

## Aufgabe 1

Implementieren Sie eine Klasse `ArrayQueue`. Schreiben Sie eine Testklasse, in der sie `ArrayQueue` mit Objekten einer von Ihnen geschaffenen Klasse verwenden.

Hinweis: dazu sollten Sie zunächst die Spezifikation von `Stack` und `Queue` verstehen und die zur Verfügung gestellte Klasse `ArrayStack` studieren. Dann überlegen Sie, wie die Idee umgesetzt werden kann, ein `Array` als zyklisch aufzufassen.

Einschub: Erläuterung des `Queue`

`Object front()`: liefert das nächste Element; das was am längsten gewartet hat

`enqueue(Object)`: fügt ein Objekt der Menge hinzu

`Object dequeue()`: liefert dasselbe Element wie `front()` und entfernt es zugleich.

`boolean isEmpty()`: Test ob noch ein Objekt wartet

## Lösung:

```
public class ArrayQueue {
    private Object elements[] = null; Array elements
    private int min = 0; Zeiger für die unterste Stelle im Stack
    private int max = 0; Zeiger für die höchste Stelle im Stack

    // Konstruktoren
    public ArrayQueue() { Default-Constructor
        elements = new Object[100];
    }

    public ArrayQueue(int size) { Constructor mit vorgegebener Array-Größe
        elements = new Object[size];
    }

    public void enqueue(Object obj) throws QueueException { Methode enqueue()
        if ((max == elements.length && min == 0) || (max == min-1))
            // Kapazitaet erschoepft überprüfen, ob Stack voll ist
            throw new QueueException();
        if (max == elements.length) wenn Ende erreicht, geht's vorne wieder los
            max = 0;
        elements[max++] = obj; max (höchste Stelle im Stack) wird inkrementiert
    }

    public Object dequeue() throws QueueException {
        if (isEmpty()) überprüfen, ob Stack leer ist
            // Queue ist leer
            throw new QueueException();
        Object o = elements[min++]; o = oberstes Element, min inkrementieren
        elements[min] = null; oberstes Element wird gelöscht
        return o;
    }

    public Object front() throws QueueException { überprüfen, ob Stack leer ist
        if (isEmpty())
            throw new QueueException();
        if (min == elements.length) wenn min das Ende des Stacks erreicht hat, an Anfang bringen
            min = -1;
        return elements[++min]; min inkrementieren
    }

    public boolean isEmpty() { Funktion isEmpty()
        return min == max; wenn die Werte von min und max identisch sind, ist davon auszugehen, dass der Stack leer ist.
    }
}
```

Weil wir die Fehler mit einer Exception namens QueueException abfangen, brauchen wir eine entsprechend bezeichnete Klassendatei:

```
import java.lang.Exception;

public class QueueException extends RuntimeException {
    public QueueException (String msg) {
        super (msg);
    }
    public QueueException () {
    }
}
```