**Python Mid-Term Assignment**

**What you need to submit : code github link**

Add [**phitron2022@gmail.com**](mailto:phitron2022@gmai.com) as collaborator

**Note: ভিডিওতে যদিও রিপো প্রাইভেট রাখতে বলা হয়েছে।আপনি গিটহাব রিপোটা পাবলিক রেখে দিবেন এবং রিপো পাবলিক রাখলে Collaborator এ এড করতে হবেনা।**

1. Make a class named **Star\_Cinema** which will have one class attribute named **hall\_list** which is an empty list initially. Make a method named **entry\_hall()** to insert an object of class **Hall** (Described below) inside its **hall\_list**. **(5)**

Class **Star\_Cinema:**

hall\_list=[ ]

def entry\_hall(self, Hall):

self.hall\_list.append(Hall)

1. Make a class named **Hall** which will have 5 instance attributes given below
   1. **seats** which is an dictionary of seats information
   2. **show\_list** which is an list of tuples
   3. **rows** which is the row of the seats in that hall
   4. **cols** which is the column of the seats in that hall
   5. **hall\_no** which is the unique no. of that hall

Initialize an object of class **Hall** with **rows**, **cols** and **hall\_no**. And insert that object to the **Star\_Cinema** class attribute named **hall\_list** inside the initializer using **inheritance**. **seats** and **show\_list** will be empty initially. **(20)**

Class **Hall:**

def \_\_init\_\_(self, seats, show\_list):

self.seats= { }

self.show\_list= [ ]

Class **SubHall(Hall):**

def \_\_init\_\_(self, rows, cols, hall\_no):

super().\_\_init\_\_()

self.rows= rows

self.cols= cols

self.hall\_no= hall\_no

Class **Star\_Cinema(Hall):**

hall\_list=[ ]

def entry\_hall(self,SubHall):

super().\_\_init\_\_()

self.hall\_list.append(SubHall)

1. Make a method in **Hall** class named **entry\_show()** which will take **id**, **movie\_name** and **time** in string format. Make a tuple with all of the information and append it to the **show\_list** attribute. Allocate seats with **rows** and **cols** using 2d list, initially all seats will be free. Make a key with **id** to the attribute **seats** and value will be the 2d list.

**(10)**

Class **SubHall(Hall):**

def \_\_init\_\_(self, rows, cols, hall\_no):

super().\_\_init\_\_()

self.rows= rows

self.cols= cols

self.hall\_no= hall\_no

def entry\_show(self, id, movie\_name, time):

info=(id, movie\_name, time)

self.show\_list.append(info)

self.seats[id]=[[0] \* self.cols for \_ in range(self.rows)]

1. Make a method in **Hall** class named **book\_seats()** which will take an **id** of the show and list of tuples where every tuple contains the **row** and **col** of the seat. You need to check the **id** of the show, and book the seats. **(10)**

Class **SubHall(Hall):**

def \_\_init\_\_(self, rows, cols, hall\_no):

super().\_\_init\_\_()

self.rows= rows

self.cols= cols

self.hall\_no= hall\_no

def entry\_show(self, id, movie\_name, time):

info=(id, movie\_name, time)

self.show\_list.append(info)

self.seats[id]=[[0] \* self.cols for \_ in range(self.rows)]

def book\_seats(self, id, seats):

if id not in self.seats:

return ‘No available shows’

for seat in seats:

row,col= map(int, seat.split(“ ”))

if self.seats[id][rows-1][cols-1]==1:

return “already booked”

self.seats[id][row-1][col-1]=(seat, 1)

return “Booking successful”

1. Make a method in **Hall** class named **view\_show\_list()** which will view all the shows running. **(5)**

Class **SubHall(Hall):**

def \_\_init\_\_(self, rows, cols, hall\_no):

super().\_\_init\_\_()

self.rows= rows

self.cols= cols

self.hall\_no= hall\_no

def entry\_show(self, id, movie\_name, time):

info=(id, movie\_name, time)

self.show\_list.append(info)

self.seats[id]=[[0] \* self.cols for \_ in range(self.rows)]

def book\_seats(self, id, seats):

if id not in self.seats:

return ‘No available shows’

for seat in seats:

row,col= map(int, seat.split(“ ”))

if self.seats[id][rows-1][cols-1]==1:

return “already booked”

self.seats[id][row-1][col-1]=(seat, 1)

return “Booking successful”

def view\_show\_list(self):

return self.show\_list

1. Make a method in **Hall** class named **view\_available\_seats()** which will take an **id** of show, and view the seats that are available in that show **(10)**

