

# Simulation & Modeling 2023-2024

---

LAB 2

# Teams and Materials

---

- Registration Form:

<https://docs.google.com/forms/d/e/1FAIpQLSfxvnL1dhnfCqZxv2n4YdHtLbNIuwQgKNfbvbDQj1Lr6OxiQ/viewform> Tasks Templates:

- Tasks Templates:

<https://drive.google.com/drive/folders/1XaQonB3ZhWntuWAKudi8mbhPiBwjCGyh?usp=sharing>

# Task (1) Delivery Rules

---

- Task support will be held next week
  - In all sections slots
- Task delivery after two weeks.
- Every team will be assigned a time slot for the task delivery. Teams should commit to their assigned time slot
- Any delay will not be accepted.
- Each team member should be there for the delivery at least once.

## Cheating Policy

- First Incident: -10% from the yearwork grades
- Second Incident: -50% from the yearwork grades
- Third Incident: -100% from the yearwork grades

# Multi-channel Queue

---

THE CALL CENTER

# Multi-channel Queue

---



**Baker**



**Able**

# Multi-channel Queue

---



# Multi-channel Queue

---



**Baker**



**Able**

# Multi-channel Queue

---





# Multi-channel Queue

---



**Baker**



**Able**

# Multi-channel Queue

---



# Multi-channel Queue

---



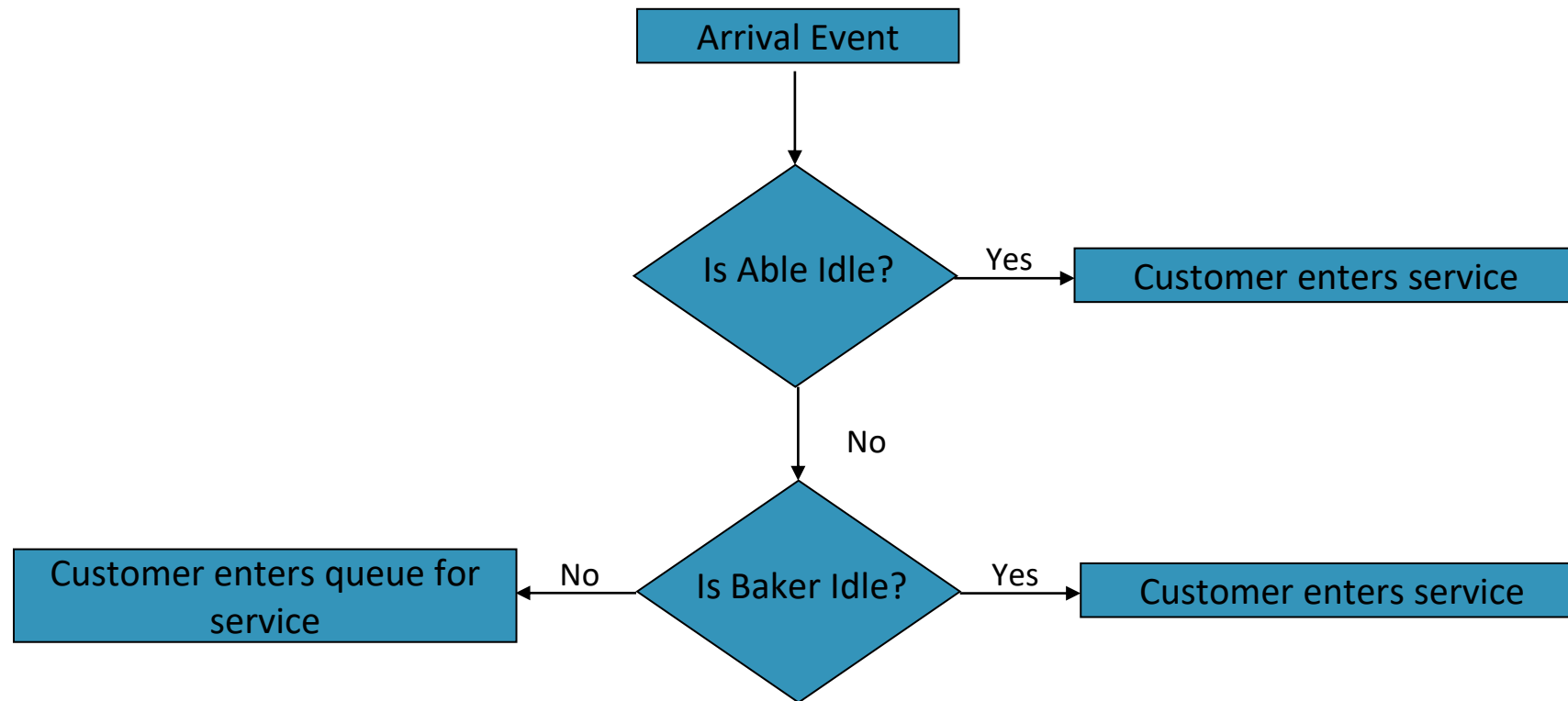
**Baker**



**Able**

# Multi-channel Queue - Arrival of a Customer

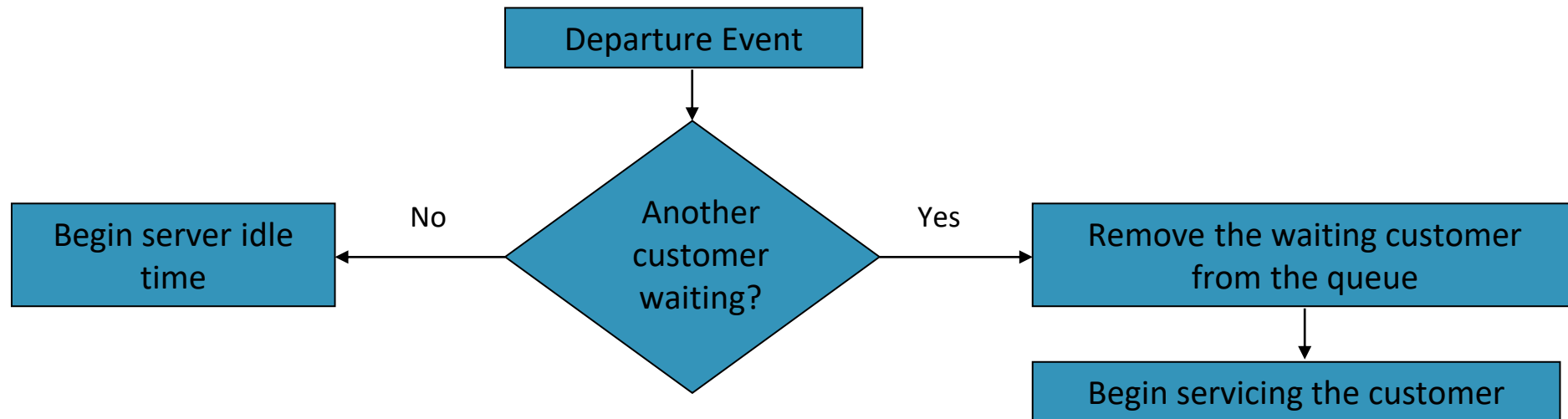
---



Unit-entering-system flow diagram

# Multi-channel Queue - Departure of a Customer

---



