

Topic 7

Stack: a Linked Implementation

5-1

Objectives

- Examine a linked list implementation of the Stack ADT

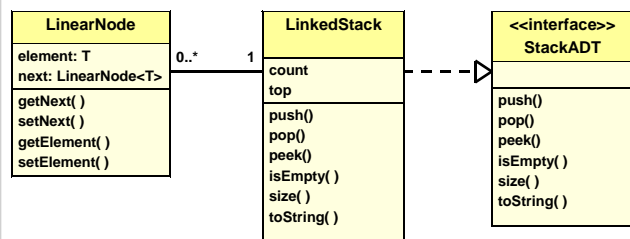
5-2

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!

5-3

UML Description of the LinkedStack Class



5-4

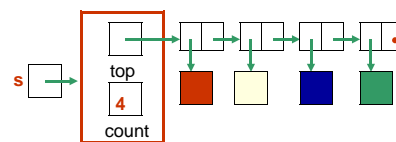
Linked Implementation of a Stack

- 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 need to keep track of the number of elements in the stack (**count**)

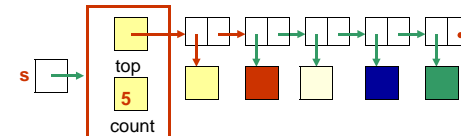
5-5

Linked Implementation of a Stack

A stack *s* with 4 elements



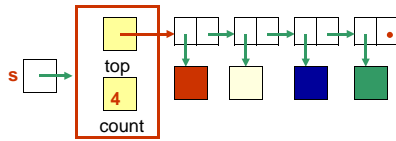
After pushing a fifth element



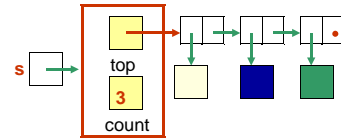
5-6

Linked Implementation of a Stack

After popping an element



After popping another element



5-7

The **LinkedList** Class

- Note that it is called "LinkedList.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

5-8

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

**The
LinkedList
constructor**

5-9

```
//-----
// 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?

5-10

```
//-----
// 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?

5-11

The Other Operations

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

5-12

Discussion

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