Class **SubHall(Hall):**

def \_\_init\_\_(self, rows, cols, hall\_no):

super().\_\_init\_\_()

self.rows= rows

self.cols= cols

self.hall\_no= hall\_no

def entry\_show(self, id, movie\_name, time):

info=(id, movie\_name, time)

self.show\_list.append(info)

self.seats[id]=[[0] \* self.cols for \_ in range(self.rows)]

def book\_seats(self, id, seats):

for seat in seats:

row,col= map(int, seat.split(“ ”))

self.seats[id][row-1][col-1]=(seat, 1)

return “Booking successful”

def view\_show\_list(self):

return self.show\_list

def view\_available\_seats(self, id):

if id not in self.seats:

return “Show not available”

Seats= self.seats[id]

Avb\_seats=[ ]

for rows in range(len(Seats)):

for cols in range(len(Seats[row])):

if Seats[rows][cols]=0:

Avb\_seats.append(Seats[rows][cols])

return Avb\_seats

1. Make a replica system so that the counter can view all shows that are running, view available seats in a show and can book tickets in a show. **(20)**

Class **SubHall(Hall):**

def \_\_init\_\_(self, rows, cols, hall\_no):

super().\_\_init\_\_()

self.rows= rows

self.cols= cols

self.hall\_no= hall\_no

def entry\_show(self, id, movie\_name, time):

info=(id, movie\_name, time)

self.show\_list.append(info)

self.seats[id]=[[0] \* self.cols for \_ in range(self.rows)]

def book\_seats(self, id, seats):

if id not in self.seats:

return ‘No available shows’

for seat in seats:

row,col= map(int, seat.split(“ ”))

if self.seats[id][rows-1][cols-1]==1:

return “already booked”

self.seats[id][row-1][col-1]=(seat, 1)

return “Booking successful”

def view\_show\_list(self):

return self.show\_list

def view\_available\_seats(self, id):

if id not in self.seats:

return “Show not available”

Seats= self.seats[id]

Avb\_seats=[ ]

for rows in range(len(Seats)):

for cols in range(len(Seats[row])):

if Seats[rows][cols]=0:

Avb\_seats.append(Seats[rows][cols])

return Avb\_seats

Class **counter**:

def \_\_init\_\_(self, SubHall):

self.SubHall=SubHall

def view\_all\_shows(self):

return self.SubHall.show\_list.

def available\_seats(self, id):

return self.SubHall.view\_available\_seats(id)

def book\_seats\_for\_show(self, id, seats):

return self.SubHall.book\_seats(id, seats)

def main():

while True:

print(“1. View movie list”)

print(“2. View available seats”)

print(“3. Book Tickets”)

print(“4. Exit”)

selection= input()

if selection == ‘1’

counter.view\_all\_shows()

elif selection ==’2’

id=input()

counter.available\_seats

elif selection == ‘3’

id=input()

seats=int(input())

counter.book\_seats\_for\_show()

elif selection == ‘4’

break

1. You need to handle the errors, for example- **(10)**
   1. If someone gives a wrong **id** of a show
   2. If someone tries to book a seat that is invalid
   3. If someone tries to book a seat that is already booked

def book\_seats(self, id, seats):

if id not in self.seats:

return ‘No available shows’

for seat in seats:

rows,cols= map(int, seat.split(“ ”))

if self.seats[id][rows-1][cols-1]==1:

return “already booked”

self.seats[id][row-1][col-1]=(seat, 1)

return “Booking successful”

1. Make the information of the classes as protected/private as you can so that the attributes can’t be accessed outside the class. **(10)**

Class **SubHall(Hall):**

def \_\_init\_\_(self, rows, cols, hall\_no):

super().\_\_init\_\_()

self.rows= rows

self.cols= cols

self.hall\_no= hall\_no

def \_entry\_show(self, id, movie\_name, time):

info=(id, movie\_name, time)

self.show\_list.append(info)

self.seats[id]=[[0] \* self.cols for \_ in range(self.rows)]

def \_\_book\_seats(self, id, seats):

if id not in self.seats:

return ‘No available shows’

for seat in seats:

row,col= map(int, seat.split(“ ”))

if self.seats[id][rows-1][cols-1]==1:

return “already booked”

self.seats[id][row-1][col-1]=(seat, 1)

return “Booking successful”

def \_view\_show\_list(self):

return self.show\_list

def \_\_view\_available\_seats(self, id):

if id not in self.seats:

return “Show not available”

Seats= self.seats[id]

Avb\_seats=[ ]

for rows in range(len(Seats)):

for cols in range(len(Seats[row])):

if Seats[rows][cols]=0:

Avb\_seats.append(Seats[rows][cols])

return Avb\_seats