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 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.

```java
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;
```
Random rand = new Random();
serviceTime = rand.nextInt(3) + 1; // 1, 2, or 3 minutes
}
	public void setArrivalTime(int arrTime)
	{
		arrivalTime = arrTime;
	}
	public int getArrivalTime()
	{
		return arrivalTime;
	}
	public void setServiceTime(int service)
	{
		serviceTime = service;
	}
	public int getServiceTime()
	{
		return serviceTime;
	}
}

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.

import java.util.*;

public class ATMSimulation
{
    Customer customer;
    int ATMisAvailable; // time the ATM is next available
    int numArrivals; // number of arrivals in any minute
    Queue<Customer> queue;

    // statistics
    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. }
55. private void displayStatistics()
56. {
57.    System.out.println("Number of customers served "+ numCustomersServed);
58.    System.out.println("Average wait is about "+
59.        totalWaitingTime/numCustomersServed + " minutes");
60.    System.out.println("Customers left in queue: "+ queue.size());
61. }
62. public void simulate()
63. {
64.    for (int time = 0; time < 60; time++) // for each minute
65.    {
66.        numArrivals = getArrivals(); // how many customers arrive?
67.        for (int i = 1; i <= numArrivals; i++) // place each arrival into the queue
68.            queue.insert( new Customer(time));
69.        if (!queue.empty() && ATMisAvailable <= time)
70.        {
71.            customer = queue.remove(); // remove the next customer from the line
72.            // Determine the next time that the ATM is available: current time + service time
73.            ATMisAvailable = time + customer.getServiceTime();
74.            // how long did this customer wait?
75.            int timeCustomerWaited = time - customer.getArrivalTime();
76.            totalWaitingTime += timeCustomerWaited; // add customer's wait to total
77.            numCustomersServed++;
78.        }
79.        }
80.     displayStatistics();
81. }
82. public static void main(String[] args)
ATMSimulation atmSim = new ATMSimulation();
atmSim.simulate();