Stacks

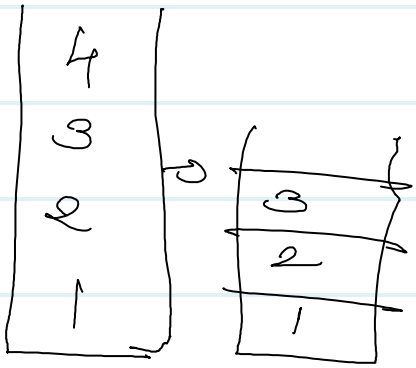
Arrays → [_ , _ , _ , _ , _] Fixed Size

Get & Set → O(1)

Array list → Dynamic Array

Stack → LIFO (Last in First out)

Stacks of Book → Example



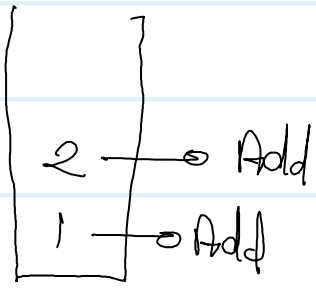
Add, Get, Remove

(4) (4)
(3) (3)

Add Get Remove

(1) Add 1

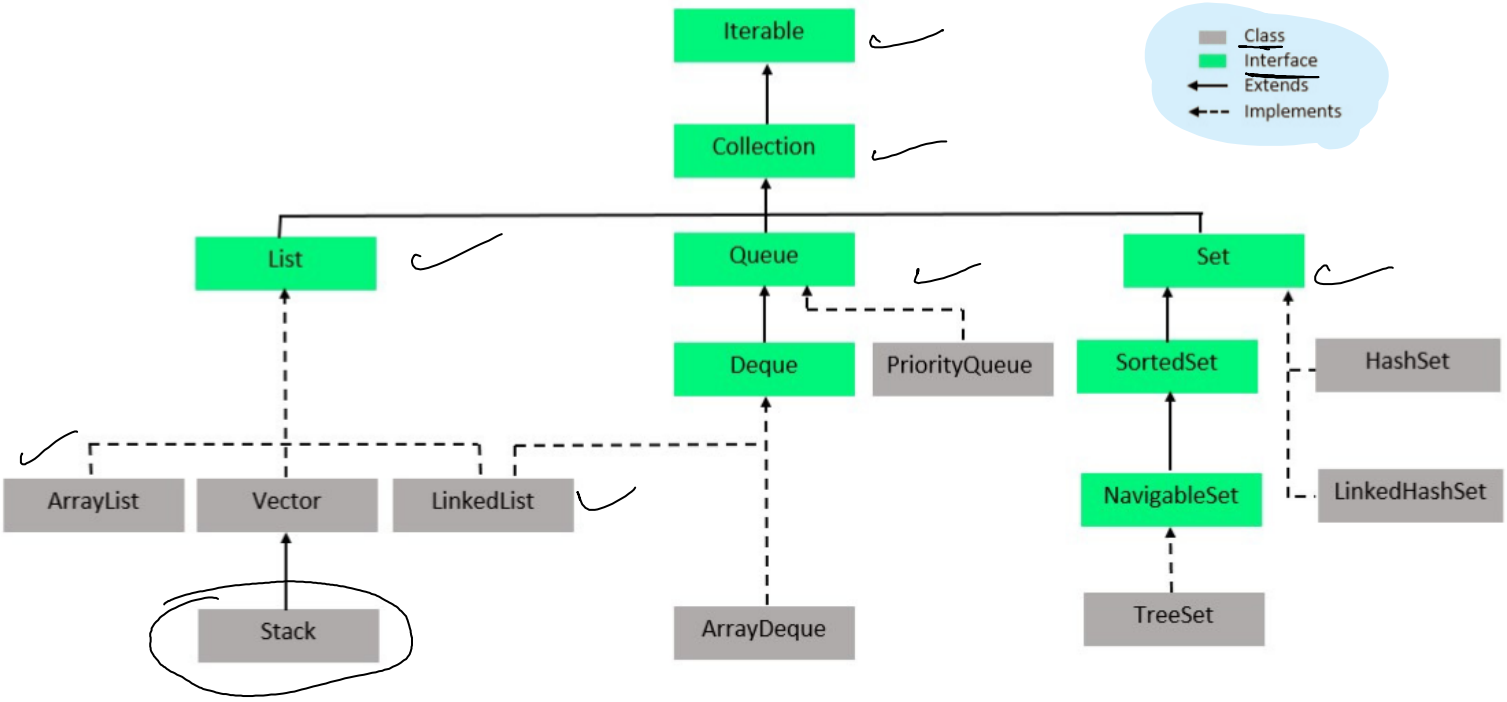
(2) Add 2 → Get (2)



