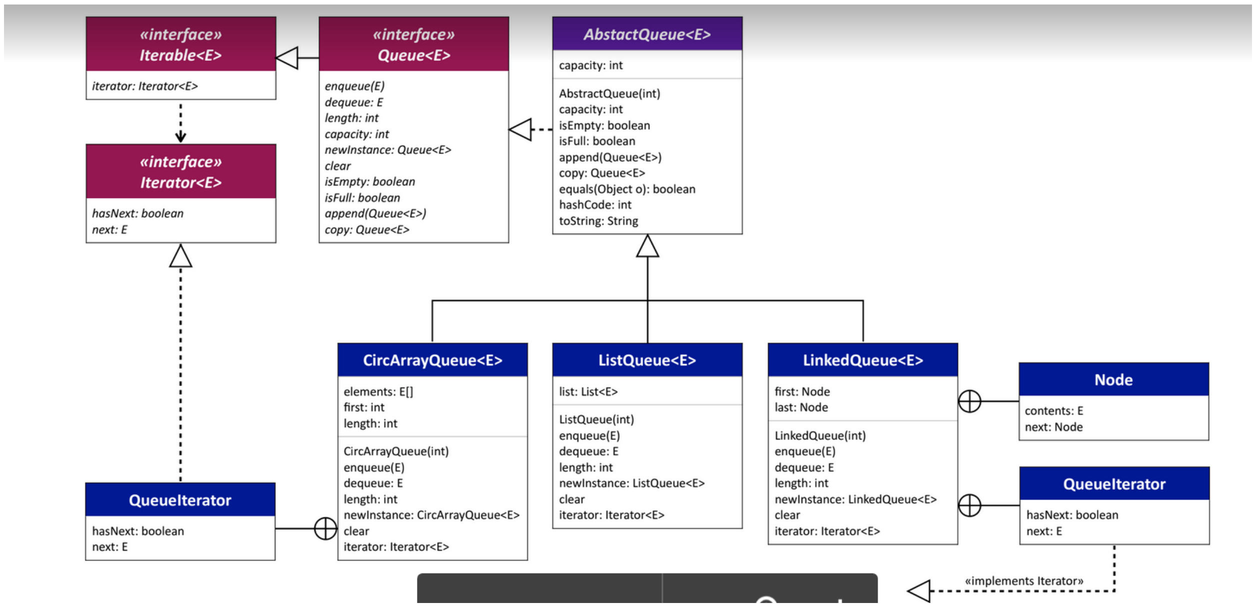
Project 2 - Components

Overview

Create a **bounded queue component** consistent with the following class diagram.



Please ensure:

- All methods in CircArrayQueue should run in linear time (no loops)
- The append method in AbstractQueue should be implemented as a recursive method (no loops and it should call itself)

The recordings on a bounded stack component should help a great deal, especially with the list-based queue implementation. The array-based queue implementation, however, is significantly more complicated than the array-based stack implementation. To get an idea of how a circular-array queue should work see this **set of slides (Links to an external site.)**.

The bounded queue interface should be named `Queue` and it should have the following methods:

```
void enqueue(E element);
E dequeue();
int length();
int capacity();
Queue<E> newInstance();
void clear();
boolean isEmpty();
boolean isFull();
void append(Queue<E> that);
Queue<E> copy();
```

The Queue interface should extend `Iterable`. And the queue should not allow null values. The interface should be fully documented with JavaDocs. The methods should list and document all exceptions. If any argument is a null value, an illegal argument exception should be thrown. If any argument is unacceptable for any reason, an illegal argument exception should be thrown. If any method is called when the queue is not in a correct state, an illegal state exception should be thrown.

The method newInstance creates a new, empty bounded queue with the same capacity as the calling queue. The method append empties its queue argument (that).

The complete queue component will also have an abstract class `AbstractQueue` and three concrete implementations: `ListQueue`, `CircArrayQueue`, and `PointerQueue`. The `toString` method should look similar to the one in the stack component. For example, `[A, B, C]:20` represents a queue with a bound of 20, where A is the element at the beginning of the queue and C is the element at the end of the queue.

Write unit tests for your classes that cover all of your code.