Homework Turnin

Account:	6G_06 (rgalanos@fcps.edu)
Section:	6G
Course:	TJHSST APCS 2016-17
Assignment:	06-09
Receipt ID:	347ee07be6de516131cbd4f8654cf5a6

Turnin Successful!

The following file(s) were received:

McRonaldPQ.java (6981 bytes)

```
1. //name :   date:
2.
3. import java.util.*;
4. public class McRonaldPQ
5. {
6.    public static final int TIME = 1079;  //18 hrs * 60 min
7.    public static void main(String[] args)
8.    {
9.       int[] numberCustomers = {0, 0, 0 ,0};
10.      int[] totalWait = {0, 0, 0 ,0};
11.      int[] longestWait = {0, 0, 0 ,0};
12.      int[] servingTime = {100, 100, 100};
13.      int[] count = {0, 0, 0};
14.
15.      PriorityQueue<Customer> customers = new PriorityQueue<Customer>();
16.      Queue<Customer> service1 = new LinkedList<Customer>();
17.      Queue<Customer> service2 = new LinkedList<Customer>();
18.      Queue<Customer> service3 = new LinkedList<Customer>();
19.
20.      for(int i=0; i<TIME; i++)
21.      {
22.         if(Math.random()<0.5)
23.         {
24.            Customer c = new Customer(i);
25.            customers.add(c);
26.            numberCustomers[12-c.getGrade()]++;
27.         }
28.
29.         if(!service1.isEmpty())
30.         {
31.            count[0]++;
32.            if(servingTime[0]==count[0])
33.            {
34.               Customer c = service1.remove();
35.               int x = i - c.getTime();
36.               totalWait[12-c.getGrade()]+=x;
37.               if(x>longestWait[12-c.getGrade()])
38.                  longestWait[12-c.getGrade()] = x;
39.            }
40.         }
41.         if(!service2.isEmpty())
42.         {
43.            count[1]++;
44.            if(servingTime[1]==count[1])
45.            {
46.               Customer c = service2.remove();
47.               int x = i - c.getTime();
48.               totalWait[12-c.getGrade()]+=x;
49.               if(x>longestWait[12-c.getGrade()])
50.                  longestWait[12-c.getGrade()] = x;
```

```
            }
         }
         if(!service3.isEmpty())
         {
            count[2]++;
            if(servingTime[2]==count[2])
            {
               Customer c = service3.remove();
               int x = i - c.getTime();
               totalWait[12-c.getGrade()]+=x;
               if(x>longestWait[12-c.getGrade()])
                  longestWait[12-c.getGrade()] = x;
            }
         }


         if(service1.isEmpty()&&!customers.isEmpty())
         {
            service1.add(customers.remove());
            servingTime[0] = (int)(Math.random()*6+2);
            count[0] = 0;
         }
         if(service2.isEmpty()&&!customers.isEmpty())
         {
            service2.add(customers.remove());
            servingTime[1] = (int)(Math.random()*6+2);
            count[1] = 0;
         }
         if(service3.isEmpty()&&!customers.isEmpty())
         {
            service3.add(customers.remove());
            servingTime[2] = (int)(Math.random()*6+2);
            count[2] = 0;
         }

         //System.out.print(i+": ");
       //  display(customers);
      //     display(merge(service1, service2, service3, customers));
      }

      int counter = 1079;
      while(!(customers.isEmpty()&&service1.isEmpty()&&service2.isEmpty()&&service3.isEmpty()))
      {
         if(!service1.isEmpty())
         {
            count[0]++;
            if(servingTime[0]==count[0])
            {
               Customer c = service1.remove();
               int x = counter - c.getTime();
               totalWait[12-c.getGrade()]+=x;
               if(x>longestWait[12-c.getGrade()])
                  longestWait[12-c.getGrade()] = x;
            }
         }
         if(!service2.isEmpty())
         {
            count[1]++;
            if(servingTime[1]==count[1])
            {
               Customer c = service2.remove();
               int x = counter - c.getTime();
               totalWait[12-c.getGrade()]+=x;
               if(x>longestWait[12-c.getGrade()])
                  longestWait[12-c.getGrade()] = x;
            }
         }
         if(!service3.isEmpty())
         {
            count[2]++;
            if(servingTime[2]==count[2])
            {
               Customer c = service3.remove();
               int x = counter - c.getTime();
               totalWait[12-c.getGrade()]+=x;
               if(x>longestWait[12-c.getGrade()])
                  longestWait[12-c.getGrade()] = x;
            }
         }

