Lecture 14: Linked Lists and ADTs Stack and Queue

CS2504/CS4092– Algorithms and Linear Data Structures

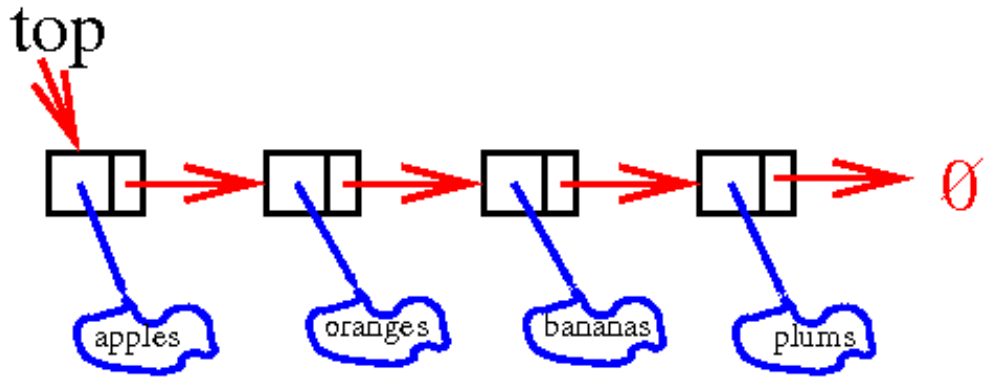
Dr. Kieran T. Herley

Department of Computer Science
University College Cork

2013/14

List Implementation of ADT Stack

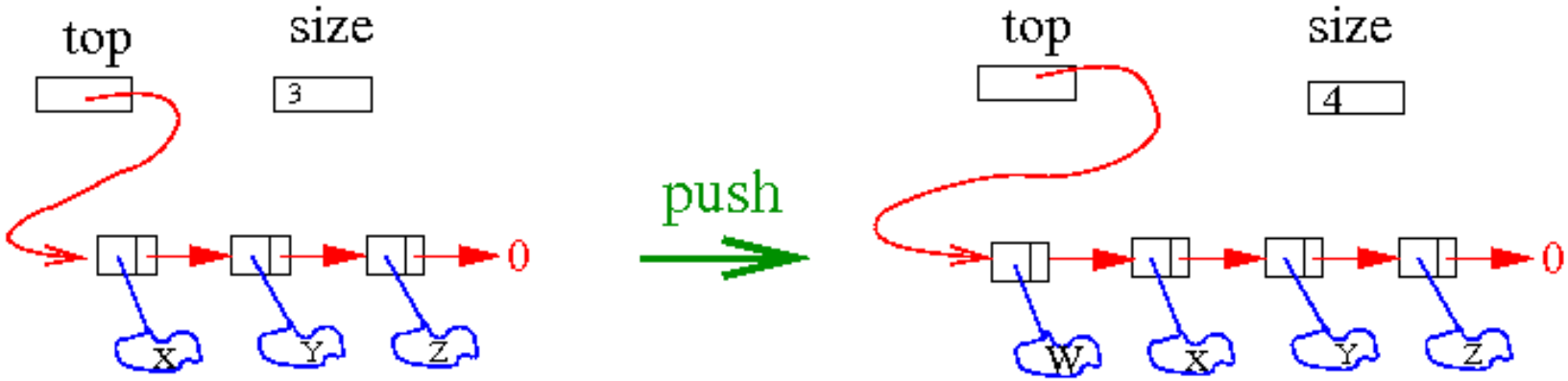
- **Representation:**



  - list of nodes (LLNode) containing stack elements, arranged top to bottom
  - reference to top node
  - int size representing list length (not shown)

List Implementation of ADT Stack

- **push:**



- **push(o):**
  - Create new node containing *o*
  - Attach new node at head of list
  - Adjust top/size

List Implementation of ADT Stack

- **pop:**

- **pop():**
  - Remove node at head of list (by adjusting top)
  - Adjust size
  - Return element of newly-deleted node

Interface for ADT Stack

Recall file Stack.java specifies methods supported by ADT:

```
public interface Stack<EltType>
{
    public int  size ();
    public boolean isEmpty();
    public EltType top()  ;
    public void push (EltType element);
    public EltType pop();
}
```

The following implementation though totally different from earlier array-based implementation of ADT shares *precisely* the same interface.

