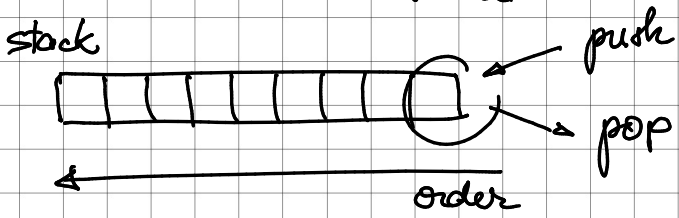
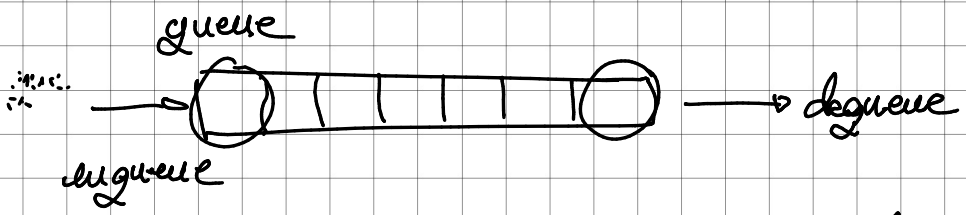
I) Stacks and Queues

- fundamental data types
  - value : collection of objects
  - operations: insert, remove, iterate + test if empty





STACK = examine item most recently added

LIFO

QUEUE = examine item least recently added

FIFO

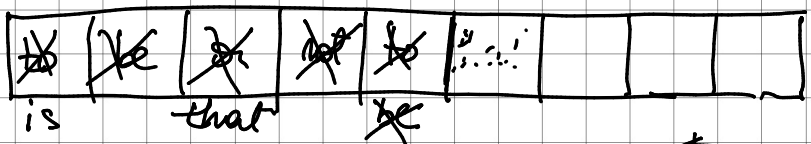
(1) Stacks

class StackOfStrings

- new
- void push (String item)
- String pop ()
- boolean isEmpty ()
- int size ()

---

eg.

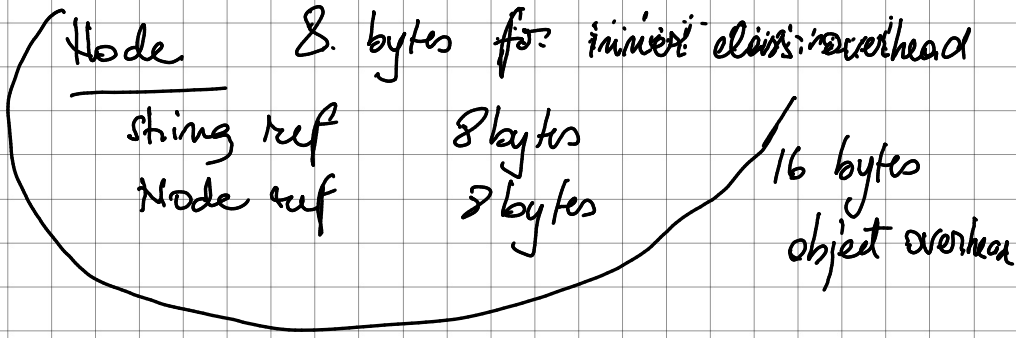


input to be or not to - be - - - that

- - - is

output to be not or that be to

* Implementation A - using a linked list
  - every operation takes constant time in the worst case (PROPOSITION)
  - a stack with N items uses ~ 40N bytes



⇒ 40 bytes per stack node

* the strings are owned by the client

Implementation B : Array

- s[]

current length

- push : add item at s[s.length]
- pop : remove item from s[s.length - 1]
  decrement length
- you need to declare fixed capacity

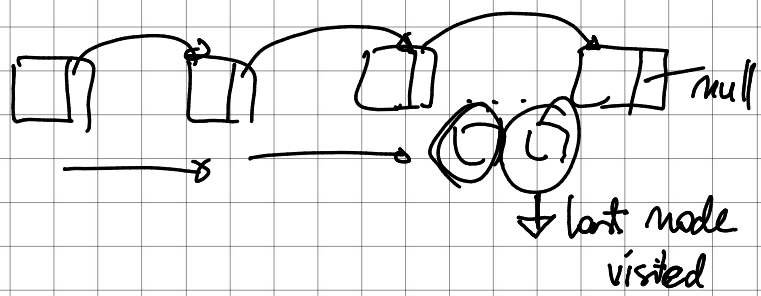
Stack considerations

underflow : throw exception if pop from an empty stack

overflow : resize array

- null items?
- loitering: holding references to an object when no longer needed

eg.



II) Resizing Arrays

1st try: push – incr. size by 1
pop – decrease size by 1

- need to copy all existing elements
- $1+2+..+N \longrightarrow N^2/2$

2nd try: repeated doubling

cost of inserting N items $\sim 2N$

N + $(2+4+8+..+N)$ $\sim 3N$

N: 1 array access per push

$(2+4+8+..+N)$: k array accesses to double to size k

- how to shrink the array ?

1st try : halve size of array when array is one half full ?

no because of push - pop - push - pop

2nd try : one quarter full → halve it

* Memory usage

8 bytes (ref. to array)
24 bytes (array overhead)
8 bytes × array size
4 bytes int
4 bytes padding

~ 8N full
~ 32N when one quarter full

Queues

class Queue Of Strings

void enqueue (String item)
String dequeue()
boolean isEmpty()

Queue w. Linked List

- 2 pointers < first / last

Exercise : Implement Queue with resizing array

- tail & head (update % capacity)

Generics

- quick hack is to use casting, everything in Java is an object
- Linke Lists Stacks can easily use Generics
- Array Stacks not : Java does not allow generic array creation
- create an array of objects and cast it to Item. (Item[])

Iterators

- make data type implement Iterable interface if the client wants to iterate over the array

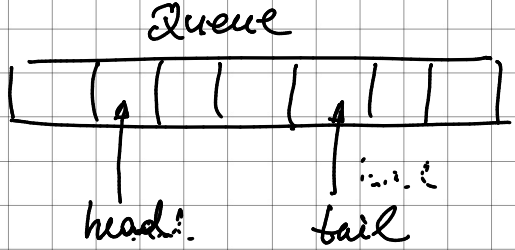
Iterable — has method iterator()
returns an iterator

Iterator — hasNext() remove()
next()

* When order doesn't matter

class Bag

- void add(Item x)
- int size()
- Iterable<Item> iterator()



array iterator is LIFO

Stack and Queues Applications

Stack

- parsing in a compiler
- browser history

Dijkstra's two stack algorithm

- value → push to the value stack
- operator → push to the operator stack
- left parenthesis → ignore
- right parenthesis → pop operator and two values; push the result to the operand stack