C S 272/463 Introduction to data structures
Fall 2019
Lab 9: Generic Programming, Implement and Use Queues

1 Learning objectives

Objective 1 (queue), Objective 5 (generic programming), Objective 6, Objective 7

2 Requirements

1 (10 points) Implement a generic interface QueueInterface in **QueueInterface.java**. It should include the `enqueue`, `dequeue`, `front`, `size`, and `isEmpty` methods.

2 (40 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) (30 points) `LinkedQueue` should implement the `QueueInterface` interface and all the methods declared in this interface. (Each method carries 6 pts)

3 (40 points) Implement **ArrayQueue.java** with the following detailed requirements.

(1) (5 points) It has 4 instance variables. The first instance variable denotes the front of a queue and it is an integer. The second instance variable denotes the rear of the queue and it is an integer. The third instance is an array for holding the queue elements. The fourth variable denotes the number of elements in the queue.

(2) (5 points) `ArrayQueue` should have a proper constructor.

(3) (30 points) `ArrayQueue` should implement the `QueueInterface` interface and all the methods declared in this interface. (Each method carries 6 pts)

4 (10 points) You need to design test cases to test your functios in `LinkedQueue.java` and `ArrayQueue.java` **thoroughly**. If your test cases cannot cover some important conditions, points may be deducted. Please put your test case to file **QueueTest.java**.

- (10 points BONUS) **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**.

```
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, cann'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.
```

3 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.

4 Submission

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

5 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.

(3) 5 points will be deducted if submitted files (including files types, file names, etc.) do not follow the instructions.

(4) At least 20 points will be deducted if your code cannot be run on CS servers.