Level: Easy

Implement a Queue using an array.

Questions to Clarify:
Q. Is the size of the array fixed?
A. Yes, the array has a fixed size provided as an input.

Q. Can we assume that the Queue will store integers?
A. Yes

Solution:
Implement a circular queue in the array. Maintain 2 pointers in the array, front and back.
Add elements to the back and remove elements from front.

The trick to making it easy: maintain a length variable. This keeps the number of elements in the queue. This way, we can easily tell if the queue is full or empty.

Pseudocode:

```
(Note: Never write pseudocode in an actual interview. Unless you're writing a
few lines quickly to plan out your solution. Your actual solution should be in
a real language and use good syntax.)


front - Where the front element of the queue is - initially 0
back - Where the next element goes - initially 0
length - number of elements int the queue
add() // also known as enqueue
   if (length == a.length)
         throw exception
   a[back] = n
   back = (back + 1) % a.length
   length ++

remove() // aka dequeue
   if (length == 0)
         throw empty queue exception
   result = a[front]
   front = (front + 1) % a.length
   length--;
   return result;
```

Test Cases:
Edge Cases: Empty array, empty queue
Base Cases: Single element in queue, 2 elements in queue
Regular Cases: Queue full, general case

Time Complexity: O(1) for insertion and deletion

Space Complexity: O(1) extra space after the initial array

```
public class Queue {
    int[] a;

    int front;
    int back;
    int length;

    public Queue(int capacity) {
        a = new int[capacity];
        front = 0;
        back = 0;
        length = 0;
    }

    public void add(int item) throws QueueFullException {
        if (length == a.length)
            throw new QueueFullException();
        a[back] = item;
        back = (back + 1) % a.length;
        length++;
    }

    public int remove() throws QueueEmptyException {
        if (length == 0)
            throw new QueueEmptyException();
        int result = a[front];
        front = (front + 1) % a.length;
        length--;
        return result;
    }
}

/*
 * Helper Code. Ask the interviewer if they want you to implement.
 */

public class QueueFullException extends Exception {
    public QueueFullException() {

    }
}
```

```
public class QueueEmptyException extends Exception {
    public QueueEmptyException() {

    }
}
```