CS 315: Program 3 Worth: 10 points Handed out: Wednesday, October 29, 2014

Due: Wednesday, November 12th

Using a Queue to simulate a Ticket Counter

The purpose of this program is to understand how queues can be used to simulate a ticket counter.

Read the Handout (pp 107 -112 from Lewis & Chase]

[p. 107]: Read carefully the assumptions about the simulated ticket counter.

[p. 112]: Note also that the manager wants to keep the total time less than 7 minutes.

**Part I: Understanding the Code : Worth 3 points**

1) a) Download the **TicketCounterEx** program and run it. Provide the output here:

|  |
| --- |
| Number of cashiers: 1  Average time: 5317  Number of cashiers: 2  Average time: 2325  Number of cashiers: 3  Average time: 1332  Number of cashiers: 4  Average time: 840  Number of cashiers: 5  Average time: 547  Number of cashiers: 6  Average time: 355  Number of cashiers: 7  Average time: 219  Number of cashiers: 8  Average time: 120  Number of cashiers: 9  Average time: 120  Number of cashiers: 10  Average time: 120 |

b) According to this simulation, how many cashiers are needed if the manager wants to keep the total average time less than 7 minute? \_\_\_\_\_\_6\_\_\_\_\_\_\_\_\_\_

c) less than 4 minutes? \_\_\_\_\_\_\_\_\_\_\_7\_\_\_\_\_\_\_\_\_

2) Look at the implementation code for the Queue. Explain how this implementation differs from the one that we used in class.

|  |  |  |
| --- | --- | --- |
|  | ADT | Implementation Code |
| Given Program: | QueueADT | LinkedQueue |
| Our TextBook | QueueInterface : p. 415 | QueueReferencedBased: pp 418 - 419 |

Answer:

ADT

The queue interface of the book is developed to support all three queue inmplementations. That is: the linked queue, the array based and the ADT list while the one in the book is that for the specific program, which is a linked queue,and does not require no input from the client. The method for the enqueue declared in the books interface has a data type of type Object opposed to the one of the program that is developed using generics. The reason why is that arrays can't be instatiated as a generic type but it can be done only by instatiating it as a Object data type and then typed cast it to a generic type. The declaration of the methods in the interface of the program is strictly for the use of the program. In fact, their is no client interaction with the program, just strict output. for this reason their are no deQueueAll() and peek() method on the interface in contrast to the books interface. Another difference, the book interface uses exceptions while in the program there are none. Maybe that is so, because there is no risk that the client can perform ellegal operations since there is no input or data manipulation to be performed.

The implementation

In the implementation the most stryking difference is that in the program the linked queue in the book is a circular queue while the one in the program is a linear queue

**Part II: Improving the Code. [Worth 7 points]**

**The program can be improved**:

The output can be printed in columns like the following:

|  |
| --- |
| Number of Cashiers Average Time (in minutes)  1 5317  .  .  10 120 |

* **[2 pts]Write Code so that the program writes the output more compactly, r**un the program with the improved formatting, and provide the output; [Hint: Lab1 from CS221 reviews how to use System.outprintf with the appropriate escape sequences such as:

%s Print the next item as a string.

%n Advance to a new line (no item is processed)]

|  |
| --- |
| /\*\*  \* TicketCounter demonstrates the use of a queue for simulating a waiting line.  \*  \* @author Dr. Lewis  \* @author Dr. Chase  \* @version 1.0, 08/12/08  \*/  public class TicketCounter  {  final static int PROCESS = 120;  final static int MAX\_CASHIERS = 10;  final static int NUM\_CUSTOMERS = 100;  public static void main ( String[] args)  {  Customer customer;  LinkedQueue<Customer> customerQueue = new LinkedQueue<Customer>();  int[] cashierTime = new int[MAX\_CASHIERS];  int totalTime, averageTime, departs;        // printing the output  System.out.printf("%20s %30s %n","Number of Cashiers","Average Time (in minutes)");  /\*\* process the simulation for various number of cashiers \*/  for (int cashiers=0; cashiers < MAX\_CASHIERS; cashiers++)  {  /\*\* set each cashiers time to zero initially\*/  for (int count=0; count < cashiers; count++)  cashierTime[count] = 0;  /\*\* load customer queue \*/  for (int count=1; count <= NUM\_CUSTOMERS; count++)  customerQueue.enqueue(new Customer(count\*15));  totalTime = 0;  /\*\* process all customers in the queue \*/  while (!(customerQueue.isEmpty()))  {  for (int count=0; count <= cashiers; count++)  {  if (!(customerQueue.isEmpty()))  {  customer = customerQueue.dequeue();  if (customer.getArrivalTime() > cashierTime[count])  departs = customer.getArrivalTime() + PROCESS;  else  departs = cashierTime[count] + PROCESS;  customer.setDepartureTime (departs);  cashierTime[count] = departs;  totalTime += customer.totalTime();  }  }  }  /\*\* output results for this simulation \*/  averageTime = (totalTime / NUM\_CUSTOMERS)/60;  System.out.printf ("%10d %30d %n",(cashiers+1),averageTime);    }  }  } |

