CSCI 230 HW#4

Collaboration policy: **Individual Assignment**
Total Points: **200**

Source Code

The Java interfaces and classes provided in the zip file attached to this Dropbox assignment are:

- `Stack.java (interface)`
- `Queue.java (interface)`
- `ConstantTimeStack.java (class)`
- `SemiConstantTimeQueue.java (class)`

Furthermore, in this coding assignment the same `List` interface (provided in HW#2) will be used. Under no circumstances are you allowed to modify or create a new `List`, `Stack`, `Queue` interface or a new `Node` class. You must use these files **as is**.

You **may only** modify the `ConstantTimeStack` and `SemiConstantTimeQueue` class. In particular, in both classes you **may only** modify the methods listed in Part 1 and 2, and under no circumstances are you allowed to remove, add, or modify any other line of code in this class (this include instance variables, class variables, constants, etc.).

Additionally, you will reuse the `ArrayList` and `SinglyLinkedList` classes developed in homework assignments two and three. You may use the coding solutions provided by the instructor (on OAKS) without penalty.

Lastly, you **may not** change the package structure! Specifically, `edu.cofc.csci230` cannot be removed or modified. If a solution is submitted with a different package structure, it will not be graded, no exceptions.

Part 1

In the `ConstantTimeStack` class please fully implement the stack interface methods listed below:

- `public void push( AnyType t );`
- `public AnyType pop() throws EmptyStackException;`
- `public AnyType peek() throws EmptyStackException;`

In each method you will see a TODO comment - this is where you add your code. Please ensure a constant time O(1) coding solution is provided for each method listed above. In the

provided source code, numerous comments are given; please ensure you read them carefully.

Part 2

In the `SemiConstantTimeQueue` class please fully implement the queue interface methods listed below:

- `public void add( AnyType t ) throws NullPointerException;`
- `public AnyType remove() throws NoSuchElementException;`
- `public AnyType peek();`

In each method you will see a TODO comment - this is where you add your code. Please ensure a constant time O(1) coding solution is provided for the add and peek methods, and a linear time O(n) coding solution is provided for the remove method. In the provided source code, numerous comments are given; please ensure you read them carefully.

Part 3

The provided `ConstantTimeStack` and `SemiConstantTimeQueue` classes both have main methods. In the main please add test cases that demonstrate that you have fully evaluated the operational correctness of the methods you implemented in Part 1 and 2. To receive full credit, these test cases **must** be included.

Submission

Create a zip file that **only** includes the completed `ConstantTimeStack.java` and `SemiConstantTimeQueue.java` files. If you have any questions about the submission policy, you must resolve before the due date. Lastly, please plan appropriately, asking questions the day the assignment is due (within 12 hours) is too late. Please try to resolve any questions at least 2 days before the due date.

The name of the zip file must be your last name. For example, *ritchie.zip* would be correct if the original co-developer of UNIX (Dennis Ritchie) submitted the assignment. Only assignments submitted in the correct format will be accepted (no exceptions).

Please submit the zip file (via OAKS) to the Dropbox setup for this assignment by the due date. You may resubmit the zip file as many times as you like, Dropbox will only keep the newest submission.

Per the syllabus, late assignments will not be accepted – no exceptions. Please do not email Hassam or I your assignment after the due date, we will not accept it.

Grading Rubric

ConstantTimeStack Compiles	10 points
ConstantTimeStack meets time complexity requirements	15 points
Thoroughness of your test cases	5 points
Instructor test cases (7 cases 10 points each)	70 points
	100 points

SemiConstantTimeQueue Compiles	10 points
SemiConstantTimeQueue meets time complexity requirements	15 points
Thoroughness of your test cases	5 points
Instructor test cases (7 cases 10 points each)	70 points
	100 points

In particular, each data structure will be graded as follows. If the submitted solution

- Does not compile: 0 of 100 points
- Compiles but does not run: 10 of 100 points
- Compiles, runs, and meets time complexity requirements: 25 of 100 points
- Thoroughness of your test cases: 30 of 100 points
- Passes all 7 test cases developed by instructor: 100 of 100 points.