Service-just-completed flow diagram

# Multi-channel Queue

---

**We need to the system measures of system performance in terms of:**

- The efficiency of Able.
- The efficiency of Baker
- The average caller delay.

# Multi-channel Queue - System Design

---

Server Selection: [How to select a server when both servers are available?]

- Highest Priority
  - Each server should be assigned a priority
  - When a customer arrives, it enters the available server having the highest priority
- Random
  - Choose any server of the available servers
- Least utilization [Bonus]
  - When a customer arrives, calculate the utilization of the available servers
  - The customer enters the server having the least utilization, i.e. the server that worked the least amount of time since the system has started running.

# Multi-channel Queue - System Design

---

## System Variables

- Arrival Time(i) =  $T_i$
- Interarrival Time(i) =  $A_i = T_i - T_{i-1}$
- Service time(i) =  $S_i$
- Waiting Time(i) =  $D_i$
- Departure Time (i) =  $C_i = T_i + D_i + S_i$

**S and A are stochastic variables (generated according to probability distribution)**



# Multi-channel Queue - System Input

---

1. Inter-arrival time distribution
2. Number of servers
3. Service time distribution for each server
4. Server Selection method (priority, random, least utilization[BONUS])
5. Stopping Condition
  - Maximum Number of customers
  - Simulation end time

