

# Queue

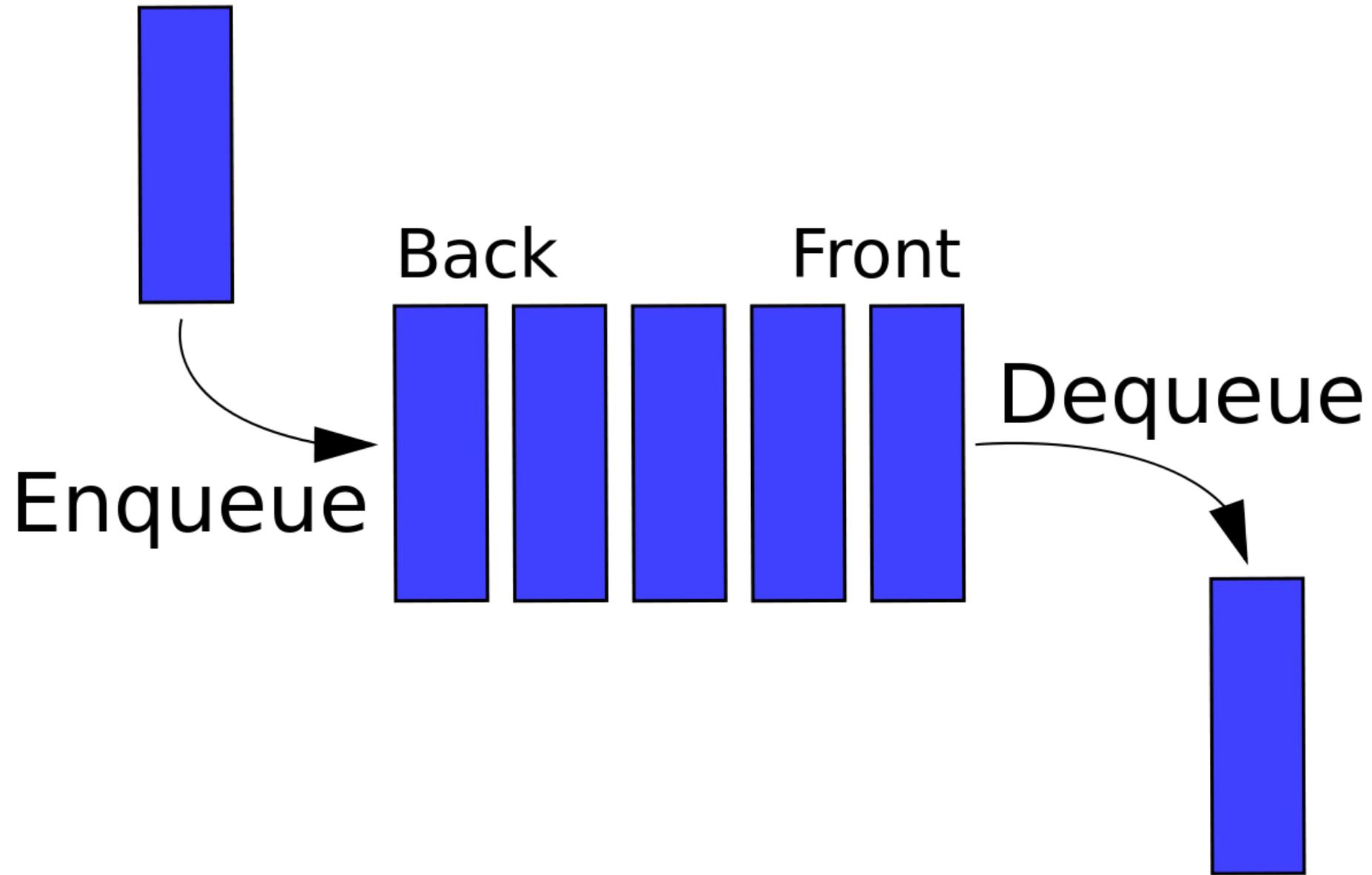
An ADT for a Line

A queue is like a line.



Back

Front



# Queue - FIFO

- An standard object type (an ADT)
- First in, first out (FIFO)
- Add to the back, remove from the front
- Used to simulate lines: Printer Queue, Banking operations, Order Placement



# Object = data + methods



1. `Queue q = new Queue ();`
2. `q.enqueue (1);`
3. `q.enqueue (2);`
4. `System.out.println (q.dequeue ());`
5. `q.enqueue (3);`
6. `q.enqueue (4);`
7. `System.out.println (q.size ());`
8. `System.out.println (q.dequeue ());`
9. `q.enqueue (5);`
10. `System.out.println (q.dequeue ());`