List Implementation of ADT Stack

Interface `public interface Stack<EltType>`

```
{
    public int  size ();
    public boolean isEmpty();
    public EltType top()  ;
    public void push (EltType element);
    public EltType pop();
}
```

Implementation `public class LinkedStack<EltType>`

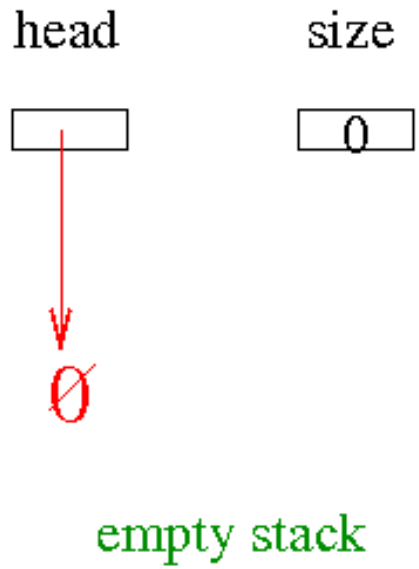
```
        implements Stack<EltType>
{    /* implementation containing instance variables ,
         constructors , methods for push, pop etc.
}
```

List Implementation cont'd

```
public class LinkedStack<EltType>
        implements Stack<EltType>
{   public LinkedStack()
    {  top = null;  size = 0;
    }
       . . .
    // Code of methods push etc
       . . .
    private LLNode<EltType> top;
    private int size ;
}
```

List Implementation cont'd

```
public class LinkedStack<EltType>
        implements Stack<EltType>
{   public LinkedStack()
    {  top = null;  size = 0;
    }
       . . .
    // Code of methods push etc
       . . .
    private LLNode<EltType> top;
    private int size ;
}
```

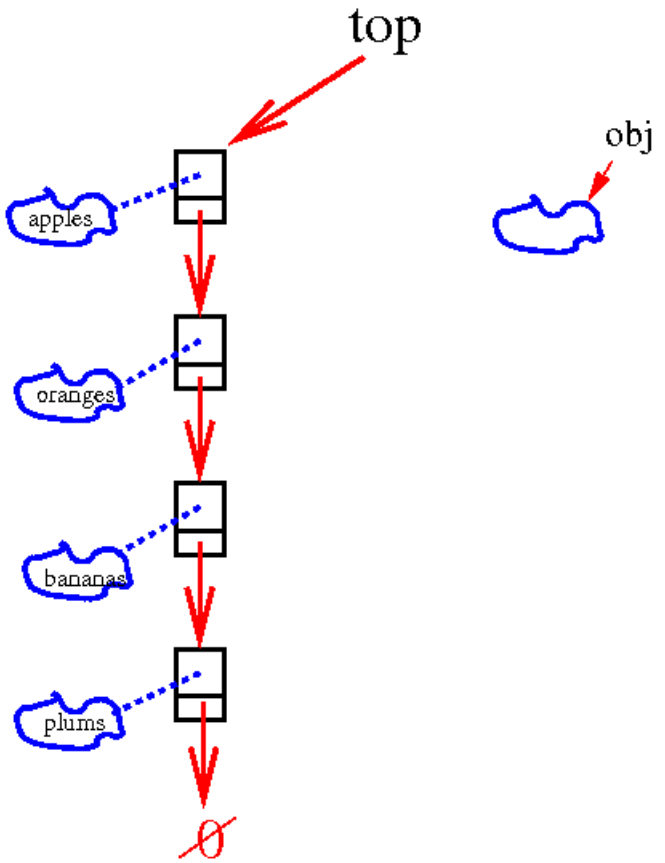


Class LLNode– Reminder

```
public class LLNode<EltType>
{   public LLNode()
    {   element = null; next = null;
    }
    public LLNNode(LLNode<EltType> n, EltType s)
    {   element = s; next = n;
    }
    // public getters and setters
    private EltType element;
    private LLNode<EltType> next;
}
```

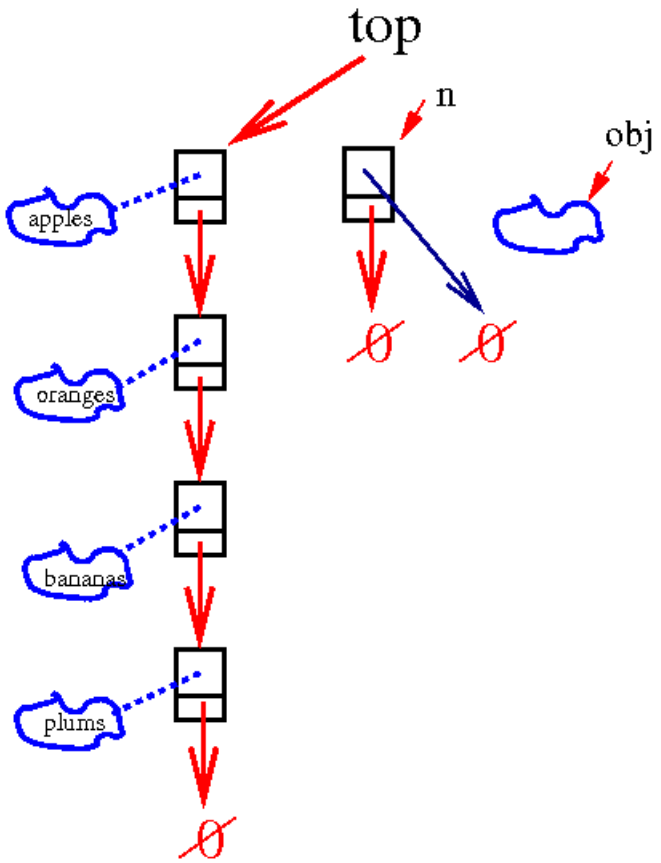
List Impl. of ADT Stack cont'd

```
public void push(EltType obj)
{
    LLNode<ElType> n =
        new LLNode<EltType>();
    n.setElement(obj);
    n.setNext(top);
    top = n;
    size++;
}
```



