Phase 1 — Evaluation Project

Ho Chi Minh City University of Technology (HCMUT)

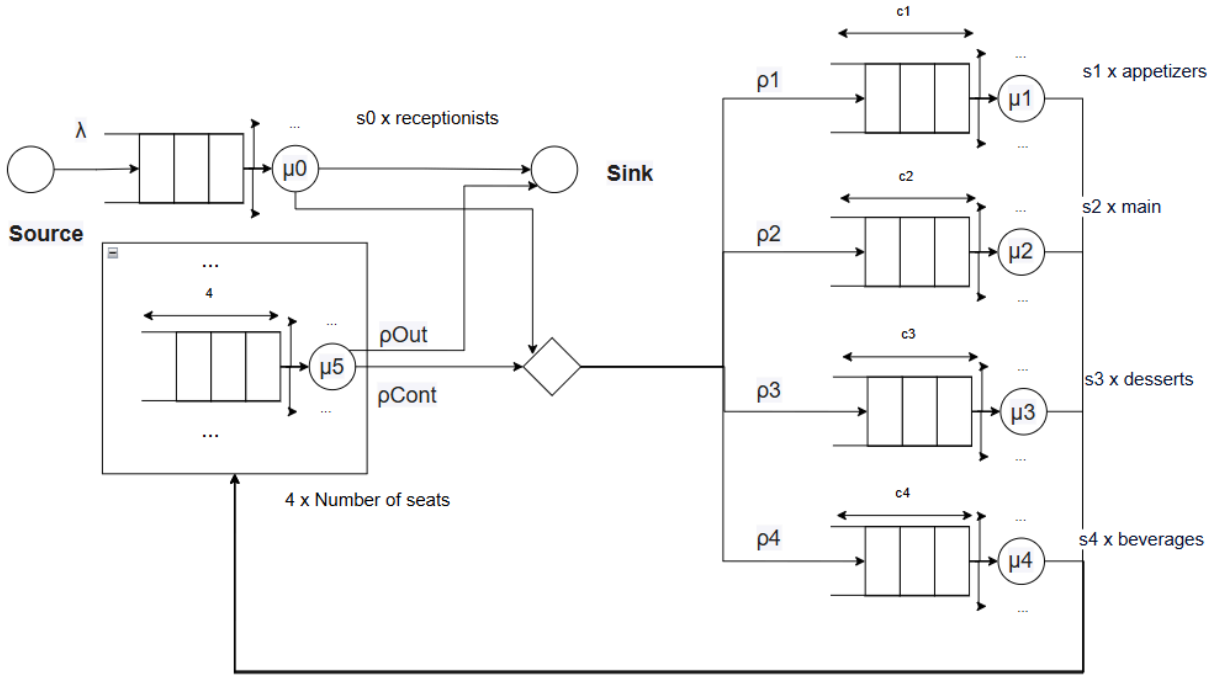
Instructor: Nguyễn Phương Duy

Team Members

No.	Name	Student ID
1	Trần Cẩm Hòa	2252240
2	Phạm Văn Bách	2252057
3	Trần Quốc Khải	2252341
4	Hoàng Phúc	2252629
5	Nguyễn Nhật Khôi	2252379
6	Nguyễn Hoàng Ngọc Tuấn	2252873

Report Phase: 1 — *Evaluation*
Group: *Obsidian Serpent*

I. Overview



Some important note to be taken, first the lamda is the rate of group in, and capacity is also in group. After reception step, that group will split up into 4 individual people. Second is the short hand in this project, s for servers (for example s2 x main mean the queue named "main" has s2 servers); p for probability and c for capacity.

The system consists of three main parts: reception, seats and buffet area.

Seats are where our guests will be eating, from Seats they can choose either to get some more food or finish their meal and exit the restaurant. For this project, seats are models of many simple M/M/4/4 queue. And other queues will be M/M/c.

For the buffet area, there are 4 rows of foods and drinks, and they will use the M/M/c/c queue.

