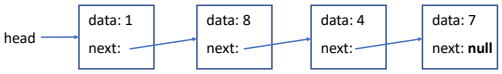
Linked list visualized

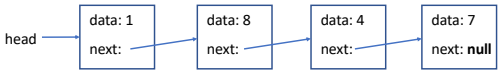
[1, 8, 4, 7]



1

Adding to the front

[1, 8, 4, 7]

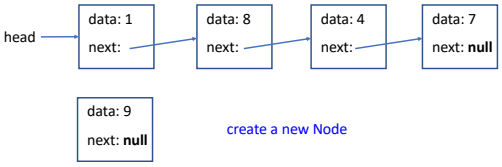


How can we add a value to the front of the list?

2

Adding to the front: add 9 to front

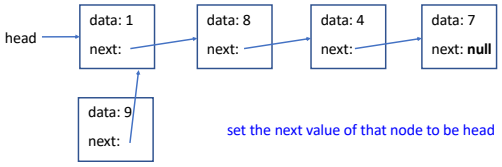
[1, 8, 4, 7]



3

Adding to the front: add 9 to front

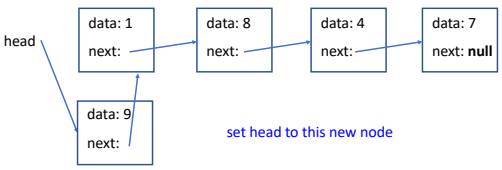
[1, 8, 4, 7]



4

Adding to the front: add 9 to front

[1, 8, 4, 7]



5

Adding to the front: add 9 to front

[1, 8, 4, 7]

6

Adding to the front: add 9 to front

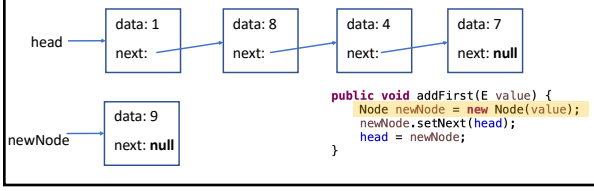
[1, 8, 4, 7]


```
public void addFirst(E value) {
    Node newNode = new Node(value);
    newNode.setNext(head);
    head = newNode;
}
```

7

Adding to the front: add 9 to front

[1, 8, 4, 7]

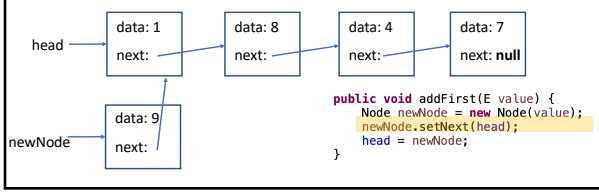



```
public void addFirst(E value) {
    Node newNode = new Node(value);
    newNode.setNext(head);
    head = newNode;
}
```

8

Adding to the front: add 9 to front

[1, 8, 4, 7]

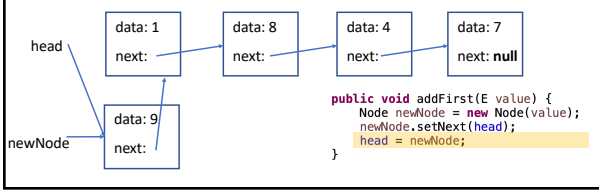



```
public void addFirst(E value) {
    Node newNode = new Node(value);
    newNode.setNext(head);
    head = newNode;
}
```

9

Adding to the front: add 9 to front

[9, 1, 8, 4, 7]




```
public void addFirst(E value) {
    Node newNode = new Node(value);
    newNode.setNext(head);
    head = newNode;
}
```

10

Adding to the front

Does this work if the linked list is empty?

head: null

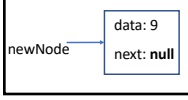
```
public void addFirst(E value) {
    Node newNode = new Node(value);
    newNode.setNext(head);
    head = newNode;
}
```

11

Adding to the front

Does this work if the linked list is empty?

head: null

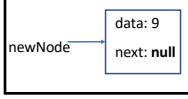



```
public void addFirst(E value) {
    Node newNode = new Node(value);
    newNode.setNext(head);
    head = newNode;
}
```

12

Adding to the front

Does this work if the linked list is empty?

