Queues for Simulation

From cars lined up at a tollbooth to print jobs waiting for a printer, queues abound in every day life. The next example uses a queue to model and simulate a customer waiting line at an ATM machine.

During lunch hour, the ATM machine in a large office complex is in heavy demand. Customers complain that the waiting time is much too long. The local bank is considering the addition of a second machine. But first, the bank needs a few statistics to justify the cost.

Problem
Simulate a waiting line at the ATM machine for a period of one hour. Make the following assumptions:

- With equal probability, a customer spends:
  one minute,
  two minutes , or
  three minutes
  at the ATM machine.
  .
- During any minute:
  no customers arrive (50% chance),
  one customer arrives (40% chance), or
  two customers arrive (10% chance).

At the end of an hour, display the following summary statistics:

- the number of customers served, i.e., the number who accessed the ATM machine,
- the average time a that customer waits in line before being served, and
- the number of customers that remain in the waiting line at the end of the simulation.

Assume that the ATM is available when the simulation begins and that no customers are waiting.

Solution:
Before considering an algorithm that simulates the comings and goings of customers at an ATM machine, we design a class that models an ATM customer.

A customer knows his/her arrival time and how much time he/she spends making an ATM transaction. The following class encapsulates a customer.

```
1.  import java.util.*;

2.  public class Customer
3.  {
4.      private int arrivalTime;        // 0..59, the minute of the hour when a customer arrives
5.      private int serviceTime;        // 1, 2, or 3 minutes

6.      public Customer()               // default constructor
7.      {
8.          arrivalTime = 0;
9.          serviceTime = 0;
10.     }

11.     public Customer(int arrTime)            // one argument constructor
12.     {
13.         arrivalTime = arrTime;
```

```
14.          Random rand = new Random();
15.          serviceTime = rand.nextInt(3)+1;  // 1, 2, or 3 minutes
16.      }

17.      public void setArrivalTime(int arrTime)
18.      {
19.          arrivalTime = arrTime;
20.      }

21.      public int getArrivalTime()
22.      {
23.          return arrivalTime;
24.      }

25.      public void setServiceTime(int ser)
26.      {
27.          serviceTime = ser;
28.      }

29.      public int getServiceTime()
30.      {
31.          return serviceTime;
32.      }
33. }
```

The algorithm that simulates an ATM waiting line uses a loop that ticks through a sixty minute simulation.

```
For each minute from 0 through 59
{
        Determine the number of new customers arriving:  0, 1, or  2;
        For each new customer
                Place the new customer in the queue;

        If there are customers waiting and the ATM is available
        {
                Remove  a customer from the queue;
                Increment the number of customers served;
                Add to the total waiting time the waiting time of the current customer;
                Update the time the ATM is next available;
        }
}
Print the summary statistics;
```

The following class implements this algorithm.

```
33. import java.util.*;

34. public class ATMSimulation
35. {
36.     Customer customer;
37.     int ATMisAvailable;  // time the ATM is next available
38.     int numArrivals;   // number of arrivals in any minute
39.     Queue<Customer> queue;

40.     // statistics
41.     int totalWaitingTime;  // for all customers
```

```
42.     int numCustomersServed;

43.     public ATMSimulation()  // default constructor
44.     {
45.         ATMisAvailable = 0; // assume the ATM is  available at time 0
46.         numArrivals = 0;
47.         totalWaitingTime = 0;
48.         numCustomersServed = 0;
49.         queue = new Queue<Customer>(200);
50.     }

51.     private int getArrivals()
52.     // generate a random integer in the range 0..9
53.     //  if the random integer is 0,1,2,3,or 4, then no arrivals ( 50% chance)
54.     //  if the random integer is 5,6,7, or 8, then 1 arrival (40 % chance)
55.     //  if the random integer is 9,  then 2 arrivals (10% chance)
56.     {
57.         Random rand = new Random();
58.         int randomInteger = rand.nextInt(10); // 0..9
59.         if ( randomInteger <= 4) // 0..4
60.             return 0;  // 50% chance of a single arrival
61.         if ( randomInteger <= 8) // 5..8
62.             return 1;  // 40% chance of a single arrival
63.         return 2; // 10% chance of 2 arrivals
64.     }

65.     private void displayStatistics()
66.     {
67.         System.out.println("Number of customers served "+ numCustomersServed);
68.         System.out.println("Average wait is about "+
69.                                 totalWaitingTime/numCustomersServed + "          minutes");
70.         System.out.println("Customers left in queue: "+ queue.size());
71.     }

72.     public void simulate()
73.     {
74.         for (int time = 0; time < 60; time++)                    // for each minute
75.         {
76.             numArrivals = getArrivals();                 // how many customers arrive?
77.             for (int i = 1; i <= numArrivals; i++)       // place each arrival into the queue
78.                 queue.insert( new Customer(time));
79.             if (!queue.empty() && ATMisAvailable <= time)
80.             {
81.                 customer = queue.remove(); // remove the next customer from the line
82.                 // Determine the next time that the ATM is available: current time+ service time
83.                 ATMisAvailable = time + customer.getServiceTime();
84.                  // how long did this customer wait?
85.                 int timeCustomerWaited = time - customer.getArrivalTime();
86.                 totalWaitingTime += timeCustomerWaited;   // add customer's wait to total
87.                 numCustomersServed++;
88.             }
89.         }
90.         displayStatistics();
91.     }

92.     public static void main(String[] args)
```

```
93.     {
94.         ATMSimulation atmSim = new ATMSimulation();
95.         atmSim.simulate();
96.     }
97. }
```

Output

Running the application three times produced the following output:

```
Number of customers served  30
Average wait is about 5 minutes
Customers left in queue: 16

Number of customers served 29
Average wait is about 8 minutes
Customers left in queue: 13

Number of customers served 32
Average wait is about 6 minutes
Customers left in queue: 6
```

Notes:
The application simulates the waiting line for each minute of an hour.

- During each minute, customers can arrive as well as gain access to the ATM machine.
- At the end of the 60 minute interval, a call to the helper method displayStatistics() prints the summary statistics.

The getArrivals() method

- One of the assumptions of the simulation is that the number of arrivals during any particular minute is 0, 1, or 2 customers with probabilities of 0.50, 0.40, and 0.10, respectively. The method first generates a random integer between 0 and 9 inclusive.
- The probability that this random integer is 0, 1, 2, 3, or 4 is 0.50. ---No arrivals
- The probability that the random number is 5, 6 ,7, or 8 is 0.40. --- One arrival
- The probability that the number is 9 is 0.10. – 2 arrivals

- If the random number is 0, 1, 2, 3 or 4, we assume that there are no arrivals. There is a 50% chance that this happens.
- If the number is 5, 6, 7, or 8 we assume that there is a single arrival. This happens 40% of the time.
- And, finally, if the random number is 9, we assume that there are two arrivals.