The normal flow of a customer will be:

1. Check in at Reception
2. Choose which row to choose and return to seats
3. Repeat step 2 until satisfied then leave.

II. Interface for system

Global variables:

- MaxSimTime: Simultation time and also simulate the operating time of our resturant
- TotalCustomer: Total customer that visit our restaurant. Noted that it also included those succesfully entering, and also those rejected because the restaurant is full.
- CustomersRateMean: λ in the picture. Noted that the rate will be in group of 4. After the "customer" get through reception, they will become 4 individuals.
- ReceptionRateMean: Rate in which reception deal with customers.
- ReceptionServer: Number of servers in the recption section.
- ReceptionRateQueueLength: Maximum number of customers waiting, including those being served.
- AppetizersRateMean: Rate at which customers finish their pick at the appetizers line.
- AppetizersServers: Number of servers in the appetizers section.
- AppetizersQueueLength: Maximum number of customers waiting, including those being served.
- MainRateMean: Rate at which customers finish their pick at the main course line.
- MainServers: Number of servers in the main course section.
- MainQueueLength: Maximum number of customers waiting, including those being served.
- DessertsRateMean: Rate at which customers finish their pick at the desserts line.
- DessertsServers: Number of servers in the desserts section.
- DessertsQueueLength: Maximum number of customers waiting, including those being served.
- BeveragesRateMean: Rate at which customers finish their pick at the beverages line.
- BeveragesServers: Number of servers in the beverages section.
- BeveragesQueueLength: Maximum number of customers waiting, including those being served.
- NumTables: Number of tables in restaurant. 1 table for 4 customers.

Class:

1. Customer (all below will be public access variable)

- PatientReception: Maximum time unit that group of customer wait in reception line.

- PatientQueue: Maximum time unit that customer wait in buffet line. If wait to long, customer will storm out, and give bad review for restaurant.
- ProbAppetizers: Probability that a customer chooses to visit the appetizers line.
- ProbMain: Probability that a customer chooses to visit the main course line.
- ProbDessert: Probability that a customer chooses to visit the desserts line.
- ProbBeverage: Probability that a customer chooses to visit the beverages line.
- ProbCont: Probability that a customer continues to the next section after finishing their current selection.
- ProbOut: Probability that a customer leaves after finishing their selection.
- EatTime: Time customers spend eating after completing all their selections.

Note:

ProbAppetizers + ProbMain + ProbDessert + ProbBeverage = 1

ProbCount + ProbOut = 1

Function

- GeneratorGroup: Create group to be serve in recption (EnterReception function)
- GeneratorCustomer: Create individual customers objects (1 group = 4 customer) to be serve in MakeDecisionWhatToEat fnction.
- EnterReception (env): A receptionists serve a group, use timeout with mean of global variable ReceptionRateMean. It will loop over tables and find free one. If yes, continue to add event (function) MakeDecisionWhatToEat. If no then do nothing.
- MakeDecisionContOrOut (env, Customer):Base on Customer class (reference), this function will choose either continue to create new event (run function) MakeDecisionWhatToEat or do nothing and return (the same as leave restaurant).
- MakeDecisionWhatToEat (env, Customer): Base on Customer class (reference), this function will create new event of entering food/drink queue according to probability.
- EnterAppetizer (env, Customer): Simulate the action of waiting and taking appetizers. Will loop over appetizers server to find which one is free. Wait until get on hold of 1 server, then delay according to AppetizersRateMean variable.
- EnterMain (env, Customer): Similar to EnterAppetizers function, with according variable.

- EnterDessert (env, Customer): Similar to EnterAppetizers function, with according variable.
- EnterBeverage (env, Customer): Similar to EnterAppetizers function, with according variable.
- StartEating (env, Customer):

Workflow

Note this section will view over how all of the above will interact in our simulation system.

