**CE 412 – PROJECT 2**

**“SIMULATION OF A GROCERY STORE”**

***Project Description***: A small grocery store has only one checkout counter. Customers arrive at this checkout counter at random from 1 to 8 minutes apart. The service times vary from 1 to 6 minutes. Analyze the system and answer the following questions through a simulation study. The distribution of inter arrival times and service times are illustrated in the tables below. Assume that the first customer arrives at the checkout counter at time *t=0* .

|  |  |
| --- | --- |
| **Distribution of time between arrivals** | |
| **Time between arrivals (mins)** | **Probability** |
| **1** | 1/8 |
| **2** | 1/8 |
| **3** | 1/8 |
| **4** | 1/8 |
| **5** | 1/8 |
| **6** | 1/8 |
| **7** | 1/8 |
| **8** | 1/8 |

|  |  |
| --- | --- |
| **Service Time Distribution** | |
| **Service time (mins)** | **Probability** |
| **1** | 1/10 |
| **2** | 1/5 |
| **3** | 3/10 |
| **4** | 1/4 |
| **5** | 1/10 |
| **6** | 1/20 |

***What do we need to find ?***

* What is the average time a customer spends in the system?
* What is the average waiting time of a customer?
* What is the average service time of a customer?
* What percentage of the customers wait at the checkout counter?
* What is the average waiting time of the customers who wait?
* What percentage of the time the checkout counter is idle?
* What is the average time between arrivals to the checkout counter?

***Let’s analyze the project code step by step;***

I have used 3 different methods in the project. Two of them are used in one method to provide random number for service time and arrival time.

public static int randomArrivalTime() {

// Arrival times have the same probability

// So, we don't need to check the probability to create random variable

int getRnd2 = ((int) (Math.random() \* (8))) + 1;

return getRnd2;

}

This method is creating random number for arrival time of customers. You can see on the above all service times have the same probability that is **0.125** so I don’t include the probabilities in this method. It is enough to create random number between 1 and 8. (Math.random() \* (8)) is creating random variable between 0 and 7 so I added +1. The result will occur numbers between 1 and 8.

public static int randomServiceTime()

{

// Probability of Service Time Distribution. Firstly, to calculate Icm(EKOK) probability of service time

// 30--->1 [1-30]--->1

// 60--->2 [31-90]--->2

// 90--->3 [91-180]--->3

// 75--->4 [181-255]--->4

// 30--->5 [256-285]--->5

// 15--->6 [286-300]--->6

int getRnd = ((int) (Math.random()\*(299))) + 1;

if (getRnd<=30)

{

getRnd=1;

}

else if (getRnd>=31 && getRnd<=90)

{

getRnd=2;

}

else if (getRnd>= 91 && getRnd<=180)

{

getRnd=3;

}

else if (getRnd>= 181 && getRnd<=255)

{

getRnd=4;

}

else if (getRnd>=256 && getRnd<=285)

{

getRnd=5;

}

else if (getRnd >= 286 && getRnd<=300)

{

getRnd=6;

}

return getRnd; }

This method is creating random variable for service time. These random variables depend on the probability of each element. Firstly, I multiplied by 300 for each variable. Then, I have got 30 for first arrival time, and 60,90,75,30,15 respectively for each arrivals.

30 equals 1 If the value is between [1-30] then return 1

60 equals 2 If the value is between [31-90] then return 2

90 equals 3 If the value is between [91-180] then return 3

75 equals 4 If the value is between [181-255] then return 4

30 equals 5 If the value is between [256-285] then return 5

15 equals 6 If the value is between [286-300] then return 6

public static void calculate()

{

int numberOfCustomer = 100000000;

int[][] array = new int[numberOfCustomer][9];

array[0][0] = 1;

array[0][1] = 0;

array[0][2] = 0;

array[0][3] = SadifePrj2.randomServiceTime(); // to create random number for service time

array[0][4] = 0;

array[0][5] = array[0][4] - array[0][2];

array[0][6] = array[0][3] + array[0][4];

array[0][7] = array[0][6] - array[0][2];

array[0][8] = 0;