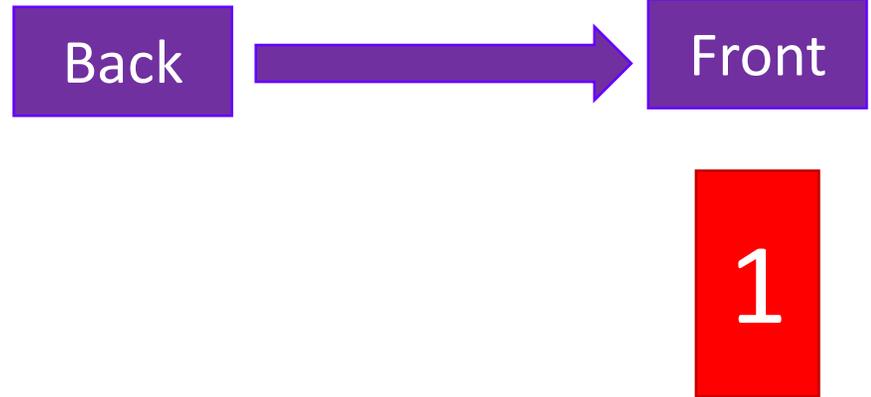
1. `Queue q = new Queue ();`
2. `q.enqueue (1);`
3. `q.enqueue (2);`
4. `System.out.println (q.dequeue ());`
5. `q.enqueue (3);`
6. `q.enqueue (4);`
7. `System.out.println (q.size ());`
8. `System.out.println (q.dequeue ());`
9. `q.enqueue (5);`
10. `System.out.println (q.dequeue ());`

Back

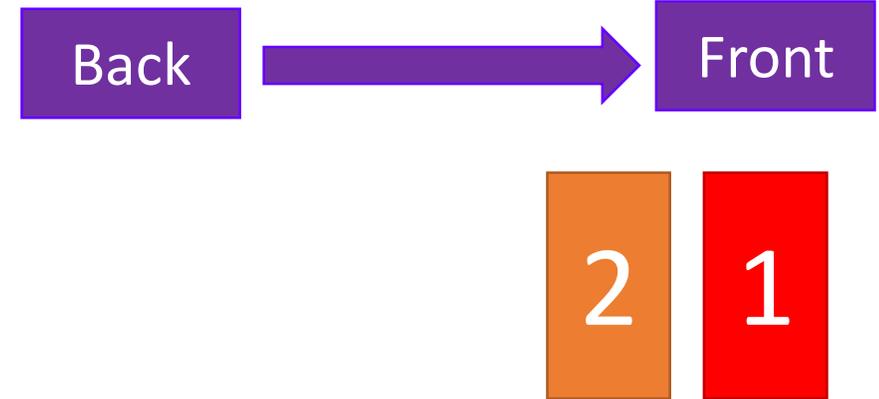


Front

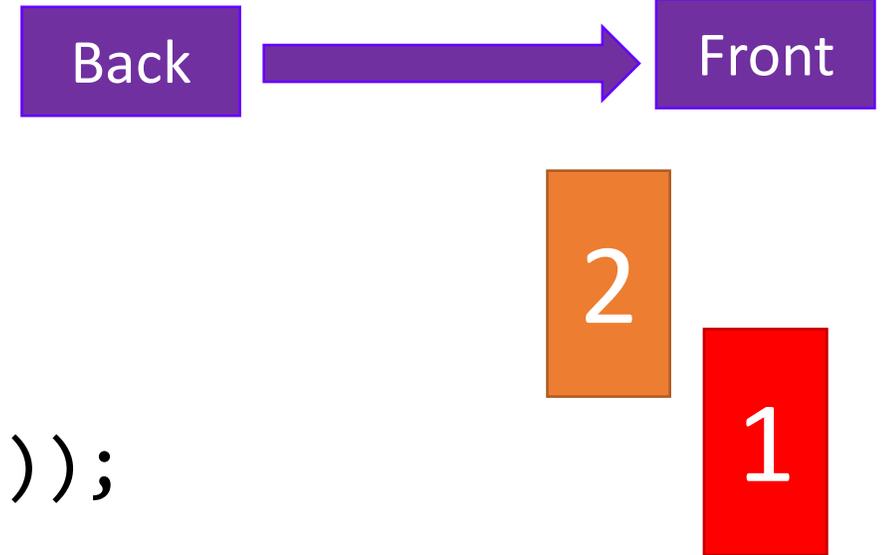
```
1. Queue q = new Queue ();
2. q.enqueue (1);
3. q.enqueue (2);
4. System.out.println (q.dequeue ());
5. q.enqueue (3);
6. q.enqueue (4);
7. System.out.println (q.size ());
8. System.out.println (q.dequeue ());
9. q.enqueue (5);
10. System.out.println (q.dequeue ());
```



```
1. Queue q = new Queue ();
2. q.enqueue (1);
3. q.enqueue (2);
4. System.out.println (q.dequeue ());
5. q.enqueue (3);
6. q.enqueue (4);
7. System.out.println (q.size ());
8. System.out.println (q.dequeue ());
9. q.enqueue (5);
10. System.out.println (q.dequeue ());
```



```
1. Queue q = new Queue ();
2. q.enqueue (1);
3. q.enqueue (2);
1 4. System.out.println (q.dequeue ());
5. q.enqueue (3);
6. q.enqueue (4);
7. System.out.println (q.size ());
8. System.out.println (q.dequeue ());
9. q.enqueue (5);
10. System.out.println (q.dequeue ());
```



