→ Infix Conversion → Postfix , Prefix.

→ Postfix Conversion → Evaluate , Infix , Prefix → for(int i=

+24 =6 → Prefix Conversion → Evaluate , Infix , Postfix.

```
{ for (int i= str.length() -1; i>=0; i--)
  {

}
```

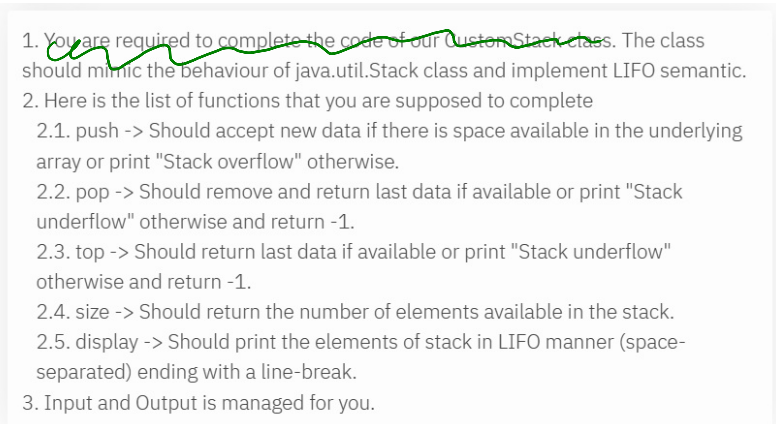
Imp:

1. Normal stack

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

2. Dynamic stack

3. Queue D.S.



1. You are required to complete the code of our CustomStack class. The class should mimic the behaviour of java.util.Stack class and implement LIFO semantic.
2. Here is the list of functions that you are supposed to complete
   2.1. push -> Should accept new data if there is space available in the underlying array or print "Stack overflow" otherwise.
   2.2. pop -> Should remove and return last data if available or print "Stack underflow" otherwise and return -1.
   2.3. top -> Should return last data if available or print "Stack underflow" otherwise and return -1.
   2.4. size -> Should return the number of elements available in the stack.
   2.5. display -> Should print the elements of stack in LIFO manner (space-separated) ending with a line-break.
3. Input and Output is managed for you.

Stack
↳ push() → overflow
↳ pop() → underflow
↳ size()
↳ peek()

↳ display() →

9:12 - 9:18

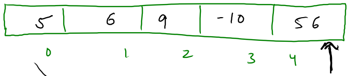
~~Just to the dry run~~

5 mins

Cap = 5

tos = ~~-1~~ ~~0~~ ~~1~~ ~~2~~ ~~3~~ 4

Stack =



→ tos++;

→ stack [tos] = 5

if(tos == Stack. length-1) → Sysa(" Stack is overflow")

$\boxed{tos}$ == Stack.length

(4 == 4) ?

Push
Push
pop()
pop()
pop()

pop()
pop()
pop()
pop()
pop()

① Push

↳ x = 5 → 0

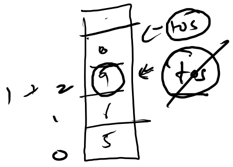
↳ x = 6 ← 1

↳ x = 9 ← 2

↳ x = -10 ← 3

↳ x = 56 ← 4

$\boxed{↳ x = -70}$



pop()

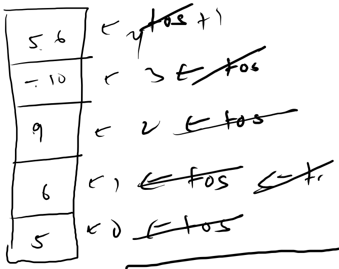
Push

② Display

for(int i = tos; i >= 0; i--)

print(stack[i]);

~~for(int i = Stack. length; i >= 0; i--)~~



$\boxed{← tos = -1}$

x = -7

③ Size()

return tos+1

⑤ Operation

④ pop() } → Class

↳ tos--

↳

if (tos == -1)
Sysa(" Stack underflow");
return -1

⑤ peek()
if (tos == -1)
{
Sysa(
}
else
Sysa(stack(tos))

Normal Stack →

1. Push
2. Display
3. Size
4. Pop
5. Peek

→ Stack / Queue Recursion T.C. + Knight Tours → Sunday.

→ Saturday no classes.

↳ backlog.

