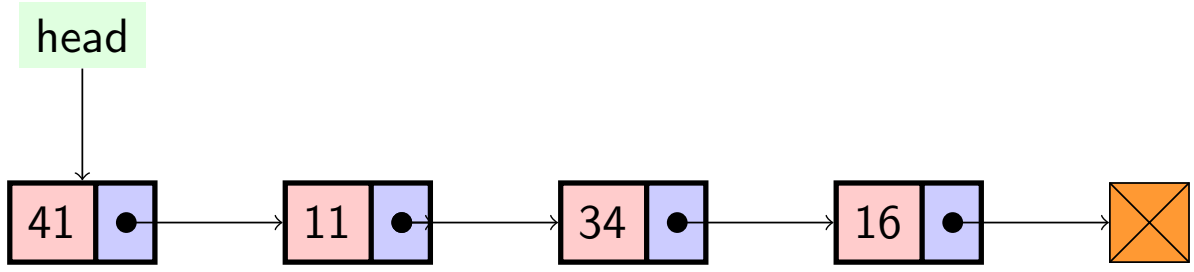


Figure: removeFirst() öncesi

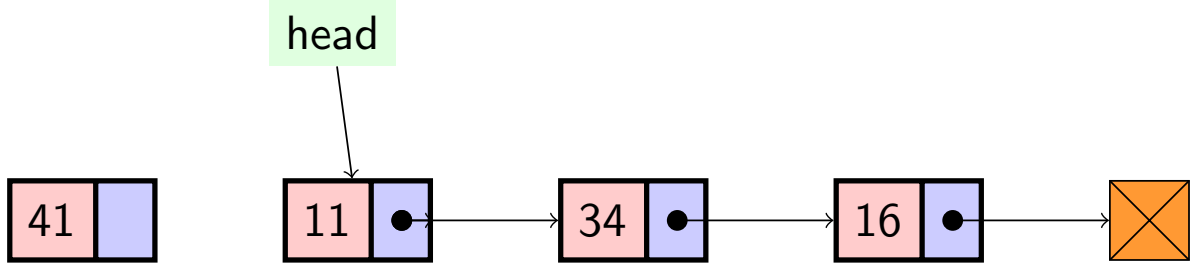


Figure: removeFirst() sonrası

removeFirst metodu

```
39      public T removeFirst(){ //Listenin başını siler
40          Node<T> toDelete=head;
41          head=head.next;
42          return toDelete.value;
43      }
```

