**Lab Task 1.**

In the theory class, we created a generic stack class supporting push and pop operations. Now, Create the generic queue data structure supporting the following operations.

* Enqueue: append element to arraylist, must throw an exception if queue is full
* Dequeue: remove first element of arraylist, must throw an exception if queue is empty
* isEmpty: check if arraylist is empty
* isFull: check if arraylist size is greater than queue capacity
* ReverseQueue: **Hint:** You can use another stack or queue for this purpose

Test different methods of your generic queue in the test class.

**Lab Task 2.**

Consider the Accounts class containing accountTitle and balance fields. Update the class so that it extends the comparable interface. The compareTo method should compare the balance of 2 account objects. Then, add a comparator class implementing the comparator interface, whose compare method compares the accounts with respect to account titles.

Create 5 account objects inside the main class and add them to any arraylist. Use the sort method in java collections framework to sort the accounts both with respect to account balance and account title.

**Home Task 1.**

(Overloaded Generic Method printArray) Overload generic method printArray of Fig. 20.3 so that it takes two additional integer arguments, lowSubscript and highSubscript. A call to this method prints only the designated portion of the array. Validate lowSubscript and highSubscript. If either is out of range, the overloaded printArray method should throw an InvalidSubscriptException; otherwise, printArray should return the number of elements printed. Then modify main to exercise both versions of printArray on arrays integerArray, doubleArray and characterArray. Test all capabilities of both versions of printArray.