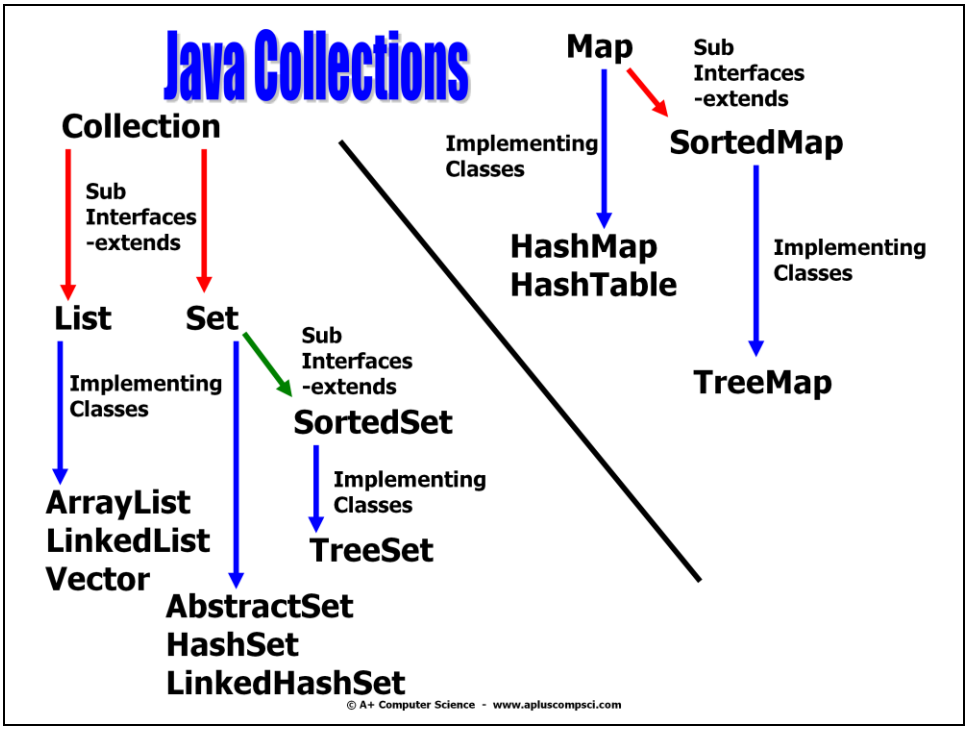
Linked Lists

Java LinkedList


Java Collections
Collection
Sub Interfaces -extends
List
Set
Implementing Classes
ArrayList
LinkedList
Vector
Sub Interfaces -extends
SortedSet
Implementing Classes
TreeSet
AbstractSet
HashSet
LinkedHashSet
Map
Sub Interfaces -extends
SortedMap
Implementing Classes
HashMap
HashTable
Implementing Classes
TreeMap
© A+ Computer Science - www.apluscompsci.com

LinkedList Methods

LinkedList

frequently used methods

Name	Use
add(x)	adds item x to the list
set(x,y)	set location x to the value y
get(x)	get the item at location x
size()	returns the # of items in the list
remove()	removes an item from the list
clear()	removes all items from the list

```
import  java.util.LinkedList;
```

add()

```
LinkedList<String> list;
list = new LinkedList<String>();

list.add("c");
list.add("b");
list.add("a");
list.add(1, "d");

out.println(list);
```

OUTPUT

[c, d, b, a]

get()

```
LinkedList<String> list;
list = new LinkedList<String>();

list.add("c");
list.add("b");
list.add("a");
list.add(1, "d");

out.println(list.get(0) );
out.println(list.get(1) );
out.println("first " + list.getFirst());
out.println("last " + list.getLast());
```

OUTPUT

```
c
d
first c
last a
```

Open
linkedlistadd.java
linkedlistget.java

Open linkedlistdemo.java

Start work on the labs

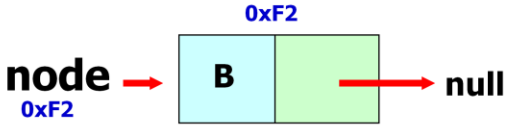
Linked Lists

A linked list is a group of nodes. Each node contains a value and a reference to the next node in the list.