2) [1 pt] We want to see how changing the number of customers in the queue at the beginning affects the simulation.

a) Double the number of customers in the queue at the beginning. Provide the **output h**ere and write an **explanation of what happens**:

|  |
| --- |
| Number of Cashiers Average Time (in minutes)  1 176  2 76  3 43  4 26  5 16  6 10  7 5  8 2  9 2  10 2 |

Explanation:

We see that the values from 1 to 7 the total time is preatty much doubled while for the rest they remain the same. That suggesting that from more then 8 cashiers their will be no improvment in the service

3) [2 pts] We would like to see how changing the assumption that customers arrive every 15 seconds affects the simulation.

a) Keep the number of customers doubled, but now change the simulation so that a new customer arrives every 10 seconds. Provide the **output** here and write an **explanation of what happens**:

Number of Cashiers Average Time (in minutes)

1 184

2 84

3 51

4 34

5 24

6 18

7 13

8 10

9 7

10 5

Explanation:

We see that from cashiers from 1 to 8 the values have an additional 8 minutes added to the total from the previus simulation while from 9 and 10 we see that the the "just-in-time" effect is vanished. So the customers have always a wait period before they can be served

b) Keep the number of customers doubled, but now change the simulation so that a new customer arrives every 20 seconds. Provide the **output** here and write an **explanation of what happens**:

Number of Cashiers Average Time (in minutes)

1 167

2 68

3 34

4 18

5 8

6 2

7 2

8 2

9 2

10 2

Explanation:

We see that from cashiers 1 to 6 the total amount of time to serve the customers have decreased while for cashiers 7 and up their will be no improvement on the service

c) Notice that each change in the arrival time of the customers required that you change code within the program. This is NOT good programming practice. **Change the code so that the change in the arrival** time is NOT in the code, but instead in a constant at the beginning of the program that you can change.

**Show all changes in your code:**

/\*\*

\* TicketCounter demonstrates the use of a queue for simulating a waiting line.

\*

\* @author Dr. Lewis

\* @author Dr. Chase

\* @version 1.0, 08/12/08

\*/

public class TicketCounter

{

final static int PROCESS = 120;

final static int MAX\_CASHIERS = 10;

final static int NUM\_CUSTOMERS = 200;

final static int ARRIVAL\_TIME = 20;

public static void main ( String[] args)

{

Customer customer;

LinkedQueue<Customer> customerQueue = new LinkedQueue<Customer>();

int[] cashierTime = new int[MAX\_CASHIERS];

int totalTime, averageTime, departs;

// printing the output

System.out.printf("%20s %30s %n","Number of Cashiers","Average Time (in minutes)");

/\*\* process the simulation for various number of cashiers \*/

for (int cashiers=0; cashiers < MAX\_CASHIERS; cashiers++)

{

/\*\* set each cashiers time to zero initially\*/

for (int count=0; count < cashiers; count++)

cashierTime[count] = 0;

/\*\* load customer queue \*/

for (int count=1; count <= NUM\_CUSTOMERS; count++)

customerQueue.enqueue(new Customer(count\*ARRIVAL\_TIME));

totalTime = 0;

/\*\* process all customers in the queue \*/

while (!(customerQueue.isEmpty()))

{

for (int count=0; count <= cashiers; count++)

{

if (!(customerQueue.isEmpty()))

{

customer = customerQueue.dequeue();

if (customer.getArrivalTime() > cashierTime[count])

departs = customer.getArrivalTime() + PROCESS;

else

departs = cashierTime[count] + PROCESS;

customer.setDepartureTime (departs);

cashierTime[count] = departs;

totalTime += customer.totalTime();

}

}

}

/\*\* output results for this simulation \*/

averageTime = (totalTime / NUM\_CUSTOMERS)/60;

System.out.printf ("%10d %30d %n",(cashiers+1),averageTime);

}

}

}

4) [2 pts] We will now change the simulation so that instead of customers taking on average 2 minutes, they instead take a value between 1 and 3 minutes. You will need to use a random integer. Define 2 constants, MIN = 60 and MAX = 180, and then define the processtime as a random integer between 60 and 240.

[Hint: To find an integer between 1 and 10, use the code

int num = (int)(10\*Math.random() + 1);

Change the code and run the program 3 times. Show the output of the 2 runs. Explain what is happening.

Process time(in seconds):69

Number of Cashiers Average Time (in minutes)

1 82

2 25

3 6

4 1

5 1

6 1

7 1

8 1

9 1

10 1

-------------------------------------------------------------------------------------------

Process time(in seconds):119

Number of Cashiers Average Time (in minutes)

1 166

2 67

3 34

4 17

5 8

6 1

7 1

8 1

9 1

10 1

Explanation:

In these simulation we see that we the process time gets larger the average time increases and the number of chashier needed to keep the process time under the 7 minutes and 4 minutes increasses. In the first simulation with a process time of 68 second we see that their is no bettering of service time after the number of cashiers are 4 while for the 119 average time there is no bettering of service after 6.