

ASSIGNMENT 5

BY SHREYASI REJA

Ticket Booking System

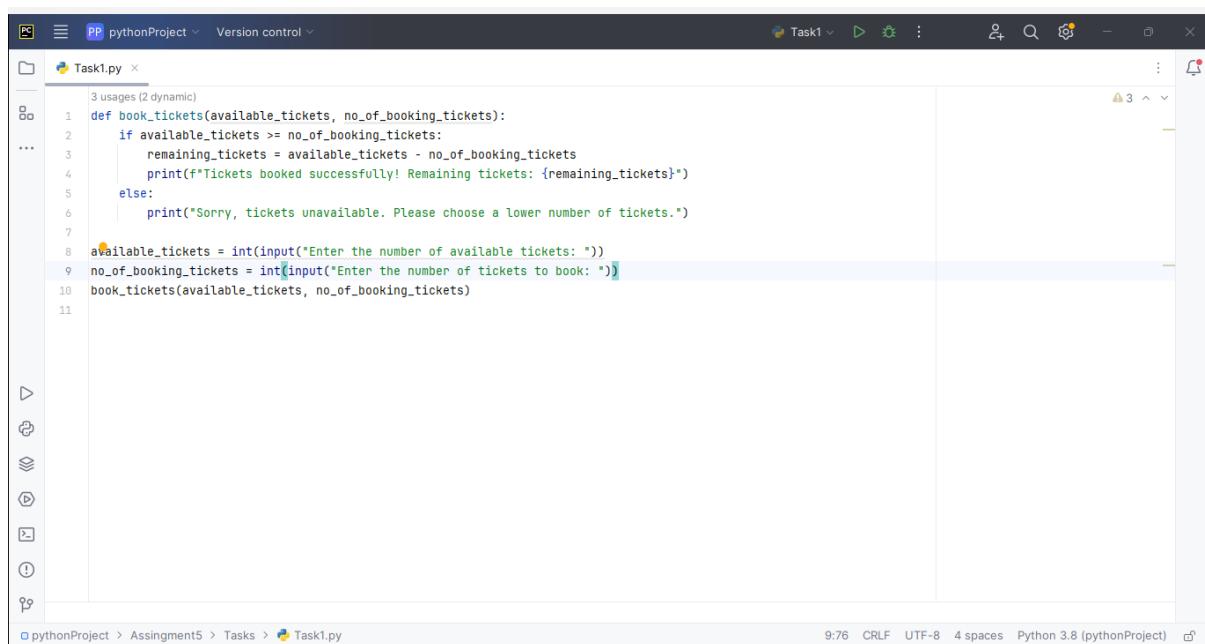
Control structure

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.



The screenshot shows the PyCharm IDE interface with the following details:

- Title Bar:** Shows "pythonProject" and "Version control".
- Toolbar:** Includes icons for file operations, search, and navigation.
- Code Editor:** Displays the Python code for "Task1.py".

```
Task1.py
1 def book_tickets(available_tickets, no_of_booking_tickets):
2     if available_tickets >= no_of_booking_tickets:
3         remaining_tickets = available_tickets - no_of_booking_tickets
4         print(f"Tickets booked successfully! Remaining tickets: {remaining_tickets}")
5     else:
6         print("Sorry, tickets unavailable. Please choose a lower number of tickets.")
7
8 available_tickets = int(input("Enter the number of available tickets: "))
9 no_of_booking_tickets = int(input("Enter the number of tickets to book: "))
10 book_tickets(available_tickets, no_of_booking_tickets)
```
- Sidebar:** Shows project structure with "pythonProject" and "Assingment5".
- Status Bar:** Shows "9:76 CRLF UTF-8 4 spaces Python 3.8 (pythonProject)".

