**Assignment 5**

**Control Structure**

**SaiPrabath Chowdary S**

**Task 1: Conditional Statements**

In a BookingSystem, you have been given the task is to create a program to book tickets. if available tickets more than noOfTicket to book then display the remaining tickets or ticket unavailable:

**Tasks:**

1. Write a program that takes the availableTicket and noOfBookingTicket as input.
2. Use conditional statements (if-else) to determine if the ticket is available or not.

3. Display an appropriate message based on ticket availability.

def check\_ticket\_availability(available\_tickets, no\_of\_tickets\_to\_book):

if available\_tickets >= no\_of\_tickets\_to\_book:

remaining\_tickets = available\_tickets - no\_of\_tickets\_to\_book

print(f"Tickets available! Remaining tickets: {remaining\_tickets}")

else:

print("Tickets unavailable!")

# Example 1

available\_tickets = 50

no\_of\_tickets\_to\_book = 3

check\_ticket\_availability(available\_tickets, no\_of\_tickets\_to\_book)

Screenshot 2024-04-25 at 3.35.14 PM

# Example 2

available\_tickets = 5

no\_of\_tickets\_to\_book = 7

check\_ticket\_availability(available\_tickets, no\_of\_tickets\_to\_book)

Screenshot 2024-04-25 at 3.39.02 PM

**Task 2: Nested Conditional Statements**

Create a program that simulates a Ticket booking and calculating cost of tickets. Display tickets options such as "Silver", "Gold", "Dimond". Based on ticket category fix the base ticket price and get the user input for ticket type and no of tickets need and calculate the total cost of tickets booked.

def calculate\_ticket\_cost(ticket\_type, num\_tickets):

base\_prices = {"Silver": 100, "Gold": 200, "Diamond": 300}

if ticket\_type in base\_prices:

base\_price = base\_prices[ticket\_type]

total\_cost = base\_price \* num\_tickets

return total\_cost

else:

return None

ticket\_type = input("Enter ticket type (Silver/Gold/Diamond): ")

num\_tickets = int(input("Enter number of tickets: "))

total\_cost = calculate\_ticket\_cost(ticket\_type, num\_tickets)

if total\_cost is not None:

print(f"Total cost for {num\_tickets} {ticket\_type} tickets: {total\_cost}")

else:

print("Invalid ticket type!")



**Task 3: Looping**

From the above task book the tickets for repeatedly until user type "Exit"

def calculate\_ticket\_cost(ticket\_type, num\_tickets):

base\_prices = {"Silver": 100, "Gold": 200, "Diamond": 300}

if ticket\_type in base\_prices:

base\_price = base\_prices[ticket\_type]

total\_cost = base\_price \* num\_tickets

return total\_cost

else:

return None

while True:

ticket\_type = input("Enter ticket type (Silver/Gold/Diamond), or type 'Exit' to quit: ")

if ticket\_type.lower() == "exit":

print("Exiting ticket booking system.")

break

num\_tickets = int(input("Enter number of tickets: "))

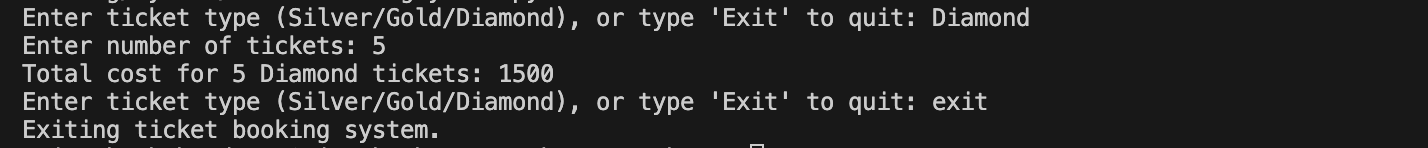
total\_cost = calculate\_ticket\_cost(ticket\_type, num\_tickets)

if total\_cost is not None:

print(f"Total cost for {num\_tickets} {ticket\_type} tickets: {total\_cost}")

else:

print("Invalid ticket type!")



**Task 4: Class & Object**

**Create a Following classes with the following attributes and methods:**

1. **Event** Class**:**

• **Attributes:**

o event\_name,

o event\_date DATE,

o event\_time TIME,

o venue\_name,

o total\_seats,

o available\_seats,

o ticket\_price DECIMAL,

o event\_type ENUM('Movie', 'Sports', 'Concert')

• **Methods and Constuctors:**

o Implement default constructors and overload the constructor with Customer

attributes, generate getter and setter, (print all information of attribute) methods for

the attributes.

o **calculate\_total\_revenue()**: Calculate and return the total revenue based on the

number of tickets sold.

o **getBookedNoOfTickets()**: return the total booked tickets

o **book\_tickets(num\_tickets)**: Book a specified number of tickets for an event. Initially

available seats are equal to the total seats when tickets are booked available seats

number should be reduced.

o **cancel\_booking(num\_tickets)**: Cancel the booking and update the available seats.

o **display\_event\_details():** Display event details, including event name, date time seat

availability.