# Multi-channel Queue - System Output

---

Results table including these columns:

- a. Customer No
- b. Random Digit for inter-arrival time
- c. Inter-arrival time
- d. Arrival time (Clock Time)
- e. Random Digit for service duration
- f. Service duration
- g. Server Index
- h. Time Service Begins
- i. Time Service Ends (Departure)
- j. total delay time for this customer (Time in queue)

N.B: Assume Server 1 has higher priority

# Sample Testcase

# Multi-channel Queue - Sample Input

---

Interarrival Distribution of calls			
Interarrival Time	Probability	Cumulative probability	Rang
1	0.25	0.25	01-25
2	0.40	0.65	26-65
3	0.20	0.85	66-85
4	0.15	1	86-00

# Multi-channel Queue - Sample Input

---

Able's Service Time distribution			
Service Time	Probability	Cumulative probability	Range
2	0.30	0.30	1-30
3	0.28	0.58	31-58
4	0.25	0.83	59-83
5	0.17	1	84-00

# Multi-channel Queue - Sample Input

---

Baker's Service Time distribution			
Service Time	Probability	Cumulative probability	Range
3	0.35	0.35	1-35
4	0.25	0.60	36-60
5	0.20	0.80	61-80
6	0.20	1	81-00

# Multi-channel Queue - Sample Output

A	B	C	D	E	F	G	H	I	J	K	L
<i>Customer</i>	<i>Random Digits</i>	<i>Time between</i>	<i>Clock Time</i>	<i>Random Digits</i>	<i>Able</i>		<i>Baker</i>				
<i>No.</i>	<i>for Arrival</i>	<i>Arrivals</i>	<i>of Arrival</i>	<i>for Service</i>	<i>Time Service</i>	<i>Service</i>	<i>Time Service</i>	<i>Time Service</i>	<i>Service</i>	<i>Time Service</i>	<i>Time in</i>
					<i>Begins</i>	<i>Time</i>	<i>Ends</i>	<i>Begins</i>	<i>Time</i>	<i>Ends</i>	<i>Queue</i>
1	—	—	0	95	0	5	5				0
2	26	2	2	21				2	3	5	0
3	98	4	6	51	6	3	9				0
4	90	4	10	92	10	5	15				0
5	26	2	12	89				12	6	18	0
6	42	2	14	38	15	3	18				1
7	74	3	17	13	18	2	20				1
8	80	3	20	61	20	4	24				0
9	68	3	23	50				23	4	27	0
10	22	1	24	49	24	3	27				0
11	48	2	26	39	27	3	30				1
12	34	2	28	53				28	4	32	0
13	45	2	30	88	30	5	35				0
14	24	1	31	01				32	3	35	1
15	34	2	33	81	35	4	39				2
16	63	2	35	53				35	4	39	0
17	38	2	37	81	39	4	43				2
18	80	3	40	64				40	5	45	0
19	42	2	42	01	43	2	45				1
20	56	2	44	67	45	4	49				1
21	89	4	48	01				48	3	51	0
22	18	1	49	47	49	3	52				0
23	51	2	51	75				51	5	56	0
24	71	3	54	57	54	3	57				0
25	16	1	55	87				56	6	62	1
26	92	4	59	47	59	3	62				0
						56			43		11

# System Output - Performance Measures

---

## **Calculate the Measures of Performance for the system:**

- Average waiting time (in the queue).
- Maximum queue length.
- Probability that a customer wait in the queue.
  
- Per Server:
  - Average service time per server.
  - Utilization of each server
  - Probability that a server is in idle state.

**Do we need extra server? Why?!!**



# System Output - Performance Measures

---

Measure system performance by calculating the following variables:

$$\text{average waiting time} = \frac{\text{total time customers waited in queue}}{\text{total number of customers}}$$

$$\text{probability (wait)} = \frac{\text{number of customers who waited}}{\text{total number of customers}}$$

Maximum queue length during simulation runtime.