1. `Queue q = new Queue ();`

2. `q.enqueue (1);`

3. `q.enqueue (2);`

**1** 4. `System.out.println (q.dequeue ());`

5. `q.enqueue (3);`

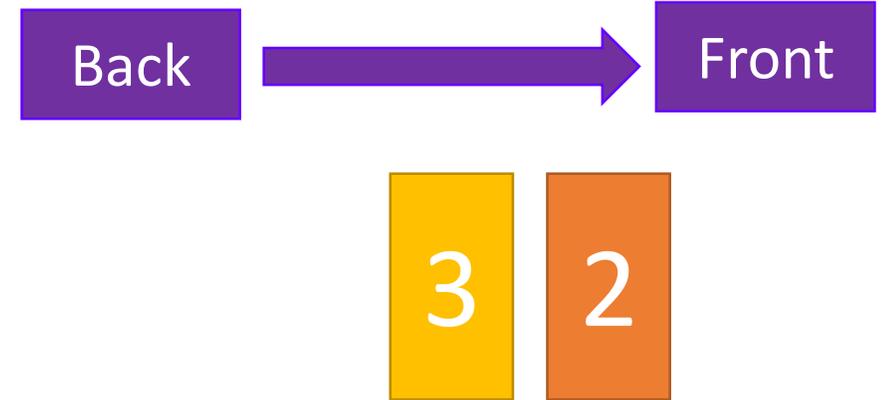
6. `q.enqueue (4);`

7. `System.out.println (q.size ());`

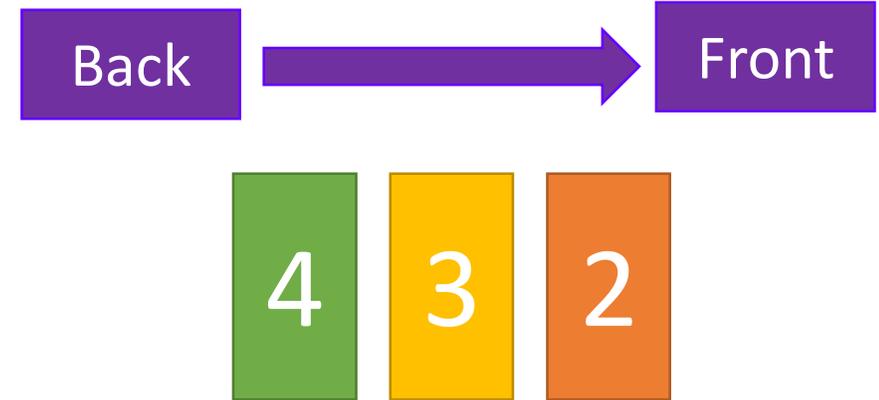
8. `System.out.println (q.dequeue ());`

9. `q.enqueue (5);`

10. `System.out.println (q.dequeue ());`



```
1. Queue q = new Queue ();
2. q.enqueue (1);
3. q.enqueue (2);
1 4. System.out.println (q.dequeue ());
5. q.enqueue (3);
6. q.enqueue (4);
7. System.out.println (q.size ());
8. System.out.println (q.dequeue ());
9. q.enqueue (5);
10. System.out.println (q.dequeue ());
```



