**COMP232 – Data Structures & Problem Solving**

**Homework #3**

**Generics & Linear Structures**

All of the code used in this assignment is available in the homework3 repo (assignment code link provided on Moodle).

1. Create a generic class named Pair in your homework3 repo. The Pair class is able to store and retrieve two values. The types of these values must be specified by type parameters. The Pair class must include the following methods:

* A constructor that allows the two values to be specified as parameters.
* getFirst – returns the first value
* getSecond – returns the value of the second value
* setFirst – changes the value of the first value
* setSecond – changes the value of the second value

It is not necessary to write any comments or tests for the Pair class.

2. Add a main method to your Pair class. In that method define the following variables and objects:

i. A variable intPair that refers to a Pair object that holds two Integer objects.

ii. A variable mixPair that refers to a Pair object that holds a Double object and a String object.

iii. A variable pairPair that refers to a Pair object that holds two Pair objects, one as defined in part i, and the other as defined in part ii.

3. Copy the COMP132Queue interface from the structures.objects package in the COMP232 sample code to your homework3 repo and rename it to MyQueue. This interface uses Object as the type for the elements on the stack. Modify your new MyQueue interface so that the type of the elements that the queue can hold is specified using a type parameter. Note: You do not have to implement a queue, just rewrite the interface so that it is generic.

4. Create a MyArrayStack class in your homework3 repo that implements the MyStack interface and uses a MyArrayList as the backing store. Use the methods of the MyArrayList to implement the stack operations. Try to make the stack operations as efficient as possible.

5. Complete the implementation of the following methods that appear in the MyDoublyLinkedList class in your homework3 repo:

* remove
* clearTo
* addAllAt

6. Complete the implementation of the following methods that appear in the DLLIterator inner class in the MyIterableDoublyLinkedList class in your homework3 repo:

* hasPrevious
* previous
* remove – Hint: use a field in the iterator to keep track of the node returned by the most recent call to next or previous and use that to determine if the call to remove is valid, and if so what element to return.

7. Modify the MyArrayList class in your homework3 repo so that it implements the parts of the MyIterable interface as describe below. You will need to:

i. Make MyArrayList implement the MyIterable interface.

ii. Add an inner class to MyArrayList that implements the MyIterator interface. This class should support the hasNext, next, hasPrevious and previous methods of MyIterator interface. You do not need to implement the insert or remove methods, have these methods throw an UnsupportedOperationException.

Hints: The cursor can simply be an integer. Take advantage of the get method in the MyArrayList class when implementing the iterator operations.