Data Structures and Analysis of Algorithms CST 225-3

Stack

Uva Wellassa University

Have you seen ?

- Cafeteria Plate Dispenser

Stacks

- A stack is a linear data structure which can be accessed only at one of its ends (called top of the stack) for storing and retrieving data.
- It behaves very much like the common stack of plates or stack of newspapers.
- A stack is a dynamic constantly changing object.

Stacks

- Stores a set of elements in a particular order
- Stack principle: **LAST IN FIRST OUT (LIFO)**
- It means: the last element inserted is the first one to be removed
- What is the first element to pick up?
- Restrict the access to most recently inserted item. (top item)

Primitive Operations

- The two changes which can be made to a stack are given special names.
- When an item is added to a stack, it is **pushed** onto the stack and when an item is removed, it is **popped** from the stack.
- Because of the push operation which adds elements to a stack, a stack is sometimes called a **pushdown list**.

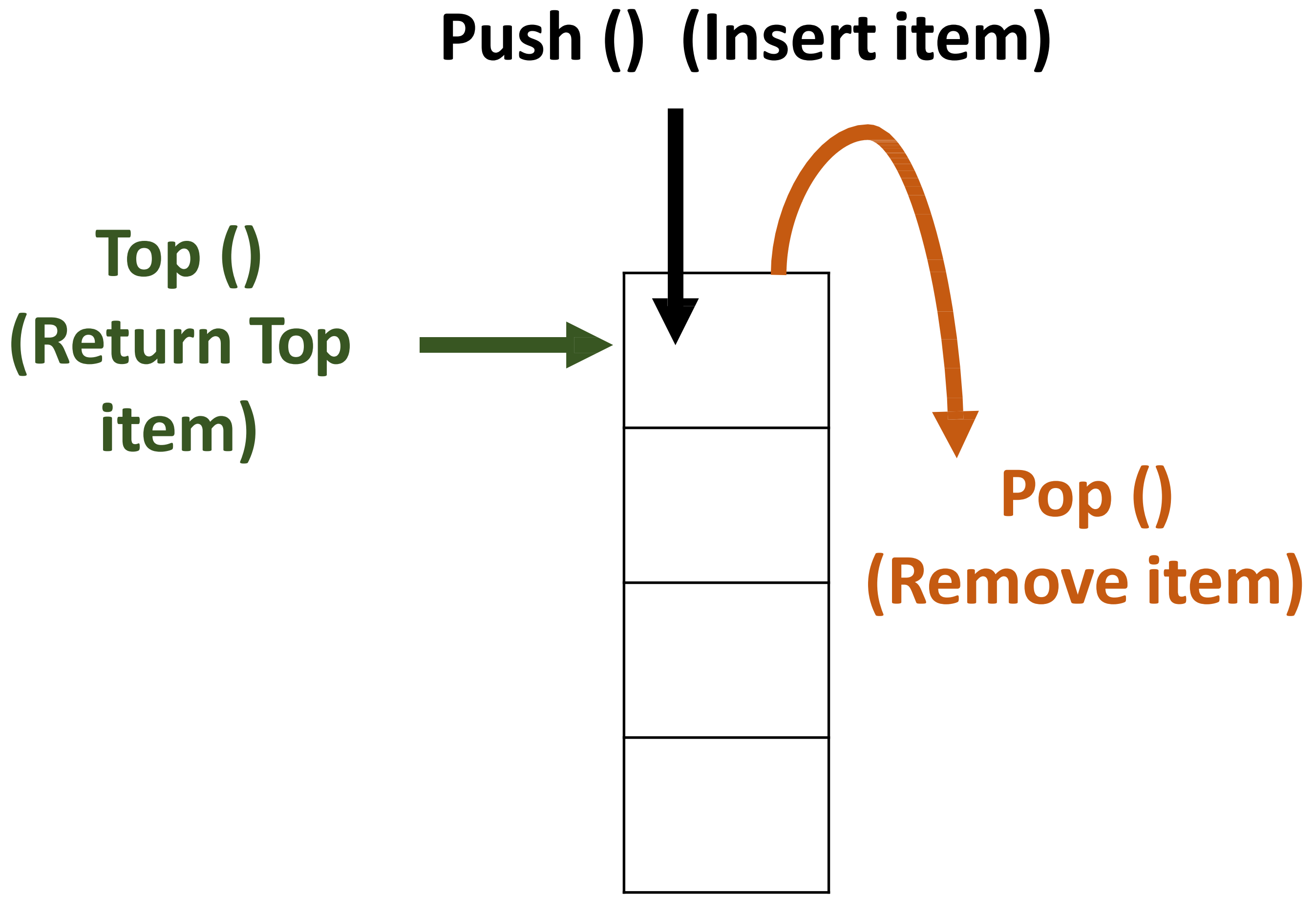
Primitive Operations cont'd…

- If a stack contains a single item and the stack is popped, the resulting stack contains no items and is called the **empty stack**.
- Although the *push* operation is applicable to any stack, *pop* operation cannot be applied to the empty stack because such a stack has no elements to pop.
- The operation *empty(s)* determines whether a stack is empty or not.

Primitive Operations cont'd …

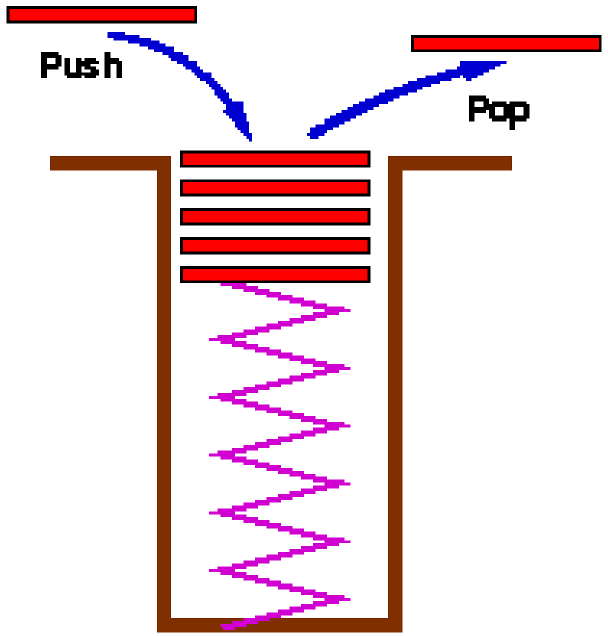
- Another operation that can be performed on a stack is to determine what the top item on a stack without removing it.
- This operation is written *top(s)* and returns the top item of the stack S.

Basic Operations of Stack



Stack Model

- A common model of a stack is a plate or coin stacker.
- Plates are "pushed “ onto to the top and "popped" off from the top.



Basic Operations of Stack cont’d..