I have created a method which is named “calculate”. This method will print result of each question. I have created two dimensional array to hold related data. Each column of the array represents;

Index Array

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Customer** | **Interarrival Time (Minutes)** | **Arrival Time** | **Service Time (Minutes)** | **Time Service Begins** | **Waiting Time in Queue (Minutes)** | **Time  Service Ends** | **Time Customer Spends in System (Minutes)** | **Idle Time of Server (Minutes)** |
|
|
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** |

So, int[][] array = new int[numberOfCustomer][9];

I used to variable named by “numberOfCustomer”, it’ll occur rows “numberOfCustomer” times. Also, I defined the first row of the matrix according to rule of each element. These rules are defined as;

* Interarrival time and service time are being created randomly without using any rules.
* Value of “Arrival time” equals value of arrival time for previous customer plus interarrival time for the current customer.
* Value of “Time service begins” equals maximum value of time service end for previous customer and arrival time in the current customer.
* Values of “Waiting time in the queue” equals value of time service begins minus arrival time in the current position.
* Value of “Time customer spends in the system” equals value of time service end minus arrival time in the current position.
* Value of “Idle time of server” equals maximum value of zero and value of arrival time for current customer minus value of time service end for previous customer.

And continue to use the same rules inside of the “for loop”;

for (int i = 1; i < numberOfCustomer; i++)

{

array[i][0] = i + 1;

array[i][1] = SadifePrj2.randomArrivalTime();// to create random number for arrival time

array[i][2] = array[i][1] + array[i-1][2];

array[i][3] = SadifePrj2.randomServiceTime();

array[i][4] = Math.max(array[i][2],array[i-1][6]);

array[i][5] = array[i][4] - array[i][2];

array[i][6] = array[i][4] + array[i][3];

array[i][7] = array[i][6] - array[i][2];

array[i][8] = Math.max((array[i][2] - array[i-1][6]) , 0);

If the value doesn’t equal to zero, it means customers will wait on the queue.

if (array[i][5] != 0) {

numberOfCustomerWhoWait++;

totalTimeCustomersWaitInQue = totalTimeCustomersWaitInQue + array[i][5];

}

totalIdleTimeOfServer = totalIdleTimeOfServer + array[i][8];

totalServiceTime = totalServiceTime + array[i][3];

}

As a result of this code we’ll see the calculations on the screen. Some variables are created for calculations. Also, you can see the formulation for each question inside of the “print” code.

double totalTimeCustomersWaitInQue = 0;

double totalIdleTimeOfServer = 0;

double totalRunTime = 0;

double totalServiceTime = 0;

double sumOfTimesBetweenArrivals = array[0][3];

double numberOfCustomerWhoWait = 0;

System.out.println("Avg time Customer spend in the system : " + (totalTimeCustomersWaitInQue/numberOfCustomer+totalServiceTime/numberOfCustomer));

System.out.println("Avg waiting time of a customer : " + totalTimeCustomersWaitInQue/numberOfCustomer);

System.out.println("Avg service time of a customer : " + totalServiceTime/numberOfCustomer);

System.out.println("Percentage of the customers wait at the checkout counter: " + numberOfCustomerWhoWait/numberOfCustomer);

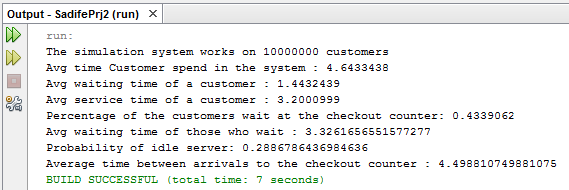
System.out.println("Avg waiting time of those who wait : " + totalTimeCustomersWaitInQue/numberOfCustomerWhoWait);

System.out.println("Probability of idle server: " + totalIdleTimeOfServer/array[numberOfCustomer-1][6]);

System.out.println("Average time between arrivals to the checkout counter : " + (double)(array[numberOfCustomer-1][2])/(numberOfCustomer-1));

}

***Output on the screen;***

******

* The result of “average service time” converges to 3.2.
* The result of “average time between arrivals” converges to 4.50.