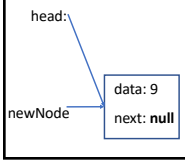



```
public void addFirst(E value) {
    Node newNode = new Node(value);
    newNode.setNext(head);
    head = newNode;
}
```

13

Adding to the front

Does this work if the linked list is empty?

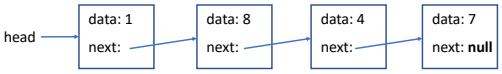



```
public void addFirst(E value) {
    Node newNode = new Node(value);
    newNode.setNext(head);
    head = newNode;
}
```

14

Removing from the front

[1, 8, 4, 7]

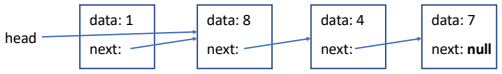


How can we delete a value to the front of the list?
(assuming the list isn't empty)

15

Removing from the front

[1, 8, 4, 7]

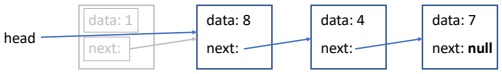


Simply move head down the list!

16

Removing from the front

[1, 8, 4, 7]

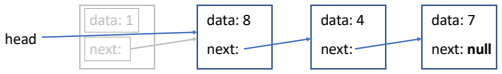


Simply move head down the list!

17

Removing from the front

[1, 8, 4, 7]

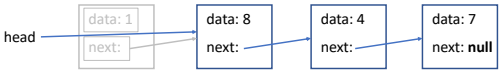


How can do this in code?

18

Removing from the front

[1, 8, 4, 7]

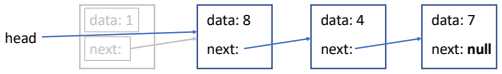


How can do this in code? `head = head.next();`

19

Removing from the front

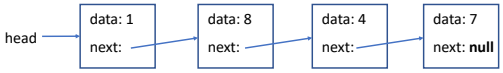
[1, 8, 4, 7]




```
public E removeFirst() {
    if( head == null ) {
        return null;
    } else {
        E returnMe = head.value();
        head = head.next(); // move head down the list
        return returnMe;
    }
}
```

20

Iterating through a linked list

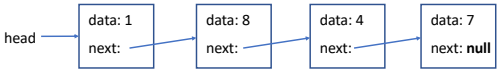



```
Node finger = head;

while (finger != null ) {
    // do something with the current node, finger
    finger = finger.next();
}
```

21

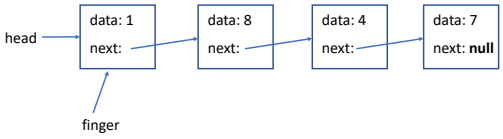
Iterating through a linked list



How can we iterate through a linked list?

22

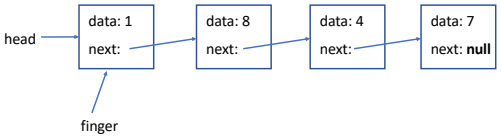
Iterating through a linked list



Use a variable starting at the head and move it down the list

23

Iterating through a linked list

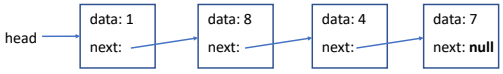


Use a variable starting at the head and move it down the list

How will we know when we're at the end?

24

Iterating through a linked list: printing it

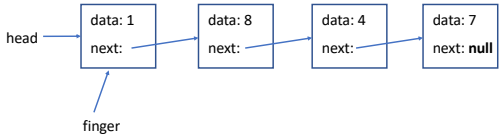



```
Node finger = head;

while (finger != null ) {
    System.out.println(finger.value());
    finger = finger.next();
}
```

25

Iterating through a linked list: printing it

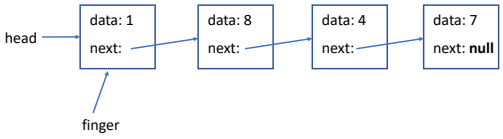



```
Node finger = head;

while (finger != null ) {
    System.out.println(finger.value());
    finger = finger.next();
}
```

26

Iterating through a linked list: printing it

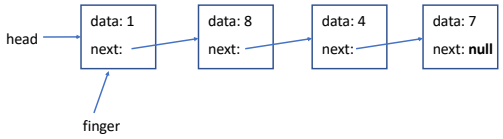



```
Node finger = head;

while (finger != null ) {
    System.out.println(finger.value());
    finger = finger.next();
}
```

