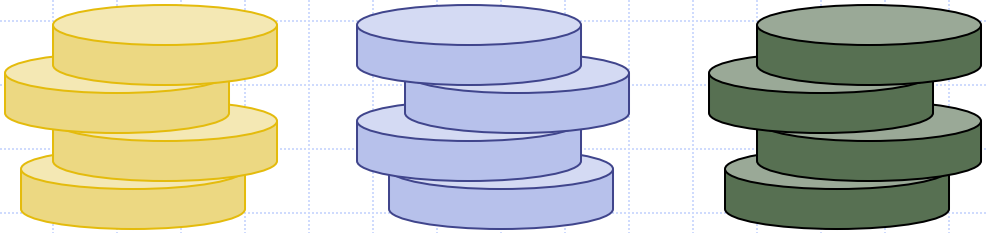
Presentation for use with the textbook Data Structures and Algorithms in Java, 6th edition, by M. T. Goodrich, R. Tamassia, and M. H. Goldwasser, Wiley, 2014

Stacks



Reading

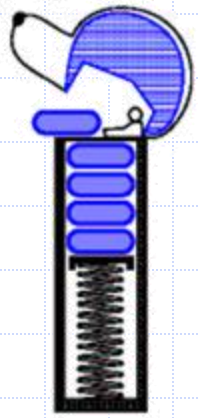
M. T. Goodrich, R. Tamassia and M. H. Goldwasser, *Data Structures and Algorithms in Java*, 6th Edition, 2014.

- **Chapter 6. Stacks and Queues**

Abstract Data Types (ADTs)

- An abstract data type (ADT) is an abstraction of a data structure
- An ADT specifies:
  - Data stored
  - Operations on the data
  - Error conditions associated with operations
- Example: ADT modeling a simple stock trading system
  - The data stored are buy/sell orders
  - The operations supported are
    - order buy(stock, shares, price)
    - order sell(stock, shares, price)
    - void cancel(order)
  - Error conditions:
    - Buy/sell a nonexistent stock
    - Cancel a nonexistent order

The Stack ADT



- The Stack ADT stores arbitrary objects
- Insertions and deletions follow the *last-in first-out* scheme
- Think of a spring-loaded plate dispenser
- Main stack operations:
  - push(object): inserts an element
  - object pop(): removes and returns the last inserted element
- Auxiliary stack operations:
  - object top(): returns the last inserted element without removing it
  - integer size(): returns the number of elements stored
  - boolean isEmpty(): indicates whether no elements are stored

Stack Interface in Java

- Java interface corresponding to our Stack ADT
- Assumes null is returned from top() and pop() when stack is empty
- Different from the built-in Java class java.util.Stack

```
public interface Stack<E> {
   int size();                  返回栈中元素个数
   boolean isEmpty();           判断栈是否为空
   E top();                     返回栈顶元素（不移除）
   void push(E element);        压入（插入）一个元素
   E pop();                     弹出（移除并返回）栈顶元素
}
```

Example

1. **栈的操作遵循“后进先出 (LIFO, Last-In First-Out)”原则：**
   - `push(x)` **插入元素**到栈顶。
   - `pop()` **移除并返回**栈顶元素。
   - `top()` **获取栈顶元素**但不移除。
   - `isEmpty()` **检查栈是否为空。**
   - `size()` **返回栈中元素的数量。**
2. **空栈的行为**
   - 当 `pop()` 操作在空栈上调用时，返回 `null`。

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)

Exceptions vs. Returning Null

- Attempting the execution of an operation of an ADT may sometimes cause an error condition
- Java supports a general abstraction for errors, called exception
- An exception is said to be “thrown” by an operation that cannot be properly executed
- In our Stack ADT, we do not use exceptions
- Instead, we allow operations pop and top to be performed even if the stack is empty
- For an empty stack, pop and top simply return null

Applications of Stacks

- Direct applications
  - Page-visited history in a Web browser
  - Undo sequence in a text editor
  - Chain of method calls in the Java Virtual Machine
- Indirect applications
  - Auxiliary data structure for algorithms
  - Component of other data structures

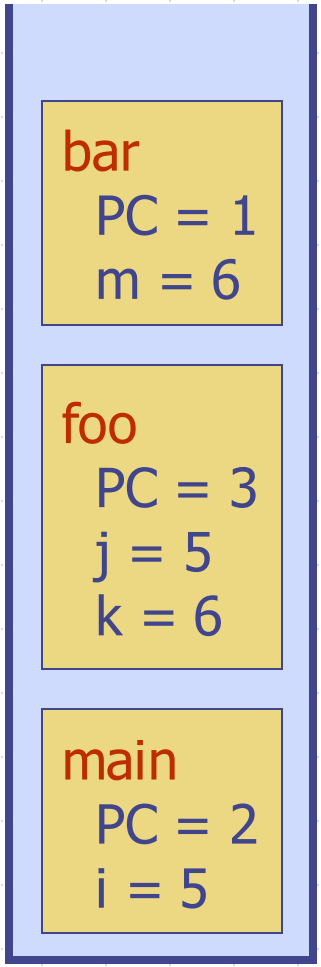
Method Stack in the JVM

- The Java Virtual Machine (JVM) keeps track of the chain of active methods with a stack
- When a method is called, the JVM pushes on the stack a frame containing
  - Local variables and return value
  - Program counter, keeping track of the statement being executed
- When a method ends, its frame is popped from the stack and control is passed to the method on top of the stack
- Allows for recursion

```
main() {
  int i = 5;
  foo(i);
  }

foo(int j) {
  int k;
  k = j+1;
  bar(k);
  }

bar(int m) {
  …
  }
```



Array-based Stack

- A simple way of implementing the Stack ADT uses an array
- We add elements from left to right
- A variable keeps track of the index of the top element

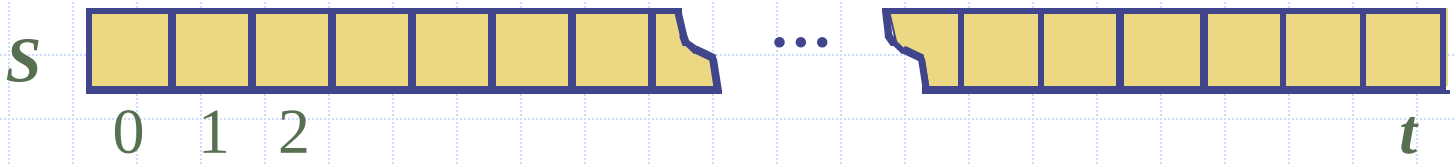
```
Algorithm size()
  return t + 1

Algorithm pop()
  if isEmpty() then
    return null
  else
    t ← t – 1
    return S[t + 1]
```

Array-based Stack (cont.)

- The array storing the stack elements may become full
- A push operation will then throw a FullStackException
  - Limitation of the array-based implementation
  - Not intrinsic to the Stack ADT

Algorithm *push*(o)
  if $t = S.length - 1$ **then**
    throw *IllegalStateException*
  else
    $t \leftarrow t + 1$
    $S[t] \leftarrow o$



Performance and Limitations

- Performance
  - Let n be the number of elements in the stack
  - The space used is $O(n)$
  - Each operation runs in time $O(1)$
- Limitations
  - The maximum size of the stack must be defined a priori and cannot be changed
  - Trying to push a new element into a full stack causes an implementation-specific exception