Simple Node Class

```
public class Node
{
  private Comparable data;
  private Node next;

  public Node(Comparable dat, Node nxt)
  {
    data=dat;
    next=nxt;
  }
}
```

A Single Node



A node typically has a data component and a reference to the next node.

Linkable Interface

```
public interface Linkable
{
  Comparable getValue();
  Linkable getNext();
  void setNext(Linkable next);
  void setValue(Comparable value);
}
```

The ListNode Class

```
public class ListNode implements Linkable
{
  private Comparable listNodeValue;
  private ListNode nextListNode;

  public ListNode(){
    listNodeValue = null;
    nextListNode = null;
  }

  public ListNode(Comparable value, ListNode next)
  {
    nextListNode = next;
    listNodeValue = value;
  }

 //other methods not shown
 //refer to the Linkable interface
}
```

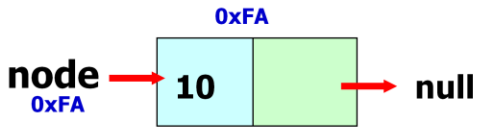
This ListNode class is similar to the AP ListNode.

You can obtain the official AP ListNode class from the college board website. You will be provided with a copy of the AP ListNode class when you take the AP Computer Science AB exam.

Creating A Single Node

```
Linkable node = new ListNode("10", null);
out.println(node.getValue());
out.println(node.getNext());
```

OUTPUT
10
null



Open onenode.java

Linking Nodes

Linking Nodes

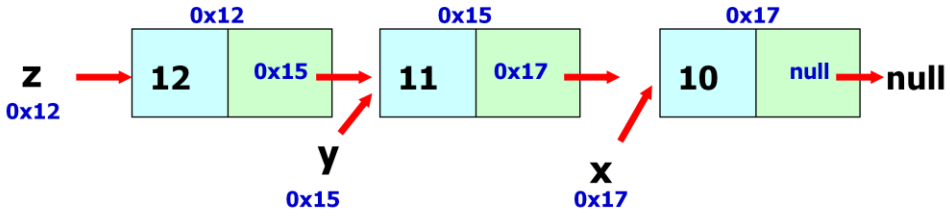
OUTPUT
10
12
11

```
ListNode x = new ListNode("10",
                 new ListNode("11",
                 new ListNode("12",null)));

out.println(x.getValue());
out.println(x.getNext().getNext().getValue());
out.println(x.getNext().getValue());
```

Open linkone.java

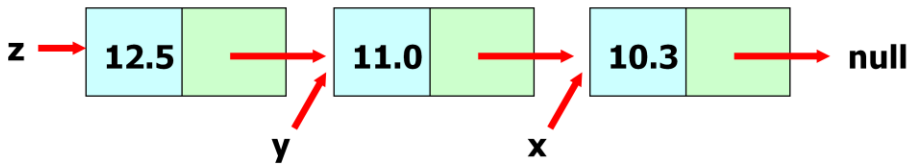
Linking Nodes

```
ListNode x = new ListNode("10", null);

ListNode y = new ListNode("11",x);

ListNode z = new ListNode("12",y);
```



Linking Nodes

```
ListNode x = new ListNode(10.3, null);

ListNode y = new ListNode(11.0, x);

ListNode z = new ListNode(12.5, y);
```



Linking Nodes

```
ListNode x = new ListNode(10.3, null);
ListNode y = new ListNode(11.0, x);
ListNode z = new ListNode(12.5, y);

out.println(z.getValue());
out.println(z.getNext().getNext().getValue());
out.println(z.getNext().getValue());
```

OUTPUT

```
12.5
10.3
11.0
```

Open linktwo.java

Using Loops With Lists

Printing All Nodes

```
ListNode x = new ListNode("10",
                 new ListNode("11",
                 new ListNode("12",null)));

while( x != null )
{
  out.println( x.getValue() );
}
```

OUTPUT

```
10
10
10
. . .
```

Open printone.java

Printing All Nodes

```
ListNode x = new ListNode("10",
                 new ListNode("11",
                 new ListNode("12",null)));

while( x != null )
{
  out.println( x.getValue() );
  x = x.getNext();
}
```

OUTPUT

```
10
11
12
```

Open printtwo.java

Summing All Nodes

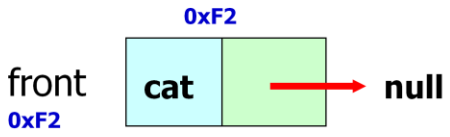
```
ListNode x = new ListNode(11,
                 new ListNode(8,
                 new ListNode(5,null)));

int sum=0;
while( x != null )
{
  sum = sum + (Integer)x.getValue();
  x = x.getNext();
}
out.println(sum);
```