class Event:

def \_\_init\_\_(self, event\_name, event\_date, event\_time, venue\_name, total\_seats, ticket\_price, event\_type):

self.event\_name = event\_name

self.event\_date = event\_date

self.event\_time = event\_time

self.venue\_name = venue\_name

self.total\_seats = total\_seats

self.available\_seats = total\_seats

self.ticket\_price = ticket\_price

self.event\_type = event\_type

# Getter and Setter methods

def get\_event\_name(self):

return self.event\_name

def set\_event\_name(self, event\_name):

self.event\_name = event\_name

def get\_event\_date(self):

return self.event\_date

def set\_event\_date(self, event\_date):

self.event\_date = event\_date

def get\_event\_time(self):

return self.event\_time

def set\_event\_time(self, event\_time):

self.event\_time = event\_time

def get\_venue\_name(self):

return self.venue\_name

def set\_venue\_name(self, venue\_name):

self.venue\_name = venue\_name

def get\_total\_seats(self):

return self.total\_seats

def set\_total\_seats(self, total\_seats):

self.total\_seats = total\_seats

def get\_available\_seats(self):

return self.available\_seats

def set\_available\_seats(self, available\_seats):

self.available\_seats = available\_seats

def get\_ticket\_price(self):

return self.ticket\_price

def set\_ticket\_price(self, ticket\_price):

self.ticket\_price = ticket\_price

def get\_event\_type(self):

return self.event\_type

def set\_event\_type(self, event\_type):

self.event\_type = event\_type

# methods

def calculate\_total\_revenue(self):

return self.ticket\_price \* (self.total\_seats - self.available\_seats)

def get\_booked\_no\_of\_tickets(self):

return self.total\_seats - self.available\_seats

def book\_tickets(self, num\_tickets):

if num\_tickets > self.available\_seats:

print("Insufficient seats available!")

return False

else:

self.available\_seats -= num\_tickets

print(f"{num\_tickets} tickets booked successfully for the event '{self.event\_name}'")

return True

def cancel\_booking(self, num\_tickets):

if self.available\_seats + num\_tickets > self.total\_seats:

print("Invalid number of tickets to cancel!")

return False

else:

self.available\_seats += num\_tickets

print(f"{num\_tickets} tickets canceled successfully for the event '{self.event\_name}'")

return True

def display\_event\_details(self):

print("Event Details:")

print(f"Event Name: {self.event\_name}")

print(f"Event Date: {self.event\_date}")

print(f"Event Time: {self.event\_time}")

print(f"Venue: {self.venue\_name}")

print(f"Total Seats: {self.total\_seats}")

print(f"Available Seats: {self.available\_seats}")

print(f"Ticket Price: {self.ticket\_price}")

print(f"Event Type: {self.event\_type.value}")

1. **Venue** Class

• **Attributes**:

o venue\_name,

o address

• **Methods and Constuctors:**

o **display\_venue\_details():** Display venue details.

o Implement default constructors and overload the constructor with Customer

attributes, generate getter and setter methods.

class Venue:

def \_\_init\_\_(self, venue\_name, address):

self.venue\_name = venue\_name

self.address = address

def get\_venue\_name(self):

return self.venue\_name

def set\_venue\_name(self, venue\_name):

self.venue\_name = venue\_name

def get\_address(self):

return self.address

def set\_address(self, address):

self.address = address

def display\_venue\_details(self):

print("Venue Details:")

print(f"Venue Name: {self.venue\_name}")

print(f"Address: {self.address}")

1. **Customer** Class

• **Attributes:**

o customer\_name,

o email,

o phone\_number

• **Methods and Constuctors:**

o Implement default constructors and overload the constructor with Customer

attributes, generate getter and setter methods.

o **display\_customer\_details()**: Display customer details.

class Customer:

def \_\_init\_\_(self, customer\_name, email, phone\_number):

self.customer\_name = customer\_name

self.email = email

self.phone\_number = phone\_number

def get\_customer\_name(self):

return self.customer\_name

def set\_customer\_name(self, customer\_name):

self.customer\_name = customer\_name

def get\_email(self):

return self.email

def set\_email(self, email):

self.email = email

def get\_phone\_number(self):

return self.phone\_number

def set\_phone\_number(self, phone\_number):

self.phone\_number = phone\_number

def display\_customer\_details(self):

print("Customer Details:")

print(f"Name: {self.customer\_name}")

print(f"Email: {self.email}")

print(f"Phone Number: {self.phone\_number}")

1. **Booking** Class to represent the Tiket booking system. Perform the following operation in main

method. Note:- Use Event class object for the following operation.

• **Methods and Constuctors:**

o **calculate\_booking\_cost(num\_tickets)**: Calculate and set the total cost of the

booking.

o **book\_tickets(num\_tickets)**: Book a specified number of tickets for an event.

o **cancel\_booking(num\_tickets)**: Cancel the booking and update the available seats.

o **getAvailableNoOfTickets()**: return the total available tickets

o **getEventDetails()**: return event details from the event class

class Booking:

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

self.event = event

self.num\_tickets = 0

self.total\_cost = 0

def calculate\_booking\_cost(self, num\_tickets):

self.num\_tickets = num\_tickets

self.total\_cost = self.num\_tickets \* self.event.get\_ticket\_price()

def book\_tickets(self, num\_tickets):

if self.event.book\_tickets(num\_tickets):

self.num\_tickets += num\_tickets

self.total\_cost += num\_tickets \* self.event.get\_ticket\_price()

print(f"{num\_tickets} tickets booked successfully!")

return True

else:

print("Failed to book tickets.")

return False

def cancel\_booking(self, num\_tickets):

if self.event.cancel\_booking(num\_tickets):

self.num\_tickets -= num\_tickets

self.total\_cost -= num\_tickets \* self.event.get\_ticket\_price()

print(f"{num\_tickets} tickets canceled successfully!")

return True

else:

print("Failed to cancel booking.")

return False

def get\_available\_no\_of\_tickets(self):

return self.event.get\_available\_seats()

def get\_event\_details(self):

return self.event.display\_event\_details()