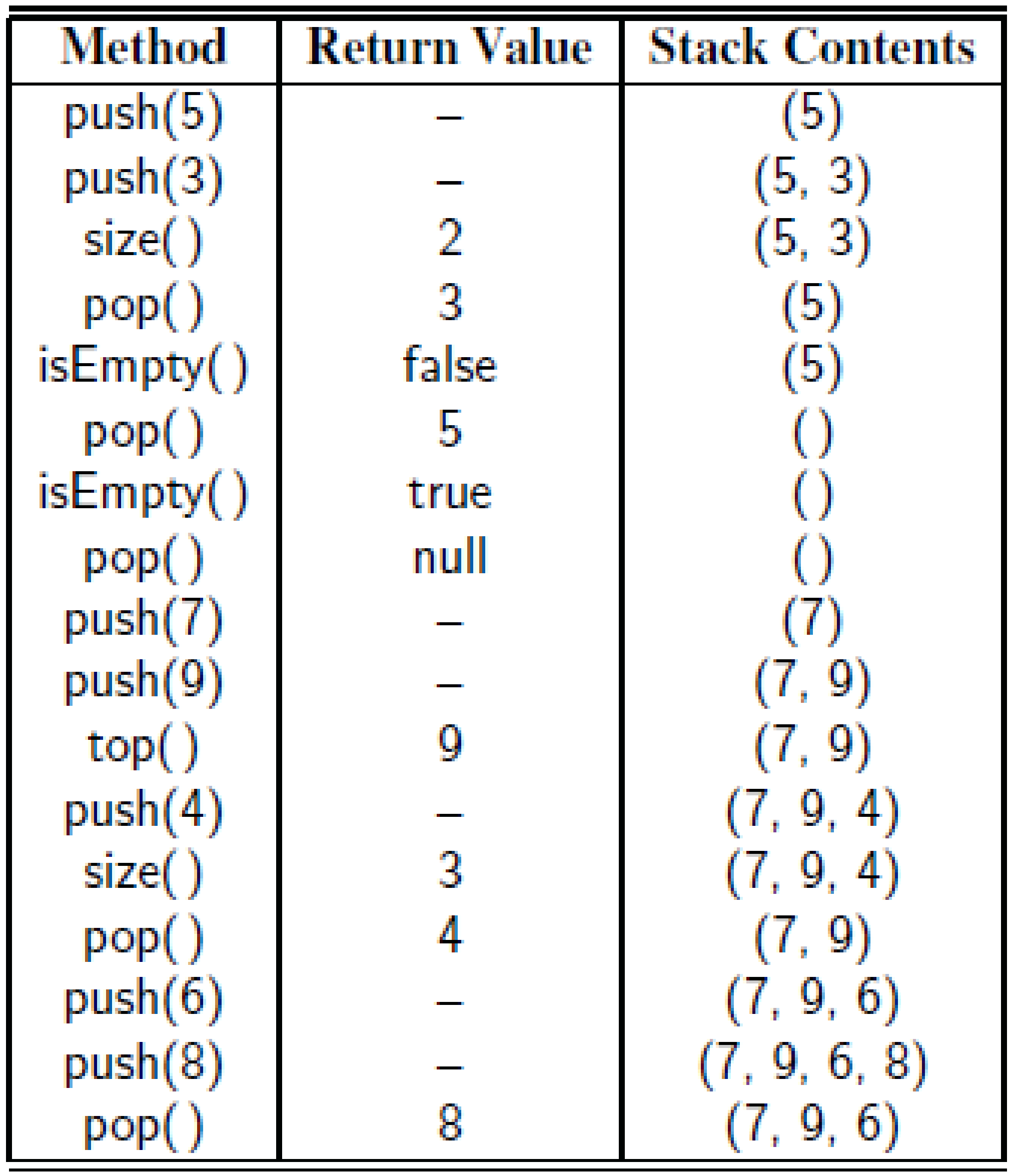
- **push(e):** Adds element e to the top of the stack.
- **pop(**): Removes and returns the top element from the stack (or null if the stack is empty).
- **top( ):** Returns the top element of the stack, without removing it (or null if the stack is empty).
- **isEmpty():** Checks whether stack is empty or no items
- **isFull():** Consider full if no other element can be inserted on top of the stack
- **size( ):** Returns the number of elements in the stack.

States of Stack

- **Underflow state :**
  - Stack is empty
  - Items can be inserted
  - IsEmpty()= TRUE

- **Overflow state :**
  - Stack is full
  - Items cannot be inserted
  - IsEmpty()=False

Series of Stack Operations-Example