1. `Queue q = new Queue ();`

2. `q.enqueue (1);`

3. `q.enqueue (2);`

1 4. `System.out.println (q.dequeue ());`

5. `q.enqueue (3);`

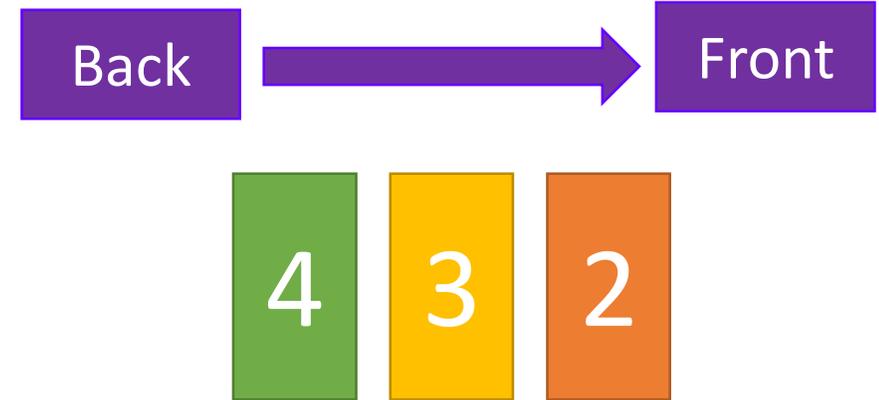
6. `q.enqueue (4);`

3 7. `System.out.println (q.size ());`

8. `System.out.println (q.dequeue ());`

9. `q.enqueue (5);`

10. `System.out.println (q.dequeue ());`



1. `Queue q = new Queue ();`

2. `q.enqueue (1);`

3. `q.enqueue (2);`

**1** 4. `System.out.println (q.dequeue ());`

5. `q.enqueue (3);`

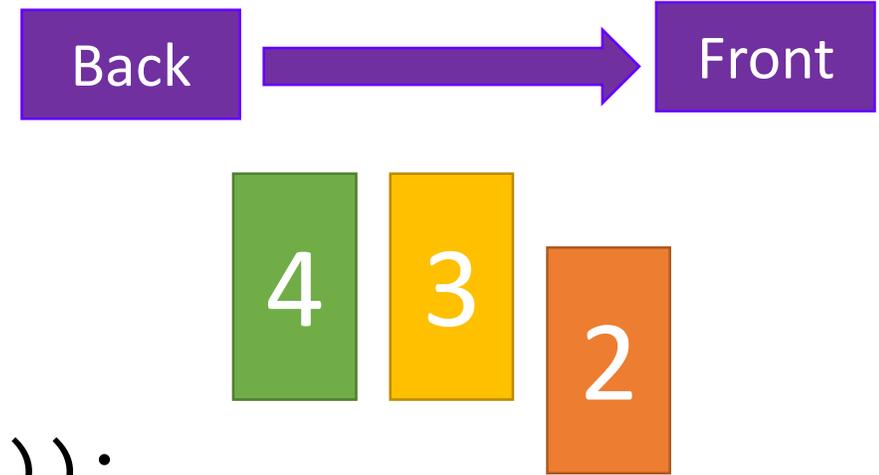
6. `q.enqueue (4);`

**3** 7. `System.out.println (q.size ());`

**2** 8. `System.out.println (q.dequeue ());`

9. `q.enqueue (5);`

10. `System.out.println (q.dequeue ());`



1. `Queue q = new Queue ();`

2. `q.enqueue (1);`

3. `q.enqueue (2);`

**1** 4. `System.out.println (q.dequeue ());`

5. `q.enqueue (3);`

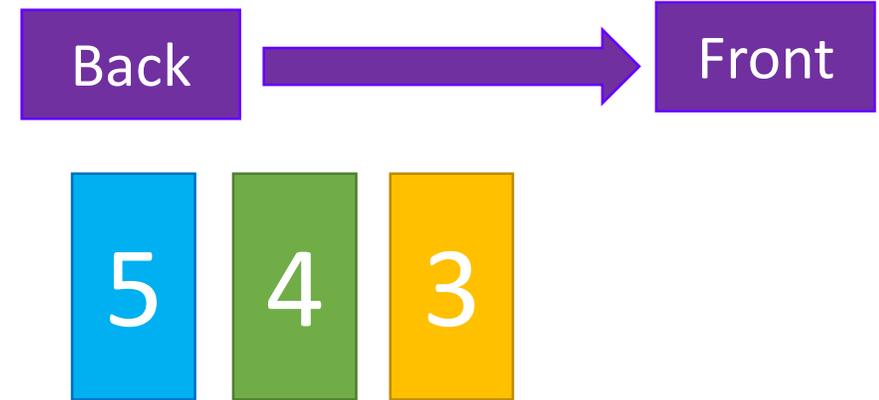
6. `q.enqueue (4);`

**3** 7. `System.out.println (q.size ());`

**2** 8. `System.out.println (q.dequeue ());`

9. `q.enqueue (5);`

10. `System.out.println (q.dequeue ());`



1. `Queue q = new Queue ();`

2. `q.enqueue (1);`

3. `q.enqueue (2);`

1 4. `System.out.println (q.dequeue ());`

5. `q.enqueue (3);`

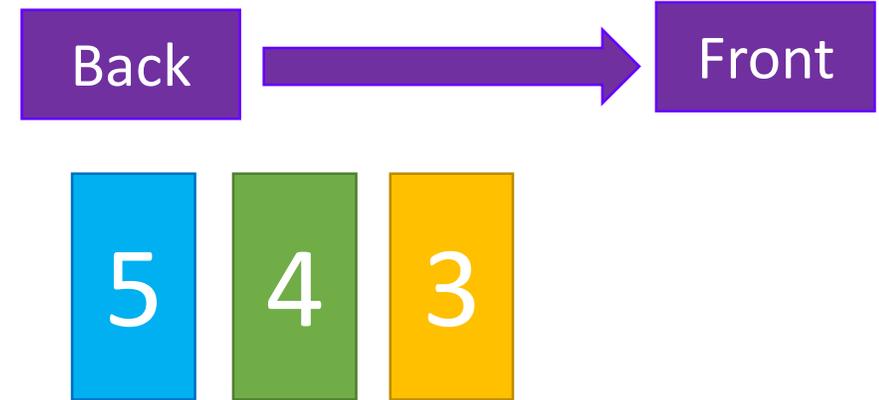
6. `q.enqueue (4);`

3 7. `System.out.println (q.size ());`

2 8. `System.out.println (q.dequeue ());`

9. `q.enqueue (5);`

3 10. `System.out.println (q.dequeue ());`



## Declaration

Setting aside RAM  
"JButton b;"



