Ama Freeman
Bank Teller Simulator

Goal

In order to help the manager of a bank, I have been tasked to help her decide whether or not she should hire new tellers and how many. After simulation and various tests, with my data I should be able to provide ample information in order for her to make the most cost and time effective decision. It is my duty to make sure the data collected is as close to real-life as possible.

Plan

1. Design a UML based solution to the problem using Object Oriented Analysis and Design. I need to know what the core of my system is before any code is implemented and tested.
2. Build up an infrastructure for my Bank. All objects need to have defined and concise roles in the system. However, my program must also be flexible; if I need to add multiple tellers or shrink the number of customers, it should be a breeze changing my code and when the user give input into these categories.
3. Provide tests and detailed analysis of my results. What happens when we have 10 customers in 10 minutes, or 100 customers in 10 minutes? How will one teller hold up; three tellers; 10 tellers? My program needs to be able to handle the pressure of multiple services as does a typical bank.
4. Collect all of my calculated results and present them to my client for further analysis that involves cost calculation and time management.

Part 1 of Plan - OOA and OOD

I have created a UML diagram displaying the attributes of each class involved in the Bank environment. The diagram shows the implementation of fives classes: Bank, Bank Manager, Customer, Teller, and Account. All banks *at least* have these five core components.

The Bank class is simply where all of the entities of a bank are encapsulated. In the most simple case, a bank has a manager, a teller, customers (but only knows about the accounts of the customers via their account number), and various accounts usually typed as savings and checking.

The Bank Manager class can learn all about the Tellers. The bank manager supervises the Tellers and can find out their names, salaries, total customers served, can schedule the availability of Tellers, and create and destroy Tellers.

The Teller class has the power to access the name and account numbers of Customers in order to access the respective account of the customer. If they are a new customer, the Teller has the ability to create a new Account associated with a new Customer.

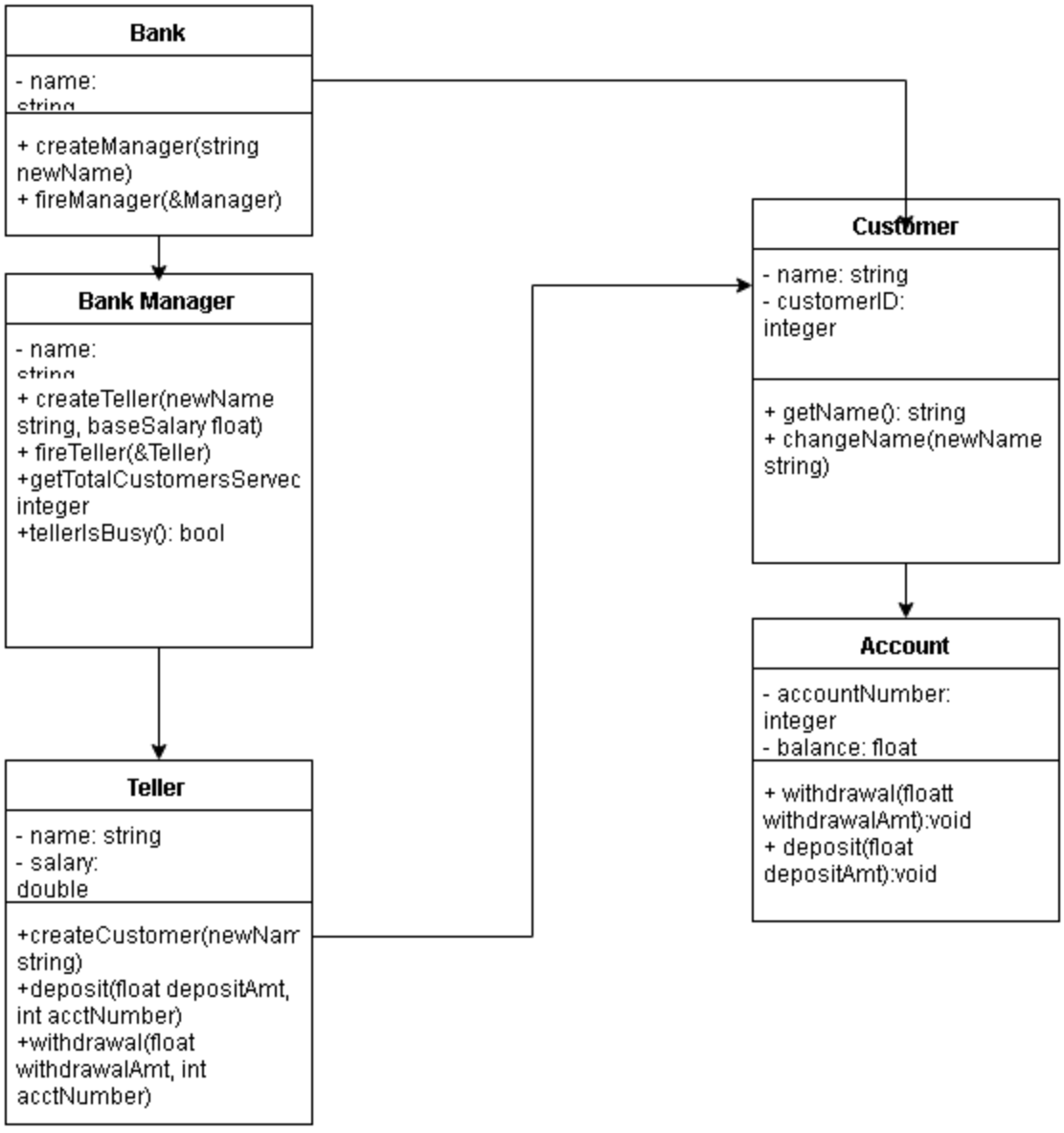
The Account class contains respective class methods: withdrawal, deposit, and check balance. Accounts of type savings and checking can also be created for the application of such in a more realistic bank.

