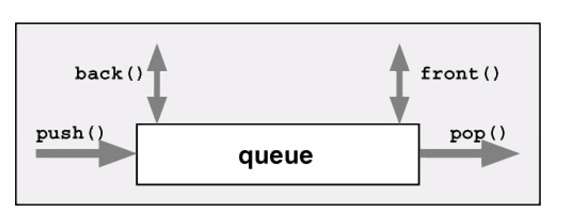
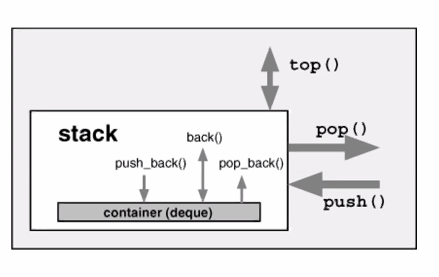
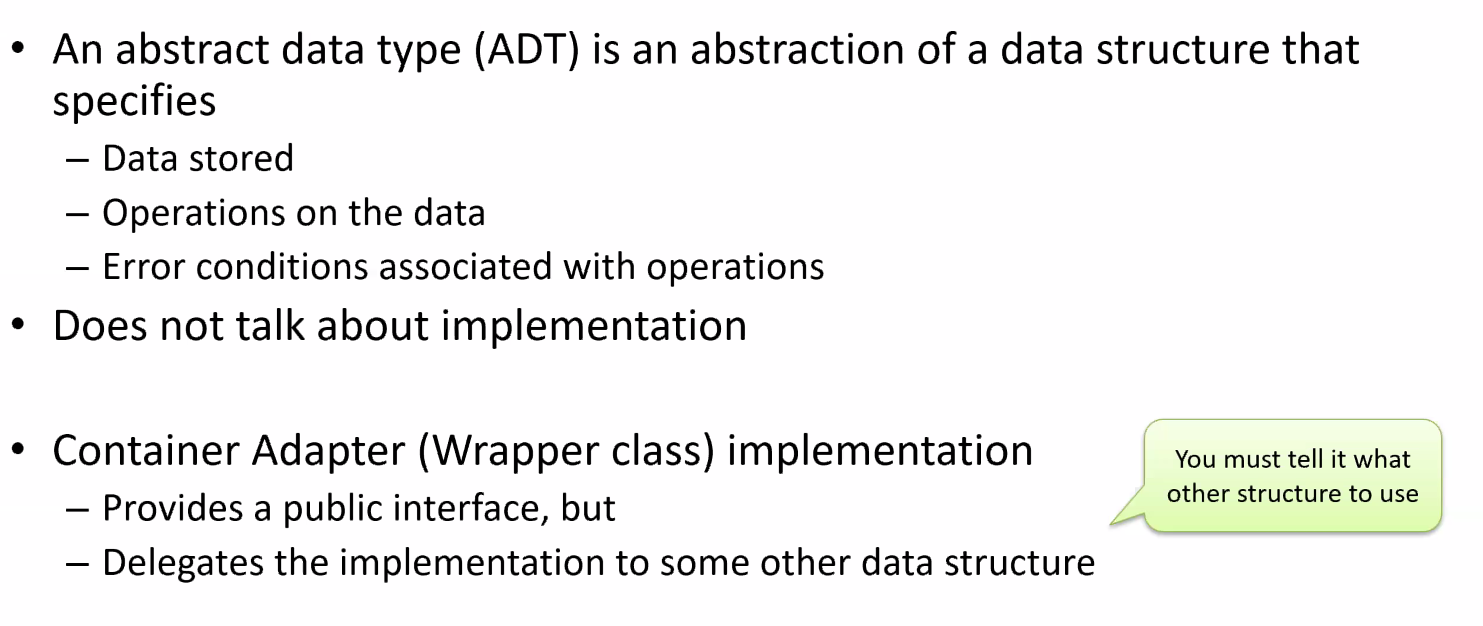
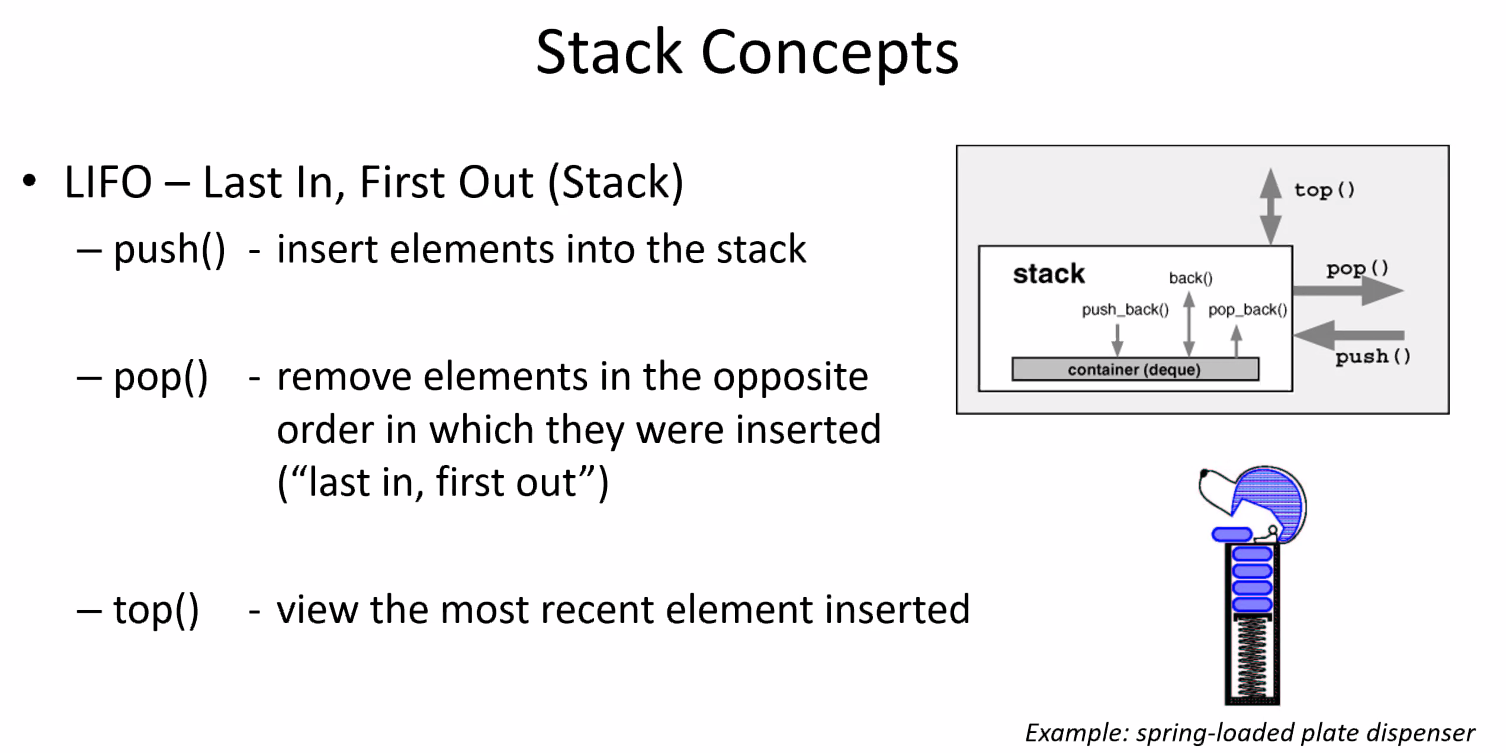
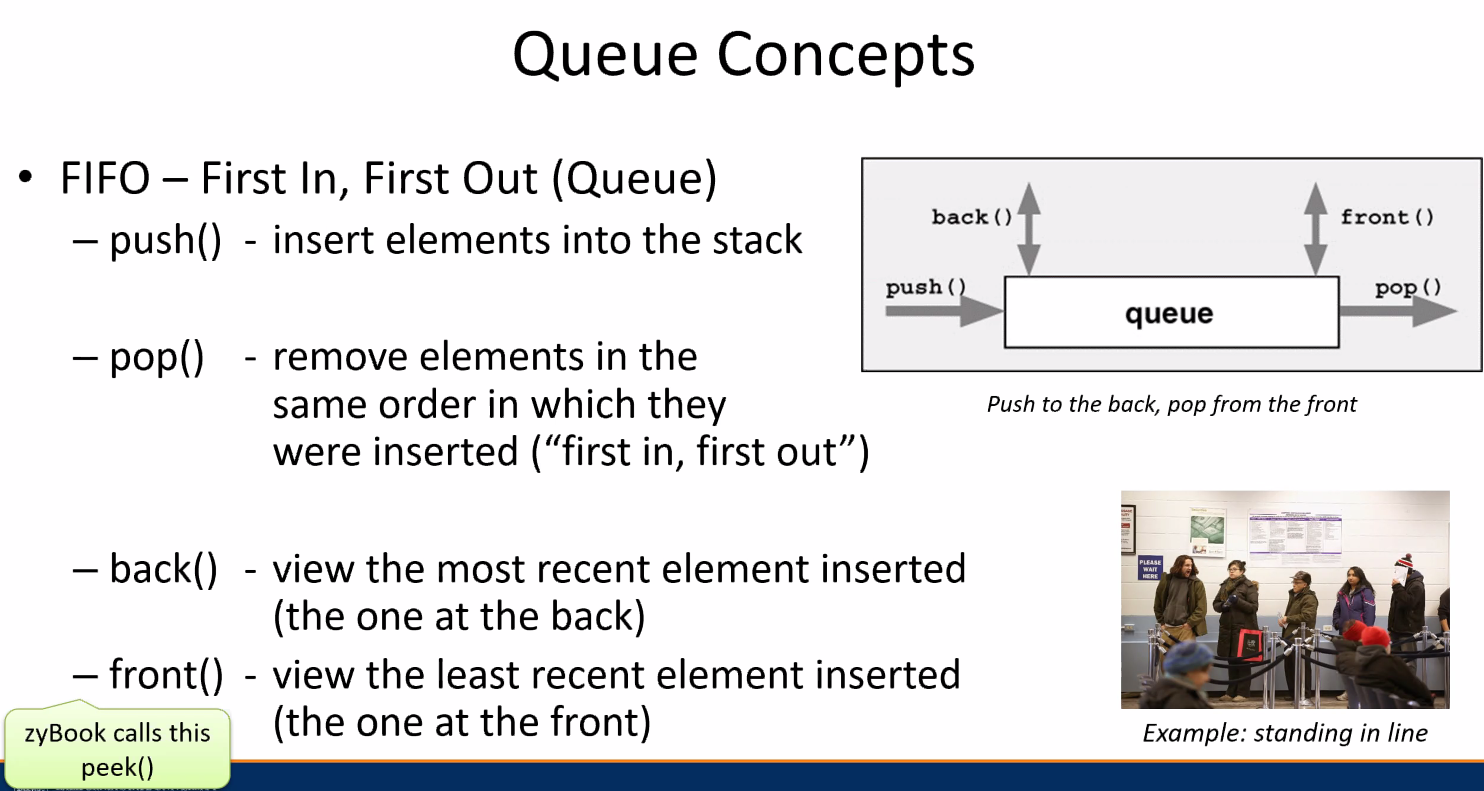
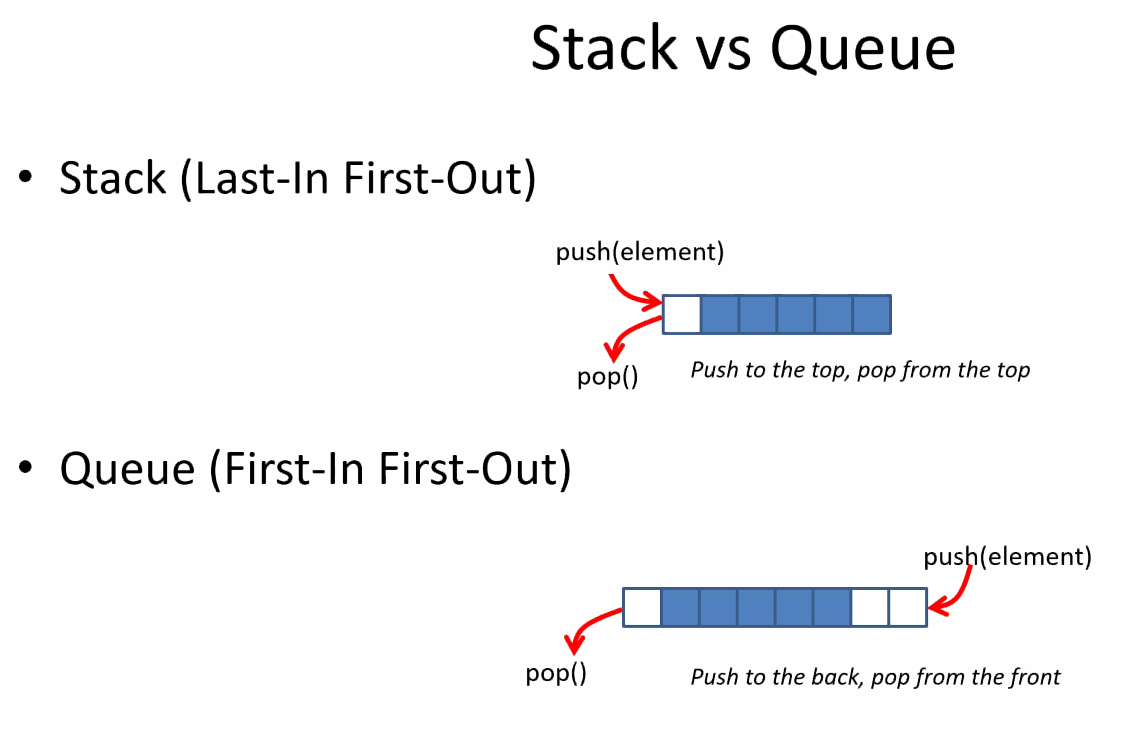
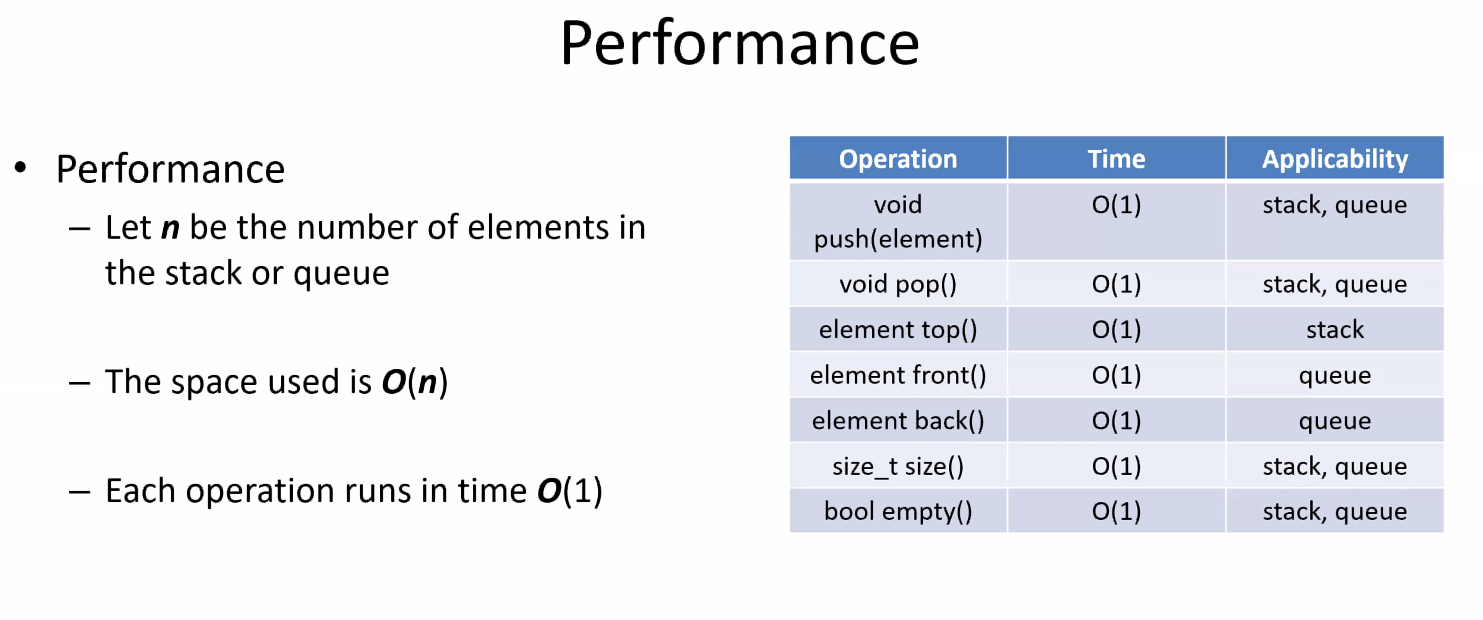
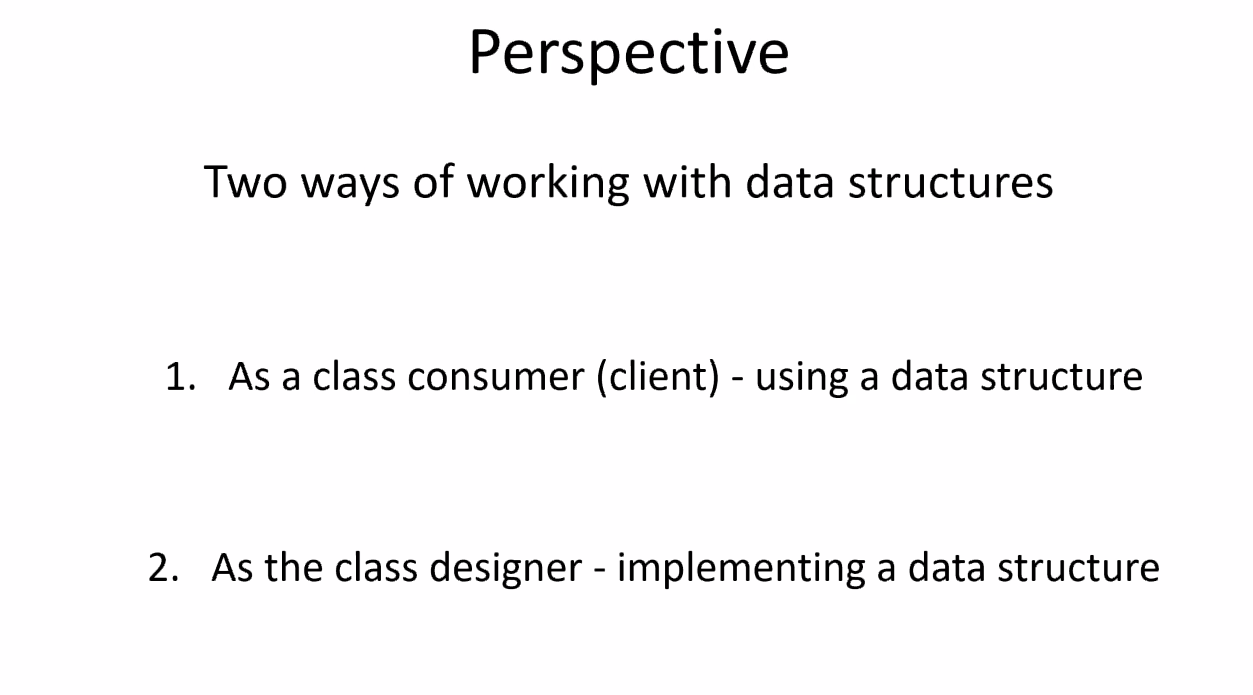
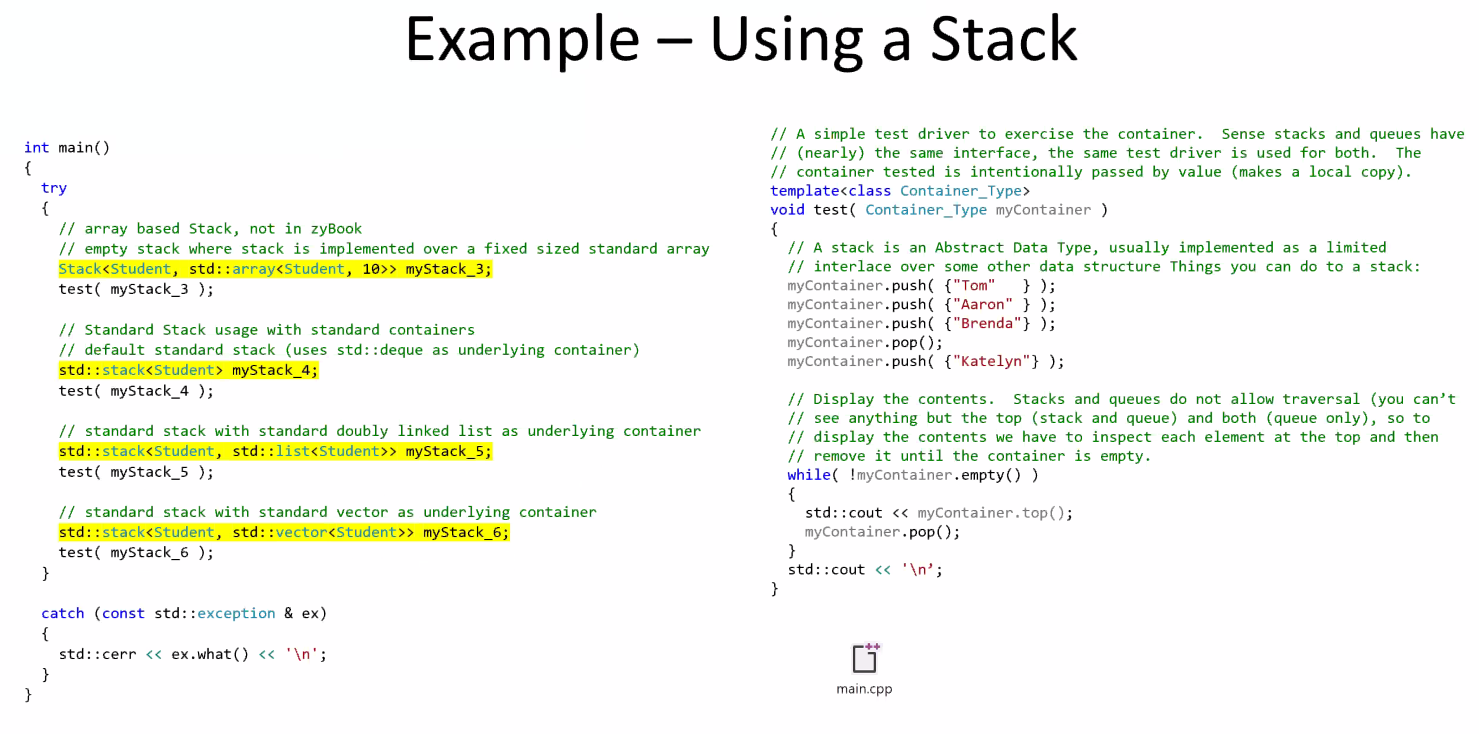
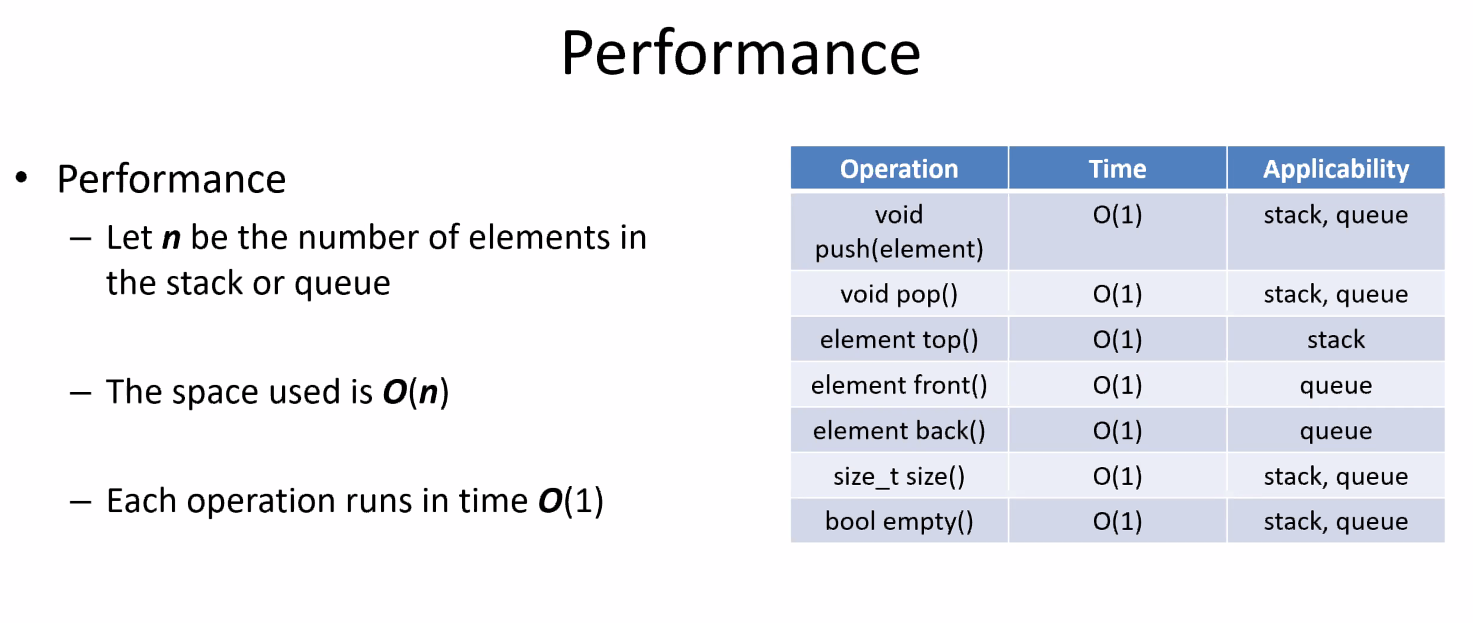
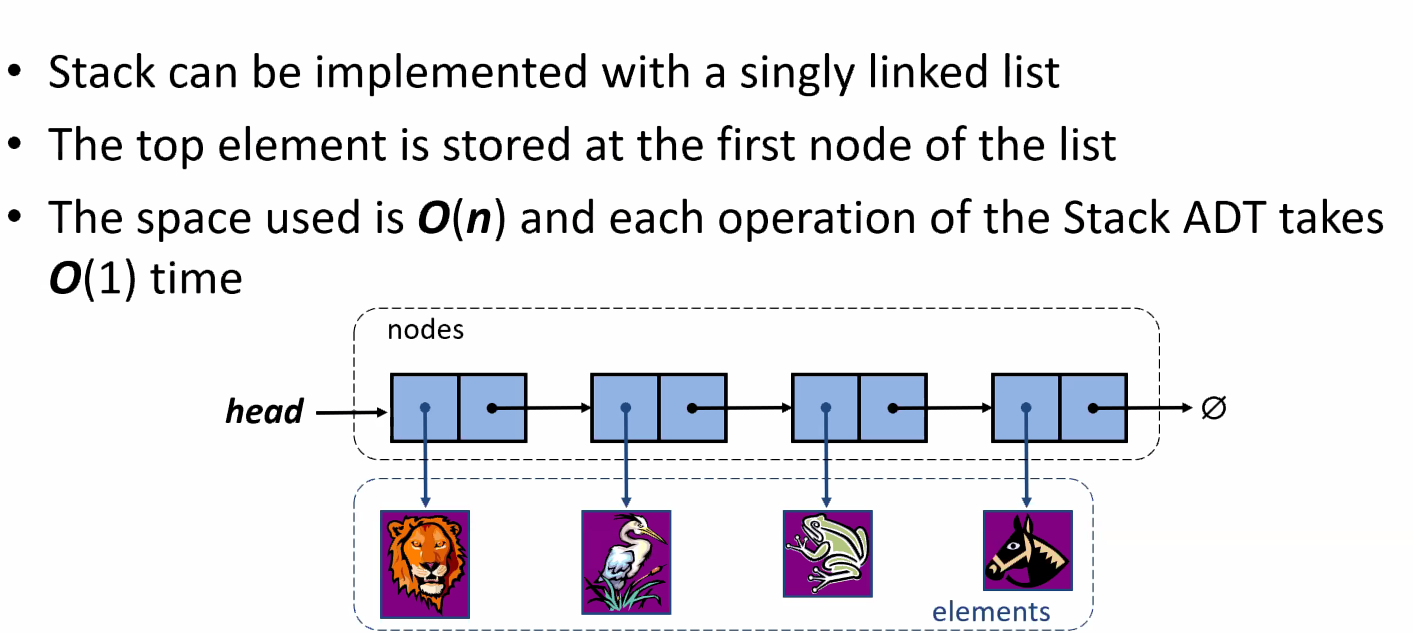