First, we need to make share resources through some of the public variables

- Receptionists = simpy.Resource(env, capacity=ReceptionServer)
- ReceptionQueue = simpy.Resource(env, capacity=ReceptionRateQueueLength)
- Tables = [simpy.Resource(env, capacity=4) for _ in range(NumTables)]
- AppetizersServer = simpy.Resource(env, capacity=AppetizersServers)
- AppetizersQueue = simpy.Resource(env, capacity=AppetizersQueueLength)

For main, desserts, and beverages will be the same of appetizers with corresponding capacity.

The workflow will be

1. GeneratorGroup(). This function will use TotalCustomer variable and CustomersRateMean
2. Reception (). This fuction will use ReceptionRateMean variable, Tables resource, Receptionists resource.
3. GeneratorCustomer(). It will create 4 new customers, each run individual to step 4
4. MakeDecisionWhatToEat(). This function will use Customer reference and randomness to choos which function to run in step 5.
5. EnterAppetizers() OR EnterMain() OR EnterDessert() OR EnterBeverage(). For EnterAppetizers(), it will use AppetizerServer resource, AppetizerQueue resource and AppetizerRateMean variable.
6. StartEating(). This function use EatTime variable.
7. MakeDecisionContOrOut(). This function will use Customer reference and randomness to choos either step 7 or step 4.
8. End.

```
!pip install simpy
```

```
Collecting simpy
  Downloading simpy-4.1.1-py3-none-any.whl.metadata (6.1 kB)
Downloading simpy-4.1.1-py3-none-any.whl (27 kB)
Installing collected packages: simpy
```

```
Successfully installed simpy-4.1.1
```

```
#Python code skeleton; 0 act as placeholder. The finished code will replace all
import simpy
import random

MaxSimTime = 0
TotalCustomer = 0
CustomersRateMean = 0
ReceptionRateMean = 0
ReceptionServer = 0
ReceptionRateQueueLength = 0
AppetizersRateMean = 0
AppetizersServers = 0
AppetizersQueueLength = 0
MainRateMean = 0
MainServers = 0
MainQueueLength = 0
DessertsRateMean = 0
DessertsServers = 0
DessertsQueueLength = 0
BeveragesRateMean = 0
BeveragesServers = 0
BeveragesQueueLength = 0
NumTables = 0

class Customer:
    def __init__(self, id):
        self.id = id
        self.PatientReception = 0
        self.PatientQueue = 0
        self.ProbAppetizers = 0
        self.ProbMain = 0
        self.ProbDessert = 0
        self.ProbBeverage = 0
        self.ProbCont = 0
        self.ProbOut = 0
        self.EatTime = 0

def GeneratorGroup(env):
    pass

def GeneratorCustomer():
    pass

def EnterReception(env, group):
    pass

def MakeDecisionWhatToEat(env, customer):
    pass
```

```
def MakeDecisionContOrOut(env, customer):
    pass

def EnterAppetizers(env, customer):
    pass

def EnterMain(env, customer):
    pass

def EnterDessert(env, customer):
    pass

def EnterBeverage(env, customer):
    pass

def StartEating(env, customer):
    pass

env = simpy.Environment()
Receptionists = simpy.Resource(env, capacity=ReceptionServer)
ReceptionQueue = simpy.Resource(env, capacity=ReceptionRateQueueLength)
Tables = [simpy.Resource(env, capacity=4) for _ in range(NumTables)]
AppetizersServer = simpy.Resource(env, capacity=AppetizersServers)
AppetizersQueue = simpy.Resource(env, capacity=AppetizersQueueLength)
MainServer = simpy.Resource(env, capacity=MainServers)
MainQueue = simpy.Resource(env, capacity=MainQueueLength)
DessertsServer = simpy.Resource(env, capacity=DessertsServers)
DessertsQueue = simpy.Resource(env, capacity=DessertsQueueLength)
BeveragesServer = simpy.Resource(env, capacity=BeveragesServers)
BeveragesQueue = simpy.Resource(env, capacity=BeveragesQueueLength)



if __name__ == "__main__":
    #env.process(GeneratorGroup(env))#Comment out due to ValueError: None is nc
    env.run()
```