Link Nodes
Introduction

• Linked node: a class containing one or more data fields that store data, and a *reference* to another linked node
• The data can be a primitive type or an object
• Linked nodes connect objects together to form a list (chain) of link nodes
• Linked nodes are the building blocks of programs (data structures) that store a large number of data without using an array.
Node class

• Below are two examples of linked nodes classes.

```java
public class Node {
    public Node next; // Point to next node
    public String data; // Value (String) for this node data
    // Constructor
    public Node(String data, Node next) {
        this.data = data;
        this.next = next;
    }

    // data fields are public
    // no need for getters and setters
}
```

```java
public class Node {
    public Node next; // Point to next node
    public Computer data; // Value (Computer) for this node data
    // Constructor
    public Node(Computer data, Node next) {
        this.data = data;
        this.next = next;
    }

    // data fields are public
    // no need for getters and setters
}
```

This node will store a String value

This node will store a Computer value
Chain of nodes

- Let's build a chain of nodes.
- Each node stores an integer value

```java
Node head = new Node(20, null);
```

The value stored in head’s pointee is a 20

The next (following) node of head’s pointee is a null reference
Chain of nodes

- `head.next = new Node(30, null);`

Memory

- The next (follower) node of head’s pointee is a `new Node` storing 30
- Update/add a new node at the end of the chain
Chain of nodes

- `head.next.next = new Node(10, null);`

- Returns head’s next (follower) node

- The next (following) node of head’s follower is a new Node storing 10
Chain of nodes

Putting everything together:

```java
Node head = new Node(20, null);
head.next = new Node(30, null);
head.next.next = new Node(10, null);
```

Will create the following chain:
Chain of nodes: iteration

• To iterate through a chain of nodes:
• We don’t need to know how many nodes are in the chain
• The last node next field points to a null reference
• Steps:
  1. Create a temporary node that points to the head of the chain (sharing)
  2. Iterate/loop by following the next references with each iteration, update the pointee of the temporary node
  3. Stop when the temporary node points to a null reference
Chain of nodes: iteration

- Given the following chain
Chain of nodes: iteration

- Create a temporary node that points to the head of the chain

Node curr = head;  // curr and head are aliases for each other
Chain of nodes: iteration

• Create a temporary node that points to the head of the chain
  `Node curr = head; // curr and head are aliases for each other`
• Start the loop we stop when `curr` points to the last node in the chain
  `while(curr != null){  // the pointee of curr is not null`
  `      curr = curr.next; // we advance curr`
  }`
Chain of nodes: iteration

• Curr now points to the node storing 30

```java
while(curr != null){  // the pointee of curr is not null
    curr = curr.next;  //we advance curr
}
```

Note that head did not move.
Chain of nodes: iteration

• Curr now points to the node storing 10

```java
while(curr != null){  // the pointee of curr is not null
    curr = curr.next;  // we advance curr
}
```
Chain of nodes: iteration

- Curr now points to the node storing 5

```java
while (curr != null) {
    // the pointee of curr is not null
    curr = curr.next;  // we advance curr
}
```
Chain of nodes: iteration

• Curr now points to a null reference

```java
while (curr != null) {
    // the pointee of curr is now null
    curr = curr.next;  // we exit the loop
}
```
Chain of nodes: iteration

- Putting everything together:
- The following code will print all the values stored in our chain

```java
Node curr = head;
while(curr != null){
    System.out.print(curr.data);
    curr = curr.next;
}
```

Will print: 20 30 10 5
Chain of nodes: iteration (for loop)

• Putting everything together:
• The following code will print all the values stored in our chain

```java
for(Node curr = head; curr != null; curr = curr.next){
    System.out.print(curr.data);
}
```

Will print: 20 30 10 5