# System Output - Performance Measures Per Server

---

Measure servers performance by calculating the following variables:

$$\textit{probability of idle server}(x) = \frac{\textit{total idle time of server}(x)}{\textit{total run time of simulation}}$$

$$\textit{average service time}(x) = \frac{\textit{total service time}(x)}{\textit{total number of customers}(x)}$$

$$\textit{Utilization}(x) = \frac{\textit{Total Time the server } x \textit{ spends on calls}}{\textit{Total run time of the simulation}}$$

# Multi-channel Queue – Graph

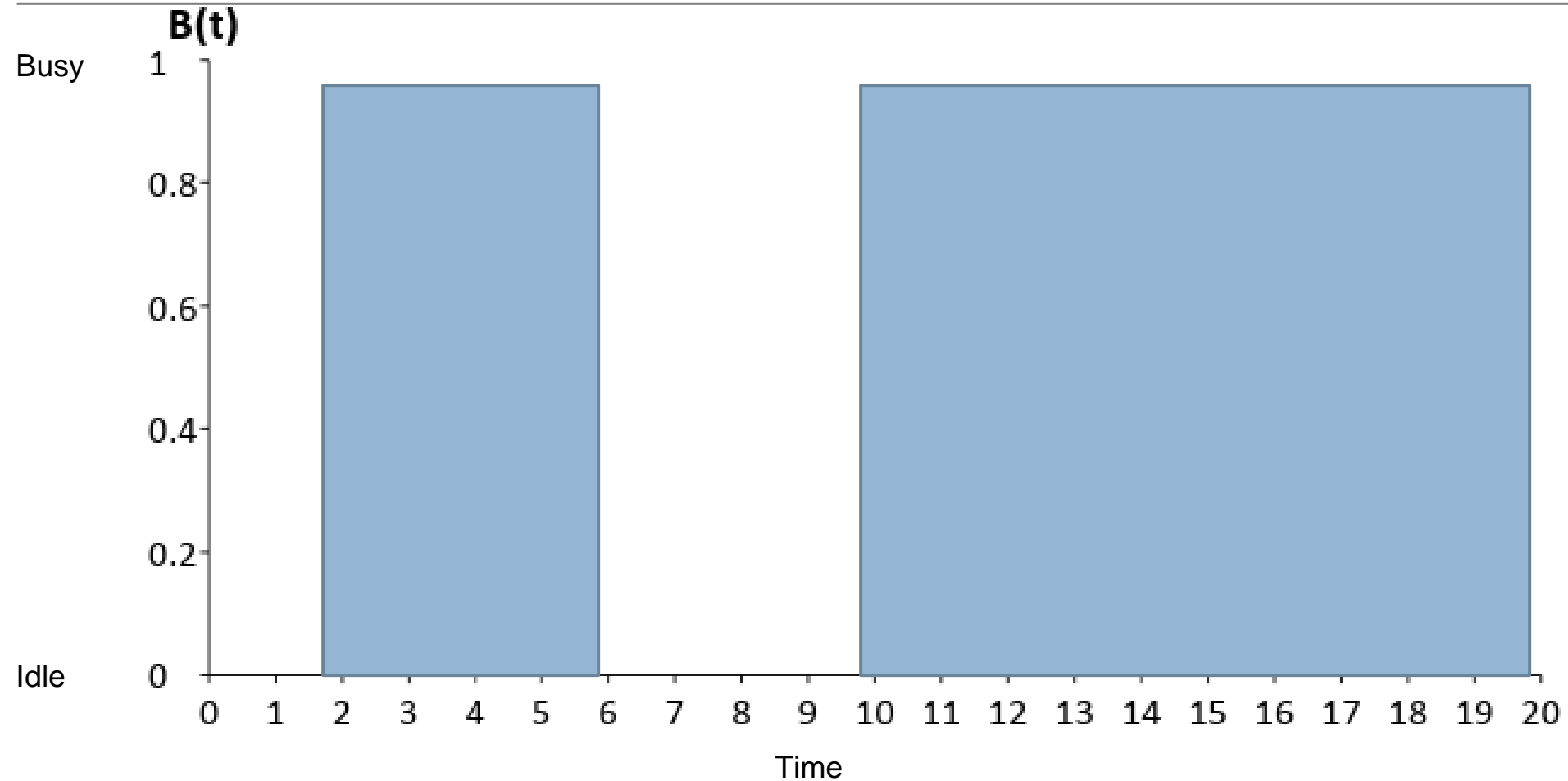
---

## Required Chart

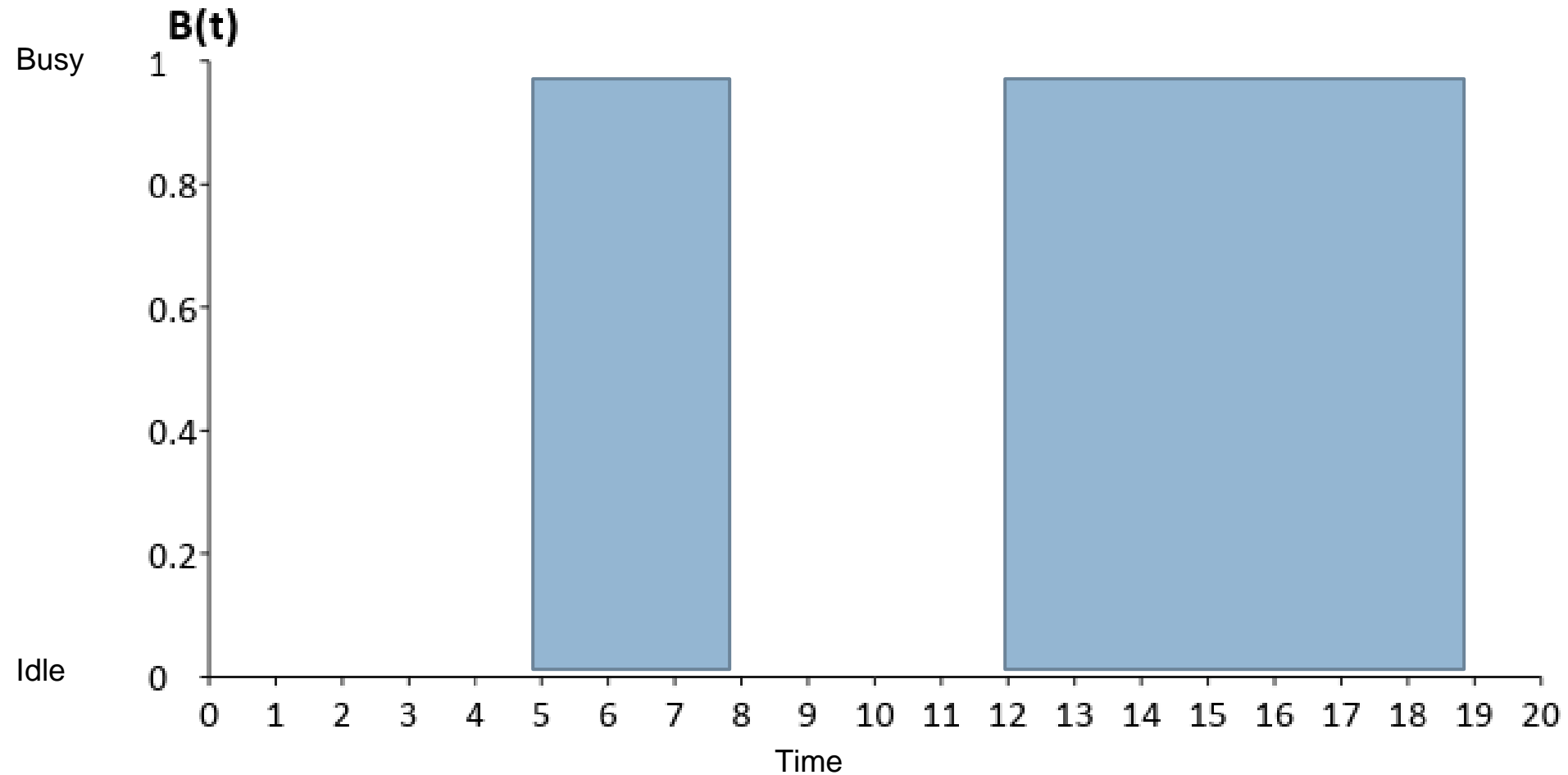
Server Busy Time [One for every server]

- X-axis : time
- Y- axis : it has a value of 1 if the server is busy or zero if the server is idle

# Server Busy Time – Server 1



# Server Busy Time – Server 2



# Task (1) Deliverables

---

1. The complete simulation table for 100 customers (refer to slide 22)
2. The performance parameters calculated from the simulation table (refer from slide 23 to 25)
3. The graph (refer from slide 26 to 28)