## Constructor

Sets up dynamic memory  
to be ready for use  
"new"

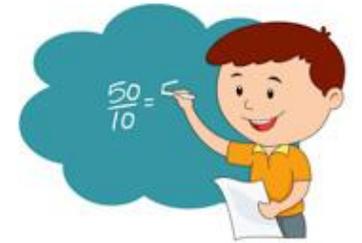
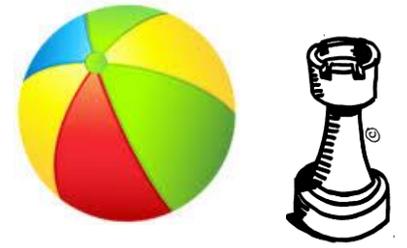


## Mutator

Changing  
dynamic memory  
"set" - setForeground

## Facilitator

Anything that isn't  
construction, accessor, mutator  
Complex functions  
"equals"  
"compareTo"



## Accessor

Checking dynamic memory  
"get" – getText  
"toString"  
"is" – isVisible



## Classifying the Method Types in the Queue Class

```
Queue ()  
void enqueue (String value)  
String dequeue ()  
String peek ()  
int size ()  
boolean isEmpty ()  
boolean isFull ()  
void clear()
```

# Classifying the Method Types in the Queue Class

constructor



Queue ()

mutator



void enqueue (String value)

mutator



String dequeue ()

accessor



String peek ()

accessor



int size ()

accessor



boolean isEmpty ()

accessor



boolean isFull ()

mutator



void clear()

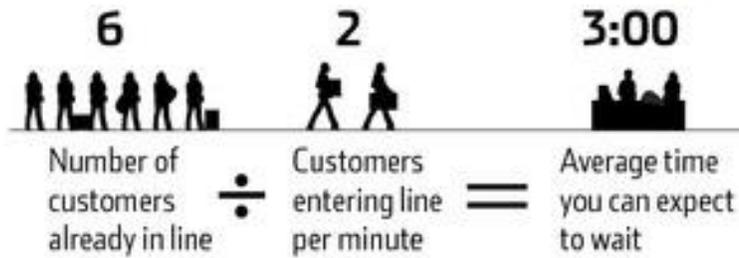
# Queue

- A common ADT, An object
- Has a FIFO structure (First in, first out)
- Functions: size, isFull, isEmpty, enqueue (add), dequeue (remove), peek (look at the top)
- All operations are  $O(1)$ , but you are limited to queue functions. The speed comes from the fact that you restricted the functionality a lot.

# The Science of Lines

## What's really happening at checkout

A shopper can use this **formula**, by John D.C. Little, to determine expected wait time: Average wait time = average number of people in line divided by their arrival rate.



### Clock watching

Once a wait lasts longer than three minutes, the perceived wait time multiplies with each passing minute. Shoppers who actually waited five minutes told surveyors they felt they had waited twice as long.

### Impulse buying

Mall retailers are copying grocery stores with items like tiny stuffed animals and gift cards next to lines to distract from the wait.



### Line jockeying

Short lines are usually short for a reason. Other shoppers may have concluded that a short line has an extremely slow or chatty cashier.

### More staff

Some stores employ 'runners' at the holidays to assist cashiers. Old Navy sends out 'line expeditors' and 'super helpers' during peak times.

