C S 272/463 Introduction to Data Structures

Lab 8: Generic Programming, Implement and Use Stacks, Queues

I. Stacks

Q1. (2.5 pts) Write a generic interface for stack and put the code in StackInterface.java. This interface should include five functions: push, pop, top, size, and isEmpty.

Q2. (2.5 pts) Write a generic class for the node in singly linked lists and put the code in SNode.java.

Q3. (15 pts) Implement LinkStack.java with the following detailed requirements.

(1) (2 pts) It has ONLY one instance variable, which is a generic node of type SNode.

(2) (13 pts) LinkStack should implement the StackInterface interface and implement all the methods declared in this interface.

Q4. (15 pts) Implement ArrayListStack.java with the following detailed requirements.

(1) (2 pts) It has one instance variable: an ArrayList with a generic data type.

(2) (13 pts) ArrayListStack should implement StackInterface interface and implement all the methods declared in this interface.

Q5. (5 pts) You need to design test cases to test your functions in ArrayListStack.java and LinkStack.java thoroughly. If your test cases cannot cover some important conditions, points may be deducted. Please put your test case files to StackTest.java.

Q6. (10 pts) NQueen.java: Use either stack that you implemented to solve the N-queen problem. Your design needs to follow the logic in the lecture notes. You can also use the program project 10 at page 358 as reference. The parameter should be N (in the range of [1, 16]). The result should print queens at proper positions. For example, the solution at page 358 should be printed as

Q - - - -

- - Q - -

- - - - Q

- Q - - -

- - - Q -

II. Queues

Q7. (5 points) Implement a generic queue interface in QueueInterface.java. It should include the enqueue, dequeue, front, size, and isEmpty methods.

Q8 (30 points) Implement LinkedQueue.java with the following detailed requirements.

(1) (5 points) It has three instance variables. The first instance variable denotes the front of a queue and it is a generic node of type SNode that you implemented. The second instance variable denotes the rear of the queue and it is a generic node of type SNode. The third instance variable denotes the number of elements in the queue.

(2) (5 points) LinkedQueue should have a proper constructor.

(3) (20 points) LinkedQueue should implement the QueueInterface interface and all the methods declared in this interface. (Each method carries 6 pts).

Q9. (5 points) You need to design test cases to test your functions in LinkedQueue.java. If your test cases cannot cover some important conditions, points may be deducted.

Please put your test case to file QueueTest.java.

Q10 (10 points) Palindrome.java: Use the LinkedQueue that you implemented to work on the following question . (which is Programming Project 1 at page 406 in your text book). Write test functions for your implementation in the main method of Palindrome.java.

(Hints: you may consider using both stack and queues) A word-by-word palindrome is a string of words such that the words read the same forward and backward. For example, the quote "You can cage a swallow, can't you, but, you can't swallow a cage, can you?" is a word-by-word palindrome.

Write a program to test an input string and tell whether or not it is a word-by-word palindrome. Consider upper- and lowercase letters to be the same letter. Define a word as any string consisting of only letters or an apostrophe and bounded at each end with one of the following: a space, a punctuation mark, the beginning of the line, or the end of the line.

III. Note

- Specifications for all your classes and methods:

  Please properly explain (1) the functionality of the methods, (2) the parameters, (3) the return values, (4) the pre-conditions if there is any;

  Please use inline comments, meaningful variable names, indentation, formatting, and whitespace throughout your program to improve its readability.

- You can (but are not required to) design and implement other facilitating methods (E.g., other get and set methods, toString method) to finish the implementation of the required methods.

IV. Submission

Submit through canvas a zipped file containing your (1) java file(s) (not .class files).

V. Grading Criteria

1. The score allocation is beside the questions.
2. Please make sure that you test your code thoroughly by considering all possible test cases. Your code may be tested using more test cases.