## Tutorials and Exercises

- Weeks 6-8, due 19/11: Sliding window

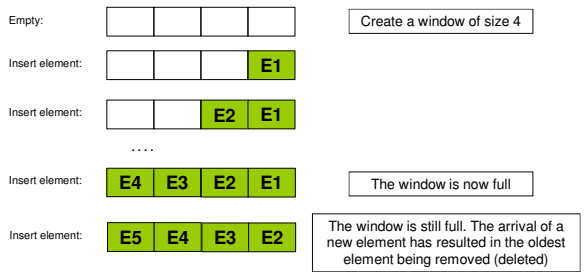
Write a **generic** Java class that implements a sliding window for elements of different types using generic methods.

A sliding window is a type of queue that is often used for buffering the last n received packets in a data stream (for example over a TCP/IP connection) without having to move a packet once it has been inserted into the buffer.

Your solution should be based on an implementation of a simple ring buffer supporting the methods outlined below. As always, you also need to write an application for testing and demonstrating your solution.

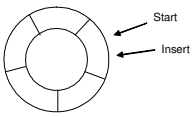


## Realising a Sliding Window

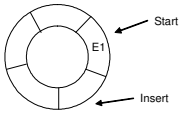





## Using a Ring Buffer




Create a buffer for 4 (!) elements




Insert an element

Once the buffer is full, start overriding oldest element.

No elements are copied!!



## Hints

- Use an array as ring buffer
- Use two “pointers” to keep track of the elements
  - Start element, current element
- The size of your array must be 1 larger than the size of your buffer (window)
  - Otherwise you can’t tell whether the buffer is full or empty
- You may want to use the stack example from the lecture notes as a starting point

## Methods

- Your sliding window must support methods for:
  - Creating a sliding window of a certain size
  - Inserting elements
  - Resetting the window
  - Displaying all elements in the window

Hint: Realise (and test) the four methods above first (they are straightforward) and then add the method below

  - Checking whether a specific element is already in the window



## Test Application

- Your test application must demonstrate that your solution works for Integer, String, and application-specific object types
  - Create a class for testing object types
  - Hint: implement your generic class for Integer and String first, then expand to handle application-specific objects
- Show that your solution works for empty, half-full and full windows of at least size 5



## Handing Up

- Printouts are to be handed up to SCSS reception staff *by 4*
- Printouts must include source code *and* terminal output of the test application
  - Source code must be readable – get page format and line breaks (reasonably) right
  - Terminal output can be captured as plain text (e.g., copying the text output into a text editor) or as a screen shot (e.g., copying a screen shot into a text editor)
  - `Alt - print screen` copies the active window to the Office Clipboard
  - `Print screen` copies the whole screen to the Office Clipboard
- All printouts must include *exercise title, student name and student number*
- Printouts must be s*tapled*. No envelopes, paperclips, folders, .. please!