# OccurrenceTimeIndex Specification

**Overview:** The *OccurrenceTimeIndex<T>* class implements a fixed-size circular buffer that stores the most recent events of type *T*. It maintains up to *N* elements, where *N* is the specified capacity of the buffer. The buffer supports adding new events and retrieving the $k^{th}$ newest event.

**Parameters and Variables**

**Capacity**: $N \in N, N \geq 1$

The fixed size of the buffer.

**Current Number of Items**: $M \in \{0,1, \dots, N\}$

The number of items currently in the buffer.

**Current Index**: $currentIndex \in \{0,1, \dots, N-1\}$

The index where the next event will be added.

**Buffer Array**: $Array = [a_0, a_1, \dots, a_N - 1]$

An array of fixed size $N$ storing elements of type $T$.

**Items Amount**: $M = ItemsAmount$

Represents the number of valid items in the buffer.

**State Representation:** At any time, the buffer contains a sequence of up to $N$ elements:

$$S = [e0_0, e_1, \dots, e_M - 1]$$

where

$e_0$ is the oldest event

$e_{M-1}$ is the newest event

The mapping between logical positions in $S$ and physical positions in the array is:

$$For\ i = 0\ to\ M-1:$$

$$PhysicalIndex(i) = (currentIndex + i - M + N)\ mod\ N$$

and

$$e_i = Array[PhysicalIndex(i)]$$

**Operations**

1. **Constructor**: *OccurrenceTimeIndex(N)*

**Purpose**: Initializes the buffer with a fixed capacity N.

**Precondition**: $N \geq 1$

**Postcondition**: $M = 0$ and $currentIndex = 0$

Array is an array of size $N$

The buffer is empty.

**Exception**: If $N \leq 0$, an exception is thrown.

2. **Add(eventValue)**

**Input**: $eventValue \in T$

**Behaviour**:

**Insert Event**: $Array[currentIndex] = eventValue$

**Update currentIndex**: $currentIndex = (currentIndex + 1)\ mod\ N$

**Update** *ItemsAmount*: $M = min\ (M + 1, N)$

**Postcondition**:

$If\ M < N,\ M\ increases\ by\ 1$

$If\ M = N,\ M\ remains\ N\ (buffer\ is\ full)$

$The\ oldest\ event\ may\ be\ overwritten\ if\ the\ buffer\ is\ full$

**Note**:

The buffer maintains the most recent N events.

3. **GetKthNewestElement(k)**

**Input**: $k \in N, k \geq 0$

**Output**:

$Returns\ e_M - 1 - k\ if\ 0 \leq k < M$

$Returns\ null\ (or\ default\ value)\ if\ k \geq M\ or\ M = 0$

**Behaviour**:

**Check for Empty Buffer**: $If\ M = 0, return\ null$

**Check for Out-of-Bounds**: $If\ k \geq M, return\ null$

**Calculate Physical Index**: $index = (currentIndex - 1 - k + N)\ mod\ N$

**Retrieve Element**:

$result = Array[index]$

Return result

**Explanation**: $k = 0$ corresponds to the newest event $e_{M-1}$

The calculation adjusts for potential negative indices due to wrap-around.

4. **ToString()**

**Purpose**: Returns a string representation of the buffer's current state.

**Behaviour**:

**Outputs:**

$Buffer: [a_0, a_1, \ldots, a_{N-1}], \quad CurrentIndex : currentIndex, \quad ItemsAmount : M$

Invariants: At all times, the following conditions hold:

**Capacity Constraint**: $0 \leq M \leq N$

**Indices Range**: $0 \leq currentIndex < N$

**Mapping Between Logical and Physical Indices**:

$For\ i = 0\ to\ M - 1$

$e_i = Array[(currentIndex + i - M + N)\ mod\ N]$

**Wrap-Around Handling**:

The modulo operation ensures indices remain within valid array bounds.

**Notes**

**Circular Buffer Behaviour**: The buffer overwrites the oldest events when it becomes full, ensuring that only the most recent $N$ events are stored.

**Event Retrieval**: Events are retrieved based on their relative age, with $k = 0$ being the most recent.

**Edge Cases**: When the buffer is empty ($M = 0$), retrieval methods return *null*.

When $k \geq M$, the requested event is out of bounds, and *null* is returned.

**Physical vs. Logical Indices**: Physical indices refer to positions in the underlying array.

Logical indices refer to the order of events based on insertion time.

Example Usage

**Initialization**: $N = 5$

Buffer is empty: $M = 0,\ currentIndex = 0$

**Adding Events**: Add events $e_a, e_b, e_c, e_d, e_e$ sequentially

After each addition, $M$ increments by 1 until it reaches $N$

**Retrieving Events**: To get the newest event ($k = 0$)

$$index = (currentIndex - 1 + N)\ mod\ N$$

For older events, increase $k$ accordingly

**Buffer Overflow**: Adding a sixth event $e_f$ when $M = N$

$e_f$ overwrites the oldest event $e_a$

$M$ remains $N$

**Conclusion:** This mathematical specification precisely defines the behaviour of the *OccurrenceTimeIndex<T>*, capturing its circular buffer mechanics, event addition, retrieval logic, and internal state management. The class ensures efficient storage and access to the most recent events within a fixed-size buffer.