In this project you will write code for Dequeue data structure. Dequeue is a special kind of a Queue where additions and deletions can be done at both ends. You are to implement Dequeue using a circular array. Dequeue has instance variables *front*, *rear* and *elements* (the array that holds the elements of the Dequeue). Go over the circular array implementation of Queue given in your book.

Recall that, for an empty Queue *rear* is one-position counter clockwise to *front* and unfortunately, the same configuration holds for a full queue. To distinguish between a full-queue and an empty queue, we could keep track of the number of elements in *count* instance variable. (if *count* == 0, the queue is empty, and if *count* == *elements.length*, queue is full).

In our implementation of Dequeue, there is no *count* variable to keep track of the number of elements. Instead, to distinguish a full-queue from an empty queue, we never allow the queue to get completely full. A full queue is one with exactly one empty spot in the array. That way we can distinguish an empty queue and full queue by looking at the relative positions of *front* and *rear*.

When Dequeue is created, it starts off with an array of 5 elements. Go over the array implementation of Stack if you need help (Week 5 tab in Blackboard).

You are doc folder with all the files generated by Javadoc utility and Junit test suite. You are to write code for Dequeue.java from the .html specifications.

Draw diagrams to understand the concepts before you start to code.

Deliverables:

Dequeue.java (or the entire src folder from Eclipse) and your discussion log.

Note: Java may not compute the % of a negative number correctly. Be aware of this when coding.