Data Structures and Algorithms

Lecture 3: Stacks

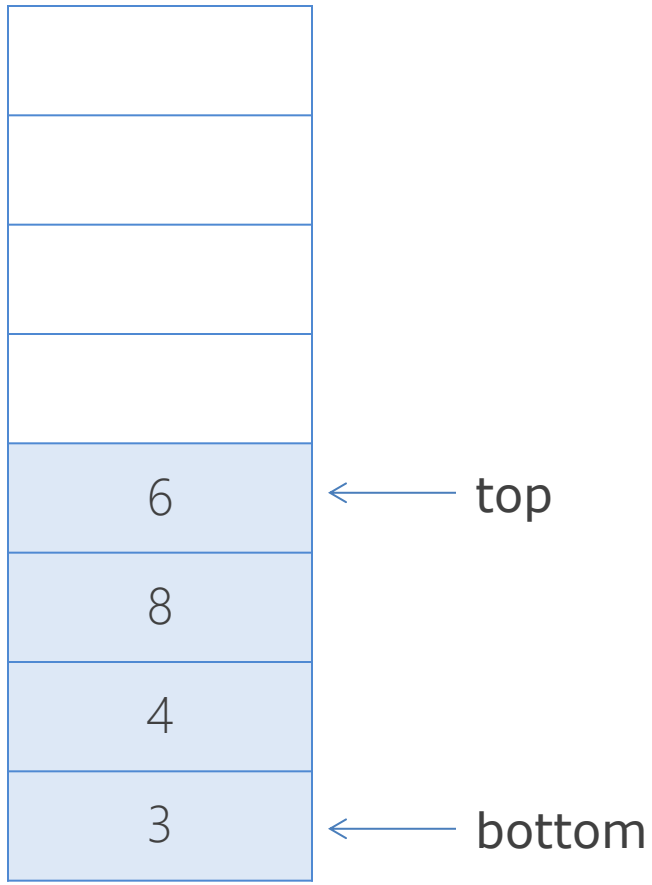
Department of Computer Science & Technology
United International College

Outline

- Stack ADT
- Basic operations of stack
  - Pushing, popping etc.
- Applications of stacks
- Implementations of stacks using
  - array
  - linked list

Stack ADT

- Stack is a special list where insertion and deletion take place at the same end
  - This end is called *top*
  - The other end is called *bottom*
- Everything happens at the top
- Nothing happens at the bottom

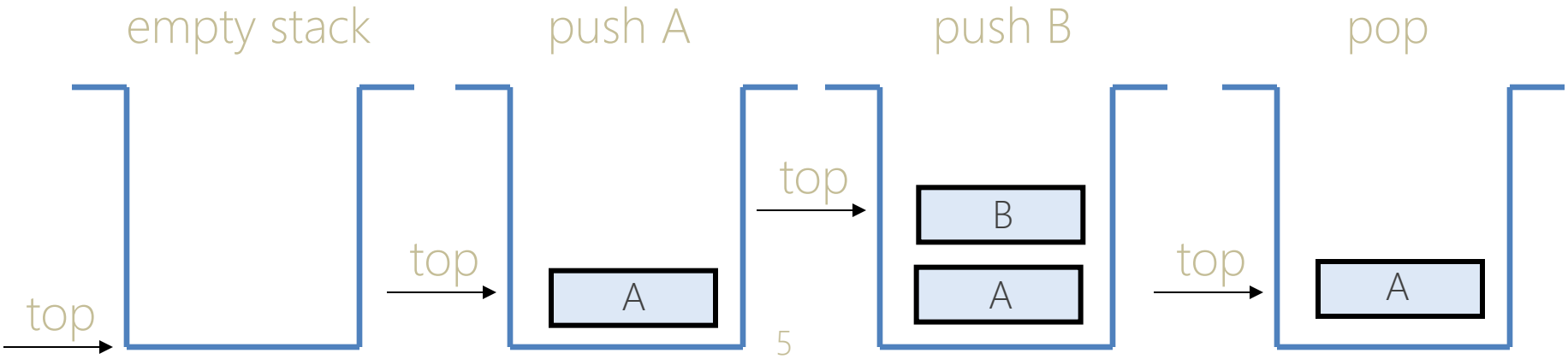


Stack Animation

- http://liveexample.pearsoncmg.com/liang/animation/animation.html
- Stacks are known as LIFO (Last In, First Out) lists.

