Project 1: Basic Data Structures
Due: 11:59 pm 7/19/19
Total Points: 8

Projects must be submitted on BlackBoard as a ZIPPED FOLDER with the folder name as X{8 Digit CUNY ID}

for example your student id is 12345678 than the folder name is X12345678

Within the files will only be source code, NO .class files. The file structure must be:

1) DLL (Folder)
   a. DLinkedList.java
   b. DNode.java
   c. List.java
   d. Main.java
2) SLL (Folder)
   a. SLinkedList.java
   b. Node.java
   c. List.java
   d. Main.java
3) Queue (Folder)
   a. Queue.java
   b. QueueIterface.java
4) Stack (Folder)
   a. LinkedStack.java
   b. List.java
   c. SLinkedList.java
   d. Node.java
   e. List.java

Any projects submitted that **DOES NOT** have this naming convention will not be graded.
If you do not submit anything, you will receive 1 point for the project. Any projects that **do not compile or work** will receive a 0. Excuses such as “It compiles on my computer” or “It worked last time” will not be accepted. Your program must work on all machines not just yours.
If you are using an IDE such as eclipse, before submitting, remove all package statements from all files.
Late penalty Any project submitted:
1 day late will receive a max possible score of 6 and extra credit will not be rewarded
2 days late will receive a max possible score of 5 and extra credit will not be rewarded
3 days late will receive a ax possible score of 4 and extra credit will not be rewarded
Any project submitted after 3 days will not be graded.
Cheating Any one caught cheating, copying code or letting others copy, will receive a 0 and reported.
Collaborating with others is encourage on a high level, but code and implementation should never be shared.

Project Specs:
There are 4 parts to this project:

A) Complete the Singly Linked List data structure. Make sure the that each method in the List interface is implemented. As well that the data structure is robust.
   a. There are errors right now. It is throwing a null pointer exception when the main function is ran. Find where is that error occurring and fix it.
   b. Finish the remove(Node target) method
   c. Complete the set method. This method takes in two nodes as the params, *old* and *replace*. Find the *old* node in the list and replace that node with the *replace* node. If the *old* node is not in the list, then just add the *replace* node onto the back of the list.
B) Complete the Doubly Linked List data structure. Implement the List interface. None of the methods in the interface is currently implemented

C) Complete the Queue data structure using the circular array implementation. Implement the QueueInterface interface.
    a. The peek() method allows the user to view the next element on the queue that is to be removed but does not remove it. This is different from dequeue() as that method actually removes that element.

D) Complete the Stack data structure using the linked list representation. Implement the StackInterface interface
    a. As we discussed in class, we can implement this stack very quickly by using a singly linked list.
    b. Use the SlinkedList class that you did in part A. However you may need to implement a 2 more methods in that class in order to accomplish this task.
    c. Remember that a stack push and pop should be running in constant time. Where on a singly linked list can we add and remove in constant time?

Each part of this project is worth 2 points.

Testing:
For SLL and DLL, I provided a main method for you to test your data structures. I did not for stack and queue. It is your job to come up with test cases for these data structures.