```

```
132.          if(service1.isEmpty()&&!customers.isEmpty())
133.          {
134.             service1.add(customers.remove());
135.             servingTime[0] = (int)(Math.random()*6+2);
136.             count[0] = 0;
137.          }
138.          if(service2.isEmpty()&&!customers.isEmpty())
139.          {
140.             service2.add(customers.remove());
141.             servingTime[1] = (int)(Math.random()*6+2);
142.             count[1] = 0;
143.          }
144.          if(service3.isEmpty()&&!customers.isEmpty())
145.          {
146.             service3.add(customers.remove());
147.             servingTime[2] = (int)(Math.random()*6+2);
148.             count[2] = 0;
149.          }
150.
151.        //  System.out.print(counter+": ");
152.        //  display(customers);
153.       //      display(merge(service1, service2, service3, customers));
154.          counter++;
155.       }
156.
157.       System.out.println("Customer\tTotal Served\tLongest Wait\tAverage Wait");
158.       System.out.println("Senior\t\t" + numberCustomers[0] + "\t\t\t" + longestWait[0] + "\t\t\t" + (double)tota
159.       System.out.println("Junior\t\t" + numberCustomers[1] + "\t\t\t" + longestWait[1] + "\t\t\t" + (double)tota
160.       System.out.println("Sophomore\t" + numberCustomers[2] + "\t\t\t" + longestWait[2] + "\t\t\t" + (double)tot
161.       System.out.println("Freshman\t" + numberCustomers[3] + "\t\t\t" + longestWait[3] + "\t\t\t" + (double)tota
162.
163.    }
164.    public static Queue<Integer> merge(Queue<Integer> a, Queue<Integer> b, Queue<Integer> c, Queue<Integer> d)
165.    {
166.       Queue<Integer>  temp = new LinkedList<Integer>();
167.       Queue<Integer>  temp1 = new LinkedList(a);
168.       Queue<Integer>  temp2= new LinkedList(b);
169.       Queue<Integer>  temp3= new LinkedList(c);
170.       Queue<Integer>  temp4 =new LinkedList(d);
171.
172.
173.       while(!temp1.isEmpty())
174.          temp.add(temp1.remove());
175.       while(!temp2.isEmpty())
176.          temp.add(temp2.remove());
177.       while(!temp3.isEmpty())
178.          temp.add(temp3.remove());
179.       while(!temp4.isEmpty())
180.          temp.add(temp4.remove());
181.
182.       return temp;
183.    }
184.    public static void display(Queue<Integer> q)
185.    {
186.       System.out.println(q);
187.    }
188. }
189. class Customer implements Comparable<Customer>
190. {
191.    int grade;
192.    int time;
193.    public Customer(int i)
194.    {
195.       int x = (int)(Math.random()*4+1);
196.       switch(x)
197.       {
198.          case 1: grade = 12;
199.             break;
200.          case 2: grade = 11;
201.             break;
202.          case 3: grade = 10;
203.             break;
204.          case 4: grade = 9;
205.             break;
206.       }
207.       time = i;
208.    }
209.
210.    public int getGrade()
211.    {
212.       return grade;
```

```
213.     }
214.     public int getTime()
215.     {
216.        return time;
217.     }
218.     public int compareTo(Customer c)
219.     {
220.        return c.getGrade() - grade;
221.     }
222. }
223.
224.
225.
```