Push and Pop

- Primary operations: Push and Pop
- Push
  - Add an element to the top of the stack
- Pop
  - Remove the element at the top of the stack
- Top
  - Return, without removing, the element at the top



Stack Applications

- Expression evaluation
- Backtracking
- Memory Management

Implementation of Stacks

- Recall the reason why we usually don't implement the list using array?

Topic		Array	Linked List
Efficiency	push		
	pop		
	Top		
space			

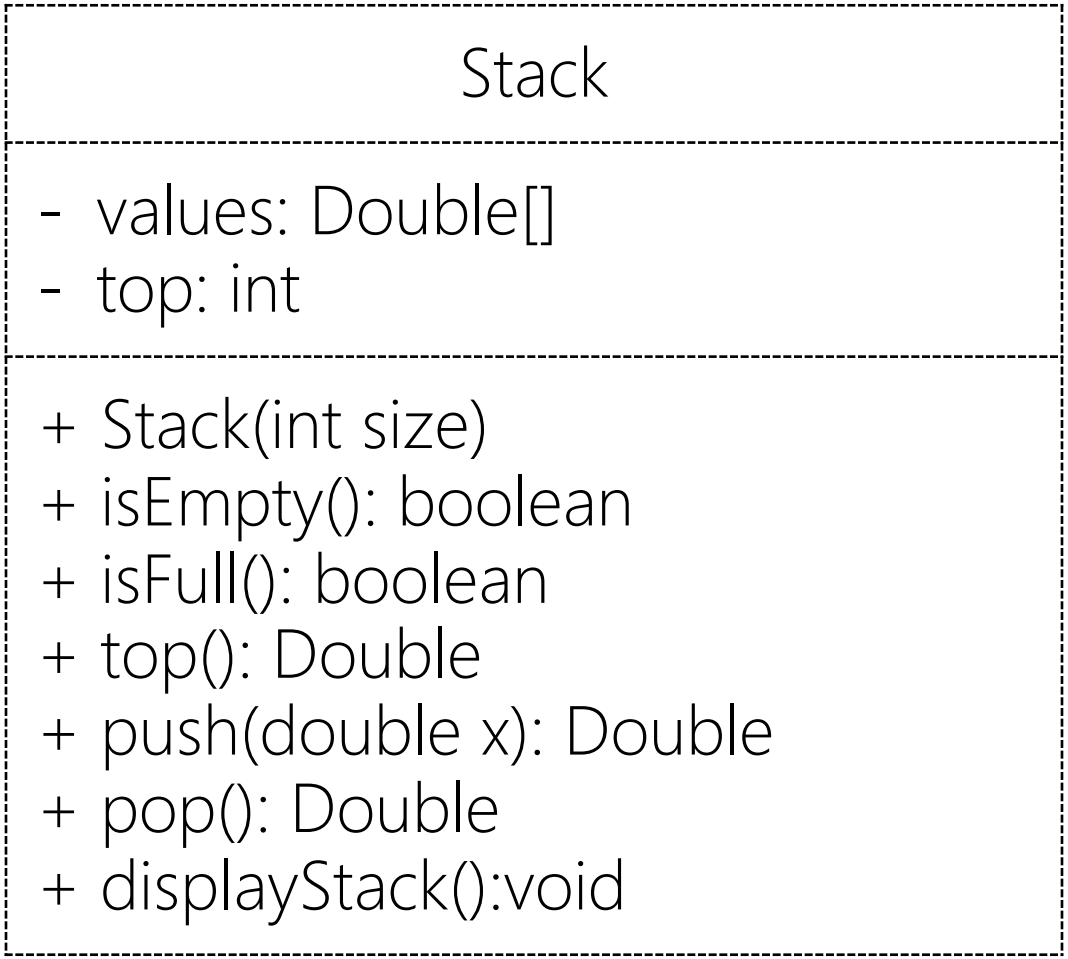
Stack Implementation

- Data are stored in an array
  - values: an array which stores elements of stack
    - values can be of any data type but we use Double for demonstration
  - top: the index of the top element of stack

Stack Implementation

Class: Stack

Setters and getters are not listed.



Methods

- public Stack(int size)
  - Creates an empty stack whose capacity is *size*
- public boolean isEmpty()
  - Returns true if the stack is empty and false otherwise
- public boolean isFull()
  - Returns true if the stack is full and false otherwise
- public Double top()
  - Returns the top element
  - Returns null if the stack is empty

Methods

- public Double push(double x)
  - Adds a new element with value *x* to the top of the stack
  - Returns the new element if the operation is successful and null otherwise
- public Double pop()
  - Removes and returns the top element of the stack
  - Returns null if the operation fails
- public void displayStack()

```
top --> |        -1.0000        |
        |         0.0000        |
        |         1.0000        |
        |         2.0000        |
        |        -3.0000        |
        +-----------------------+
```

The Constructor

```
public Stack(int size) {
    values = new Double[size];
    top = -1;
}
```

Why?

Push

```
public Double push(double x) {
    if(isFull())
        return null;
    values[++top] = Double.valueOf(x);
    return top();
}
```

This explains.

displayStack

```
public void displayStack() {
    System.out.print("top -->");
    for(int i =top; i >= 0; i --)
      System.out.println("\t|\t " +
