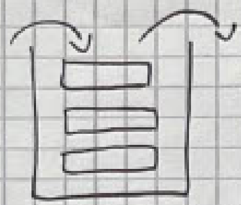
Dynamische Datenstrukturen

Stack



nur oben hinzu
oder
wegnehmen

LIFO (Last In-First Out)

Push (oben drauf)

```
public void push (String s) {
    Node temp = new Node (s);
    temp.setNextNode (top);
    top = temp;
    size ++;
}
```

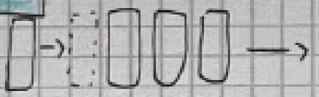
Pop (oben weg)

```
public String pop () {
    String s = top.getContent ();
    if (top != null) {
        top = top.getNextNode();
        size --;
        return s;
    } else {
        return null;
    }
}
```

Clear (löscht alles)

```
public void clear () {
    top = null;
    size = 0;
}
```

Queues



hinten ran (tail),
vorne weg (head)

FIFO (First In - First Out)

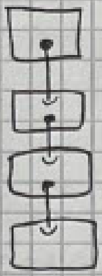
Offer (hinten dran)

```
public void offer (String s) {
  Node temp = new Node (s);
  if (size == 0) {
    head = temp;
    tail = temp;
  } else {
    tail.setNextNode (temp);
    tail = temp;
  }
  size ++;
```

Poll (vorne weg)

```
public String poll () {
  if (head != null) {
    String s = head.getContent();
    head = head.getNextNode ();
    if (size == 1) tail = null;
    size --;
    return s;
  } else {
    return null;
  }
}
```

List



einfach verkettet

Methoden: **add** (fügt an beliebiger Stelle String ein)

get (gibt String an beliebiger Stelle zurück)

remove (wie get, nur löscht es gleich Element)

PerformanceTest

LinkedList : - doppelt verkettet
- dynamisch

ArrayList : - auf jedes Element zugreifen
- festgelegte Grösse

- ArrayList schneller bei "Zugriff auf zufällig gewähltes Element"
- LinkedList schneller bei "Einfügen am Anfang"