**Requirement**

**Part I:**

1. Design a Stack <E> that support the following operations. You cannot use the Java’s Stack interface. You have to design it from the scratch. *Make sure that stack operations are implemented efficiently*.
   * A constructor that creates an empty stack.
   * pop() – if the stack is non-empty, it removes the top element off the stack. If the stack is empty, an exception is thrown, but it should be caught and handled gracefully.
   * push(int) – pushes an element onto the top of the stack.
   * top() – if the stack is non-empty, returns the top element. If the stack is empty, an exception is thrown, but it should be caught and handled gracefully. This method does not remove the top element.
   * isEmpty() returns true or false depending on whether the stack is empty or not.

Test your Stack operations by a JUnit test suite by creating a Stack of Student objects.

1. Design a Queue <E> data structure that supports the following operations. You cannot use Java’s Queue class. You have to design it from the scratch. *Make sure that Queue operations are implemented efficiently*.
   * Constructor – to create an empty queue
   * enqueue(int) – adds an element to the end of the queue
   * dequeue() – if the queue is non-empty removes the element from the front of the queue. If the queue is empty, an exception is thrown, caught and handled gracefully.
   * front() – if the queue is non-empty, returns the element at the front of the queue. If the queue is empty, an exception is thrown and handled gracefully.
   * isEmpty() – returns true or false depending on whether the queue is empty or not.

Test your queue operations by a JUnit test suite by creating a Queue of Student objects

**Part II:**

* Implement a Queue using the Stack data structure that you created in part I. This implementation should support all the queue operations, but internally it delegates the work to a stack. The driver module remains the same and should work the same way. If your code works the way it is supposed to, then the same Junit test suite (in 2) should work with this implementation as well.
* Implement a Stack using the Queue data structure from part I. This implementation should support all the stack operations, but internally it delegates the work to a queue. The driver module remains the same and should work the same way. If your code works the way it is supposed to, then the same Junit test suite (in 1) should work with this implementation as well.