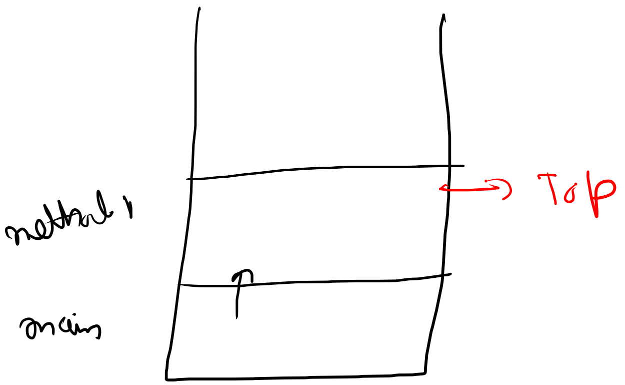
Stacks → internally → array

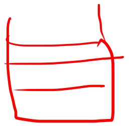
Linear data Struct



Syntax

Stack < Integer > st = new Stack<>();

(i) Fixed Stack



(ii) dynamic

Properties

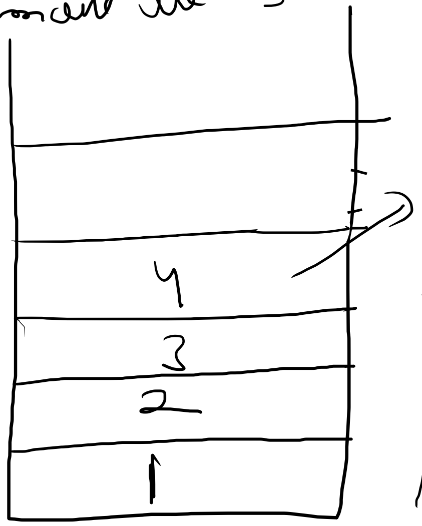
1) LIFO → Last in first out behav

Push → add element → always at top

Pop → remove ele → always at top

int

stack



top of the stack (peek stack)

Bottom to top

Push → 1

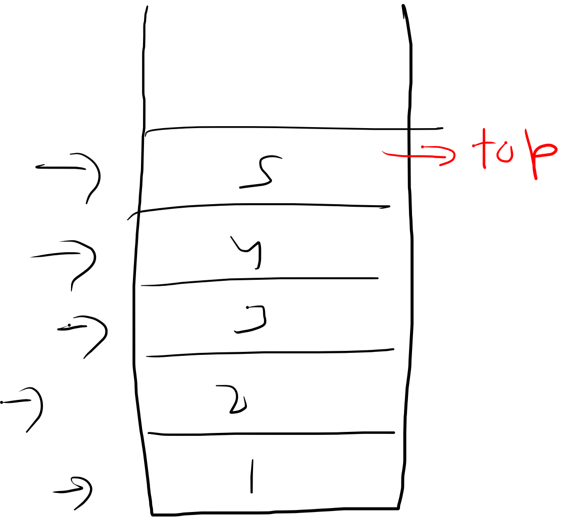
Push → 2

push 3

push 4

push 5

Pop() → 5



Basics operation ~~on~~ Stacks

single element

1). Push() → add → O(1)

2). Pop() → remove → O(1) → top

3). peek() → top element (Peek) → O(1)

4). Is Empty() → stack is empty or not → O(1) → T/F

5). size() → size of stack → O(1)

Stack Class {
int size

}

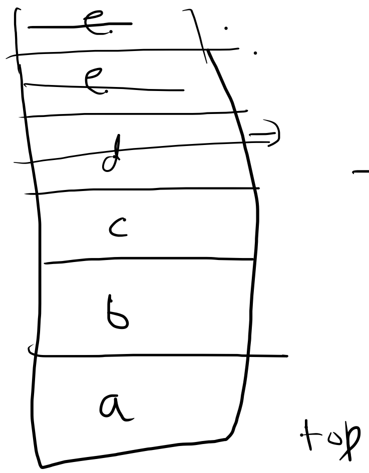
+ ~~2~~ ~~3~~ ~~4~~ 5

Reverse string using stack (Day 34)