Method	Return Value	Stack Contents
push(5)	–	(5)
push(3)	–	(5, 3)
size( )	2	(5, 3)
pop( )	3	(5)
isEmpty( )	false	(5)
pop( )	5	( )
isEmpty( )	true	( )
pop( )	null	( )
push(7)	–	(7)
push(9)	–	(7, 9)
top( )	9	(7, 9)
push(4)	–	(7, 9, 4)
size( )	3	(7, 9, 4)
pop( )	4	(7, 9)
push(6)	–	(7, 9, 6)
push(8)	–	(7, 9, 6, 8)
pop( )	8	(7, 9, 6)

Implementation of Stack

- Can be done in two ways:
- **Array-** Today discussion
  - Limited in size
  - Fast
- **Linked List-** Later discussion
  - Not limited in size
  - Overhead to allocate link, unlink and deallocate

Position Variable

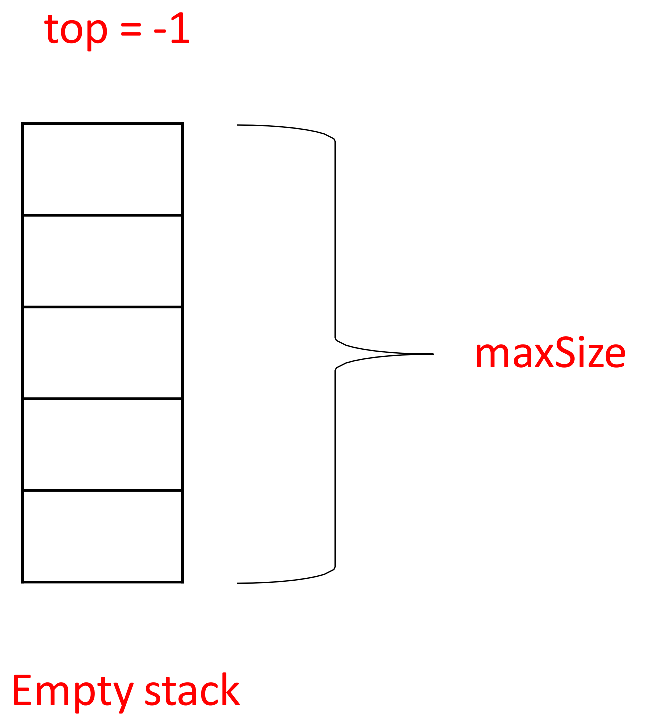
- In stack, top is considered as the position variable.
- Its values change in the following situations of the stack.