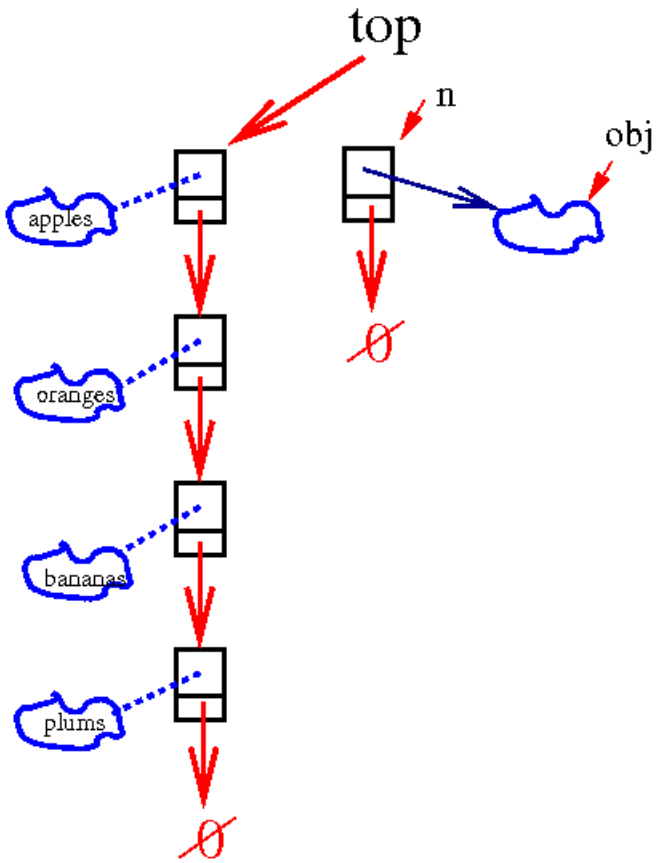
List Impl. of ADT Stack cont'd

```
public void push(EltType obj)
{
    LLNode<ElType> n =
        new LLNode<EltType>();
    n.setElement(obj);
    n.setNext(top);
    top = n;
    size++;
}
```



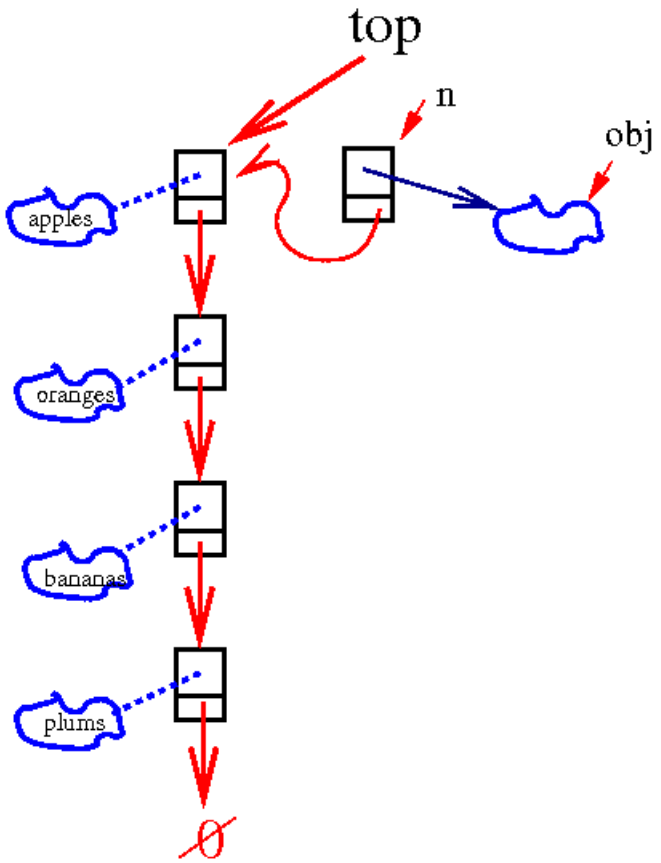
List Impl. of ADT Stack cont'd

```
public void push(EltType obj)
{
    LLNode<ElType> n =
        new LLNode<EltType>();
    n.setElement(obj);
    n.setNext(top);
    top = n;
    size++;
}
```