The Customer class has the attributes of a name and account number. They have access to their Accounts with or without the use of a Teller.

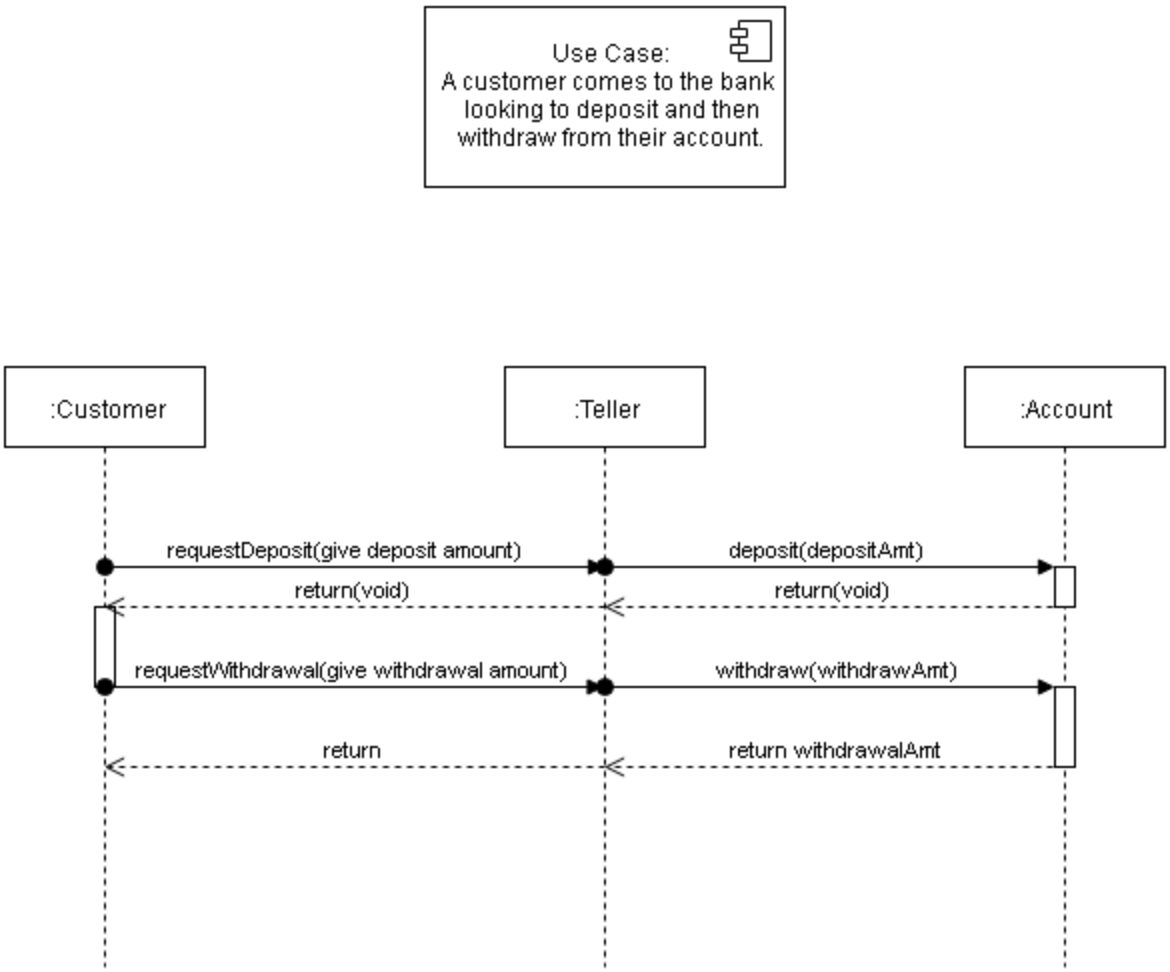
While most of these features will not be implemented in my simple queue program, I find that it is imperative to include this information along with my UML diagram.