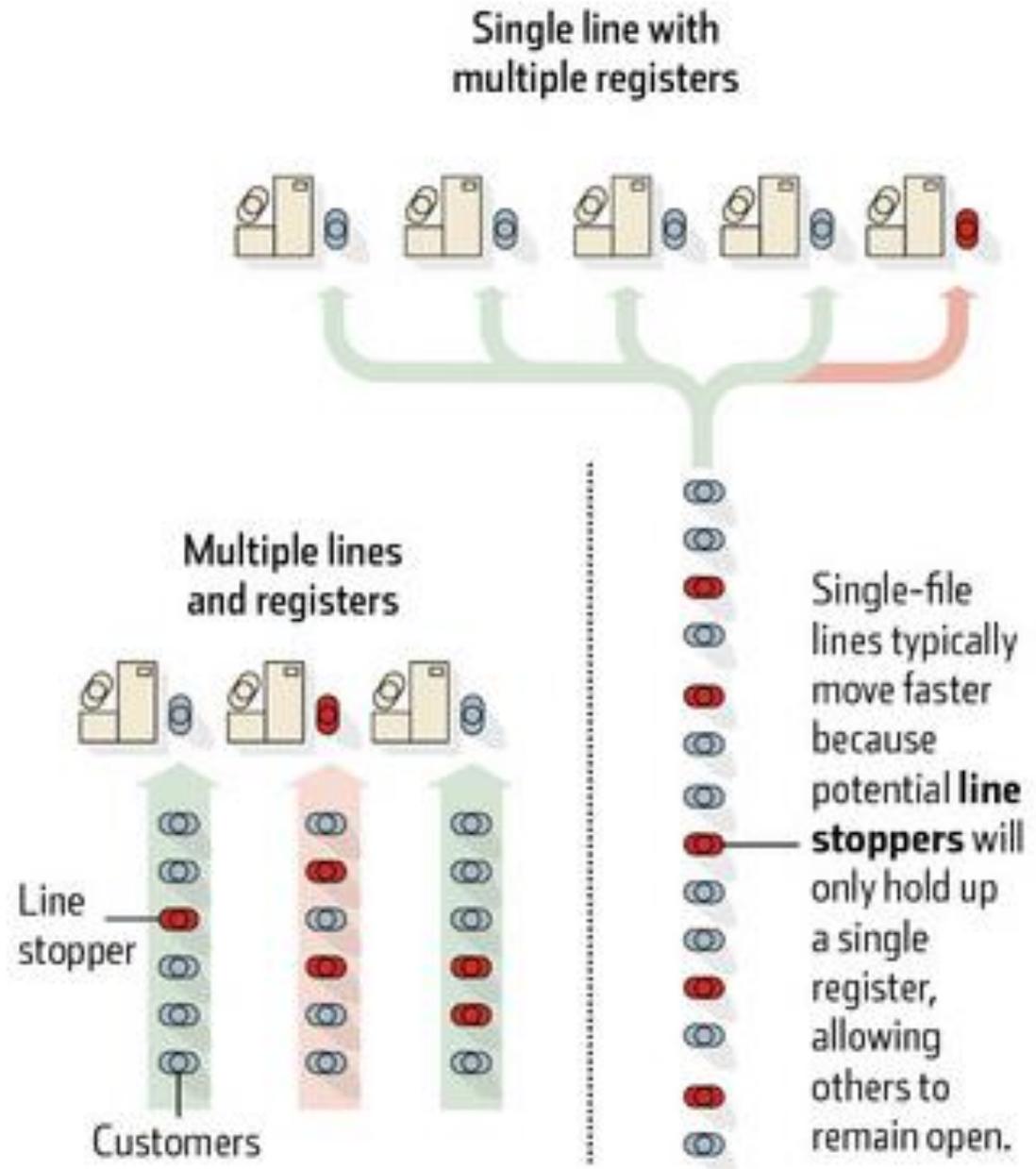
A common introductory programming problem is to use a queue to **simulate** lines.