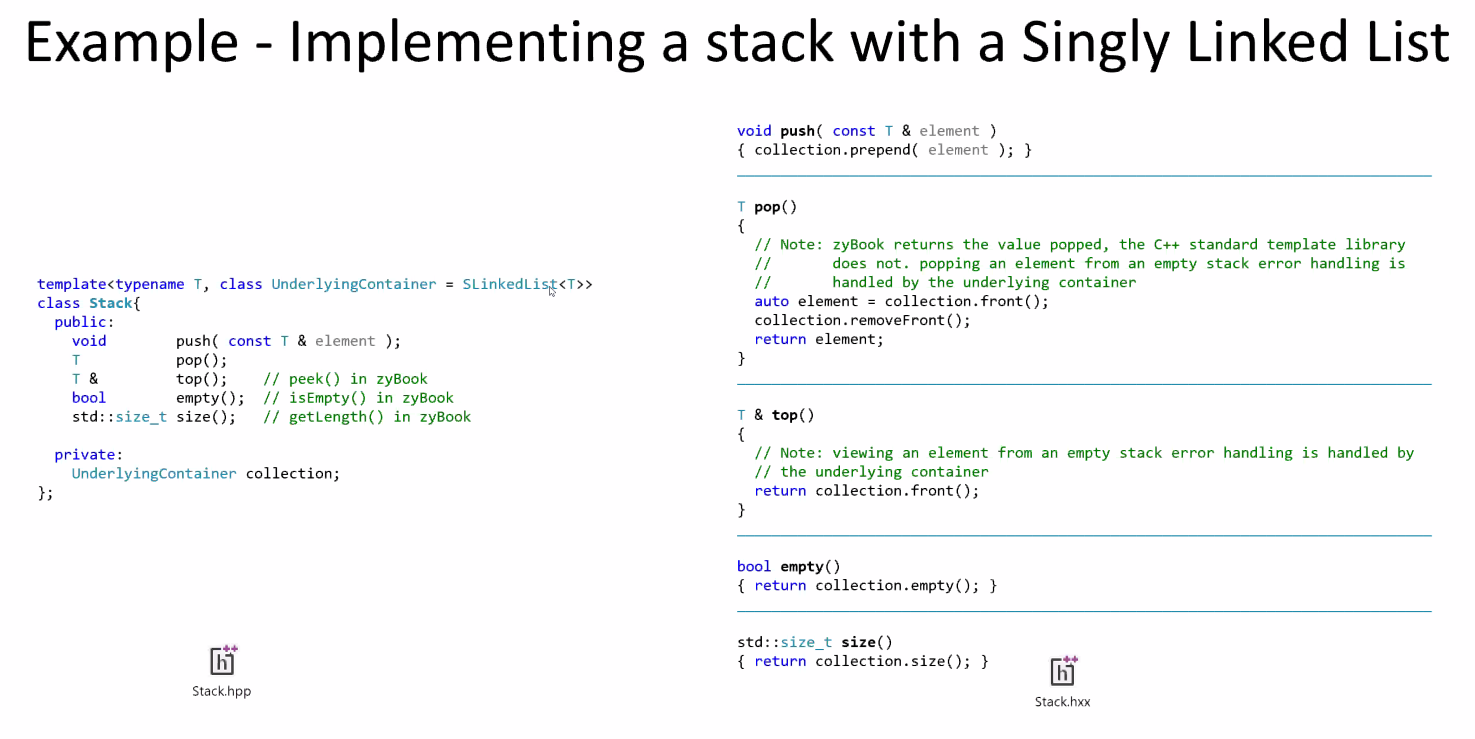
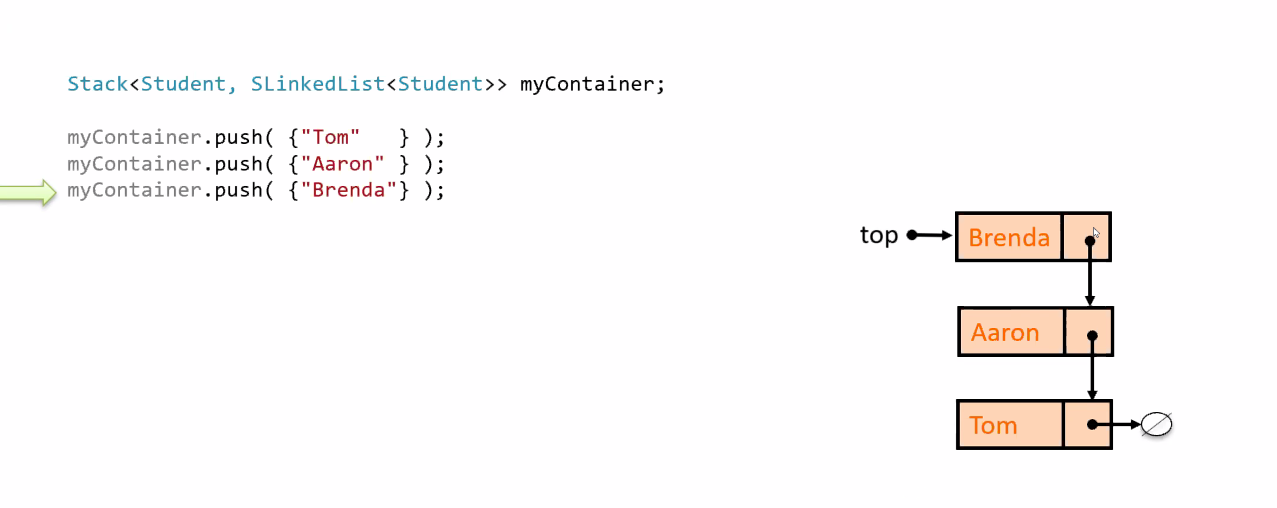
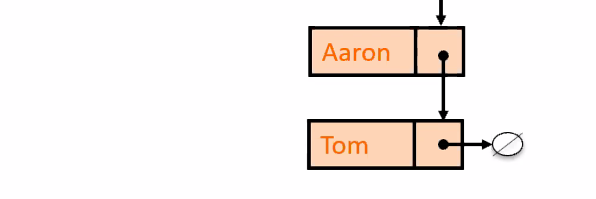
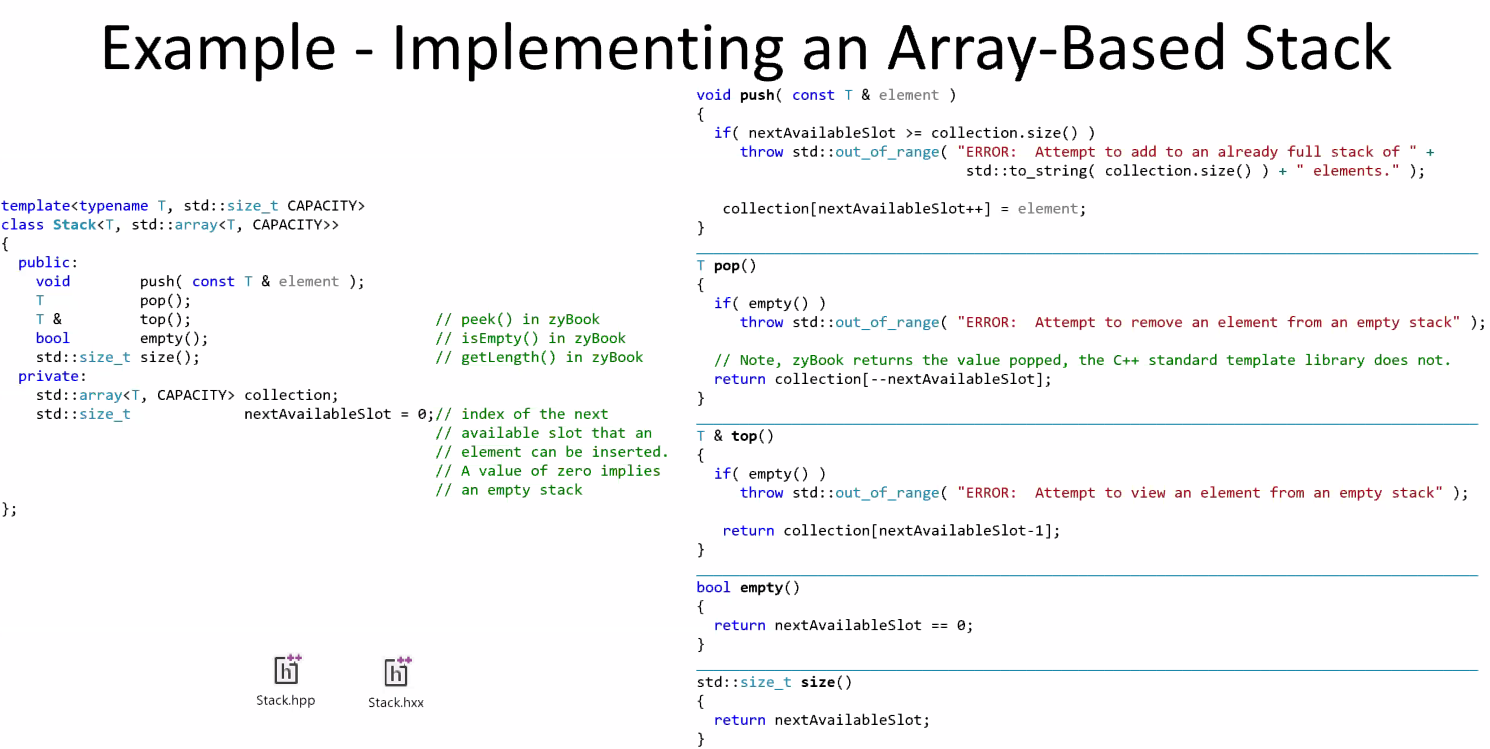
Lecture 12

CPSC 131  
10/5/2020

1. Stack and Queue
   1. Queue: FIFO
      1. Like a line at the Starbucks, first person in line as opposed to the last person at the end.  
         
   2. Stack: LIFO
      1. Like taking pile of books, you take the last book on top of the stack off rather than the first one at the bottom.