Four plausible use cases are displayed in the attached documents. Standard UML is used to define the sequence of interaction in each use case.
Use case one has a new Customer wishing to create a new account with the Bank through the Teller. Use case two is a non-optimal, but completely plausible case, where the Customer wishes to leave the Bank. In such a case, the money will be released to the Customer and various pointers and data will be released from memory in order to delete the Customer's data. The last two cases involve the Customer withdrawing and depositing cash into their respective account. As mentioned before, these cases are extremely simple yet seen in every Bank environment.

UML Class Diagram



Example Use Case Sequence Diagram



Part 2 of Plan - Pseudocode

The program will implement a queue based data structure. The Tellers will have access to this queue and the queue will be made up of Customer objects. In order to keep the program simple, the three following guidelines will be followed:

1. A customer is simply a number and a randomly generated transaction time. Every time a new customer is created and added to the queue, it is simply the order number of said customer and their designated transaction time. A newly created customer will be pushed into the queue. The queue stack will follow the standard first in, first out protocol.
2. Tellers will pop customers with every iteration of the unit time. The Tellers are kept in a vector of Tellers. All Tellers are initialized with zero customers helped and their busy status is set to false. If the Teller is not currently helping a customer, they will pop the next Customer from the queue and set the Teller status function, which is a bool, to true. The status function is called “getStatus().” This function returns true if the Teller has a customer, and false otherwise. If false, a Customer will be popped from the queue.
3. The Customer’s transaction time with a Teller is added to the total transaction time of the simulation. This value will be used to calculate the average transaction time using this formula: totalServiceTime / totalCustomersServed.
4. A while loop is used to continue the simulation. When the simulation time given by the user is equal to the start time (which is always 0), the while loop will break, retaining all information for calculations.
5. Various collections data are displayed to the user. All of this data is for recording and calculation purposes. Stats that will be calculated for the user: average transaction time for all tellers, average time a customer waits to be served, average wait time for 100 customers.

Part 3 of Plan - Testing

Three tests will be implemented. All cases will be required to flawlessly serve 10 Customers in 10 minutes, 100 Customers in 10 minutes, and finally, a random user input. Please note that the minutes are place holders for unit time. Essentially with each iteration, 1 or 10 or whatever number given by the user divided by ten, will be the number of Customers queued with each unit time. Please observe the results below.