27

Iterating through a linked list: printing it



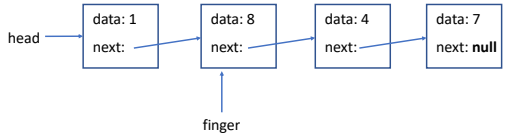
1

```
Node finger = head;

while (finger != null ) {
    System.out.println(finger.value());
    finger = finger.next();
}
```

28

Iterating through a linked list: printing it



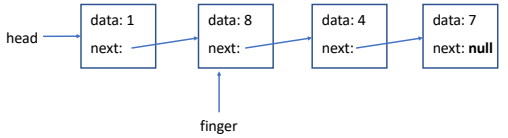
1

```
Node finger = head;

while (finger != null ) {
    System.out.println(finger.value());
    finger = finger.next();
}
```

29

Iterating through a linked list: printing it



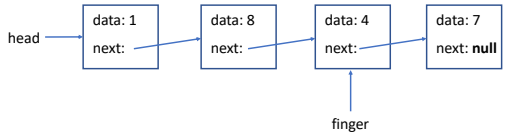
1
8

```
Node finger = head;

while (finger != null ) {
    System.out.println(finger.value());
    finger = finger.next();
}
```

30

Iterating through a linked list: printing it



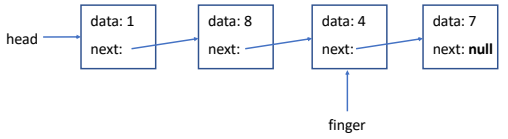
1
8

```
Node finger = head;

while (finger != null ) {
    System.out.println(finger.value());
    finger = finger.next();
}
```

31

Iterating through a linked list: printing it



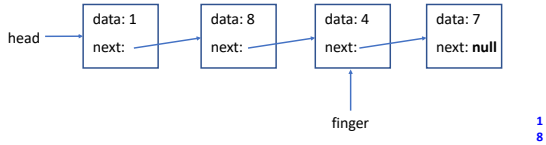
1
8
4

```
Node finger = head;

while (finger != null ) {
    System.out.println(finger.value());
    finger = finger.next();
}
```

32

Iterating through a linked list: printing it



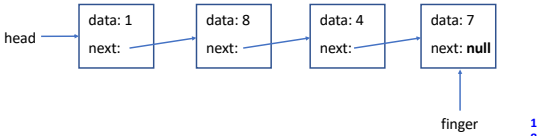
1
8
4

```
Node finger = head;

while (finger != null ) {
    System.out.println(finger.value());
    finger = finger.next();
}
```

33

Iterating through a linked list: printing it



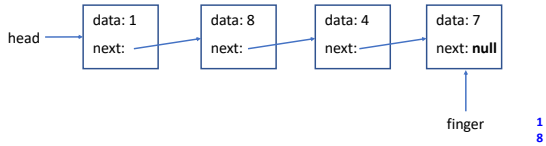
1
8
4

```
Node finger = head;

while (finger != null ) {
    System.out.println(finger.value());
    finger = finger.next();
}
```

34

Iterating through a linked list: printing it



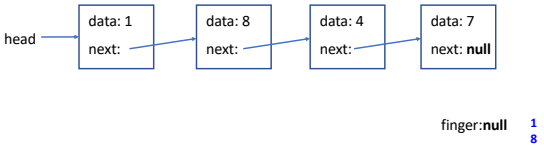
1
8
4
7

```
Node finger = head;

while (finger != null ) {
    System.out.println(finger.value());
    finger = finger.next();
}
```

35

Iterating through a linked list: printing it



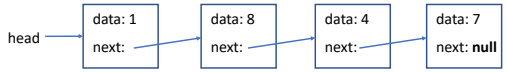
1
8
4
7

```
Node finger = head;

while (finger != null ) {
    System.out.println(finger.value());
    finger = finger.next();
}
```

36

Iterating through a linked list: printing it



finger:**null**

```
Node finger = head;

while (finger != null ) {
    System.out.println(finger.value());
    finger = finger.next();
}
```

```
1
8
4
7
```

37