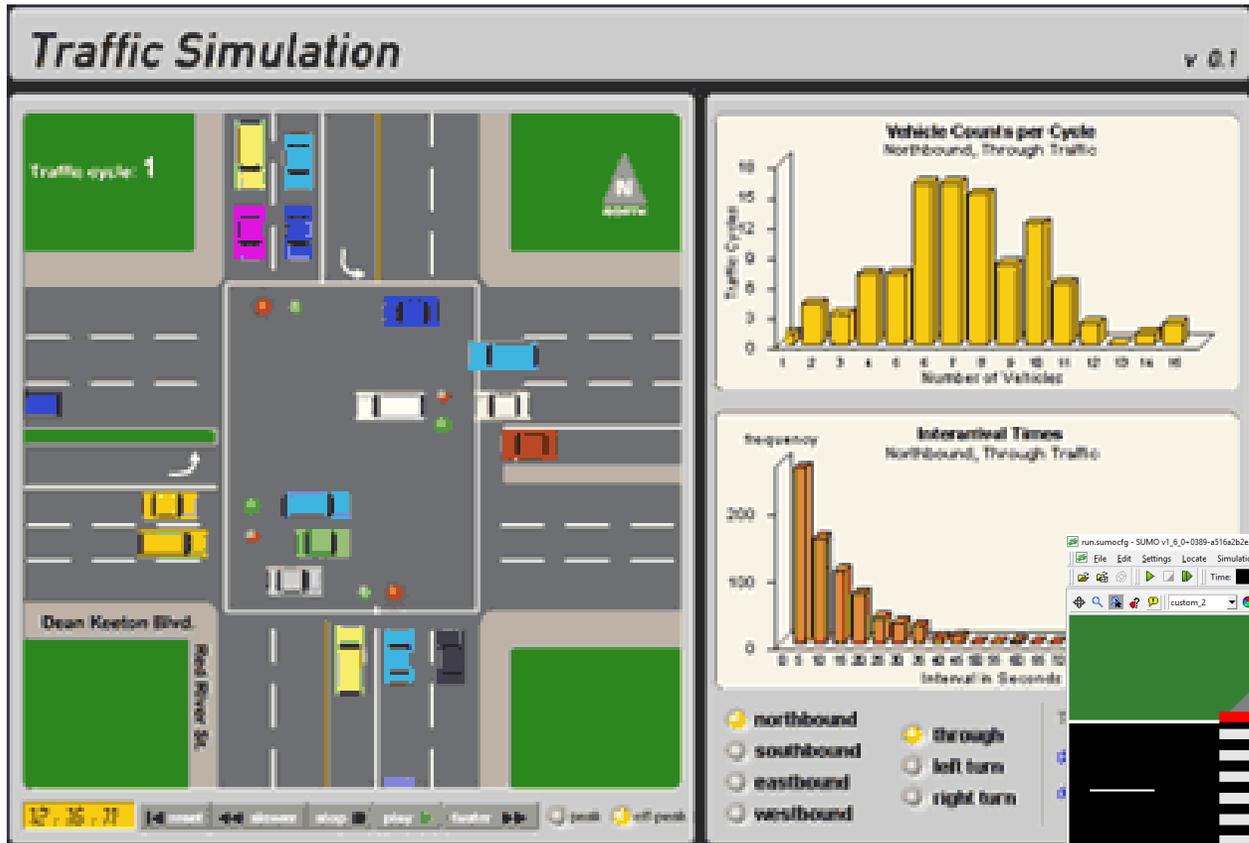


These types of **simulations** have lead us to discover a lot about supermarket lines.

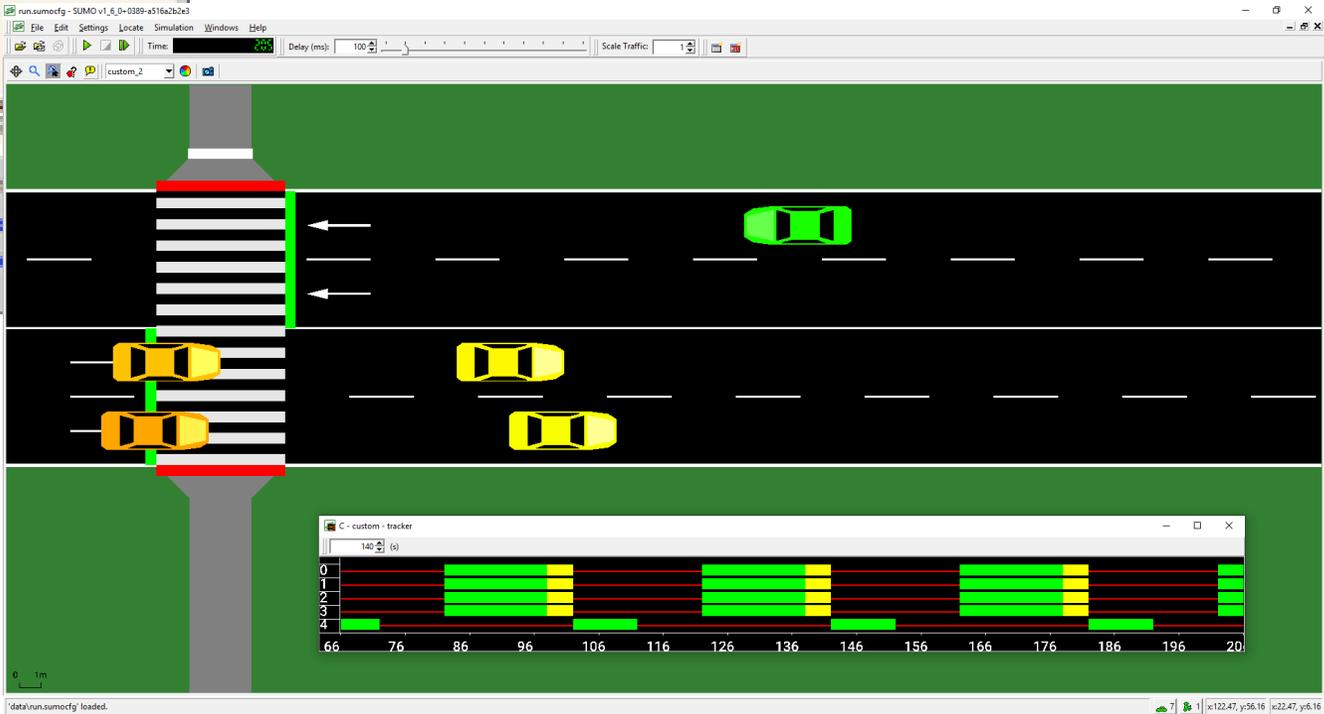
### Check It Out

A single-file line leading to three cashiers is about three times faster than having one line for each cashier. At least one of the three lines could have a random event, such as a price check, that would slow the line.