Stack Status	**Value of “top”**
Stack is empty/underflowed	top = -1
First element is added	top = 0
Stack is full	top = stack_size - 1
Stack overflowed	top = stack_size

Stack Creation using Array

- *Create stack structure object using the newStack class constructor*

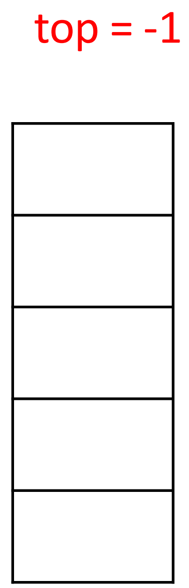
```
Class newStack
{           private int myStack[];
            private int top;
            private maxSize;
            public newStack(int size)
            {
                        maxSize = size;
                        myStack = new int[maxSize];
                         top = -1;
            }
}
```



Check stack is empty : isEmpty()

- Create a function called isEmpty() which return true if stack is empty.

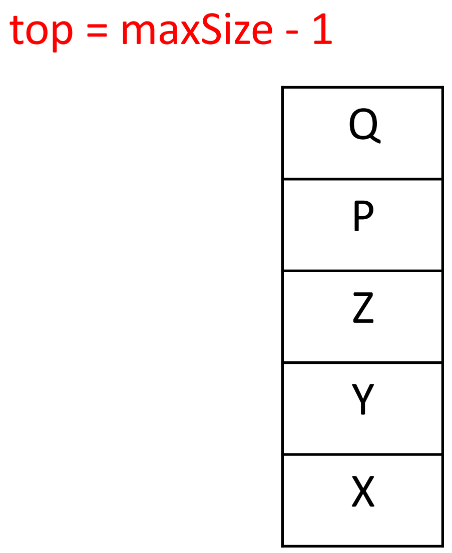
```
public boolean  isEmpty()
        {
                return [top == -1] ;
        }
```



Check stack is Full : isFull()

- Create a method called isFull() which returns true if the stack is full.

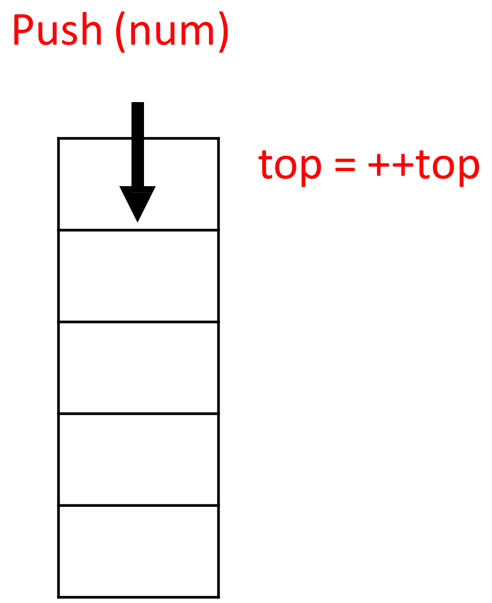
```
public boolean  isFull()
        {       return [top ==  maxSize -1 ] ;
        }
```



Insert an Element : push()

- Create new method called push()