removeLast metodu

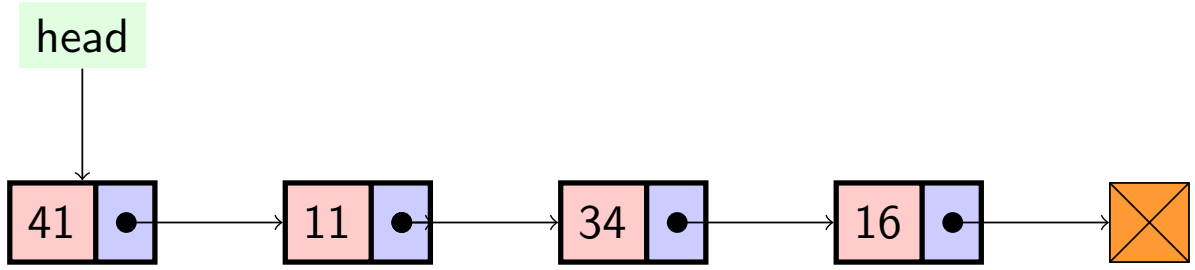


Figure: removeLast() öncesi

removeLast metodu

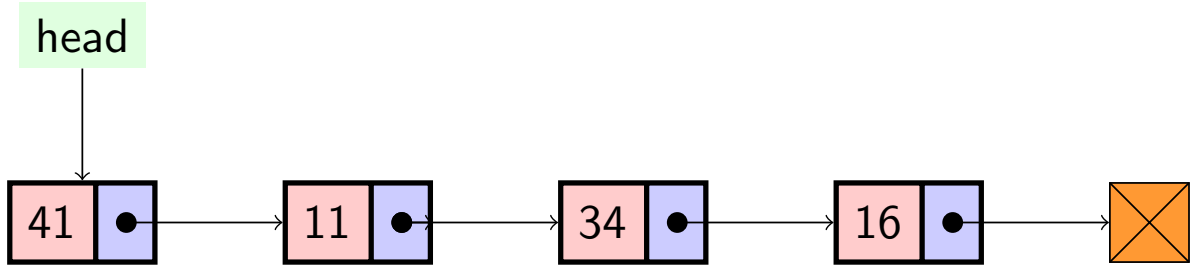


Figure: removeLast() öncesi

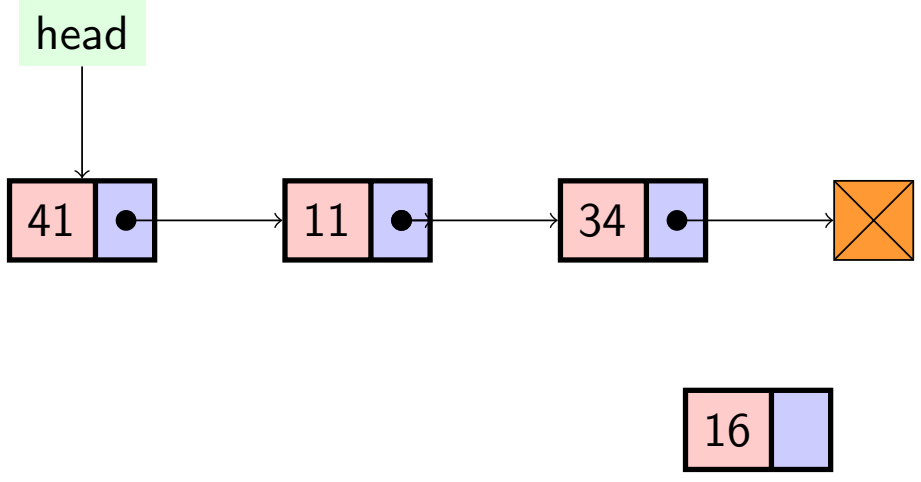


Figure: removeLast() sonrası

removeLast metodu

```
44      public T removeLast(){ //Listenin sonunu siler
45          Node<T> current=head;
46          Node<T> previous=null;
47          T r;
48          while (current.next != null){
49              previous=current;
50              current=current.next;
51          }
52          if(previous!=null){
53              r=previous.next.value;
54              previous.next=null;
55          }
56          else {
57              r=head.value;
58              head=null;
59          }
60          return r;
61      }
```

Çift Bağlı Liste

- Her düğümde hem sonraki(next) hem de önceki(previous) düğüme bağlantı vardır
- Listenin başını(head) ve sonunu gösteren bağlantılar mevcuttur
- Liste üzerinde ileri ve geri yönlü gezinme yapılabilir

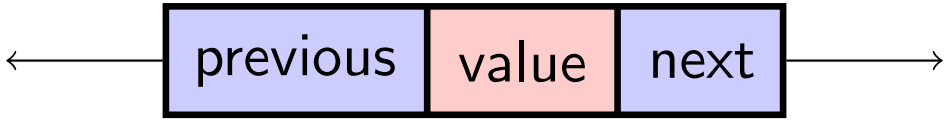
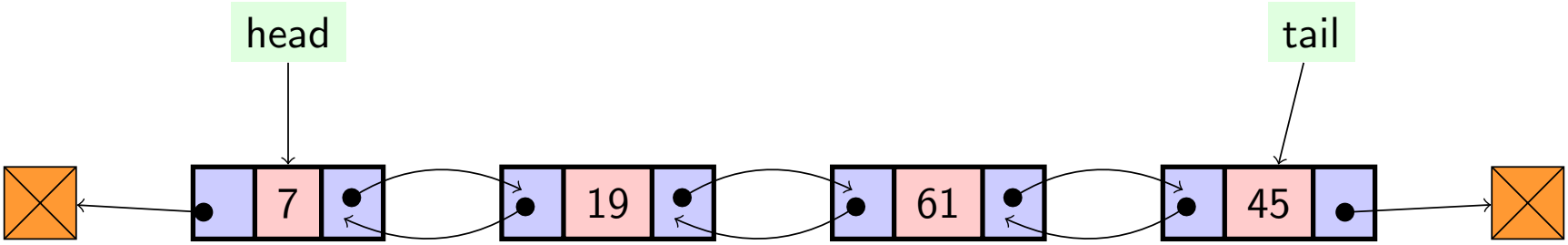


Figure: Çift Bağlı Liste Düğümü(Node)