Case One - 1 Teller, 100 Customers, 10 Minutes

In this case, 10 customers arrive every minute for 10 minutes. In other words, every minute, a 10 new customers are added to the queue. The service time of each customer is assumed to be 1 unit time, or one minute. Results follow:

```
How long would you like to run the simulation? (Time is in minutes.): 10
The program will be run for 10 minutes.
How many customers would you like? Enter an integer: 100
Now creating a line of 100 customer(s).
How many Tellers would you like? Enter an integer: 1
Created 1 teller(s) for simulation.
Would you like to a. generate random transaction times or b. generate a common t
ransaction time of 1 minute? (Type a or b): b
All customers will have a transaction time of 1 minute.
Teller #1 is taking Customer #1. Customer's Transaction Time = 1
Customer #1 waited 1 minutes before departing.
Minute 1. Adding 10 more customers to the queue.
Teller #1 is taking Customer #2. Customer's Transaction Time = 1
Customer #2 waited 2 minutes before departing.
Minute 2. Adding 10 more customers to the queue.
Teller #1 is taking Customer #3. Customer's Transaction Time = 1
Customer #3 waited 3 minutes before departing.
Minute 3. Adding 10 more customers to the queue.
Teller #1 is taking Customer #4. Customer's Transaction Time = 1
Customer #4 waited 4 minutes before departing.
Minute 4. Adding 10 more customers to the queue.
Teller #1 is taking Customer #5. Customer's Transaction Time = 1
Customer #5 waited 5 minutes before departing.
Minute 5. Adding 10 more customers to the queue.
Teller #1 is taking Customer #6. Customer's Transaction Time = 1
Customer #6 waited 6 minutes before departing.
Minute 6. Adding 10 more customers to the queue.
Teller #1 is taking Customer #7. Customer's Transaction Time = 1
Customer #7 waited 7 minutes before departing.
Minute 7. Adding 10 more customers to the queue.
Teller #1 is taking Customer #8. Customer's Transaction Time = 1
Customer #8 waited 8 minutes before departing.
Minute 8. Adding 10 more customers to the queue.
Teller #1 is taking Customer #9. Customer's Transaction Time = 1
Customer #9 waited 9 minutes before departing.
Minute 9. Adding 10 more customers to the queue.
Teller #1 is taking Customer #10. Customer's Transaction Time = 1
Customer #10 waited 10 minutes before departing.
Minute 10. Adding 10 more customers to the queue.
```

```
------------------------------------------------------------
Results:
Total customers served: 10
Total transaction time: 10 minutes.
Number of Customers: 100
Number of Tellers: 1
Simulation time (in minutes): 10
Average Transaction time is 1 minutes.
Average departure time is 5 minutes.
Average wait time is: 17 minutes.
Press any key to continue . . .
```

In another case, every condition in the setting above is followed, but each customer has a random transaction time. Results follow:

```
How long would you like to run the simulation? (Time is in minutes.): 10
The program will be run for 10 minutes.
How many customers would you like? Enter an integer: 100
Now creating a line of 100 customer(s).
How many Tellers would you like? Enter an integer: 1
Created 1 teller(s) for simulation.
Teller #1 is taking Customer #1. Customer's Transaction Time = 1
Customer #1 waited 1 minutes before departing.
Minute 1. Adding 10 more customers to the queue.
Teller #1 is taking Customer #2. Customer's Transaction Time = 7
Customer #2 waited 8 minutes before departing.
Minute 2. Adding 10 more customers to the queue.
Minute 3. Adding 10 more customers to the queue.
Minute 4. Adding 10 more customers to the queue.
Minute 5. Adding 10 more customers to the queue.
Minute 6. Adding 10 more customers to the queue.
Minute 7. Adding 10 more customers to the queue.
Minute 8. Adding 10 more customers to the queue.
Teller #1 is taking Customer #3. Customer's Transaction Time = 9
Customer #3 waited 17 minutes before departing.
Minute 9. Adding 10 more customers to the queue.
Minute 10. Adding 10 more customers to the queue.
------------------------------------------------------------
------------------------------------------------------------
Results:
Total customers served: 3
Total transaction time: 17 minutes.
Number of Customers: 100
Number of Tellers: 1
Simulation time (in minutes): 10
Average Transaction time is 5 minutes.
Average departure time is 8 minutes.
Average wait time is: 23 minutes.
Press any key to continue . . .
```

Case Two - 2 Tellers, 100 Customers, 10 Minutes

The two cases above are tested, but this time, there are two tellers.