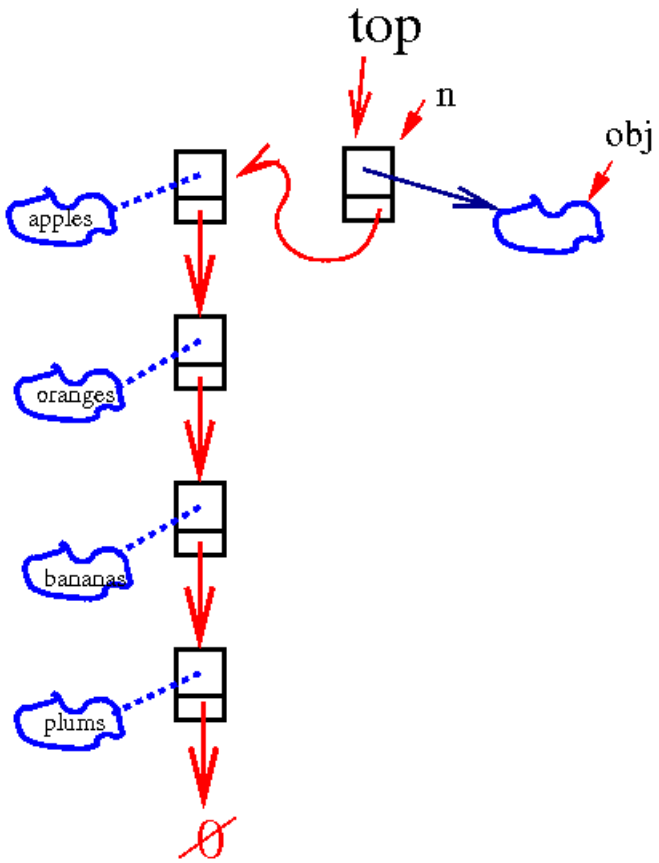
List Impl. of ADT Stack cont'd

```
public void push(EltType obj)
{
    LLNode<ElType> n =
        new LLNode<EltType>();
    n.setElement(obj);
    n.setNext(top);
    top = n;
    size++;
}
```



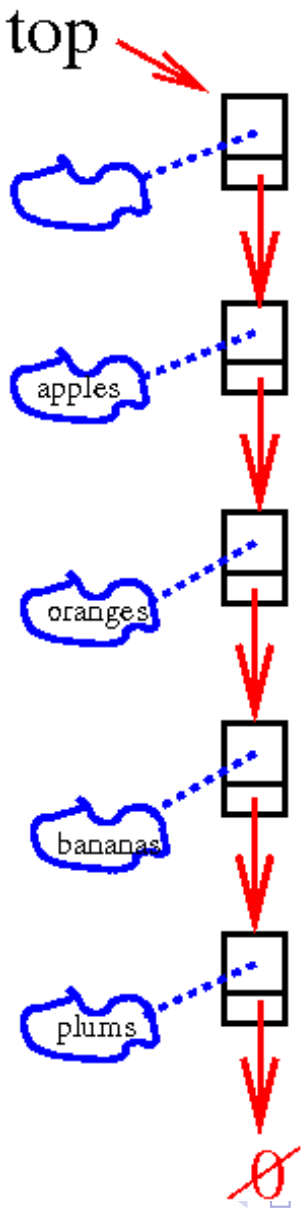
List Impl. of ADT Stack cont'd

```
public void push(EltType obj)
{
    LLNode<ElType> n =
        new LLNode<EltType>();
    n.setElement(obj);
    n.setNext(top);
    top = n;
    size++;
}
```



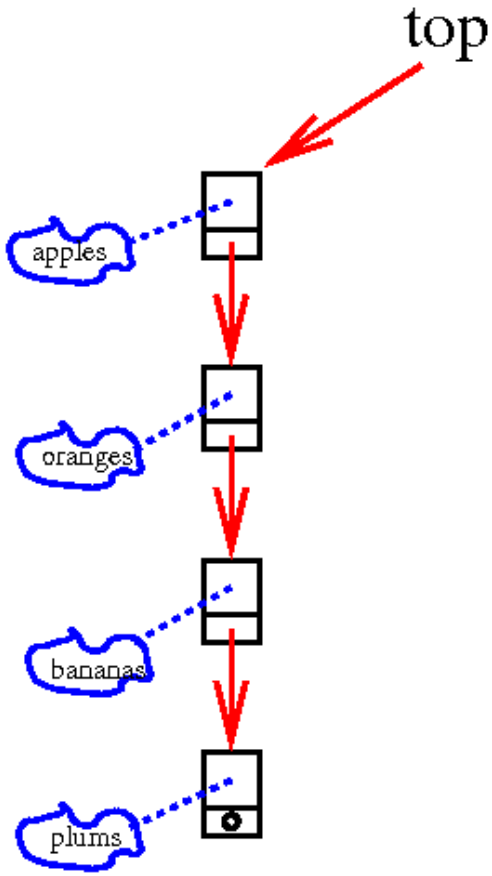
List Impl. of ADT Stack cont'd

```
public void push(EltType obj)
{
    LLNode<ElType> n =
        new LLNode<EltType>();
    n.setElement(obj);
    n.setNext(top);
    top = n;
    size++;
}
```



