Queue

Based on the notes from David Fernandez-Baca and Steve Kautz

Bryn Mawr College
CS206 Intro to Data Structures
A queue is a list that operates under the **first-in first-out (FIFO)** policy:

- read or remove only the item at the **front (head)** of the queue
- add an item only to the **back (tail)** of the queue
- examine the front item.

java.util contains a Queue<E> interface that contains all the methods you would expect from a FIFO queues, as well as other kinds of queues. Java offers several implementations, for example the LinkedList class.
The Java Queue Interface

- **E element()**: Retrieves, but does not remove, the head of this queue. Throws `NoSuchElementException` if this queue is empty.
- **E peek()**: Retrieves, but does not remove, the head of this queue, or returns null if this queue is empty.
- **boolean add(E e)**: Inserts the specified element into this queue if it is possible to do so immediately without violating capacity restrictions, returning true upon success and throwing an `IllegalStateException` if no space is currently available.
• **boolean offer(E e)**. Inserts the specified element into this queue if it is possible to do so immediately without violating capacity restrictions.

• **E poll()**. Retrieves and removes the head of this queue, or returns null if this queue is empty.

• **E remove()**. Retrieves and removes the head of this queue. Throws NoSuchElementException if this queue is empty.

Since Queue<E> extends Collection<E>, it inherits all of the latter’s methods, including isEmpty(), size(), and iterator().
A queue is easily implemented as a singly-linked list with a tail pointer.

It is perhaps even better to use a *circular* list. In this case, a pointer to the last node also gives easy access to the first node, by following one link. Thus we can handle the structure by a single pointer, instead of two.
Implementation – Array-Based

We use an array a to store the elements. Additionally, we have two indices:

- first: points to first element of queue (front)
- last: points to first available slot in the array (just before the back)

We initialize first = last = 0. The queue is empty when first == last.

- To enqueue, put the new item in A[last] and increment last.
- To dequeue, return A[first] and increment first.
Implementation – Circular Array

• A potential problem using array:
  After a series of enqueue/dequeue operations, both first and last are at the end of the array even though the array is not full.

• Solution: treat the array as being circular.
  That is, when last == A.length and we need to increment last to insert a new item, we just reset last to 0.
  \[
  \text{last} = (\text{last} + 1) \mod \text{A.length};
  \]
  Note that we still use the convention that the queue is empty when first == last. This means that at least one entry of array A will always have to be left unused. Otherwise, we wouldn’t be able to distinguish between an empty queue and a full one.