First case results follow:

```
How long would you like to run the simulation? (Time is in minutes.): 10
The program will be run for 10 minutes.
How many customers would you like? Enter an integer: 100
Now creating a line of 100 customer(s).
How many Tellers would you like? Enter an integer: 2
Created 2 teller(s) for simulation.
Would you like to a. generate random transaction times or b. generate a common t
ransaction time of 1 minute? (Type a or b): b
All customers will have a transaction time of 1 minute.
Teller #1 is taking Customer #1. Customer's Transaction Time = 1
Customer #1 waited 1 minutes before departing.
Teller #2 is taking Customer #2. Customer's Transaction Time = 1
Customer #2 waited 1 minutes before departing.
Minute 1. Adding 10 more customers to the queue.
Teller #1 is taking Customer #3. Customer's Transaction Time = 1
Customer #3 waited 2 minutes before departing.
Teller #2 is taking Customer #4. Customer's Transaction Time = 1
Customer #4 waited 2 minutes before departing.
Minute 2. Adding 10 more customers to the queue.
Teller #1 is taking Customer #5. Customer's Transaction Time = 1
Customer #5 waited 3 minutes before departing.
Teller #2 is taking Customer #6. Customer's Transaction Time = 1
Customer #6 waited 3 minutes before departing.
Minute 3. Adding 10 more customers to the queue.
Teller #1 is taking Customer #7. Customer's Transaction Time = 1
Customer #7 waited 4 minutes before departing.
Teller #2 is taking Customer #8. Customer's Transaction Time = 1
Customer #8 waited 4 minutes before departing.
Minute 4. Adding 10 more customers to the queue.
Teller #1 is taking Customer #9. Customer's Transaction Time = 1
Customer #9 waited 5 minutes before departing.
Teller #2 is taking Customer #10. Customer's Transaction Time = 1
Customer #10 waited 5 minutes before departing.
Minute 5. Adding 10 more customers to the queue.
Teller #1 is taking Customer #1. Customer's Transaction Time = 1
Customer #1 waited 6 minutes before departing.
Teller #2 is taking Customer #2. Customer's Transaction Time = 1
Customer #2 waited 6 minutes before departing.
Minute 6. Adding 10 more customers to the queue.
Teller #1 is taking Customer #3. Customer's Transaction Time = 1
Customer #3 waited 7 minutes before departing.
Teller #2 is taking Customer #4. Customer's Transaction Time = 1
Customer #4 waited 7 minutes before departing.
Minute 7. Adding 10 more customers to the queue.
Teller #1 is taking Customer #5. Customer's Transaction Time = 1
Customer #5 waited 8 minutes before departing.
Teller #2 is taking Customer #6. Customer's Transaction Time = 1
Customer #6 waited 8 minutes before departing.
Minute 8. Adding 10 more customers to the queue.
```

```
Teller #1 is taking Customer #7. Customer's Transaction Time = 1
Customer #7 waited 9 minutes before departing.
Teller #2 is taking Customer #8. Customer's Transaction Time = 1
Customer #8 waited 9 minutes before departing.
Minute 9. Adding 10 more customers to the queue.
Teller #1 is taking Customer #9. Customer's Transaction Time = 1
Customer #9 waited 10 minutes before departing.
Teller #2 is taking Customer #10. Customer's Transaction Time = 1
Customer #10 waited 10 minutes before departing.
Minute 10. Adding 10 more customers to the queue.
------------------------------------------------------------
Results:
Total customers served: 20
Total transaction time: 20 minutes.
Number of Customers: 100
Number of Tellers: 2
Simulation time (in minutes): 10
Average Transaction time is 1 minutes.
Average departure time is 5 minutes.
Average wait time is: 19 minutes.
Press any key to continue . . .
```