OUTPUT
24

Open sumone.java

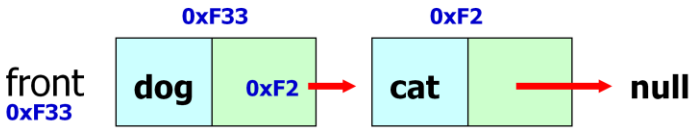
Adding Nodes

add



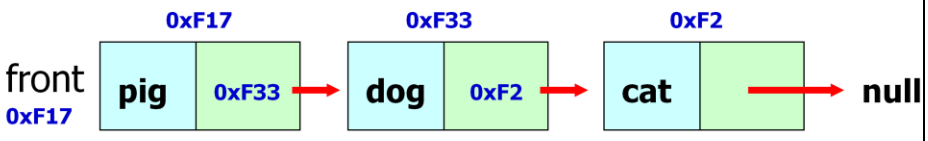

```
front = new ListNode(word, null);
```

add




```
front = new ListNode(word, front);
```

add




```
front = new ListNode(word, front);
```


```
ListNode front=null;
front = new ListNode("10", front);
front = new ListNode("11",front);
front = new ListNode("12",front);

out.println(front.getValue());
out.println(front.getNext().getNext().getValue());
out.println(front.getNext().getValue());
```

OUTPUT

```
12
10
11
```

Open add_java

Finding A Node

Finding a Node

```
ListNode list = front;
while ( there are more nodes to check )
{
  if( a node containing the value was found )
    return true;
  move to the next node to check
}
return false;
```

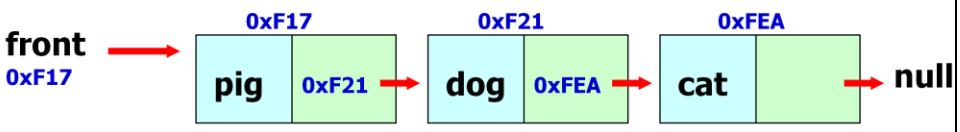
Open contains.java

Removing Nodes

Removing the First Node

Removing the first node

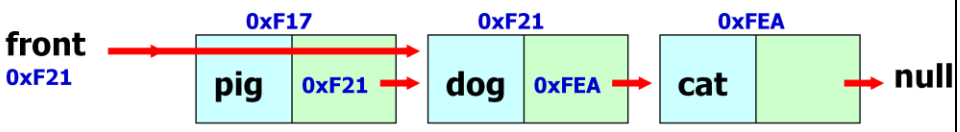
pig is to be removed.



front refers to the 1st node in the list.

Removing the first node

```
front = front.getNext();
```

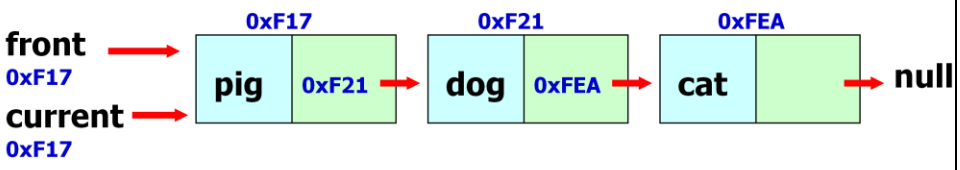


front moves up one node.

Removing any Node

Removing any node

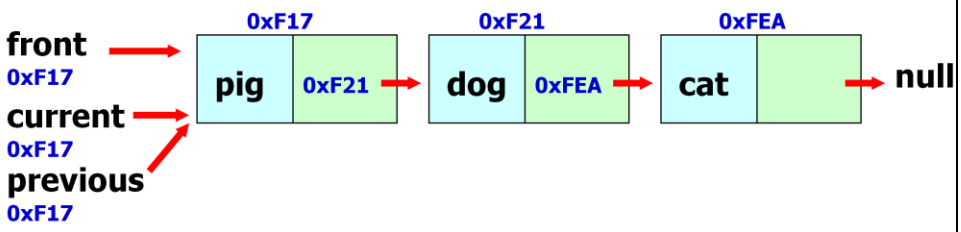
dog is to be removed.
current = front;



front and current store the same memory address.