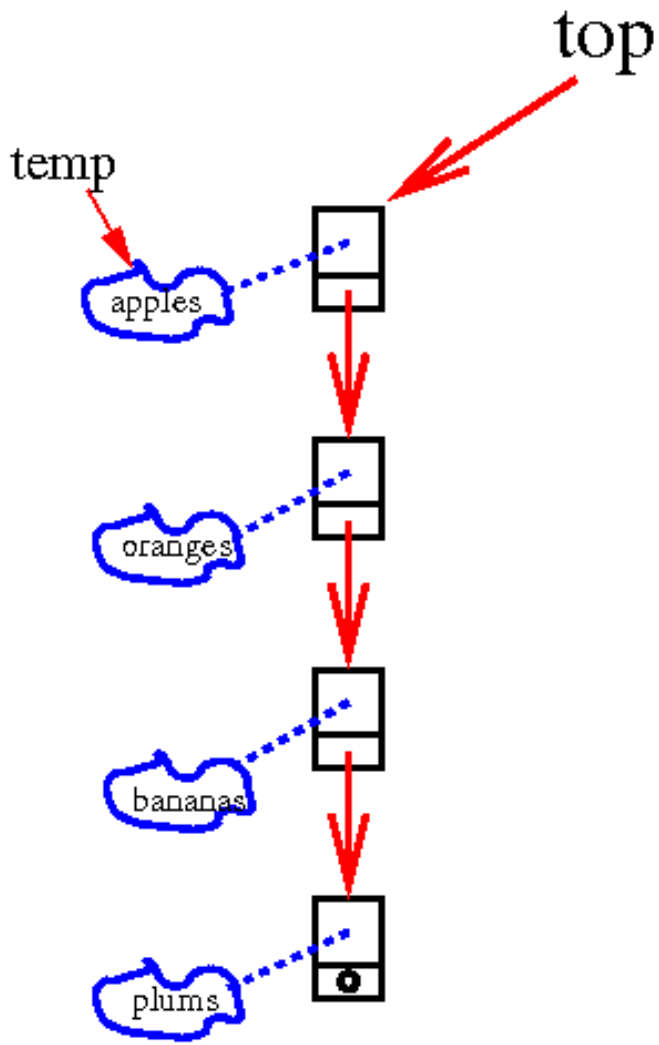
List Impl. of ADT Stack cont'd

```
public EltType pop()
{    EltType temp;
     if (isEmpty()){. . . }
     temp = top.getElement();
     top = top.getNext();
     size−−;
     return temp;
}
```



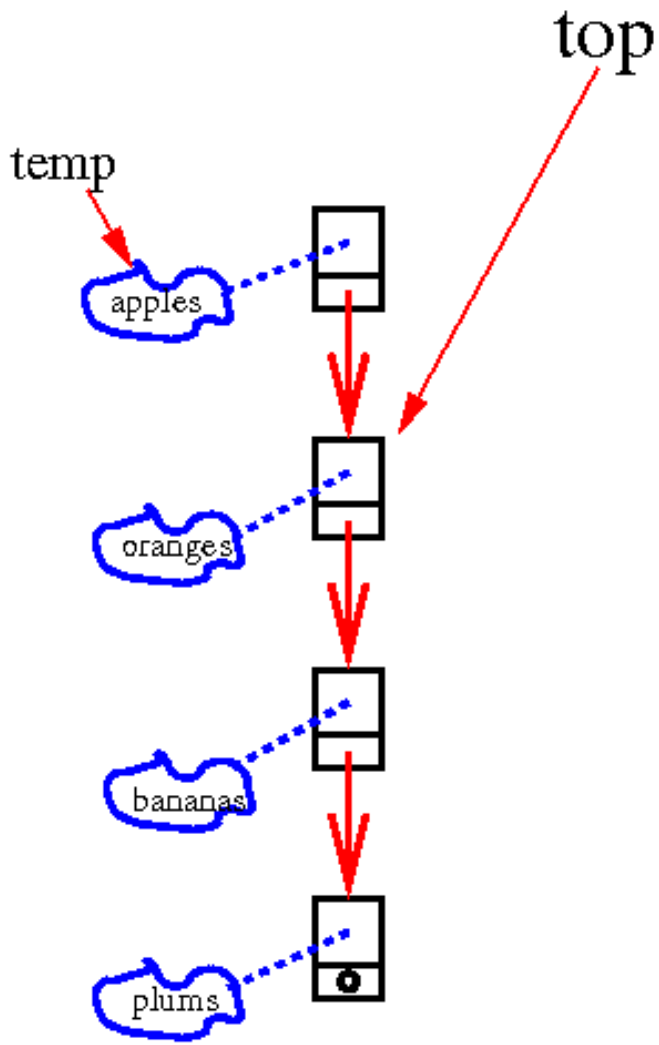
List Impl. of ADT Stack cont'd

```
public EltType pop()
{    EltType temp;
     if (isEmpty()){. . . }
     temp = top.getElement();
     top = top.getNext();
     size−−;
     return temp;
}
```



