Stack: a Linked Implementation

Objectives

- Examine a linked list implementation of the Stack ADT

Another Stack Implementation

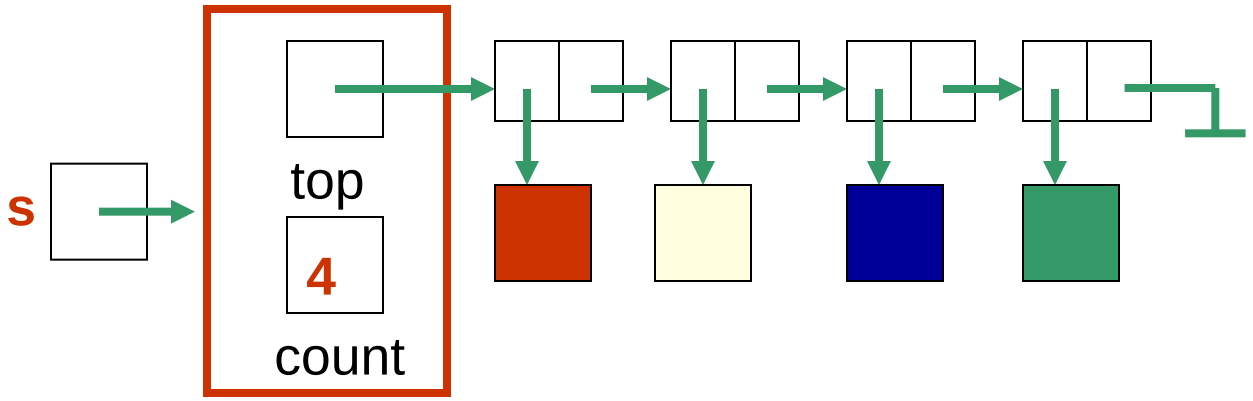
- We will now explore a ***linked list implementation*** of the Stack collection
  - The elements of the stack are stored in *nodes of a linked list*
- It will implement the same interface (Stack ADT) as the array-based implementation; only the underlying data structure changes!

Linked Implementation of a Stack

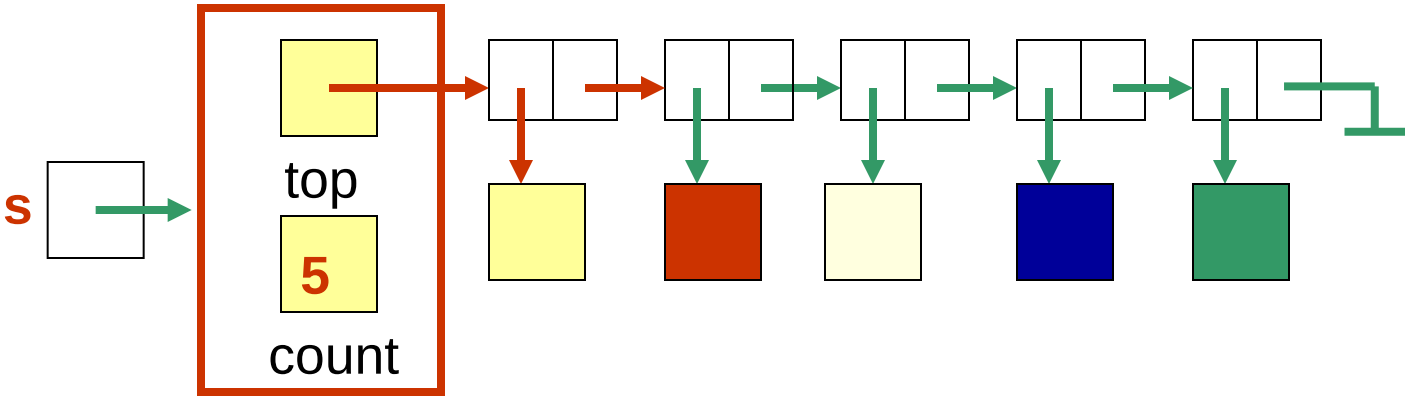
- Recall that we need a container to hold the data elements, and something to indicate the top of the stack.
- Our *container* will be a ***linked list*** **of** ***nodes***, with each node containing a data element.
- The *top of the stack* will be the *first node* of the linked list.
  - So, a reference to the *first node* of the linked list (***top***) is also the reference to the whole linked list!
- We will also keep track of the number of elements in the stack (***count***)