# More Notes

---

1. GUI is mandatory.
2. Reading data from files and view the data is mandatory.
3. Well OOP design is mandatory.
4. **Bonus:** Least Utilization

---

# Template Walkthrough



# Multi-channel Queue - Sample Input

---

Interarrival Distribution of calls			
Interarrival Time	Probability	Cumulative probability	Rang
1	0.25	0.25	01-25
2	0.40	0.65	26-65
3	0.20	0.85	66-85
4	0.15	1	86-00

# Multi-channel Queue - Sample Input

---

Able's Service Time distribution			
Service Time	Probability	Cumulative probability	Range
2	0.30	0.30	1-30
3	0.28	0.58	31-58
4	0.25	0.83	59-83
5	0.17	1	84-00

# Multi-channel Queue - Sample Input

---

Baker's Service Time distribution			
Service Time	Probability	Cumulative probability	Range
3	0.35	0.35	1-35
4	0.25	0.60	36-60
5	0.20	0.80	61-80
6	0.20	1	81-00

```
namespace MultiQueueModels
{
    10 references
    public class TimeDistribution
    {
        3 references
        public int Time { get; set; }
        2 references
        public decimal Probability { get; set; }
        3 references
        public decimal CummmProbability { get; set; }
        3 references
        public int MinRange { get; set; }
        4 references
        public int MaxRange { get; set; }
    }
}
```

# System Output - Performance Measures Per Server

---

Measure servers performance by calculating the following variables:

$$\text{probability of idle server}(x) = \frac{\text{total idle time of server}(x)}{\text{total run time of simulation}}$$

$$\text{average service time}(x) = \frac{\text{total service time}(x)}{\text{total number of customers}(x)}$$

$$\text{Utilization}(x) = \frac{\text{Total Time the server } x \text{ spends on calls}}{\text{Total run time of the simulation}}$$

```

public class Server
{
    3 references
    public Server()
    {
        this.TimeDistribution = new List<TimeDistribution>();
    }

    24 references
    public int ID { get; set; }
    7 references
    public decimal IdleProbability { get; set; }
    7 references
    public decimal AverageServiceTime { get; set; }
    13 references
    public decimal Utilization { get; set; }

    public List<TimeDistribution> TimeDistribution;

    //optional if needed use them
    8 references
    public int FinishTime { get; set; }
    2 references
    public int TotalWorkingTime { get; set; }
}

```

# Multi-channel Queue - Sample Output

A	B	C	D	E	F	G	H	I	J	K	L
						<i>Able</i>			<i>Baker</i>		
<i>Customer</i>	<i>Random Digits</i>	<i>Time between</i>	<i>Clock Time</i>	<i>Random Digits</i>	<i>Time Service</i>	<i>Service</i>	<i>Time Service</i>	<i>Time Service</i>	<i>Service</i>	<i>Time Service</i>	<i>Time in</i>
<i>No.</i>	<i>for Arrival</i>	<i>Arrivals</i>	<i>of Arrival</i>	<i>for Service</i>	<i>Begins</i>	<i>Time</i>	<i>Ends</i>	<i>Begins</i>	<i>Time</i>	<i>Ends</i>	<i>Queue</i>
1	—	—	0	95	0	5	5				0
2	26	2	2	21				2	3	5	0
3	98	4	6	51	6	3	9				0
4	90	4	10	92	10	5	15				0
5	26	2	12	89				12	6	18	0
6	42	2	14	38	15	3	18				1
7	74	3	17	13	18	2	20				1
8	80	3	20	61	20	4	24				0
9	68	3	23	50				23	4	27	0
10	22	1	24	49	24	3	27				0
11	48	2	26	39	27	3	30				1
12	34	2	28	53				28	4	32	0
13	45	2	30	88	30	5	35				0
14	24	1	31	01				32	3	35	1
15	34	2	33	81	35	4	39				2
16	63	2	35	53				35	4	39	0
17	38	2	37	81	39	4	43				2
18	80	3	40	64				40	5	45	0
19	42	2	42	01	43	2	45				1
20	56	2	44	67	45	4	49				1
21	89	4	48	01				48	3	51	0
22	18	1	49	47	49	3	52				0
23	51	2	51	75				51	5	56	0
24	71	3	54	57	54	3	57				0
25	16	1	55	87				56	6	62	1
26	92	4	59	47	59	3	62				0
						<u>56</u>			<u>43</u>		<u>11</u>

```

public class SimulationCase
{
    3 references
    public SimulationCase()
    {
        this.AssignedServer = new Server();
    }

    3 references
    public int CustomerNumber { get; set; }
    7 references
    public int RandomInterArrival { get; set; }
    8 references
    public int InterArrival { get; set; }
    22 references
    public int ArrivalTime { get; set; }
    7 references
    public int RandomService { get; set; }
    13 references
    public int ServiceTime { get; set; }
    24 references
    public Server AssignedServer { get; set; }
    13 references
    public int StartTime { get; set; }
    11 references
    public int EndTime { get; set; }
    11 references
    public int TimeInQueue { get; set; }
}

