CE4702 Exercise Sheet 5

Ex5.1: *Generic LinkedList*
Code a class for a generic linked list, i.e. a linked list class that can store objects of any class. Create your own linked list class or modify the sample linked list class discussed in the lecture. Do not use any of the container classes provided by the Java language !!!

Ex5.2: *Generic Dynamic Stack*
Based on the Linked List class from the previous exercise, implement a generic Stack class.

Ex5.3: *Generic Queue*
Based on the Linked List class from the previous exercise, implement a generic Queue class.

Ex5.4: *Generic Dynamic Stack 2*
Using a class from the Java Collection Framework, implement a generic Stack class.

Ex5.5: *Generic Queue 2*
Using a class from the Java Collection Framework, implement a generic Queue class.

Ex5.6: *Generic Dequeue*
Using a class from the Java Collection Framework, implement a generic Dequeue class.

Marked Exercise

Ex5.7: *ADT Application*

This is a group project for teams of 3-5 students.

The task is to create your own application utilizing ADTs. I assume you are writing an application to maintain a store inventory - however, any application meeting the criteria summarized below will do:

Follow the "Five Steps of Good ADT Design" to create an ADT for the items in the store (anything you can think of) and then create an ADT "StoreInventory" (what operations do you deem useful?). Base the implementation of your ADT StoreInventory on any class from the Java Collection Framework. Provide a simple application that uses your ADT and allows users to add, remove, update and sort inventory items.

Application Properties

- Your application must provide at least two ADTs - one ADT for some sort of item and another ADT to store objects of your first ADT. Provide formal specifications (as discussed in the lecture) as comments at the beginning of each class).
- You must use exceptions to deal with errors in your application. Provide at least one exception class yourself (feel free to use existing exceptions for additional errors).
- Provide one sorting functionality based on any `java.util.Collections` sorting method.
- Provide a second sorting functionality where you implement a sorting algorithm yourself.
- Provide a "Sorting Details" option in your application that displays 1) name of your selected sorting algorithm, 2) short description of the operation of your selected algorithm in plain English, 3) pseudocode for your sorting algorithm, 4) Efficiency of your sorting algorithm in Big-O Notation.
- Feel free to provide additional ways of sorting, for example, you could let the user pick by what property you sort.
- A JOptionPane user interface is sufficient, but feel free to provide a full swing/AWT GUI.