Stack

i ↓	0	1	2	3	4	5
str	a	b	c	d	e	e
o/p	e	e	d	c	b	a

String



⇒ Pop ele & update new String

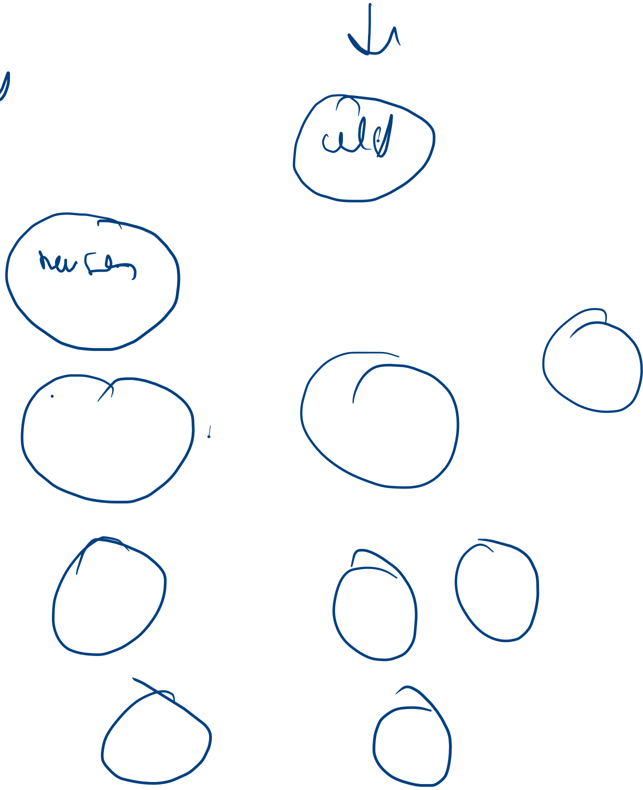
e e d c b a

Strings = immutable in nature

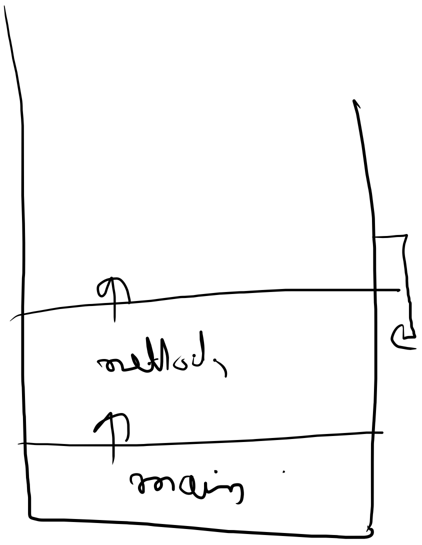
str = str + 'a' → 10 lines

update direct string

↳ memory inefficient



Stack mem



Fn call

Duplicate Brackets (Day 34)

Exp1 $(a+b) + \overbrace{((c+d))}^{\rightarrow \text{true}} \rightarrow T$

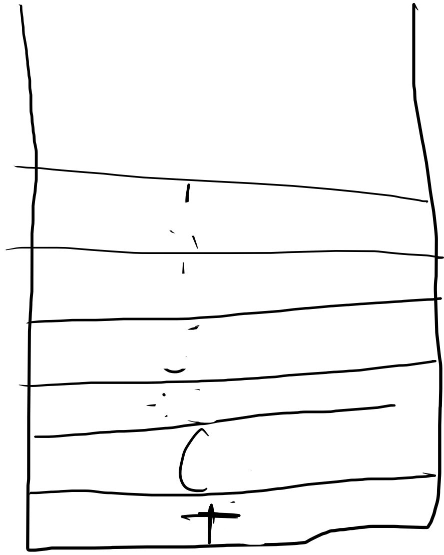
Exp2 $((a+b) + (c+d)) \rightarrow F$

for any operator (+ , - , / , *)

→ you have one pair of bracket → duplicate ✗

0 1 2 3 4 5 6 7 8 9 10 11 12

(a + b) + ((c + d))



(, char, op → push

) → is → top → (→ duplicate

→ pop till ~~top~~ ≠ (