```



# System Output - Performance Measures

---

Measure system performance by calculating the following variables:

$$\text{average waiting time} = \frac{\text{total time customers waited in queue}}{\text{total number of customers}}$$

$$\text{probability (wait)} = \frac{\text{number of customers who waited}}{\text{total number of customers}}$$

Maximum queue length during simulation runtime.

```
public class PerformanceMeasures
{
    7 references
    public decimal AverageWaitingTime { get; set; }
    6 references
    public int MaxQueueLength { get; set; }
    7 references
    public decimal WaitingProbability { get; set; }
}
```

# Multi-channel Queue - System Input

---

1. Inter-arrival time distribution
2. Number of servers
3. Service time distribution for each server
4. Server Selection method (priority, random, least utilization[BONUS])
5. Stopping Condition
  - Maximum Number of customers
  - Simulation end time

# Multi-channel Queue - System Output

---

Results table including these columns:

- a. Customer No
- b. Random Digit for inter-arrival time
- c. Inter-arrival time
- d. Arrival time (Clock Time)
- e. Random Digit for service duration
- f. Service duration
- g. Server Index
- h. Time Service Begins
- i. Time Service Ends (Departure)
- j. total delay time for this customer (Time in queue)

N.B: Assume Server 1 has higher priority

```

public class SimulationSystem
{
    2 references
    public SimulationSystem()
    {
        this.Servers = new List<Server>();
        this.InterarrivalDistribution = new List<TimeDistribution>();
        this.PerformanceMeasures = new PerformanceMeasures();
        this.SimulationTable = new List<SimulationCase>();
    }

    //////////// INPUTS ////////////
    2 references
    public int NumberOfServers { get; set; }
    3 references
    public int StoppingNumber { get; set; }
    6 references
    public List<Server> Servers { get; set; }
    4 references
    public List<TimeDistribution> InterarrivalDistribution { get; set; }
    3 references
    public Enums.StoppingCriteria StoppingCriteria { get; set; }
    2 references
    public Enums.SelectionMethod SelectionMethod { get; set; }

    //////////// OUTPUTS ////////////
    15 references
    public List<SimulationCase> SimulationTable { get; set; }
    5 references
    public PerformanceMeasures PerformanceMeasures { get; set; }
}

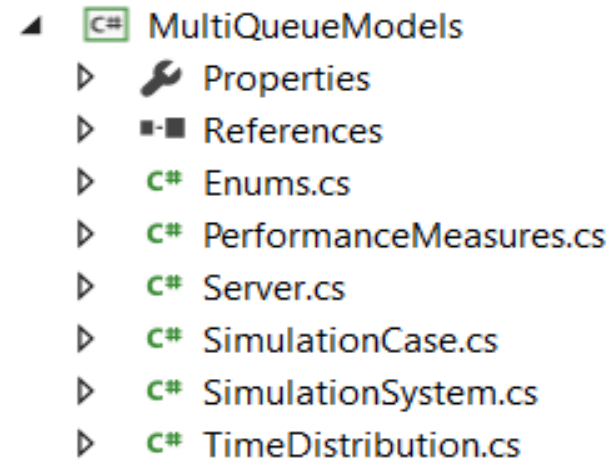
```

```
public class Enums
{
    8 references
    public enum SelectionMethod
    {
        HighestPriority = 1,
        Random = 2,
        LeastUtilization = 3
    }

    6 references
    public enum StoppingCriteria
    {
        NumberOfCustomers = 1,
        SimulationEndTime = 2
    }
}
```

