L.L. Ques. after implementation stuffs.

Ajanta
Page No.
Date 26/06/25

Stacks and Queues

Implementation → real quick

→ Interface

Stacks → FILO ; Queue → FIFO

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

class — push → pop

Queue<Integer> queue = new LinkedList<>(); → rem OOPs

add, remove, offer, peek, poll

(since interfaces are not instantiated directly.)

Deque<Integer> deque = new ArrayDeque<>();

addFirst, addLast → removeFirst, removeLast (O(1))

Custom Stack Implementation :-

fixed size

Basic declaration :-

unlike two diff. class in LL → LL, Node here we just customStack.

```
public class CustomStack {
    int[] data;        → data of stacks are stored in array
    private static final int DEFAULT_SIZE = 10;
```

declaration → const^r (2) → ptr initialized

static → no need to initialise in other fn. (private)

final → fixed value; has convention variable name in capital

non-parameterised constr for if size not mentioned then we default size of 10

```
public CustomStack() {
    this(DEFAULT_SIZE);
}
```

const^r if size is mentioned → if will used for data length

```
public CustomStack(int size) {
    this.data = new int[size];
}

int ptr = -1;
```

Write less to avoid mess

Don't close the class bracket as it is public class → contains all based on file

Try → Can just write Exception except stack exception

Ajanta
Page No.
Date

✓ add (push) :→ return type = boolean , arg. int item

① if (isFull()) { → make this fn.
print ("Stack is full");
return false;

② ptr++ ; → item aaya hai to uss use point karo to badhao!

③ jo item aaya hai us data m daalo
data [ptr] = item;

④ return true ; → just showing yes you can do it add ✓ it's done.

✓ isFull fn. → return type = boolean , no arg. → ∵ made just for check.
↳ ① sab ptr last m hoga tab.
directly → return ptr == data.length - 1 ; " ptr started from -1

✓ isEmpty → ditto same bs ptr at ~~0~~ -1 ;
since yaha se initialize hua tha.
first element index/ptr is 0.

✓ pop → return type → int , arg → nothing , throws ~~Stack~~ Exception {
↳ just to let you know which one popped (not imp.)
use without this (void) too!

① if (isEmpty()) {
throw new ~~Stack~~ Exception ("Cannot pop from empty stack");

② ~~ptr~~ int removedItem = data [ptr]

③ ptr--; → main thing

④ return removed item;

on directly return data[ptr--];
show then decrement

✓ peek → check top element → return type int , arg → nothing , throws StackException

① if (isEmpty()) {
throw new StackException ("Can't peek from empty");

② return data [ptr];

✓ In psvm throws ~~Stack~~ Exception { focus here only.

Ajanta
Page No.
Date

Stack

Dynamic stack not imp.

revise & check reasons for all conditions

all conditions!

Custom Queue implementation :-

*Basic Declaration same as of Custom stack but here fir name is end (since FIFO) & it's initialized with 0.

i.e. int end = 0;

same as stack

add :- return type → boolean, arg. → int item

① if (isFull()) { → make fn. → Full when end == data.length (since started from 0.)
return false;
}

② data[end] = item;

③ end++;

return true

same steps.

Similarly Empty when end == 0.

viewing purpose

remove → return type → int, arg → nothing throws Exception {

① if Empty throw new Exception("queue is empty");

② directly return data[end--] ✗ → wrong, Diff. from stack (∵ FILO) (FIFO only)

② removed will be first → int removed = data[0];

③ ek ek karke ab data values fir location piche lao :-

for(int i=1; i<end; i++) { → not started with i=0 ∵ data[0-1] = data[-1] ho jata

data[i-1] = data[i];
}

ek kam hua hai na!

④ end--;

⑤ return removed;

front → return data[0]; empty → exception

Display :- → return type void; arg nothing

for(i=0 to end) {

print(data[i] + "←")

}

print("End");

→ since jo last me aaya wo sabse piche hoga & last me niklega

0←0←0←0←0

first last

same for stack just opposite

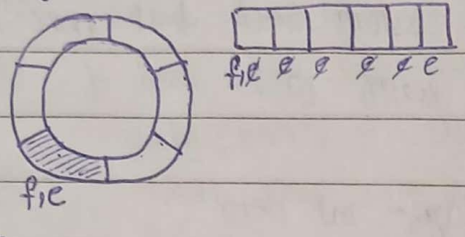
↓ arrow diff.

+1
5, 1, 2, 3, 0, 0, 1

Ajanta
Page No.
Date 26/06/25

Circular Queue implementation:-

- first and end ~~ptr~~ will be declared and all the items in circular queue will be b/w first and end (they'll be considered only).



✓ ★★★ concept → • at first, "first" and "end" will be at 0th index.
• item inserted → end + 1
• item removed → front + 1 (∵ items b/w front + end only considered)

except end-ptr;

anything is increasing ∴ add safety

• Now if size of the array is reached for the end the to repeat its path we use remainder concept → easy → [end = end % size] → if end is at data.length

↓
V. popular concept + easy

for array size of 5 if end reached till 6 i.e. (>5) ↳ explanation
then modulo it by size to revert from 1 → 6%5 = 1 ; 7%5 = 2 ....

• Since it is a queue, ∴ we'll use a end_ptr also to track.

✓ Basic declaration → same as of queue + initialize front + end as 0 with end-ptr = 0 (this is already in queue)

✓ isFull & isEmpty → " " " " for end-ptr

✓ insert → return type = boolean , arg = item

① if(isFull) return false

② end aage badhega on insertion (read concept) to usme item daalo.
data[end] = item;
end++;

③ end = end % data.length; → !! For safety purposes if end > data.length then only used otherwise no use i.e. 3%5 = 3 only

④ parampara → end-ptr++;

⑤ return true;

Insert
• full → false
• data jaga & end badhao
• data safety
• ptr badhao

*I've mistakenly used front in place of first kindly understand accordingly.

Ajanta
Page No.
Date

Please just stick to concept.

just showing purpose

* remove → int = return type, args = nothing, throws exception
① if empty throw exception
② get value for returning purposes → int removed = data[front];
front++; → concept
∴ item aage se nikalega (FIFO)
③ * safety for front too → front %= data.length;
(Circle, wo bhi aage badh rha na)
④ end-ptr --;
⑤ return removed;

* offer → return type int, arg → nothing throws exception.
① if empty throw exception.
② ~~dat~~ return data[front]; → not data[0]; ★
done mistake while coding

* Display → void, no args
① if empty → print empty
② DO while loop → int i = front;
do {
print(data[i] + " -> ");
i++; → increasing ∴ add safety
i = i % data.length;
} while(i != end);
print("END");

★★★★★
Cannot write as
while(i <= end)
↓
because front can be greater than or less than or equal to end.