Second case results follow:

```
How long would you like to run the simulation? (Time is in minutes.): 10
The program will be run for 10 minutes.
How many customers would you like? Enter an integer: 100
Now creating a line of 100 customer(s).
How many Tellers would you like? Enter an integer: 2
Created 2 teller(s) for simulation.
Teller #1 is taking Customer #1. Customer's Transaction Time = 8
Customer #1 waited 8 minutes before departing.
Teller #2 is taking Customer #2. Customer's Transaction Time = 6
Customer #2 waited 6 minutes before departing.
Minute 1. Adding 10 more customers to the queue.
Minute 2. Adding 10 more customers to the queue.
Minute 3. Adding 10 more customers to the queue.
Minute 4. Adding 10 more customers to the queue.
Minute 5. Adding 10 more customers to the queue.
Minute 6. Adding 10 more customers to the queue.
Teller #2 is taking Customer #3. Customer's Transaction Time = 8
Customer #3 waited 14 minutes before departing.
Minute 7. Adding 10 more customers to the queue.
Minute 8. Adding 10 more customers to the queue.
Teller #1 is taking Customer #4. Customer's Transaction Time = 8
Customer #4 waited 16 minutes before departing.
Minute 9. Adding 10 more customers to the queue.
Minute 10. Adding 10 more customers to the queue.
------------------------------------------------------------
------------------------------------------------------------
Results:
Total customers served: 4
Total transaction time: 30 minutes.
Number of Customers: 100
Number of Tellers: 2
Simulation time (in minutes): 10
Average Transaction time is 7 minutes.
Average departure time is 11 minutes.
Average wait time is: 45 minutes.
Press any key to continue . . .
```

Case Three - 10 Tellers, 100 Customers, 10 Minutes

First case results follow: (Note: all customers were served in this case. The output was far too long to include within the report, but the results are accurate and conclusive.)

```
Results:
Total customers served: 100
Total transaction time: 100 minutes.
Number of Customers: 100
Number of Tellers: 10
Simulation time (in minutes): 10
Average Transaction time is 1 minutes.
Average departure time is 5 minutes.
Average wait time is: 21 minutes.
Press any key to continue . . .
```

Second case results follow:

```
How long would you like to run the simulation? (Time is in minutes.): 10
The program will be run for 10 minutes.
How many customers would you like? Enter an integer: 100
Now creating a line of 100 customer(s).
How many Tellers would you like? Enter an integer: 10
Created 10 teller(s) for simulation.
Teller #1 is taking Customer #1. Customer's Transaction Time = 9
Customer #1 waited 9 minutes before departing.
Teller #2 is taking Customer #2. Customer's Transaction Time = 7
Customer #2 waited 7 minutes before departing.
Teller #3 is taking Customer #3. Customer's Transaction Time = 3
Customer #3 waited 3 minutes before departing.
Teller #4 is taking Customer #4. Customer's Transaction Time = 4
Customer #4 waited 4 minutes before departing.
Teller #5 is taking Customer #5. Customer's Transaction Time = 5
Customer #5 waited 5 minutes before departing.
Teller #6 is taking Customer #6. Customer's Transaction Time = 1
Customer #6 waited 1 minutes before departing.
Teller #7 is taking Customer #7. Customer's Transaction Time = 6
Customer #7 waited 6 minutes before departing.
Teller #8 is taking Customer #8. Customer's Transaction Time = 4
Customer #8 waited 4 minutes before departing.
Teller #9 is taking Customer #9. Customer's Transaction Time = 1
Customer #9 waited 1 minutes before departing.
Teller #10 is taking Customer #10. Customer's Transaction Time = 8
```

```
Teller #10 is taking Customer #10. Customer's Transaction Time = 8
Customer #10 waited 8 minutes before departing.
Minute 1. Adding 10 more customers to the queue.
Teller #6 is taking Customer #1. Customer's Transaction Time = 9
Customer #1 waited 10 minutes before departing.
Teller #9 is taking Customer #2. Customer's Transaction Time = 7
Customer #2 waited 8 minutes before departing.
Minute 2. Adding 10 more customers to the queue.
Minute 3. Adding 10 more customers to the queue.
Teller #3 is taking Customer #3. Customer's Transaction Time = 3
Customer #3 waited 6 minutes before departing.
Minute 4. Adding 10 more customers to the queue.
Teller #4 is taking Customer #4. Customer's Transaction Time = 4
Customer #4 waited 8 minutes before departing.
Teller #8 is taking Customer #5. Customer's Transaction Time = 5
Customer #5 waited 9 minutes before departing.
Minute 5. Adding 10 more customers to the queue.
Teller #5 is taking Customer #6. Customer's Transaction Time = 1
Customer #6 waited 6 minutes before departing.
Minute 6. Adding 10 more customers to the queue.
Teller #3 is taking Customer #7. Customer's Transaction Time = 6
Customer #7 waited 12 minutes before departing.
Teller #5 is taking Customer #8. Customer's Transaction Time = 4
Customer #8 waited 10 minutes before departing.
Teller #7 is taking Customer #9. Customer's Transaction Time = 1
```

```
Customer #9 waited 7 minutes before departing.
Minute 7. Adding 10 more customers to the queue.
Teller #2 is taking Customer #10. Customer's Transaction Time = 8
Customer #10 waited 15 minutes before departing.
Teller #7 is taking Customer #1. Customer's Transaction Time = 9
Customer #1 waited 16 minutes before departing.
Minute 8. Adding 10 more customers to the queue.
Teller #4 is taking Customer #2. Customer's Transaction Time = 7
Customer #2 waited 15 minutes before departing.
Teller #9 is taking Customer #3. Customer's Transaction Time = 3
Customer #3 waited 11 minutes before departing.
Teller #10 is taking Customer #4. Customer's Transaction Time = 4
Customer #4 waited 12 minutes before departing.
Minute 9. Adding 10 more customers to the queue.
Teller #1 is taking Customer #5. Customer's Transaction Time = 5
Customer #5 waited 14 minutes before departing.
Teller #8 is taking Customer #6. Customer's Transaction Time = 1
Customer #6 waited 10 minutes before departing.
Minute 10. Adding 10 more customers to the queue.
------------------------------------------------------------
```

```
------------------------------------------------------------
Results:
Total customers served: 26
Total transaction time: 125 minutes.
Number of Customers: 100
Number of Tellers: 10
Simulation time (in minutes): 10
Average Transaction time is 4 minutes.
Average departure time is 8 minutes.
Average wait time is: 41 minutes.
Press any key to continue . . .
```

