Whiteboard 12

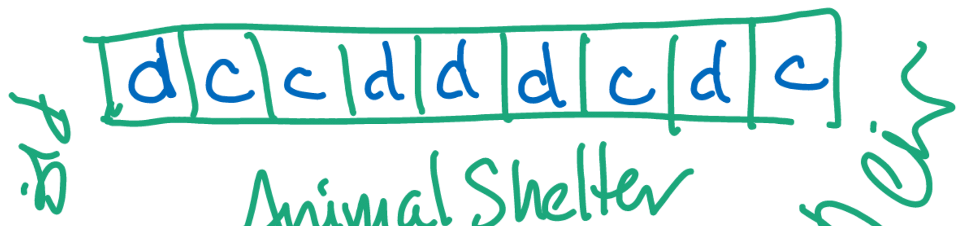
Whiteboard 12

Create an Animal Shelter class that uses a first in / first out approach using enque (add in Java) and deque (remove in Java) methods.

Also for "pref" return longest waiting preferred animal (cat or dog)

Stretch if no preffered animal type return longest waiting animal

Visual

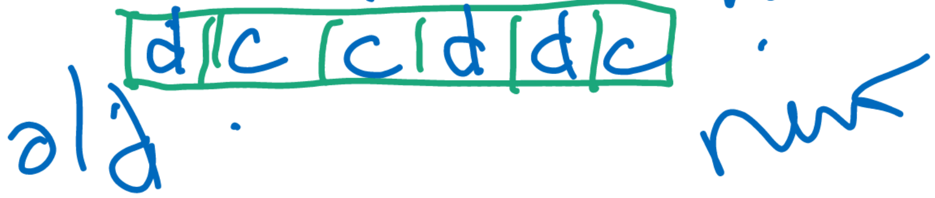


Animal Shelter
Queue Class

① Scenario - General first in/ first out

FIFO* imp

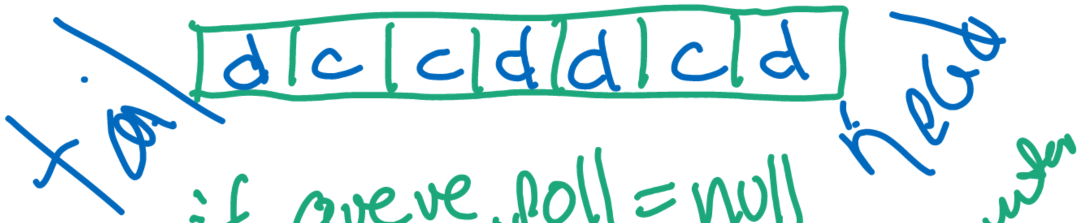
Queue



Queue.poll to get animal

* First in First Out implementation

② Scenario ~~2~~ ③
only a cat w/ first in/out queue



```
if queue.poll = null
    "no animals"
counter = Q size
while(queue.poll != pref) && counter > 0 {
    Keep pull off tail &
    putting in head
    counter--;
}
```

~~then, when c found~~
~~empty StackB~~
~~enqueue to StackA~~
~~and push to Queue~~

~~(3) Scenario 3 - dog desired~~
~~same as Scenario 2~~
~~only w/ d=dog~~
~~not c=cat~~
see #2

(4) Scenario 4 longest
waiting animal
Queue.poll ⟸ if empty
class!

Algorithm

1: for tests build custom toString method for Queue

2: instead of using stacks for scenario 2 & 3 use while loop w/ poll & size & counter

3: if #2 doesnt work, instantiate 2 empty stacks to help w/ scenarios 2 & 3

4: Actually write
simple scenario
1 & test

5. Write scenario 2 $\updownarrow$
& test

6. " Scenario 3
& test

7. " Scenario 4
& test

PseudoCode - Deque version

toString method
    Queue.toArray.toString

(Scenario One - Get FIFO animal)

```
public class Queue () {
animal = ~~Deque~~ Q. poll~~First~~ }
    return animal }}
```

(Scenario Two Get pref FIFO ~~[illegible]~~)

~~Deque extends Queue~~

```
while loop while
    pollFirst != ~~[illegible]~~ pref
```

and while not pref
{ contains())
queue.poll

return pref
or "no pref avail"
(to preserve FIFO
order)

~~(Scenario 3 - get 1st dog)~~
~~Repeat code above~~
~~but for dog~~
value

(Scenario for FIFO)
~~FILO = first In last out~~
queue.pol ;

Code

```
public class Queue<string>
animalShelter {
private int size = Queue.size();

public String toString() {
return Queue.toArray
toString;

}
```

```
public String getAnimal(){
      if(queue.poll ==
            null){
         return" no animals
            available";
         }
      int counter= size-1;
   while(counter!=size-1){
      String animal
         = Queue.poll;
      String shelter
         = queue.add
      counter--;
   }return animal;
}
```

```
public String getPref(String pref)
{
   if(queue.poll==null){
      return "no "+ pref +
        " available.";
   }
   while (queue.poll!=pref
     && count>0) {
        pref = que.poll;
        queue.offer(pref);
        count--;
   } return pref;
}
```

```
public String getlongestStay
() {
    if(queue.poll==null){
        retun "no animals
            available ";
    }
    return queue.poll;
}

// end program
```

Big O

BigO(n) time
O(1) space

Tests

for e. method

- emptyshelter
- get Cat
- get Dog
- getcat & nocats
- getdog & ndogs
- get newestanimal
- get longest Stay animal