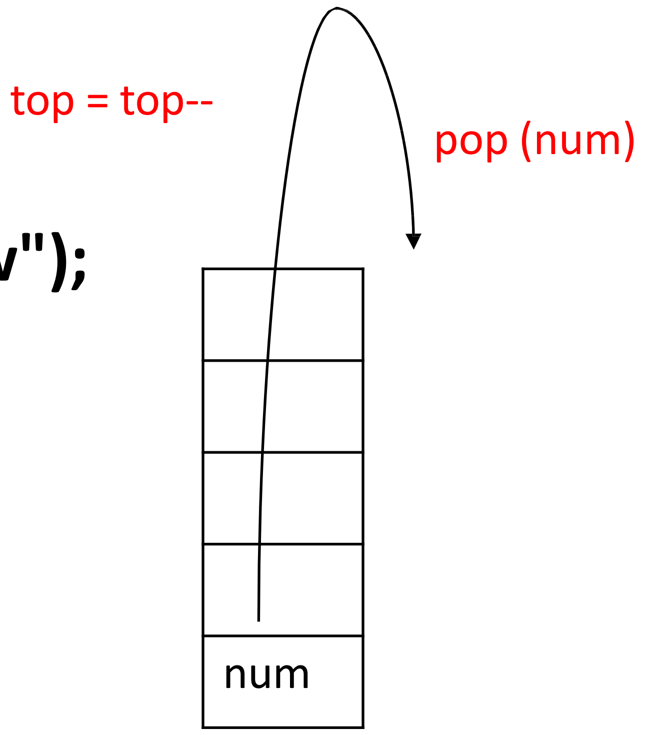
```
public  void push(int num)
        {
        if (isFull()) {
                System.out.println("stack overflow");
                } else {
                        myStack[++top] = num;

                }
        }
```



Remove an Element : pop()

- Create new method called pop()

```
public  int pop()
        {
        if (isEmpty()) {
                System.out.println("stack underflow");
                return 0;
                } else {
                        return myStack[top--] ;
                }
        }
```



Stack size: size()

- Create a function called size to output number of all elements of the stack.

```
public  int size()
        {
                return (top +1);


        }
```

Drawbacks of Array-Based Stack Implementation

- Stack relies on a fixed-capacity array, which limits the ultimate size of the stack.
- If an attempt is made to push an item onto a stack that has already reached its maximum capacity, it cannot be done.

Performance of Stack

- Items can both pushed and popped from the stack in constant time.
- That is, time is not dependent on how many items are in the stack, and is therefore very quick.

How the stack routines work: empty stack; push(a), push(b); pop?

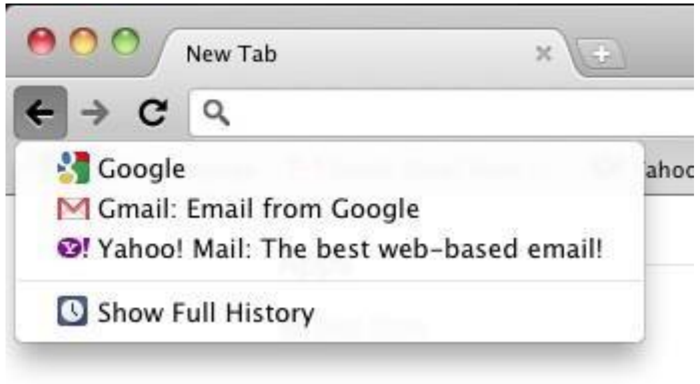
To be discussed in the class

Applications of Stacks

- Stacks are used extensively in compilers
- Internet web browsers – Forward and backward features
- Text editors – “Undo” and “Redo” features
- Simple calculators – save previous results
- Mathematical expression evaluation
- Balanced spell checker

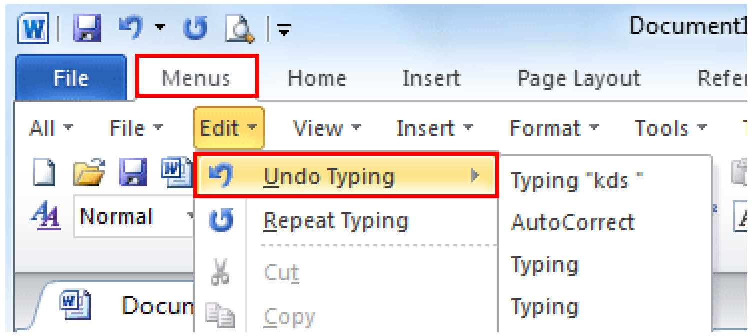
Applications of Stacks cont'd...

- Browsers when loading new web pages, push the address of the current page into a stack.
- Then address of the previous page can be popped out of the stack later when pressing the go back button.



Applications of Stacks cont'd…

- Undo operation of an text editor.
- The recent changes are pushed into a stack, and the undo operation pops it from the stack.



Exercise:
Can we use stack to reverse a word?

- Can we use a stack to reverse the letters in a word?

Applications of Stacks
Ex: Reversing a Word

- We can use a stack to reverse the letters in a word.
- How?

  - Read each letter in the word and push it onto the stack

  - When you reach the end of the word, pop the letters off the stack and print them out.

Array vs Stack

- To be discussed in the class

Questions?