      2. Functions, parameters, addresses gets pushed to the stack
      3. You push and pop on opposite ends as opposed to the same ends like stack  
         
      4. No insert, no erase. Just push, pop, top
         1. Pops puts it on top of the stack
         2. Push places it further down towards the bottom of the stack
         3. Top enables you to look down and see what’s in the stack
      5. Dynamic memory comes from the heap
2. Key Terms
   1. Key Terms  
      
3. Stack Concepts
   1. 
   2. The interface
      1. Push,pop,top from the same end
4. Queue Concepts
   1. 
5. Stack vs queue
   1. 
   2. Vector: he mentioned that it could be so that, if we can push/pop from different ends, it could be queue
6. Performance
   1. Remember, we want constant time
   2. 
7. Perspective
   1. 
8. Using a Stack
   1. Example – Using a Stack  
      
      1. In this case we would use a doubly linked list or a vector
      2. These template structures take 2 structures
      3. There is no stack/queue containers; they are data types.
         1. Arrays, lists are structures; stacks simply tell how the structures are structured lol
   2. Performance  
      
9. Implementing the Stack with: Singly Linked List
   1. Stacks as a linked list  
      
   2. Example  
      
      1. This shows how we map push, pop, top
      2. We don’t make new data structures; we just adapt these functions to templated iterators
      3. That’s why they’re called *adaptors*
      4. We take the containers to the front of the list using front()
      5. All we’re doing id delegating to whatever the container calls the operation.
   3. Example – Sketching a stack with a Singly Linked List  
      
      1. Step 1) Push top: You create a new node, you push the node to the top, the size of your container becomes 1
      2. Step 2) Grab a new node, adjust top, increase size to 2. Make the new node point to top.
      3. Step 3) Grab a new node, adjust top, adjust size, increase size to 3
      4. Step 4) Pop, remove the new node, change the top, decrease the size  
           
         top -> (Aaron)
      5. Step 5) Grab a new node, change top, increase size  
         
      6.   
         pop the top  
         



1. Implementing a Stack with: Fixed Size Array
   1. Array-Base Stack – Size and pop  
      1. Take your stack and add elements from left to right(back of the queue)
   2. Example  
      
      1. Basically, the one that doesn’t change the array is top() because it is pointing to the position of the next available “slot”.
      2. Slot here is defined as element indices.
      3. Remember, arrays run on O(1) so accessing an element of an array/implementation of top is O(1) and size likewise