CS 158 Lab 13 March 16, 2010
Barbershop Simulation

1. In this assignment, you will build a simulation of a simple barbershop that has one barber who works for ten hours a day (600 minutes). Customers arrive randomly. More precisely, the time between customer arrivals is a random number between 10 and 30 minutes. Each customer requires a haircut that takes a random amount of time, also between 10 and 30 minutes. Because of the randomness involved, sometimes when the barber finishes with a customer, there will not be anyone waiting (no one has arrived yet). That is called "slack time". At other times, there will be people waiting when the barber finishes with a customer. We will build a simulation in a simple way that is not the best. But it will be easy to compute the total slack time during the day (total time the barber is idle). We will also print the number of customers waiting at various times, although not in a very efficient way. The program will print the times when a customer begins receiving a haircut, the time the customer leaves the shop, the total number of customers served, and the total slack time for the barber. This will be the assignment for today and Thursday.

2. Create a new project called Barbershop. Create a package called **containers**, and copy the author's **Queue** interface and **ListQueue** class into that package from the folder at http://faculty.valpo.edu/jcaristi/cs158/queues. You will also need the author's two exception classes in that package. Make sure everything works.

3. Create another package called **barbershop**. We will need customer objects, so create a class in this package called **Customer**. Each customer will have a name that is a **String** and an arrival time.

4. Also in the **barbershop** package create a **Barber** class. The barber object must have its own random number generator, and a method that returns a random service time between 10 and 30 minutes. Write a good test class for this class.

5. In the **barbershop** package create a **Shop** class that contains a main method. This is the class that will be responsible for setting everything up and running the simulation. It will have to create the barber and a queue for the customers to wait in. It will need its own random number generator. It will have to keep track of the current time of day (starting at zero).

6. In the **Shop** class create a method that returns the number of minutes until the next customer arrives. This is to be a random number between 10 and 30 minutes using the random number generator from the **Shop** class. Write a good test for this method.

7. In the **Shop** class add a method that will enqueue all of the customers that could possibly arrive during the 10 hour day. This is not exactly the right way to do the simulation, but it is easy, and will give us the answers we need for this lab. As you enqueue them, you will need to (of course) create them, give them unique names (hint: involve the current time that this method will keep track of), and set their arrival times. Write a good test for this method. When this method is finished, you

should have a number of customers in the queue, and their arrival times should be increasing, and there should not be more than 30 minutes or less than 10 minutes between them. (Aside: notice that on the average, you're creating a customer every 20 minutes, for about 3 per hour, so about 30 per day).

8. The main method will actually handle the whole simulation. It should instantiate a **Shop** object, enqueue all of the customers that could arrive in a day, and then enter a loop that processes customers until the time of day is beyond the end of the day. Processing a customer is complicated, and will require its own method. But before you can process a customer, you have to make sure there is one waiting. Even though our queue has a lot of customers in it, it's very possible that at the current time of day in the simulation, the next customer has not arrived yet. Also, you do not want to process a customer if the queue is empty. It is possible for this to happen (although not likely) depending on how you wrote the method that enqueues all of the customers and how efficient (lucky) the barber happens to be. So you should make sure the queue is not empty before you try to process a customer. And if it is empty, set the time of day to the end of the day so that your main loop will end.

9. Add the **processACustomer** method to the **Shop** class. This method will take the next customer in line, figure out whether it has arrived yet, and if not, advance the time of day to the time the customer actually arrived. Either way, this method is responsible for advancing the time of day for the amount of time this customer takes in getting a haircut (which the barber knows). Write a good test for this method. To do this, load the queue with known customers and times, and then assert things about the queue and the time of day before and after running the method. Be sure to exercise both possibilities concerning whether the next customer in the queue has actually arrived before the current time or not.

10. Each time a customer begins a haircut, print a line that gives the current time and the customer name and says that the haircut is beginning. Each time the barber finishes a haircut, print a line that gives the current time and the customer name and says that that customer has left the shop. You may want to use a helper method for this. At this point you should be able to run the main method and see the printed list of times, customer names, and actions (starting haircut or leaving the shop).

11. Modify the program so that after everything else is printed, it prints the total number of customers served. Run the program repeatedly to see whether the results make sense.

12. Modify the program so that after everything else is printed, it prints the total amount of time that the barber was idle (slack time).

13. Add a method to **Shop** called **countCustomersWaiting** that will return the number of customers who are waiting for a haircut at the current time. This is NOT the number of customers in the queue, because many of them have not arrived yet. So to do this, use a temporary queue to transfer customers into as you determine if they have arrived prior to the current time. Reset the actual queue. Then each time a customer is processed, print the number of customers waiting.