Linked Implementation of a Stack

A stack s with 4 elements



After pushing a fifth element

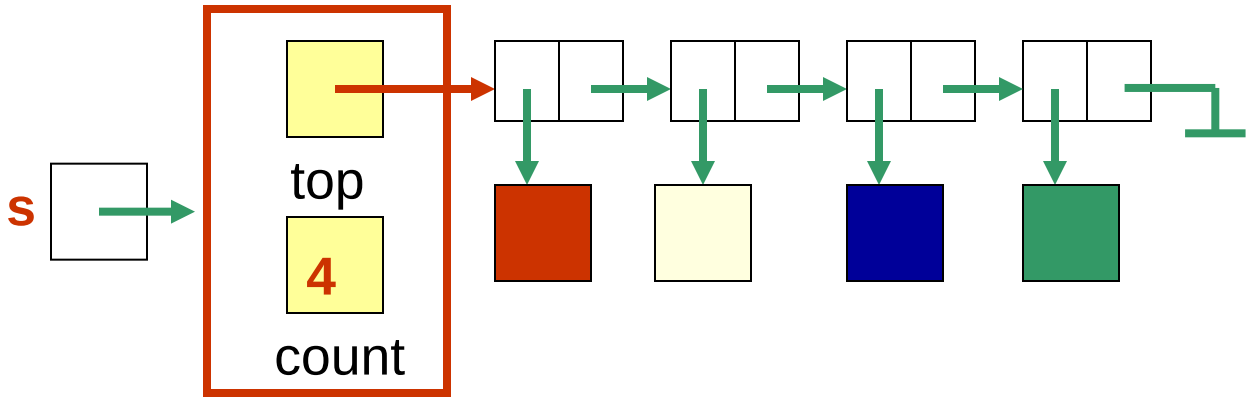


Discussion

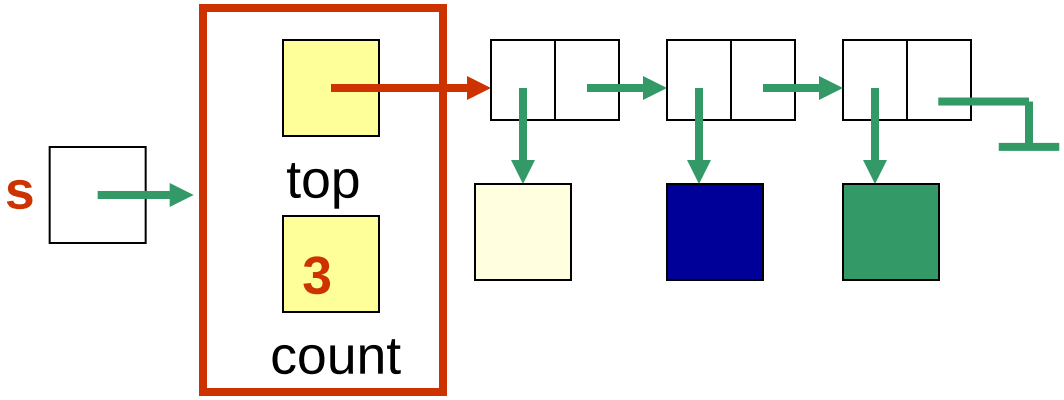
- Where does all the activity take place in a stack (i.e. the pushes and the pops)?
- So, where is this happening in the linked list implementation?

Linked Implementation of a Stack

After popping an element



After popping another element



The **LinkedStack** Class

- Note that it is called "LinkedStack.java" only to differentiate it for us from the array implementation "ArrayStack.java"
- The nodes in the linked list are represented by the LinearNode class defined in the previous topic.
- The attributes (instance variables) are:
  - **top**: a reference to the first node (i.e. a reference to the linked list)
    - So it is of type LinearNode<T>
  - **count**: a count of the current number of elements in the stack

```
//-----------------------------------------------------------
//  Creates an empty stack.
//-----------------------------------------------------------
public LinkedStack ()
{
    top = null;
    count = 0;
}
```

The LinkedStack constructor

```
//-----------------------------------------------------------
//  Adds the specified element to the top of the stack.
//-----------------------------------------------------------
public void push (T element)
{
   LinearNode<T> temp = new LinearNode<T> (element);

   temp.setNext(top);
   top = temp;
   count++;
}
```

The push( ) operation

Where in the linked list is the element added?

```
//-----------------------------------------------------------
//  Removes the element at the top of the stack and returns
//  a reference to it. Throws an EmptyCollectionException if
//  the stack is empty.
//-----------------------------------------------------------
public T pop( ) throws EmptyCollectionException
{
   if (isEmpty( ))
      throw new EmptyCollectionException("Stack" );
   T result = top.getElement( );
   top = top.getNext( );
   count--;
   return result;
}
```

The pop( ) operation

From where in the linked list is the element removed?

The Other Operations

- Write the code for the methods
  - ***peek***
  - ***isEmpty***
  - ***size***
  - ***toString***

Discussion

- Where does the stack grow and shrink?
- What happens when the stack is empty?
- Can the stack be full?