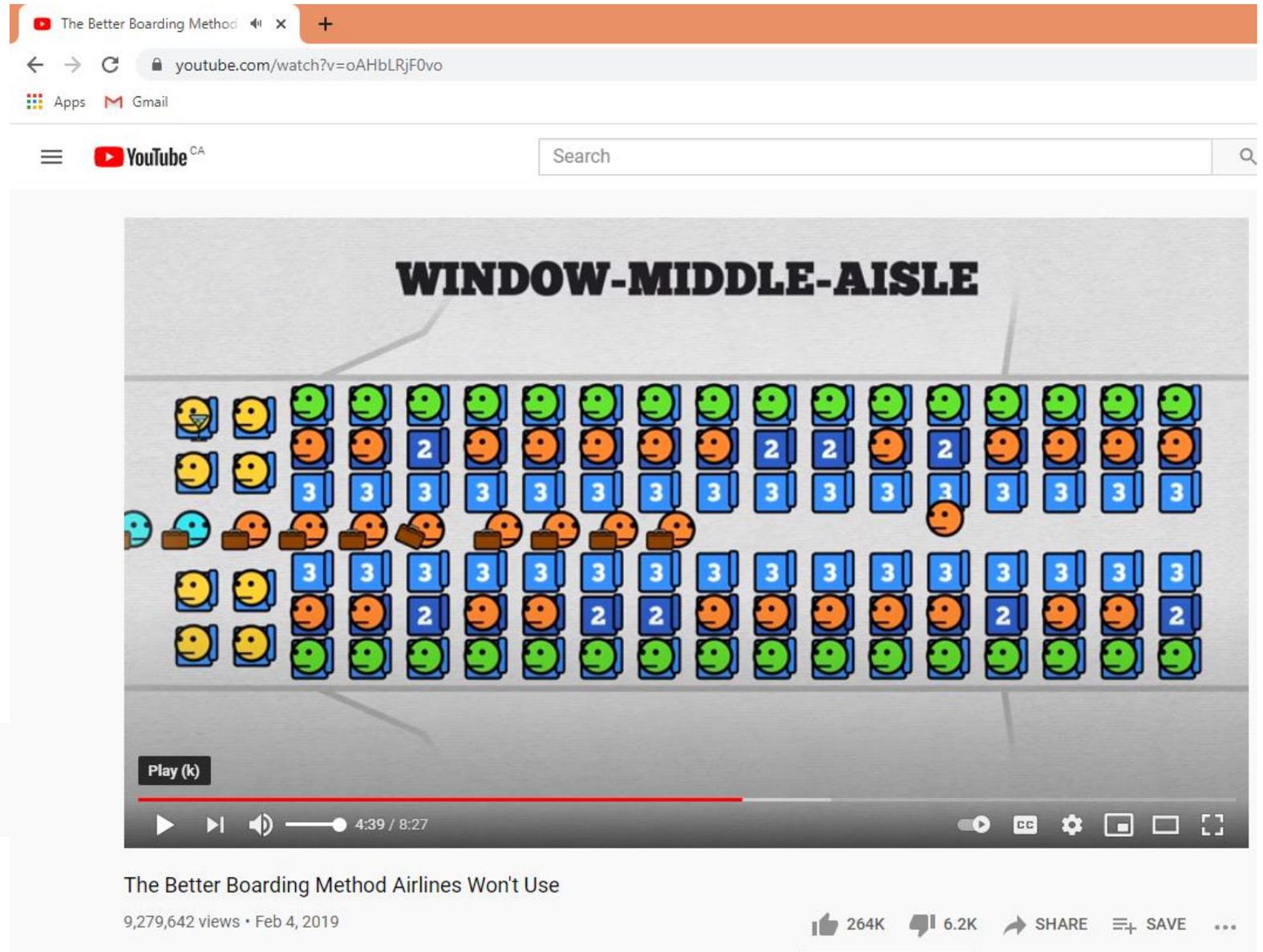


Queues are also used in traffic simulations



<https://www.youtube.com/watch?v=oAHbLRjF0vo>

An airline boarding queue simulation.  
Finds a lot of efficiencies.



The screenshot shows a YouTube video player. The browser address bar displays the URL <https://www.youtube.com/watch?v=oAHbLRjF0vo>. The video player interface includes a search bar, a play button, and a progress bar showing 4:39 / 8:27. The video content features a diagram of an airplane cabin layout with the title **WINDOW-MIDDLE-AISLE**. The diagram shows a 16-row cabin with 16 seats per row. The seats are color-coded: green for window seats, orange for middle seats, and blue for aisle seats. Some seats are marked with numbers 2 or 3. The video player shows 264K likes, 6.2K comments, and options to share and save.

**CGP Grey** ✓  
4.61M subscribers

**The Better Boarding Method Airlines Won't Use**  
9,279,642 views • Feb 4, 2019

264K 6.2K SHARE SAVE ...