CSc 210: Software Development
Section 6: Stacks and Queues

October 9th, 2017

In this section we will be using the StringBuilder Object to implement stacks and queues that hold characters.

Exercise 1: StringBuilder Stack

1. For our representation of a stack of characters, we're going to use a mutable string object. Why use this instead of an ordinary String object?

2. We are going to use a StringBuilder object. Look at the Java API to familiarize yourself with this class.

3. On the class web page is the shell of StackSec6.java. Download and open the file.

4. At the moment, all it contains is a main() method that contains some simple tests for instance methods that do not yet exist. Your job for the rest of this part: Write those methods. Specifically:
    - A no-argument constructor that creates the object representing the stack. You need a suitable instance variable to reference it, of course.
    - A boolean-returning, no-argument isEmpty() method.
    - An int-returning, no-argument getOccupancy() method. (Remember, occupancy and capacity are different things.)
    - The standard push(), pop(), and peek() methods. You should be able to figure out the arguments, if any, for each. For those that return a value, code them to return an int, so that both stack elements and error codes (PEEK POP ERROR is already defined for you) can be returned as necessary.

5. Compile and run your completed StackSec6 code, and verify that the results match the expected output given at the top of the file

Exercise 2: StringBuilder Queue

1. On the class web page is the shell of QueueSec6.java. Download and open the file.

2.In your editor of choice, implement QueueSec6 that uses a StringBuilder object to hold characters in a queue. Here are the method signatures of the methods that your class needs to provide:

- QueueSec9() — the constructor; creates an empty queue
- boolean isEmpty() — true if no characters in the queue; false otherwise
- int getOccupancy() — quantity of characters in the queue
- void enqueue(char) — add char to the rear of the queue
- int dequeue() — removes the front character, returns -1 if empty
- int peek() — shows the character at the front of the queue, returns -1 if empty

3. Compile and run QueueSec6. If necessary, debug your methods until you're getting the output specified in the comment.

Part III: Palindromes

A palindrome, as you worked with on the first assignment, is a word or phrase whose letters appear in the same order left-to-right and right-to-left. For example, I'm a lasagna hog, go hang a salami. is a palindrome (we ignore spaces and punctuation symbols). One way to test a string to see if it is a palindrome: Take each letter in the string and both push it onto a stack and enqueue it into a queue. Then, repeatedly pop the stack and dequeue from the queue. If the pairs of characters removed do not match, then the string is not a palindrome. But, if you empty both data structures and all of the pairs of characters have matched, the string is a palindrome.

1.You will be using your Stack and Queue classes you just implemented to solve this problem.

2. Using a pen/pencil and paper, use the algorithm described at the top of this part, and drawings of a stack and a queue, to test each of these four small strings:

(a) kayak (a palindrome)
(b) abcb (one 'a' short of being a palindrome)
(c) bcba (same, but on the opposite end)
(d) abcdeba (not a palindrome; 'c' doesn't pair with 'e')

(The point is to get a feel for how the algorithm works, which should make it easier to write a program that carries out that algorithm.)

3. Write a Java program named PalPal.java (for 'Palindrome Pal') that takes a string from the command line and tests it to see if it is a palindrome using the stack/queue

algorithm. The following execution examples show the behaviors your SL will expect your program to demonstrate:

```
> java PalPal
Usage example: java PalPal "Yo, banana boy!"
 > java PalPal ""
"" is a palindrome!
> java PalPal "Sit on a potato pan, Otis!"
"Sit on a potato pan, Otis!" is a palindrome!
> java PalPal "abcb"
"abcb" is not a palindrome;
the problem is that 'a' doesn't match 'b'.
> java PalPal "bcba"
"bcba" is not a palindrome;
the problem is that 'b' doesn't match 'a'.
> java PalPal "abcdeba" "abcdeba" is not a palindrome;
the problem is that 'c' doesn't match 'e'.
```

Hint: Remember that the Character wrapper class has static methods that can test characters to see if they have various properties.

Once you finish solving this problem, show your SL and you are done!