|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Implementation of StackADT | Array | | Singly Linked List | |
|  |  | |  | |
| Class and Data members | public class **ArrayStack**<T> implements StackADT<T> {  **T stack[];**  **int top; //next available index**  final static int DEFAULT\_CAPACITY = 10;  } | | public class LinkedStack<T> implements StackADT<T> {    **int count;**  **SingleLinkedNode<T> top;**    } | |
| Access    **T peek()** | O(1) | Retrieve element in the index of top-1 | O(n) | Use the top reference to get to the first node, get the element from it |
| public T peek(){  if (isEmpty()) {  throw new EmptyCollectionException();  }    return stack[top-1];  } | public T peek(){     |  |  | | --- | --- | | if (isEmpty()) {  //throw new EmptyCollectionException();  return null;  }  return top.getElement(); |  | |  |  |     } |
| Push    **void push(T element)** | O(1) | 1. if array (stack) is full, expand capacity.    2. Place the given element to the index top of the array (stack).    3. Increment top. | O(n) | 1. Create a new linear node holding the element reference  2. Assign the new node’s next to the same as top  3. Assign top to point to the new node  4. Increment count |
| public T push(T element){   |  |  | | --- | --- | |  |  | | if (top == this.stack.length)  expandCapacity();    this.stack[top] = element;  top++; |  |   } | public T push(T element){   |  |  | | --- | --- | | SinglyLinkedNode<T> node = top;  SinglyLinkedNode<T> current = new SinglyLinkedNode<T>(element);  current.setNext(top);  top = current;  count++; |  | |  |  |   } |
| pop    **T pop( )** | O(1) | 1. If stack is empty, throw an EmptyCollectionException exception.    2. Retrieve element temp in the index of top -1.  3. Set stack[top-1] to null.  4. Decrement top  Return temp. | O(n) | 1. If stack is empty, throw an EmptyCollectionException exception.  2. Declare a temporary node, temp and set it to be the same as top  3. Set the top to its next reference  4. Set the temporary node’s next reference to null  5. Decrement top  6. Return the data element of them temporary node |
| public T pop(T element){   |  |  | | --- | --- | |  |  | | if (isEmpty()) {  throw new EmptyCollectionException();  }    T element = stack[top-1];  stack[top - 1] = null;  top--;  return element; |  |   } | public T pop(T element){   |  |  | | --- | --- | | T result;  if (count != 0) {  SinglyLinkedNode<T> node = top;  result = top.getElement();  top = top.getNext();  count--;  node.setNext(null);  return result; //node.getElement();  } else {  throw new EmptyCollectionException("Linked List Stack");  } |  | |  |  |   } |
| **size()** | O(1) | Return top | O(n) | Return count |
| public int size(){   |  |  | | --- | --- | |  |  | | Return size; |  |   } | public int size(){   |  |  | | --- | --- | |  |  | | return count; |  |   } |