# Customers in a Queue

|  |  |  |
| --- | --- | --- |
| Service area 1 | Service area 2 | Service area 3 |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |

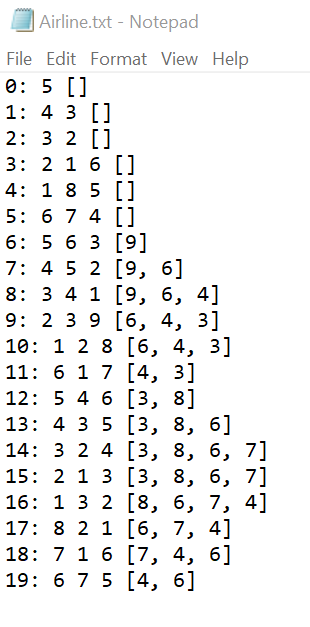
Enter here

Customers at airlines and banks typically join a single queue. They move through the queue and eventually take their place at one of the open counters or "service areas." Each customer then conducts his/her business and leaves. The situation is diagrammed here:

Suppose an airline or a bank wants to measure the wait-times to serve each customer, from the minute they enter to the minute they leave. The airline hires you to program a simulation of the situation.

* You need one queue.
* You need an array of the correct size to represent the service areas.
* You need to know how often, the frequency per minute, that customers join the queue.
* You need to know how long it takes (a random number) to conduct the customer's business after they arrive at the service area. We will count each minute down to 0, after which the customer leaves.
* You need to store the time at which the customer entered the queue.
* You need to calculate how long the customer spent getting served (the total wait time), from the minute they entered to the minute they left.
* You need to know how long to run the simulation.

For each minute write to a file:



* Each minute, starting from 0.
* The service areas serving their customers. Each customer shows the time remaining until they complete their business and leave.
* The queue with its customers, showing the time they need to complete their business.

11 outfile.print(min + ": ");  
 12 for(Customer c : a)  
 13 if( c != null )  
 14 outfile.print(c.toString()+" ");  
 15 outfile.println("" + q);

Display these calculations to the screen:

* The total customers served.
* The average wait time per customer.
* The longest wait time of any customer.
* The length of the longest queue of customers.

Customers in a Queue Simulation!   
How many service areas? 3  
How long, in minutes, should the simulation run? 20  
Total customers served = 9  
Average wait time = 7.5  
Longest wait time = 12  
Longest queue = 5

You need to generate two random numbers in this lab:

* whether a customer joins the queue during that minute. Let's set the frequency at an 80% chance per minute.
* how long it takes to conduct the customer's business after they get to the service area. Let's generate a random integer between 3 and 9 minutes, inclusive.

**The Customer class**

It seems like a Customer class would be a good way to model the situation.

What private data should the Customer store?

What methods do you need?

**Algorithm in the serveTheCustomers method**

What happens in each minute? What might happen in each minute?

What happens to customers in the queue at the end of the day? Do we empty the queue or just not serve them?

**Hints for passing the autograder’s tests:**

Your outfile must be in the right format! The autograder will process your outfile to count your number of customers, longest wait time, longest queue, and total wait time.

1. Use the given variables and methods, not your own.
2. Start the outfile from minute 0.
3. Outfile every minute, even if there are empty queues, []
4. First do the processing on the queue, then outfile the queue.
5. Debugging strategy: change the CHANCE\_OF\_CUSTOMER to 0.01 and run it. Now the activity recorded in your outfile is so small that you can look and count for yourself.