**Linear Data Structures**

**ArrayList**

* Supported operations and complexity:
  + **size()**, **isEmpty()**, **get()**, **set()** – **O(1)**
  + **add()** – the operation runs in **amortized constant** time
  + adding **n** elements requires **O(n)** time
  + all of the other operations like: **add(int index, E element)**, **contains()**, **indexOf(), remove(int index) etc**., run in **linear** **time** **O(n)** (roughly speaking)

**Stack**

* Supported operations and complexity:
  + **size**(), **isEmplty**(), **push**(), **pop**(), **peek**()– **O(1)**
  + all of the other operations run in linear time (roughly speaking):
    - **forEach**()
    - **contains**()
    - etc…

**Queue**

* Supported operations and complexity:
  + **size**(), **isEmplty**(), **poll**(), **peek**()– **O(1)**
  + **offer**():
    - if we keep the reference to the that node – **O(1)**
    - If we have to chase pointers to that node – **O(n)**
  + all of the other operations run in linear time (roughly speaking):
    - **forEach**(), **contains**(),etc…

**Singly Linked List**

* Supported operations and complexity:
  + **addFirst**(), **removeFirst**(), **getFirst**(), **size**()– **O(1)**
  + How about operations on the **last element**?
    - **addLast**(), **removeLast**(), **getLast**() – again depends if we keep the reference to the last node or no can be constant – **O(1)** or linear – **O(n)**
  + operations that **index** into the list will run in **linear** **time** **O(n)** (roughly speaking)