Ch. 12 Review Assignment

Part 1:

Take the `Stack` class that you made for section 12.3 and modify it so that the stack stores `String`s instead of `int`s.

Part 2:

You will be provided with a class called `StackShell` that gives you the shell of the program you need to write. It will handle the GUI aspects of this program for you. All you need to do is complete the `execute` method, making use of your `Stack` class.

The program will allow a user to enter a "stack program". This consists of several lines of pushes and pops. Here is a sample program:

```
push Hello there.
pop
```

This would produce output of:

```
Hello there.
```

Each statement in a "stack program" must begin with `push` or `pop` (your program should accept upper or lower case or any combination). Whatever follows `push` will get pushed onto a stack. If the `String` ends after `push` then push an empty string. There should be at least one space between `push` and what gets pushed. `Pop` shouldn't be followed with anything.

There are two types of errors that can be generated. If the program tries to `pop` an empty stack the output should be:

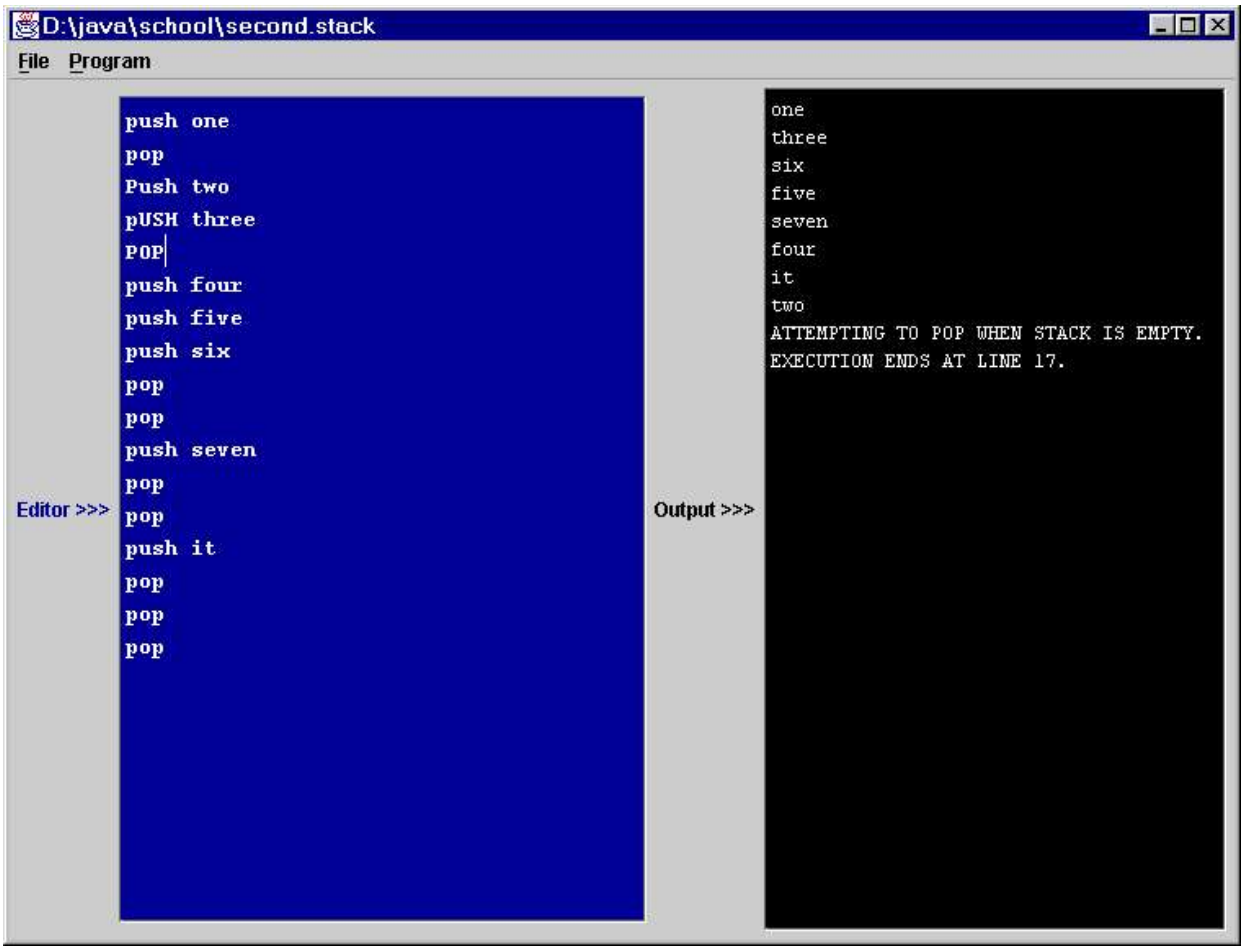
```
ATTEMPTING TO POP WHEN STACK IS EMPTY.
EXECUTION ENDS AT LINE x.
```

The x is whatever line the empty `pop` occurred on. Note that an exception should NOT be thrown. The user can correct the program and try running it again. If a line starts with any thing other than a `push` or a `pop` the output should have:

```
INVALID COMMAND ENCOUNTERED ON LINE x.
```

In either case you should stop executing the "stack program".

Here is an example of what the program should look like when you run it:



Here is what you have in the `execute` method:

```
public void execute()
{
   String[] code = getCode();
   outputArea.setText("");
}
```

You will need to complete the `execute` method. The first statement gets all the lines that the user has entered (in the blue area on the left) and returns them as an array of `String`. Each line will be a separate `String`. The second line clears the output area (the black area on the right). The object `outputArea` is a `JTextArea`. Another method you'll need from `JTextArea` is `void append(String)`. This appends the string parameter to the JTextArea. The method `getCode()` makes a global variable `codeLength` equal to the number of lines in the "stack program".

You'll need to copy `StackShell.java` from the website.