Chapter 02, 2.5 – Array-Based Stack Implementation, Review Problem(s): ***25, 26, 31***

Chapter 02, 2.6 – Stack Applications, Review Problem(s): ***35, 36, 37***

The textbook presents two array-based implementations of a stack. The first is the `ArrayBoundedStack` Class of a bounded Stack ADT. The second is an unbounded implementation that uses the Java Library `ArrayList` class.

For any implementation, the methods which need to be created are:

- isEmpty
- isFull
- push
- pop
- top

The top down approach, understanding the functionality and parameters first makes the implementations easier.

Now, we consider an application of a stack to determine if in a set of grouping symbols (we'll use the "normal" ones, (), [], and {}) the open and close versions of each symbol are matched correctly.

- Any number of other characters may appear in the input expression, before, between, or after a grouping pair, and an expression may contain nested groupings.
- Each close symbol must match the last unmatched opening symbol and each open grouping symbol must have a matching close symbol.

We use an integer to indicate the result:

- 0 means the symbols are balanced, such as (([xx])xx)
- 1 means the expression has unbalanced symbols, such as (([xx}xx))
- 2 means the expression came to an end prematurely, such as (([xxx])xx

Here's the algorithm:

```
Create a new stack of size equal to the length of input
expression

Set stillBalanced to true

Get the first character from expression

while (the expression is still balanced AND there are
still more characters to process)
  Process the current character

  Get the next character from expression

if (!stillBalanced)
  return 1
else if (stack is not empty)
  return 2
else
  return 0
```

Processing the next character:

```
if (the character is an open symbol)
  Push the open symbol character onto the stack

else if (the character is a close symbol)
  if (the stack is empty)
    Set stillBalanced to false
  else
    Set open symbol character to the value at the top
of the stack
    Pop the stack
    if the close symbol character does not "match" the
open symbol character
      Set stillBalanced to false

else
  Skip the character
```

Now, in Lab 06, you will take existing code, and add functionality.