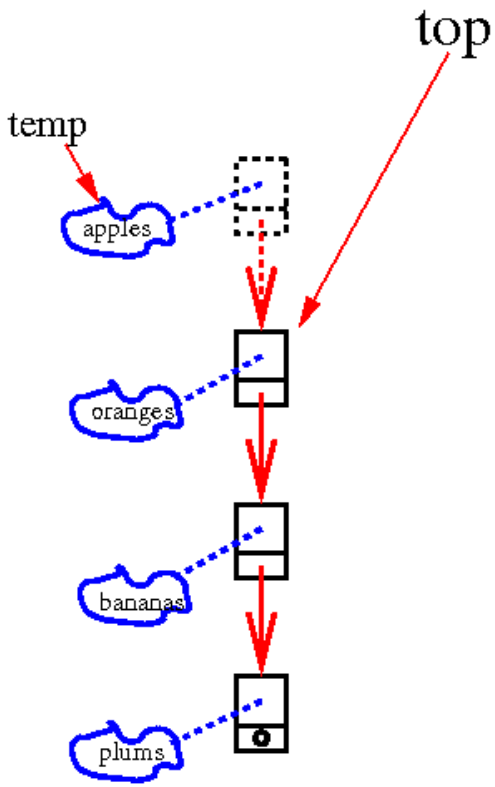
List Impl. of ADT Stack cont'd

```
public EltType pop()
{    EltType temp;
     if (isEmpty()){. . . }
     temp = top.getElement();
     top = top.getNext();
     size−−;
     return temp;
}
```



List Impl. of ADT Stack cont'd

```
public EltType pop()
{    EltType temp;
     if (isEmpty()){. . . }
     temp = top.getElement();
     top = top.getNext();
     size−−;
     return temp;
}
```



Array vs Linked Stacks

Note ArrayBasedStack.java, LinkedStack.java both implement Stack.java; programmer can "plug in" whichever he prefers; stack-manipulating application code same for either implementation

Version 1

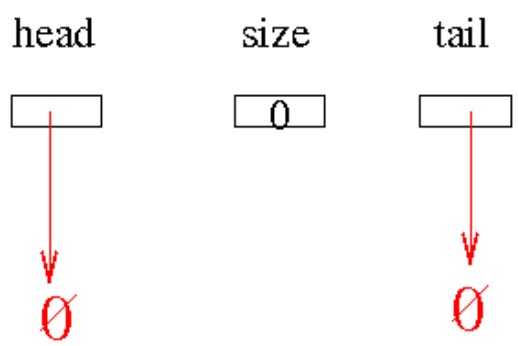
```
Stack<Integer> myStack =
    new ArrayBasedStack<Integer>();
myStack.push(. . .);
myStack.pop();
```

Version 2

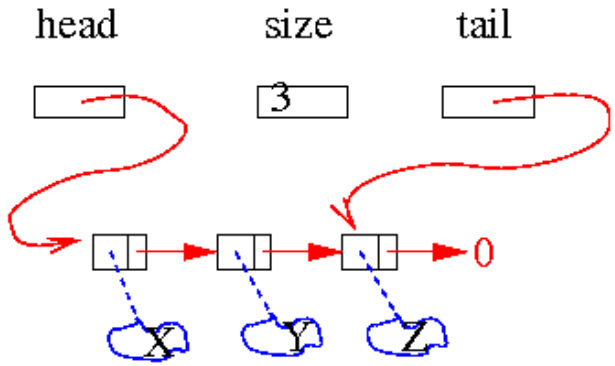
```
Stack<Integer> myStack =
    new LinkedStack<Integer>();
myStack.push(. . .);
myStack.pop();
```

List Implementation of ADT Queue

Representation



empty queue



three–item queue

Details

- list of Nodes containing queue elements, arranged front to rear
- references to head(front) **and tail(rear)** nodes
- int size representing list length.