        String.format("%, .4f",values[i].doubleValue()) +
        "\t|");
    System.out.println("\t+----------------------+");
}
```

Check it online!

Using Stack

```
public static void main(String[] args) {
    Stack myStack = new Stack(4);
    System.out.println(myStack.isEmpty());
    myStack.push(-3);
    myStack.push(5);
    System.out.println("The stack has 2 items:");
    myStack.displayStack();
    myStack.push(1);
    myStack.push(2);
    myStack.push(-1);
    System.out.println("The stack has 4 items:");
    myStack.displayStack();
    System.out.println("The top is: " + myStack.top());
    System.out.println(myStack.isFull());
    myStack.pop();
    myStack.pop();
    myStack.pop();
    myStack.pop();
    System.out.println("The stack is empty:");
    myStack.displayStack();
}
```

Using Stack

```
public static void main(String[] args) {
    Stack myStack = new Stack(4);
    System.out.println(myStack.isEmpt
    myStack.push(-3);
    myStack.push(5);
    System.out.println("The stack has
    myStack.displayStack();
    myStack.push(1);
    myStack.push(2);
    myStack.push(-1);
    System.out.println("The stack has
    myStack.displayStack();
    System.out.println("The top is: "
    System.out.println(myStack.isFull
    myStack.pop();
    myStack.pop();
    myStack.pop();
    myStack.pop();
    System.out.println("The stack is empty:");
    myStack.displayStack();
}
```

```
true
The stack has 2 items:
top --> |          5.0000          |
        |         -3.0000          |
        +--------------------------+
The stack has 4 items:
top --> |          2.0000          |
        |          1.0000          |
        |          5.0000          |
        |         -3.0000          |
        +--------------------------+
The top is: 2.0
true
The stack is empty:
top --> +--------------------------+
```

Task

- Complete *Stack.java* which implements the stack class
  - The class is defined on slide 9.
  - A *main* function has been given for the class which tests 6 functions: push, pop, top, isEmpty, isFull, displayStack
- Submit *stack.java* to iSpace.