Part 4 of Plan - Evaluation and Presentation

Unit Tests

The solution has many concise comments throughout the files of the program. Since the core of the system requires creating a queue of a custom class of customers and a vector of tellers, it is imperative to make sure the input from the user created the desired products.

Creating the teller vector was simple, as I have created vectors made from struct and class objects before. However, the queue gave me issues with passing the queue by reference while also attempting to create the queue outside of my main function. I had to settle with creating the queue in the Main.cpp file and passing the queue by reference to my simulation file. It was not the ideal situation, but it had to be done in order for my system to work as I intended it to.

After the vector and queue creation system were manifested, a simple print function was built in order to show the results of the completed modules. As expected, the queue was fully functional and so was the vector. All attributes of the customer and teller objects were correctly displayed.

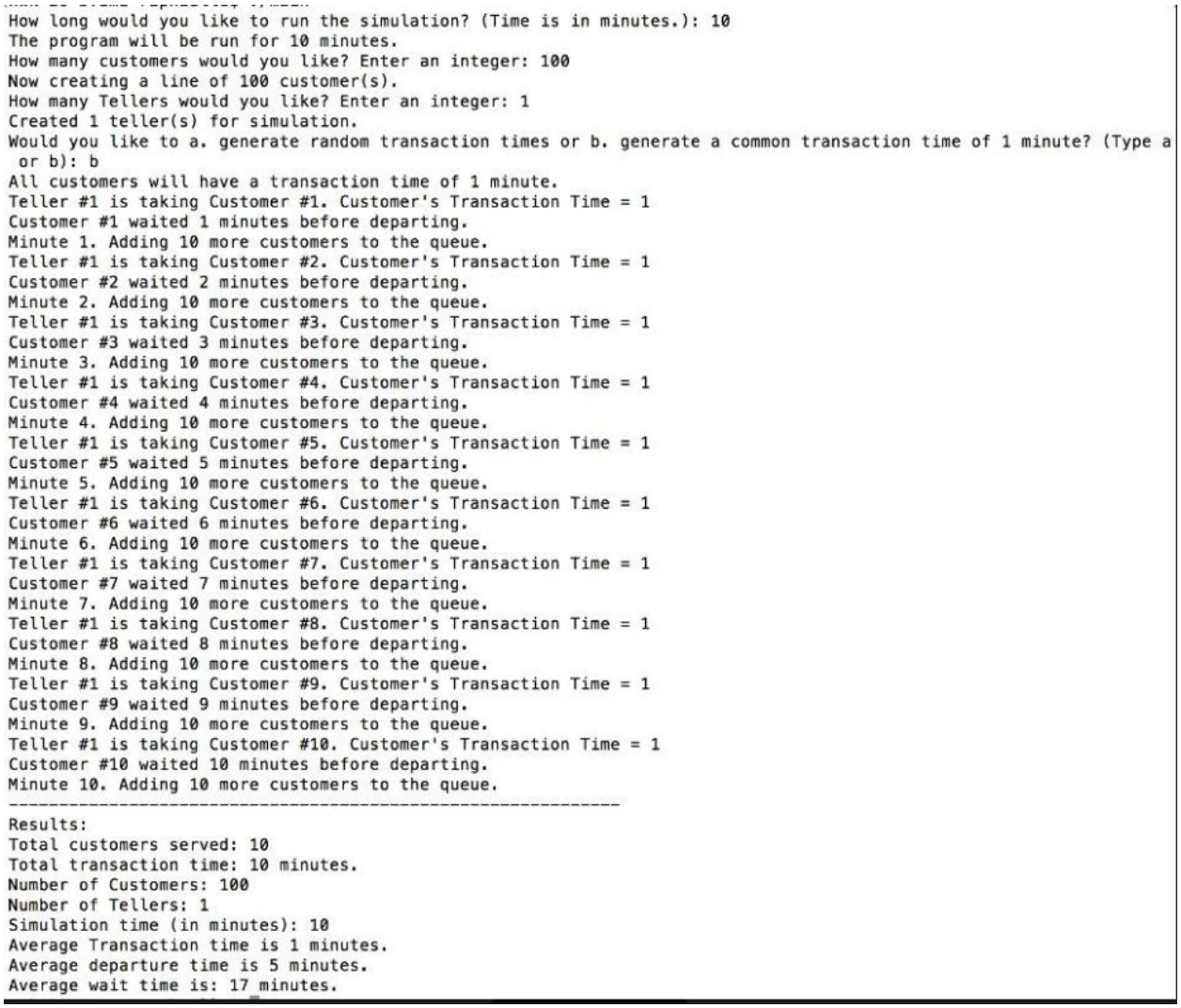
Integration Tests

This was the largest and most complicated test. This test required large amounts of debugging in order to display logical results and also not run into errors. The main question of this test was: How will I allow every teller a chance to take customers of various transaction times simultaneously? Please observe in the runSimulation() method how I implemented this very idea. At a start time of zero minutes, a for loop is integrated into the system. This for loop is used to parse through every hired teller's status at the very same minute. Conditional statements are used to decide whether or not a teller is ready for another customer. If they are not busy, they must pop a customer from the queue and proceed to help them for the customer's randomly generated transaction time. Once the vector has been parsed and every teller has been addressed, the for loop finishes and the start time is incrmented by one minute. This method proves the most fruitful and realistic in results. Essentially, the actions of the tellers are not too affected by the tick of the simulation time's clock. Each teller can "move" independently within the time before another tick and carried out.

System Tests

In order to test whether this program is usable on various systems, it will be tested on a Unix/Linux system alongside having been developed on a Windows system.

The results of the test are as predicted. The following image shows the result when the inputs are: 10 minute simulation time, 100 customers, 10 tellers, and transaction times of 1 minute for all customers:



Acceptance Test

Here, the data collected and program are evaluated in order to determine of the requests of the client were met.

As seen in the data collected above, the program was successful in simulating a Bank Teller interaction system. Multiple cases such as random transaction times and different numbers of tellers are all capable of being input into the system and analyzed. Every event in the system is shown as output to the user in order to clarify what is going on between the Tellers and Customers. The end results are calculated to further make the output readable and usable in any other calculations. Alongside the readability of the output, implementation of more events

are possible as the modules are not too dependent upon one another. The crux of the program, however, depends on the close relationship Tellers and Customers have. Despite being separate entities, or objects, a Teller's busy time is dependent upon the Customer's transaction time. On the other hand, a Customer's wait time is heavily influenced by the amount of Tellers provided by the bank.

This program should be sufficient for use in order to estimate various factors that affect the services of a bank to its customers. The program is also intuitively flexible and integratable in systems that are similar to those of banks and implement a queue data structure, such as a restaurant where customers are seated based on availability.