

Ringpuffer mit Arrays

Ein **Ringpuffer** (engl. *Buffer*) ist eine Warteschlange einer vorgegebenen Größe. Er kann Daten auf einer Seite anfügen und auf der anderen Seite wieder auslesen. Die folgenden Methoden müssen Sie zur Verfügung stellen:

`init(maxCount: integer)`: initialisiert den Puffer mit einer maximalen Elementzahl.

`add(o: Object)`: fügt ein Objekt am Ende des Puffers ein.

`get()`: `Object`: liest das Element aus, das schon am längsten im Puffer ist.

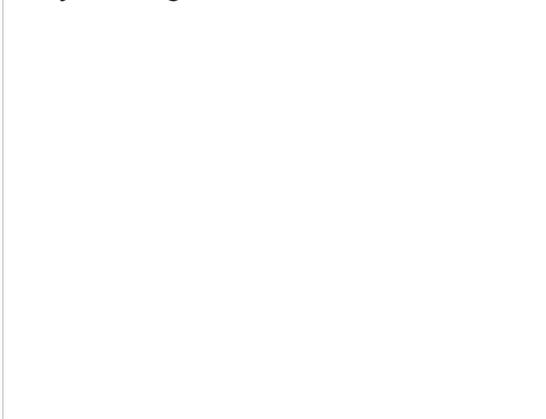
Implementieren Sie diese Schnittstelle mit einem Array. Daneben gibt es zwei (versteckte) Attribute: `erster`: `integer` und `anzahl`: `integer`.

Die Variable `erster` zeigt auf das Element, das zuerst eingefügt wurde, also dasjenige, das nun als nächstes ausgelesen werden soll (First in/First out = FiFo).

Die Variable `anzahl` gibt an, wie viele Elemente zurzeit gültig sind und somit auch, wo das nächste Element in den Array eingefüllt werden muss.

Das nächste einzufügende Element kann (falls `erster > 0`) die Array-Grenze sprengen, obschon wieder freier Platz auf den ersten Feldern vorhanden ist (s. Grafik). Die nächste freie Position ist somit $(\text{erster} + \text{anzahl}) \text{ MODULO } \text{maxCount}$. MOD maxCount ist insofern wichtig, denn der nächste freie Platz kann auch vor dem ersten zu liegen kommen. Daher der Name **Ringpuffer** (s. Grafik).

Array als Ringbuffer



Wenn wir nun diesem Datentypen Ringpuffer jedes erdenkliche Objekt (String, integer, boolean, Verbund-Variable, ...) hinzufügen können, sprechen wir von einem **abstrakten Datentypen**.

Author: Philipp G. Freimann
(BBW
(Berufsbildungsschule
Winterthur)
<https://www bbw.ch>)