Çift Bağlı Liste için Node Sınıfı

```
1  package doublyll;
2
3  public class Node<T> {
4      public T value;
5      public Node<T> next;
6      public Node<T> previous;
7
8      public Node(T value, Node<T> next, Node<T> previous) {
9          this.value = value;
10         this.next = next;
11         this.previous = previous;
12     }
13 }
```

Çift Bağlı Liste Temsili Gösterim



DoublyLinkedList Sınıfı

```
1  public class DoublyLinkedList<T> implements Iterable<T> {
2      private Node<T> head;//Listenin başı
3      private Node<T> tail;//Listenin sonu
4      private int size;//Listedeki eleman sayısı
5      public DoublyLinkedList() {}//Yapıcı metod
6      public int size(){ }//boyutu döndürür
7      public boolean isEmpty(){ }//Boş mu
8      public void addFirst(T value){ }//Başa ekler
9      public void addLast(T value){ }//Sona ekler
10     public T removeFirst(){ }//Başı siler
11     public T removeLast(){ }//Sonu siler
12     public void print(){ }//Listeyi yazdırır
13     public void reversePrint(){ }//Tersten yazdırır
14     public Node<T> getHead() { }//Başı döndürür
15     public Node<T> getTail() { }//Sonu döndürür
16     public Iterator<T> iterator() { }
17     public abstract void insertAt(int index, T value);
18     public abstract T removeAt(int index);
19 }
```

print metodları

```
47      public void print(){
48          Node<T> node=head;
49          while(node!=null) {
50              System.out.print(node.value + " ");
51              node=node.next;
52          }
53          System.out.println();
54      }
55      public void reversePrint(){
56        Node<T> node=tail;
57        while(node!=null) {
58            System.out.print(node.value + " ");
59            node=node.previous;
60        }
61        System.out.println();
62      }
```

addFirst metodu

```
15      public void addFirst(T value){
16          Node<T> node=new Node(value, head, null);
17          if (head!=null)
18              head.previous=node;
19          head=node;
20          if (tail==null)
21              tail=node;
22          size++;
23      }
```

addLast metodu

```
24      public void addLast(T value){
25          Node<T> node=new Node<>(value, null, tail);
26          if (tail!=null)
27              tail.next=node;
28          tail=node;
29          if(head==null)
30              head=node;
31          size++;
32      }
```

```
33      public T removeFirst(){
34          Node<T> node=head;
35          head=head.next;
36          head.previous=null;
37          size--;
38          return node.value;
39      }
```

removeLast metodu

```
40      public T removeLast(){
41          Node<T> node=tail;
42          tail=tail.previous;
43          tail.next=null;
44          size--;
45          return node.value;
46      }
```

iterator metodu

```
72      @Override
73      public Iterator<T> iterator() {
74          //Interface gerçekleştiren nesne oluşturma
75          Iterator<T> iterator=new Iterator<T>() {
76              private Node<T> node=head;
77              @Override
78              public boolean hasNext() { //Sonraki değer var mı
79                  return node!=null;
80              }
81              @Override
82              public T next() { //Sonraki değeri döndür
83                  T rval=node.value;
84                  node=node.next; //iterator'u ilerlet
85                  return rval;
86              }
87          };
88          return iterator;
89      }
```

Doubly Linked List İşlem Karmaşıklığı

Table: Mevcut uygulamada işlem karmaşıklıkları

İşlem	**Karmaşıklık**
addFirst, addLast	$O(1)$
removeFirst,removeLast	$O(1)$
insertAt	$O(n)$
removeAt	$O(n)$
size	$O(1)$
print, printReverse	$O(n)$

LinkedList< T > Sınıfı

java.util içinde

- Diziler kullanılarak gerçekleştirilen bir liste yapısıdır
- Bazı metodları ve özellikleri:
  - `size, getFirst, getLast`
  - `addFirst, addLast, removeFirst, removeLast`
  - `add(int index, T element)`
  - `remove(int index)`
  - `contains, indexOf, lastIndexOf, toArray, push, pop`

Listeler ile Yığıt ve Kuyruk

Aşağıdaki işlemler liste kullanılarak nasıl gerçekleştirilebilir?

- Yığıt
  - *push*
  - *pop*
- Kuyruk
  - *enqueue*
  - *dequeue*


prev->next = toDelete->next;
delete toDelete;
// if only forgetting were
// this easy for me.
SNIFF
assert "It's going to be okay.";

1

[1] xkcd