CSC 1052 – Algorithms & Data Structures II: Stacks

Professor Henry Carter
Spring 2017

Recap

- Abstraction allows for information to be compartmentalized and simplifies modular use
- Interfaces are the Java construction for formal abstraction
- Generic collections allow for simplified implementation of data structures that may hold a variety of classes
- The stack is a LIFO data structure that allows efficient access to the top element

Stacks

- LIFO structure
- Collection of elements
- Good for:
  - Save state
  - Nested data
  - Backtracking

The Development Process

- The development process
  - Write code (lather)
  - Test code (rinse)
  - Repeat (repeat)

Yes, Elizabeth Taylor uses Lustre-Creme Shampoo. It's the favorite of 4 out of 5 top Hollywood movie stars!

It never dries your hair! Lustre-Creme Shampoo is blessed with lanolin . . . foams into rich lather, even in hardest water . . . leaves hair so easy to manage.

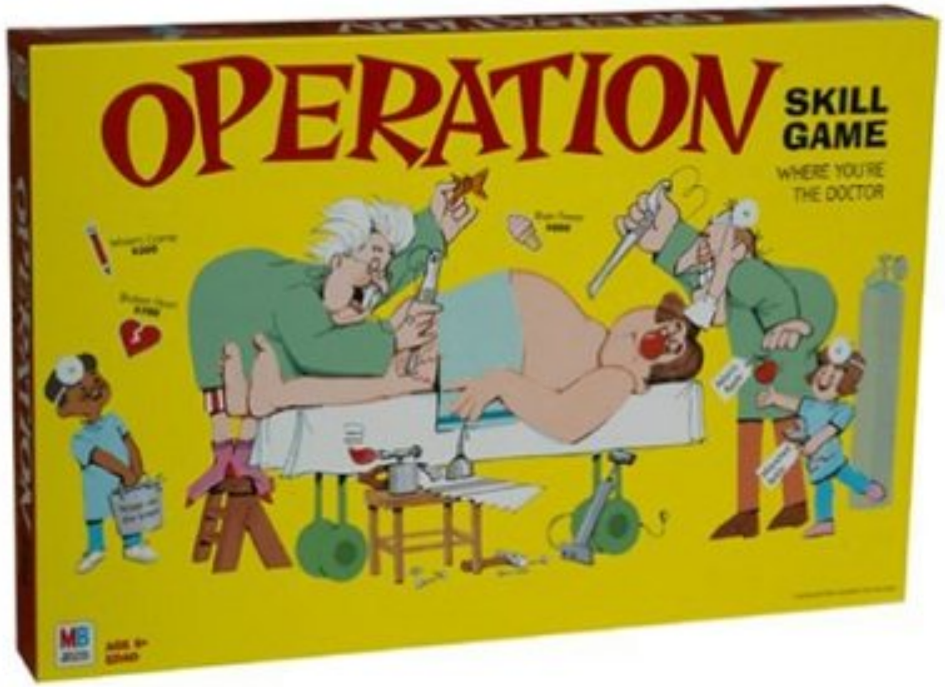
It beautifies! For soft, bright, fragrantly clean hair—without special after-rinses—choose the shampoo of America's most glamorous women. Use the favorite of Hollywood movie stars—Lustre-Creme Shampoo.

Setup

- Create three files:
  - I application (StackDriver class)
  - I interface (StackInterface interface)
  - I class (ArrayBoundedStack class)
- Does it compile?

Stack Operations

- Push
- Pop
- Top



Implementation

- Skeleton
- Class variables
- Basic functions

Test suite #1

- Declare and instantiate ArrayBoundedStack<Integer>(3)
- Push(1)…Print Top()…Pop()
- Print Top()

Exceptions

- Create StackUnderflowException Class
- New method declarations
  - Base constructors on super()
  - Declare Interface methods will throw exceptions
- New throw calls
  - Can this step be simplified?

New Observer

- Interface method header
- Implementation method body
- Test suite #2:
  - Declare and instantiate ArrayBoundedStack<Integer>(3)
  - Push(1)… Print isEmpty()... Print Top()…Pop()… Print isEmpty()...Pop()
- Repeat the addition of StackUnderflowExceptions to the Pop() method interface and implementation

Exceptions round 2

- Note that too many calls to Push(3) produces an exception
- Create StackOverflowException class
- New method declarations
  - Base constructors on super()
  - Declare Interface methods will throw exceptions
- New throw calls
  - Add isFull() observer

Bells and Whistles

- Second constructor
- Auto-sizing
- Layered abstraction

ArrayList Implementation

- ArrayList is an ADT that re-sizes automatically and is based on arrays
- Example declaration:
- To simplify our stack, we can implement our ADT on another ADT

Code Part I

```
//---------------------------------------------------------------------------
// ArrayListStack.java by Dale/Joyce/Weems Chapter 2
//
// Implements an unbounded stack using an ArrayList.
//---------------------------------------------------------------------------
package ch02.stacks;

import java.util.*;

public class ArrayListStack<T> implements StackInterface<T>
{
  protected ArrayList<T> elements; // ArrayList that holds stack elements

  public ArrayListStack()
  {
    elements = new ArrayList<T>();
  }
```

Code Part II

```
public boolean isEmpty()
// Returns true if this stack is empty, otherwise returns false.
{
  return (elements.size() == 0);
}

public boolean isFull()
// Returns false - an ArrayListStack is never full.
{
  return false;
}
```

Code Part III

```
public void push(T element)
{
  elements.add(element);
}

public void pop()
{
  if (isEmpty())
    throw new StackUnderflowException("Pop attempted on empty stack.");
  else
    elements.remove(elements.size() - 1);
}

public T top()
{
  T topOfStack = null;
  if (isEmpty())
    throw new StackUnderflowException("Top attempted on empty stack.");
  else
    topOfStack = elements.get(elements.size() - 1);
  return topOfStack;
}
```

Recap

- Stack implementation using a bounded array
- Three phase development
  - Lather, rinse, repeat
- ADT can be layered to simplify each step
  - ADT inception

Next Time...

- Dale, Joyce, Weems Chapter 2.6, 2.9
  - Remember, you need to read it BEFORE you come to class!
- Check the course webpage for practice problems
- Peer Tutors
  - http://www.csc.villanova.edu/help/