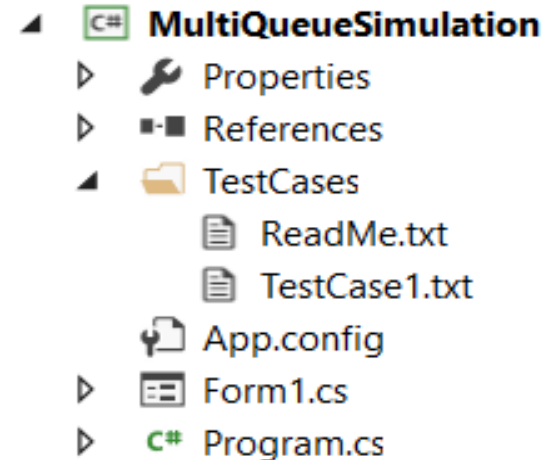
# Model Project

Layer where project entities are defined



# Windows Form Project

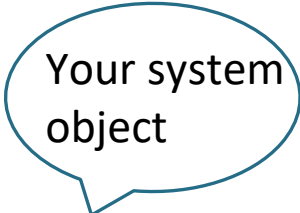
Main project where forms are created to input/output data



# Template Walkthrough

---

- You are committed to use the provided template in C#.
- You can add constructor if needed but don't override the default one.
- You are provided with only one testcase.
- You are given a testcase to run that will provide with a message
  - Success message if your code runs correctly.
  - Error message describing the failed part.
- Running using testcases



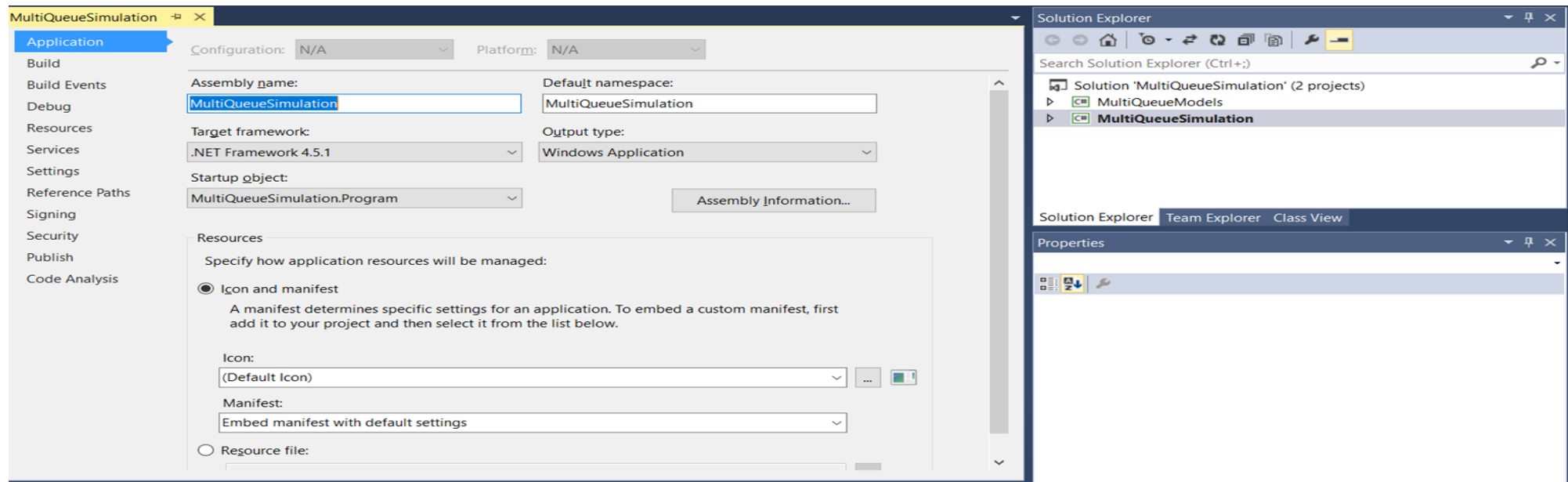
Your system  
object

```
string testingResult = TestingManager.Test(system, Constants.FileNames.TestCase1);  
MessageBox.Show(testingResult);
```



# Note

- **Error:** The type or namespace name " could not be found (are you missing a using directive or an assembly reference?)
- **Solution:** Make sure that your program target framework is .net framework 4.5.1.



Thank you...