

Data Structures

CSSE 221

Fundamentals of Software Development Honors

Rose-Hulman Institute of Technology

Announcements

- Moench F217/F225 Lab:
 - Everyone has 24/7 keycard access
 - Assistants from 7-9 pm, Sun - Thurs
- Do ANGEL > Lessons > Other > VectorGraphics > Performance Evaluations tonight

**The gem cannot be polished without friction,
nor man perfected without trials.**

-- Chinese Proverb

This week: Markov

- Monday:
 - Stacks and Queues
 - Sets and Maps
- Tuesday:
 - Introduction to *Markov*, a cool statistical text program with lots of data structures
 - File I/O
- Thursday:
 - Recursion

Stacks

- A last-in, first-out (LIFO) data structure
- Real-world stacks
 - Plate dispensers in the cafeteria
 - Pancakes!
- Some uses:
 - Tracking paths through a maze
 - Providing “unlimited undo” in an application

Operations Provided	Efficiency
Push item	$O(1)$
Pop item	$O(1)$

Implemented by Stack, LinkedList, and ArrayDeque in Java. An ArrayList also works

Queues

- A first-in, first-out (FIFO) data structure
- Real-world queues
 - Waiting line at the BMV
 - Character on Star Trek TNG
- Some uses:
 - Scheduling access to shared resource (e.g., printer)

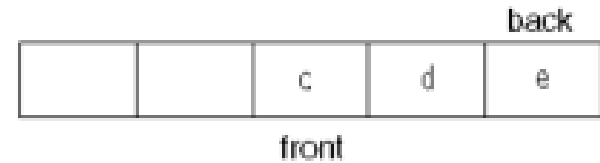
Operations Provided	Efficiency
Enqueue item	$O(1)$
Dequeue item	$O(1)$

Implemented by
LinkedList and
ArrayDeque in Java

Array implementation of a queue

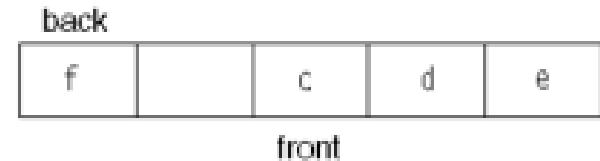
After 3 enqueues

size = 3



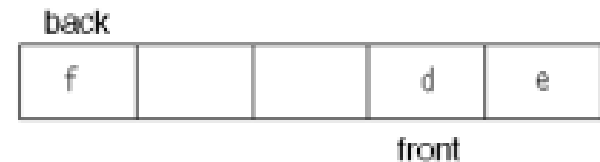
enqueue(f)

size = 4



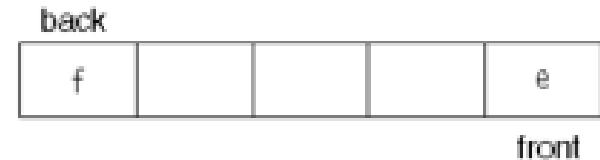
dequeue()

size = 3



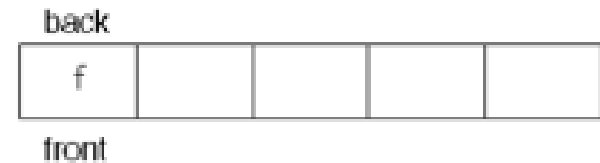
dequeue()

size = 2



dequeue()

size = 1



```
private int increment( int x )  
{  
    if( ++x == theArray.length )  
        x = 0;  
    return x;  
}
```

What if we run out of
room?

Demo

A tale of two interfaces: Set<E>...

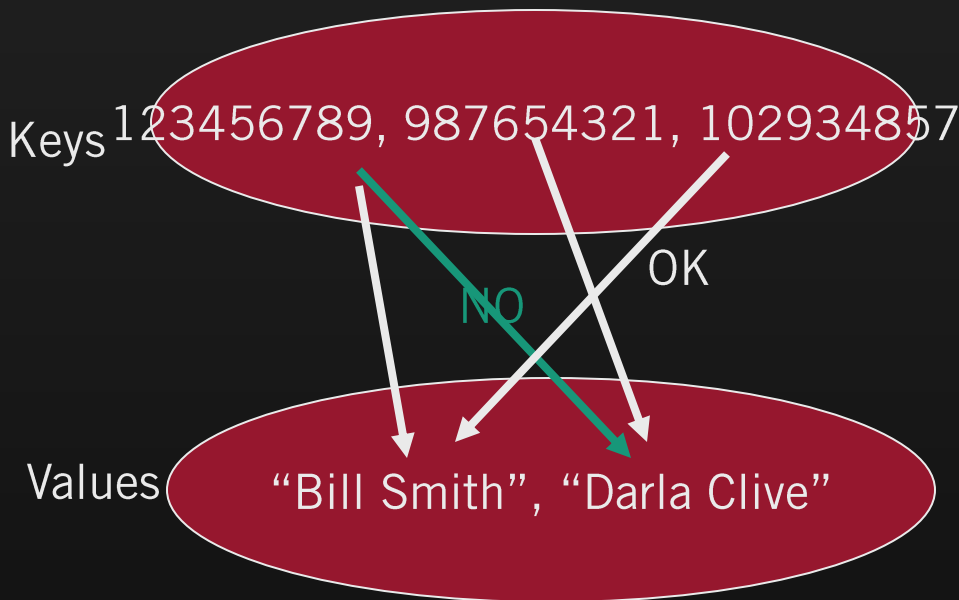
- A collection with **no duplicates**
- If **obj1** and **obj2** are both in the set, then **obj1.equals(obj2)** returns false.
- Can **.add()** and **.remove()**
- Subinterface: **SortedSet**
- Even has intersection (**retainsAll()**) and union (**addAll()**) methods

“Bob”, “Flo”, “Gary”, “Lisa”, “Marie”

...and Map<K,V>

```
HashMap<Integer, String> map =  
new HashMap<Integer, String>();
```

```
map.put(123456789, "Bill Smith");  
map.put(987654321, "Darla Clive");
```



- An object that maps keys to values. Duplicates?
 - A map cannot contain duplicate keys; each key can map to at most one value.
`V get(Object key)`
 - Multiple keys **can** have the same value
- Other operations:
 - `put(K key, V value)`
 - `containsKey(Object key)`
 - `V remove(Object key)`

TreeSets and TreeMaps

- ...are java.util implementations of SortedSet and Map.
- **Sorted** elements.
- In a tree, average time is $O(\log n)$,
 - and with complex algorithms, worst case can also be $O(\log n)$
- Also support taking ordered subsets from head, tail, or interior of set or map
- Implement in CSSE230

HashSets and HashMaps

- ...are java.util implementations of Set (not SortedSet) and Map.
- Average time for lookup, insertion, or deletion is $O(1)$.
 - but worst case is $O(N)$.
 - A quick view of how it works:
 - hashCode function maps object to an integer, which is used to find an index into an array
 - Resolve collisions
 - Fast search, but unordered
 - Need to use a class for your keys that implements .equals() and .hashCode() [or implement them]
- More details in CSSE230

Demo
