## Part B: Queue & Deque Implementations

1) **ArrayQueue**: Create a new Java class called `ArrayQueue` that realises the array-based Queue implementation strategy. As with the Stack implementations (Assignment 4), remember to implement a toString() method to help you debug / visualise the operation of the class.

public class ArrayQueue<E> implements Queue<E> {.... include all operations here...}

Queue Interface is given..

2) **LinkedDeque**: Create a new Java class called `LinkedDeque` that realises the link-based Deque implementation strategy. Your implementation should be based on the pseudo code you completed in questions A4-6. Again, remember to implement a toString() method to help you debug / visualise the operation of the class.

public class LinkedDeque<E> implements Deque<E>{.... include all operations here...}

Deque Interface is given..

EmptyDequeException and EmptyQueueException are provided.

## Part C: Evaluation

The final part of this worksheet requires that you test your implementations against the dry runs that you performed in part A. For each Queue question, you should write 2 program start classes: one for each implementation strategy; and for each Deque question, you should write 1 program start class. As a naming convention you should use the following: PCQxA.java for the ArrayQueue implementation strategy and PCQxL.java for the LinkedQueue / LinkedDeque implementation strategys. x should be the question number.

(4) Write two main methods that correspond to the series of insertion and removal operations outlined in question A7.

You can use one main method here as you have implemented only the LinkedDeque and not the LinkedQueue... Do not forget to import your LinkedDeque class

```
public class PCQ4L {

    public static void main(String[] args) {
        LinkedDeque<String> a = new LinkedDeque<String>();

.....}
```