Removing any node

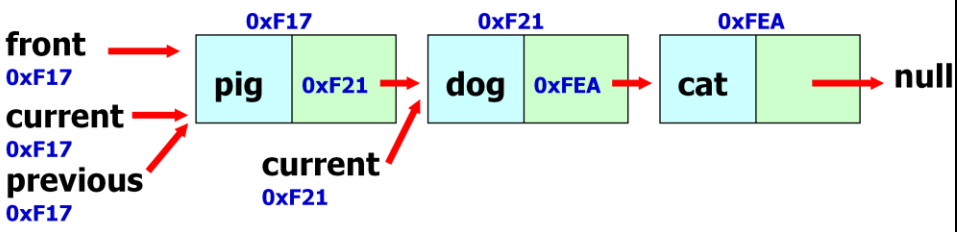
```
previous = current;
```



front, current, and previous all store the same memory address.

Removing any node

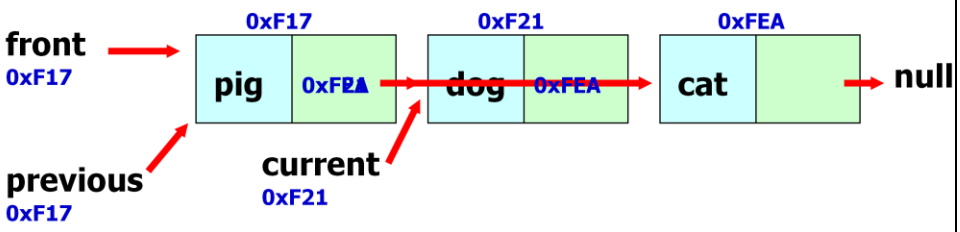
```
current=current.getNext();
```



current moves forward one node.

Removing any node

```
previous.setNext(current.getNext());
```



Found dog. Removed dog from the list.

Removing any node

Some things you have to account for!

1. What if the linked list is null?
2. What if I need to remove the 1st node?
3. How do I process the remaining nodes?
4. Do I remove more than 1 occurrence of the same value or just the 1st one?

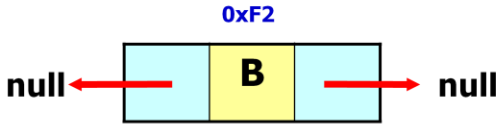
Open remove.java

Doubly Linked Lists

DoublyNode Class

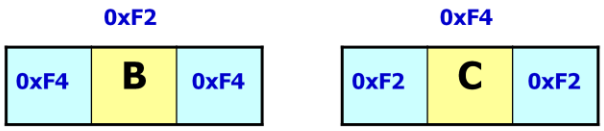
```
class DoublyNode
{
  private Comparable data;
  private DoublyNode next;
  private DoublyNode prev;

  public DoublyNode(Comparable dat,
                    DoublyNode prv, DoublyNode nxt)
  {
    data=dat;
    prev=prv;
    next=nxt;
  }

  //other methods not shown
}
```

A DoublyNode



A doubly node typically has a data component and a reference to the next node and the previous node.

A DoublyNode



Doubly nodes can be used to make a circular linked list where the front points at the back and vice versa.

BigO

Big-O Notation

Big-O notation is an assessment of an algorithm's efficiency. Big-O notation helps gauge the amount of work that is taking place.

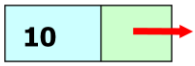
Common Big O Notations :

$O(1)$	**$O(\text{Log}_2 N)$**
$O(2^N)$	**$O(N^2)$**
$O(N \text{ Log}_2 N)$	**$O(N)$**
$O(\text{Log}_2 N)$	**$O(N^3)$**

Single LL BigO

traverse all nodes	O(N)
search for an item	O(N)
remove any item location unknown	O(N)
get any item location unknown	O(N)
add item at the end	O(N)
add item at the front	O(1)

A single linked list node has a reference to the next node only. A single linked list node has no reference to the previous node.



Java LL BigO

traverse all spots	O(N)
search for an item	O(N)
remove any item location unknown	O(N)
get any item location unknown	O(N)
add item at the end	O(1)
add item at the front	O(1)

LinkedList is implemented with a double linked list.

Continue work on the labs