enqueue

- 

- - Create new node containing *o*
  - Attach new node at rear of list
  - Adjust tail/size

dequeue

- 

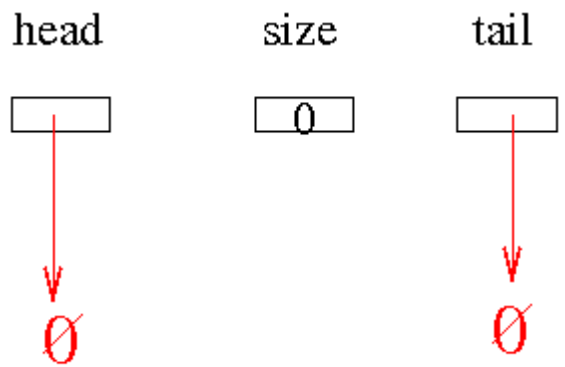
- - Remove node at head of list (by adjusting head)
  - Adjust size
  - Return element of newly-deleted node

The Queue Interface

```
public interface Queue<EltType>
{
  public int  size ();
  public boolean isEmpty();
  public EltType  front ();
  public void enqueue (EltType element);
  public EltType dequeue();
}
```

List Impl. of ADT Queue

```
public class LinkedQueue<EltType>
        implements Queue<EltType>
{   public LinkedQueue()
    {   head = null;
        tail = null;
        size = 0;
    }
    // other methods
    private LLNode<EltType> head;
    private LLNode<EltType> tail;
    private int size;
}
```



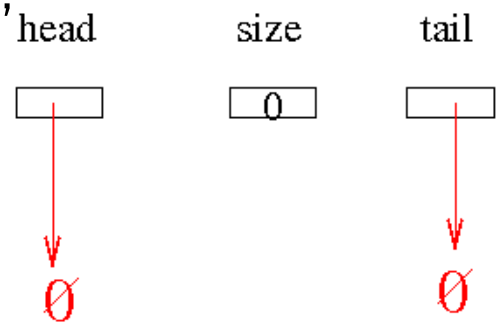
empty queue

Implementation of enqueue

```
public void enqueue(EltType obj)
{    LLNode<EltType> node = new LLNode<EltType>();
     node.setElement(obj);
     node.setNext(null);
     if (size == 0)
       head = node;
     else
        tail.setNext(node);
     tail = node;
     size++;
}
```

Illustration 1

```
public void enqueue(EltType obj)
{    LLNode<EltType> node =
       new LLNode<EltType>();
     node.setElement(obj);
     node.setNext(null);
     if (size == 0)
       head = node;
     else
        tail .setNext(node);
     tail = node;
     size++;
}
```



empty queue

Illustration 1

```
public void enqueue(EltType obj)
{    LLNode<EltType> node =
       new LLNode<EltType>();
     node.setElement(obj);
     node.setNext(null);
     if (size == 0)
       head = node;
     else
        tail .setNext(node);
     tail = node;
     size++;
}
```

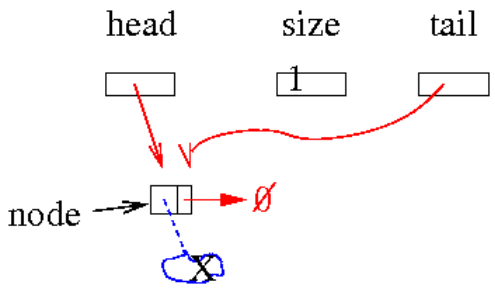


Illustration 2

```
public void enqueue(EltType obj)
{    LLNode<EltType> node =
       new LLNode<EltType>();
     node.setElement(obj);
     node.setNext(null);
     if (size == 0)
       head = node;
     else
        tail .setNext(node);
     tail = node;
     size++;
}
```

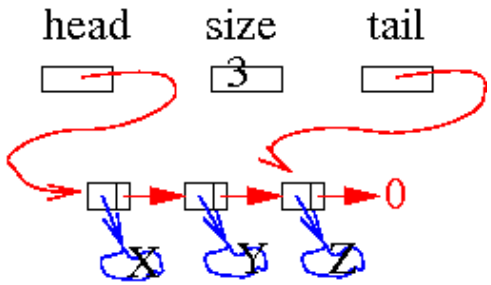
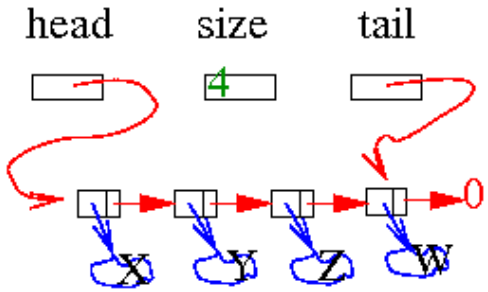


Illustration 2

```
public void enqueue(EltType obj)
{    LLNode<EltType> node =
       new LLNode<EltType>();
     node.setElement(obj);
     node.setNext(null);
     if (size == 0)
       head = node;
     else
        tail .setNext(node);
     tail = node;
     size++;
}
```



