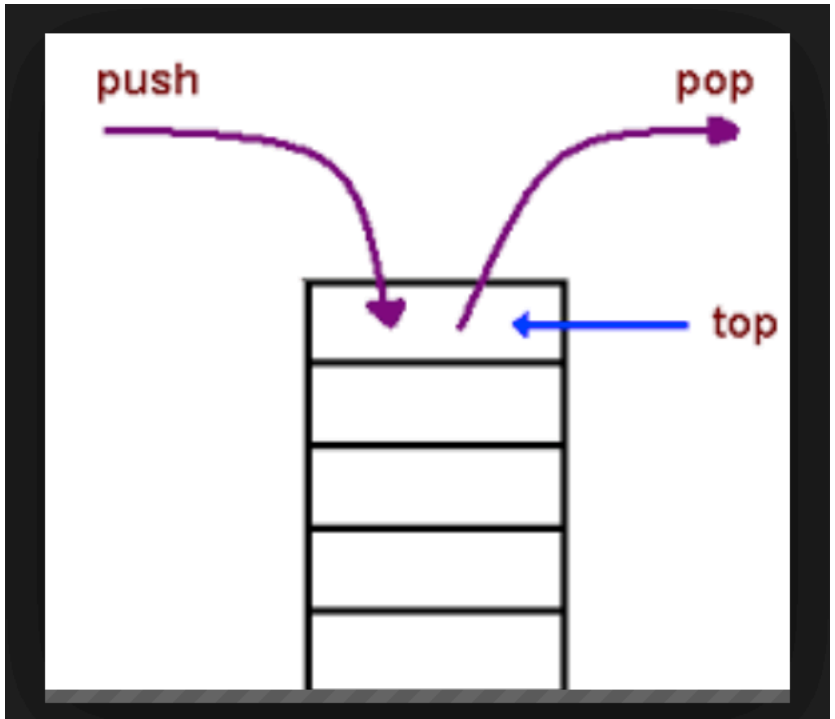
Elementary Data Structures
Stacks

T#9

Stack

- Dynamic Sets in which the element deleted from the set is the one most recently inserted.
- Stacks implements a **LAST IN FIRST OUT** policy



Stack in Java

- Java includes a Stack class as **an extension of the Vector class** (a dynamic array).
- Unfortunately, this implementation is somewhat problematic (Stack inheirit's Vector's methods, too!)

Online stack documentation:

https://docs.oracle.com/javase/8/docs/api/java/util/Stack.html

Write a Stack Class

- write a class MyStack that implements the cpsc331 Stack interface.
  - **Array Implementation**
    - A variable **top** that keeps track of top element in array
    - How to add an element?
    - How to check if the stack is empty?
    - How to check if the stack is full?
  - **Linked List Implementation**
    - Keep track of the size
    - How to add an element?
    - How to check if the stack is empty?
    - How to check if the stack is full?
- The stack operations can be implemented with a few lines of code

```
/**
 * The cpsc331Stack interface represents the Stack ADT as described
 * in CPSC 331.
 *
 * @author  Mike Jacobson
 * @version 1.0
 */
public interface cpsc331Stack<T> {

    /**
     * Tests whether the stack is empty.
     *
     * @return true if the stack is empty, false otherwise
     */
    public boolean isEmpty();

    /**
     * Pushes the object x onto the top of the stack.
     *
     * @param x object to be pushed onto the stack.
     */
    public void push(T x);

    /**
     * Returns the object at the top of the stack.
     *
     * @return reference to the item at the top of the stack
     * @throws EmptyStackException if the stack is empty
     */
    public T top();

    /**
     * Removes and returns the object at the top of the stack.
     *
     * @return reference to the item at the top of the stack
     * @throws EmptyStackException if the stack is empty
     */
    public T pop();
}
```

You can add to this interface:

- Size
- Capacity
- IsFull

Pseudocodes:

Stack-Empty(S)

```
If top[S} = 0
    Then return True
    Else return False
```

Push (S, x)

```
Top[S] <- top [s] +1
S[top[S]] <- x
```

Pop(S)

```
If Stack-Empty(S)
    then error “underflow”
Else top[S]<- top[S]-1
    return S[top[S]+1]
```

- Finally, think about (and, as time permit, design) tests that you would write to test this class.