Linked Implementation of Stacks
Objectives

• Examine a linked list implementation of the Stack ADT
Another Stack Implementation

• We will now explore a *linked list implementation* of the Stack ADT
  • The data items of the stack are stored in the nodes of a linked list
• This linked list implementation 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 items and a variable to indicate the top of the stack.
• Our container will be a linked list of nodes, with each node containing a data item.
• 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

\[ \text{count} \]

\[ \text{top} \]

\[ 4 \]

\[ s \]

\[ 5 \]

\[ \text{count} \]
Linked Implementation of a Stack

After popping an element

After popping another element
The **LinkedStack** Class

- Note that this class is called “`LinkedStack.java`” only to differentiate it from the array implementation “`ArrayStack.java`”
- The nodes in the linked list are represented by the `LinearNode` class.
- 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 data items in the stack
public LinkedStack ()
{
    top = null;
    count = 0;
}
public void push (T element) {
    LinearNode<T> temp = new LinearNode<T> (element);
    temp.setNext(top);
    top = temp;
    count++;
}
public T pop() throws EmptyCollectionException {
    if (isEmpty())
        throw new EmptyCollectionException("Stack");
    T result = top.getElement();
    top = top.getNext();
    count--;
    return result;
}
The Other Operations

• Write the code for the methods

  • peek
  • isEmpty
  • size
  • toString
Discussion

• What happens when the stack is empty?
• Can the stack be full?