Implementation of dequeue

```
public EltType dequeue()
{    EltType obj;
     if (size == 0){. . .}
     obj = head.getElement();
     head = head.getNext();
     size--;
     if (size == 0)
              tail = null;
     return obj;
}
```

Illustration 1

```
public EltType dequeue()
{   EltType obj;
    if (size == 0){. . .}
    obj = head.getElement();
    head = head.getNext();
    size--;
    if (size == 0)
            tail = null;
    return obj;
}
```

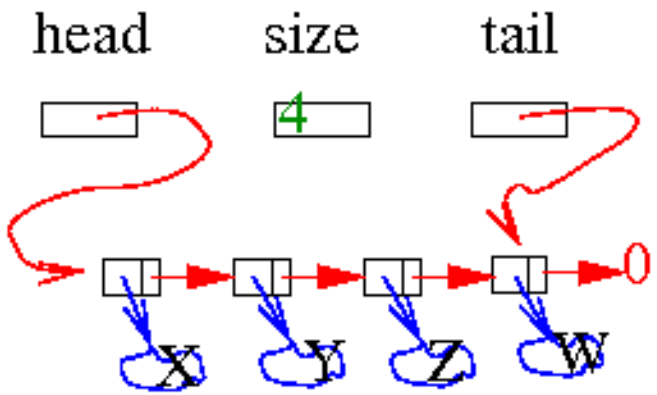


Illustration 1

```
public EltType dequeue()
{   EltType obj;
    if (size == 0){. . .}
    obj = head.getElement();
    head = head.getNext();
    size--;
    if (size == 0)
            tail = null;
    return obj;
}
```

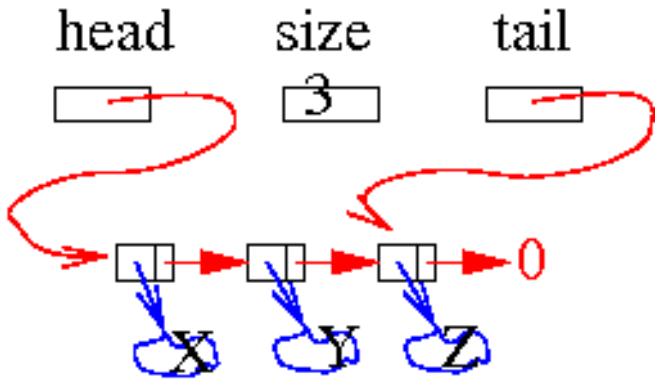


Illustration 2

```
public EltType dequeue()
{   EltType obj;
    if (size == 0){. . .}
    obj = head.getElement();
    head = head.getNext();
    size--;
    if (size == 0)
            tail = null;
    return obj;
}
```

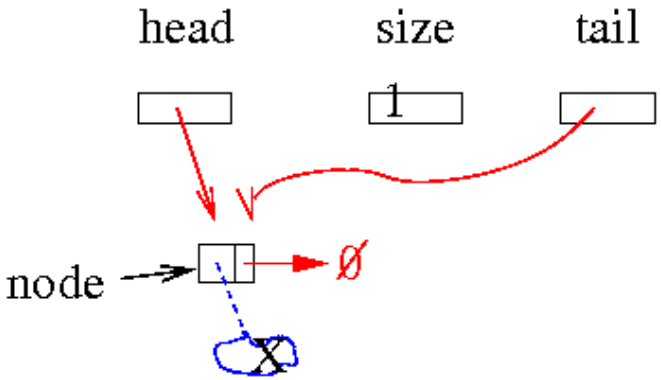
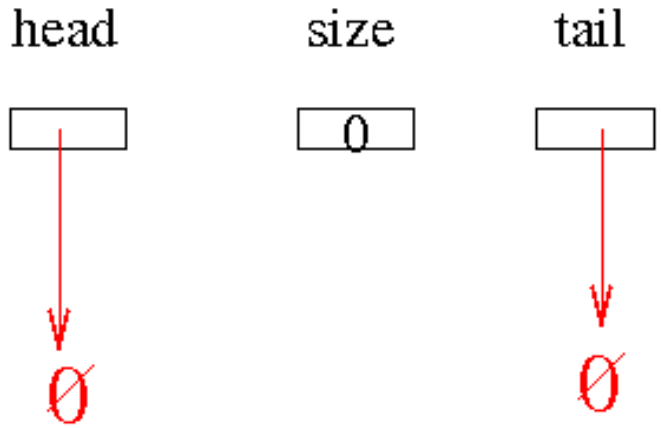


Illustration 2

```
public EltType dequeue()
{   EltType obj;
    if (size == 0){. . .}
    obj = head.getElement();
    head = head.getNext();
    size--;
    if (size == 0)
        tail = null;
    return obj;
}
```



empty queue