Add → Push
Get → Peek
Remove → Pop

→ Main Methods of Stack
Push, Peek, Pop, Size

Note → Java Collection framework provides the impl.



public class Stack extends Vector

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

st.push(1);

st.push(2);

st.peek();

st.remove();

Implementation →

```
Stack<Integer> st = new Stack<>();
st.push( item: 1);
st.push( item: 2);
st.push( item: 3);
st.push( item: 4);
st.push( item: 5);
st.push( item: 6);
st.push( item: 7);

Integer pop = st.pop();
System.out.println(pop);
System.out.println(st.peek());
```

Q1) Check for duplicate brackets

a) (1+2)+((4+5)) → True

o o c o o c c

b) ((1+2)+(4+5)) → False

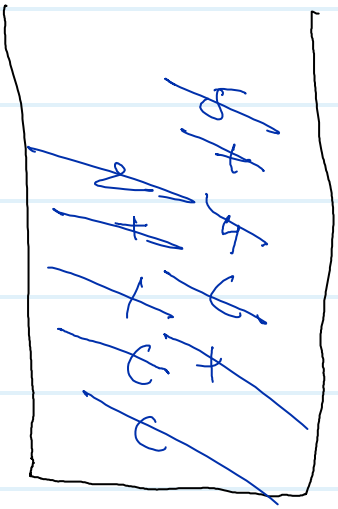
data

o o c o c c

Example → ((1+2)+(4+5))

① Push for open brackets

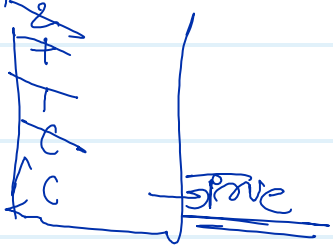
② As soon as we got the closing bracket, then pop till we not came across any opening bracket & then pop that opening bracket too.



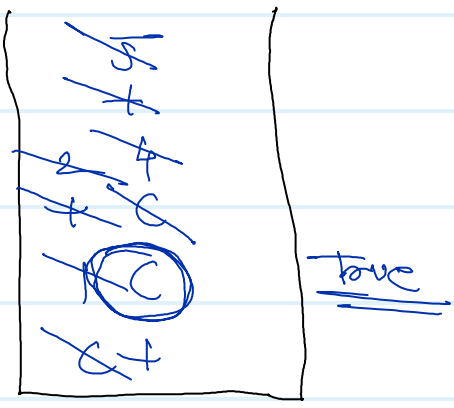
No Duplicates

Example → (1+2)+((4+5))

((1+2))+(4+5)



true



true

```java
String data = "((1+2)+(4+5))"; //True
Stack<Character> st = new Stack<>();
for (int i=0 ; i<data.length() ; i++) {
    char ch = data.charAt(i);
    if (ch == ')') {
        if (st.peek()=='(') {
            System.out.println(true); //Duplicates Found
            return;
        } else {
            while(st.peek()!='(') {
                st.pop();
            }
            st.pop();
        }
    } else {
        st.push(ch);
    }
```

→ Stack

i = 0, 1, 2, 3, 4, 5, 6, 7 ...

→ '(', '(', '1', '+', '2', ')', '+' ...

→ Condition Closing bracket

→ found the opening bracket as soon as we enter.

→ Returning.

→ Not found the concurrent opening Bracket

→ Pop till not getting opening bracket

→ Pop that opening bracket too.

→ Opening bracket

→ Push that to the Stack.