15 mins   10:00   ①   Dynamic

```java
public static class Stack {
    int arr [];
    int cap;
    int tos;

    Stack(int cap){
        this.cap = cap;
        arr = new int[cap];
        tos = -1;
    }

    int size(){
        return tos+1;
    }

    void display(){
        for(int i =tos;i>=0;i--){
            System.out.print(arr[i] + " ");
        }
    }

    int peek(){
        if(tos == -1){
            System.out.println("Stack underflow");
            return -1;
        }

        return arr[tos];
    }
```

②   tos--

```java
int pop(){
    if(tos == -1){
        System.out.println("Stack underflow");
        return -1;
    }

    int val = arr[tos];
    tos--;
    return val;
}

void push(int val){
    if(tos == arr.length-1){
        System.out.println("Stack Overflow");
        return;
    }

    tos++;
    arr[tos] = val;
}
```

③

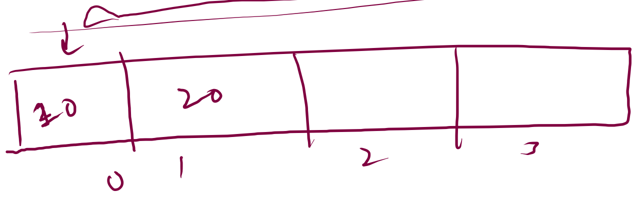
```java
public static void main(String[] args) {
    /* Enter your code here. Read input from STDIN. Print output
    Scanner scn = new Scanner(System.in);
    int t = 15;
    Stack st = new Stack(5);
    for(int i=0;i<t;i++){
        String operation = scn.next();
        if (operation.equals("Push")){
            int x = scn.nextInt();
            st.push(x);
        } else if (operation.equals("Pop")){
            System.out.println(st.pop());
        } else if (operation.equals("Size")){
            System.out.println(st.size());
        } else if (operation.equals("Display")){
            st.display();
        } else if (operation.equals("Peek")){
            System.out.println(st.peek());
        } else {
            System.out.println("Performing wrong opertaion");
        }
    }
}
```

tos = ~~0~~ 1

arr[0] = 10

arr[1] = 20

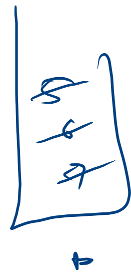
push



→ Stack underflow → pop() pop() pop() pop()

5 → 10

→ Stack overflow → size ⇒ 2 * size



Dynamic stack

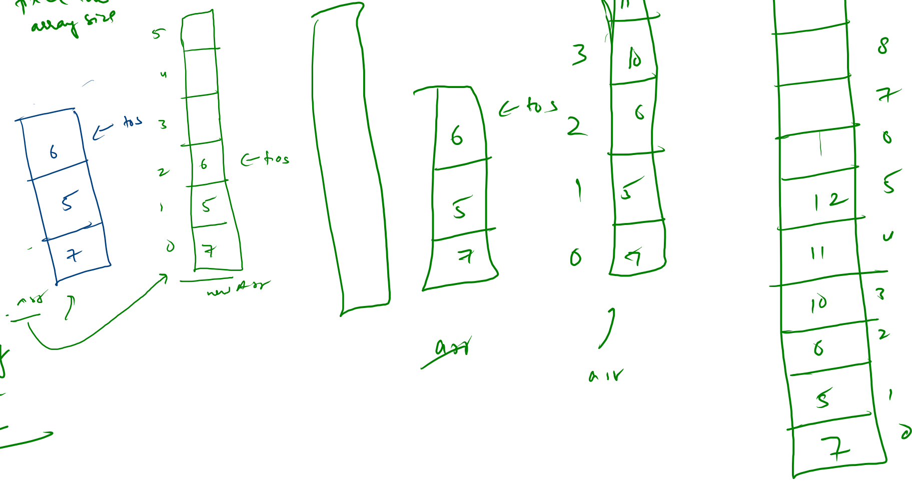
```
↳ if( tos == arr.length-1)
{
    int newArr = new int[ 2* arr.length];
    for(int i=0; i<=tos ; i++)
    {
        newArr[i] = Arr[i];
    }
    arr = newArr.
}
```

cap = 3

$\boxed{2 * \text{cap}}$ ✗ → fixed the array size

2* cap = 2×3=6

dynamically change the array size



Dynamic Stack :-

Sunday

↳ $\boxed{\text{Queue}}$

```
void push(int val){
    if(tos == arr.length-1){
        int newArr[] = new int[2*arr.length];
        for(int i=0;i<=tos;i++){
            newArr[i] = arr[i];
        }

        arr = newArr;
    }

    tos++;
    arr[tos] = val;

}
```

$\Rightarrow$ $\boxed{\text{Tuesday}}$ → OK