COMP20040: Data Structures and Algorithms II

Practical 1: Stacks and Queues

IF ANYTHING BELOW DOES NOT MAKE SENSE, PLEASE CONTACT A DEMONSTRATOR TO ASK FOR GUIDANCE.

Preliminaries:

Download the latest class codebase from moodle.

Part 1: Link-based Stack **30%**

Create a link-based implementation of a Stack that uses a combination of the Stack interface provided and the course notes.

The Stack implementation should be called LinkedStack, and the test code should be written in a file called LinkedStackTests.

Marks will be given for:

- A correct implementation of all ADT operations. 20%
- Creation of a toString() method to display the state of the stack 5%
- Use of a static class to implement a Node. 5%
- Creation/Use of appropriate exceptions for all boundary cases. 5%
- Adherence to the Stack interface and correct commenting/formatting of code 5%
- Creation of a test program that generates all test cases (see note iii) 10%

Part 2: Array-based Queue **30%**

Create an array-based implementation of a Queue that uses a combination of the Queue interface provided ad the course notes.

Marks will be given for:

- A correct implementation of all ADT operations. 15%
- Use of a circular array. 10%
- Creation of a toString() method to display the state of the stack 5%
- Creation/Use of appropriate exceptions for all boundary cases. 5%
- Adherence to the Queue interface and correct commenting/formatting of code 5%
- Creation of a test program that generates all test cases (see note iii) 10%

NOTES:

i) See the next page for a brief description of what is expected of you when you are asked to create a toString() method.

ii) The marks on the right are MAXIMUM values. You will marks for sloppy coding / incomplete work.

iii) For the test class, create a method for each test you wish to perform. Use comments immediately BEFORE the start of the method implementation to describe what test your method will perform. Call each method testXXX() where XXX relates to the test you are carrying out.

Note on Creating a toString() method:

`toString()` methods are used to generate a string representation of the current state of the associated object. A default implementation is provided as part of the Object class which prints out a combination. This default implementation returns a String that is a combination of the fully-qualified class name and the hash code of the object. The idea is that you can look at the output of this method for two different objects and will know whether the objects are semantically equivalent.

Sematic Equivalence is where two different objects represent the same concept. For two objects to be semantically equivalent they must be instances of the same class and the state (the values assigned to each field) should be the same. For example, the objects created by `Integer.parseInt("50")` and `new Integer(50)` would be semantically equivalent.

To override this method, simply declare the method in the class (note – you do not need to declare it in the interface because it is already defined in the `Object` class.

For Stacks and queues, it is common to print out the values stored in them.

For a link-based Stack a toString() method would output something like:

```
<size> : <space delimited list of items*>

*the leftmost item being the bottom of the stack
```

For an array-based Queue a toString() method would output something like:

```
<size> / <capacity> : <space delimited list of items*>

*the leftmost item being the rear of the stack
```