Output:-

The screenshot shows the PyCharm IDE interface with the project 'pythonProject' selected. In the top navigation bar, 'Task1' is chosen. The left sidebar shows a file tree with 'Task1.py' and a 'Run' section containing 'Task1'. The main editor area displays the following code and its execution output:

```
C:\Users\SHREYASI\OneDrive\Desktop\Hexaware technical training code\pythonProject\.venv\Scripts\python.exe" "C:\Users\SHREYASI\OneDrive\Desktop\Hexaware t
Enter the number of available tickets: 2
Enter the number of tickets to book: 1
Tickets booked successfully! Remaining tickets: 1

Process finished with exit code 0
```

The status bar at the bottom indicates the file is 'Task1.py', encoding is 'CRLF', and Python version is '3.8 (pythonProject)'.

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.

The screenshot shows the PyCharm IDE interface with the project 'pythonProject' selected. In the top navigation bar, 'Task2' is chosen. The left sidebar shows a file tree with 'Task1.py' and 'Task2.py'. The main editor area displays the following Python code:

```
1 usage
2 def calculate_ticket_cost(ticket_type, no_of_tickets):
3     base_prices = {"Silver": 50, "Gold": 100, "Diamond": 150}
4
5     if ticket_type in base_prices:
6         base_price = base_prices[ticket_type]
7         total_cost = base_price * no_of_tickets
8         return total_cost
9     else:
10        return None
11 print("Ticket Options:")
12 print("1. Silver")
13 print("2. Gold")
14 print("3. Diamond")
15 ticket_type_input = input("Enter the ticket type (Silver/Gold/Diamond): ")
16 no_of_tickets = int(input("Enter the number of tickets needed: "))
17
18 total_cost = calculate_ticket_cost(ticket_type_input, no_of_tickets)
19
20 if total_cost is not None:
21     print(f"Total cost for {no_of_tickets} {ticket_type_input} tickets: ${total_cost}")
22 else:
23     print("Invalid ticket type. Please choose from Silver, Gold, or Diamond.")
24
25 calculate_ticket_cost() > else
```

The status bar at the bottom indicates the file is 'Task2.py', encoding is 'CRLF', and Python version is '3.8 (pythonProject)'.

Task 3: Looping

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

The screenshot shows the PyCharm IDE interface with the following details:

- Title Bar:** Shows "pythonProject" and "Version control".
- Toolbars:** Standard PyCharm toolbars for file operations, search, and navigation.
- Code Editor:** The "Task3.py" tab is active. The code defines a function to calculate ticket cost based on type and quantity, and a loop to accept user input until "exit".

```
1 usage
2 def calculate_ticket_cost(ticket_type, no_of_tickets):
3     base_prices = {"Silver": 50, "Gold": 100, "Diamond": 150}
4
5     if ticket_type in base_prices:
6         base_price = base_prices[ticket_type]
7         total_cost = base_price * no_of_tickets
8         return total_cost
9     else:
10        return None
11
12 # Display ticket options
13 print("Ticket Options:")
14 print("1. Silver")
15 print("2. Gold")
16 print("3. Diamond")
17 total_cost = 0
18
19 while True:
20     ticket_type_input = input("Enter the ticket type (Silver/Gold/Diamond), or type 'Exit' to stop: ")
21
22     if ticket_type_input.lower() == 'exit':
23         break
24
25     no_of_tickets = int(input("Enter the number of tickets needed: "))
26
27     current_cost = calculate_ticket_cost(ticket_type_input, no_of_tickets)
28
29     if current_cost is not None:
30         total_cost += current_cost
31         print(f"Total cost for {no_of_tickets} {ticket_type_input} tickets: ${current_cost}")
32     else:
33         print("Invalid ticket type. Please choose from Silver, Gold, or Diamond.")
34
35 print(f"Overall total cost for all booked tickets: ${total_cost}")
```

- Status Bar:** Shows "pythonProject > Assingment5 > Tasks > Task3.py", "15:20", "CRLF", "UTF-8", "4 spaces", and "Python 3.8 (pythonProject)".

The screenshot shows the PyCharm IDE interface with the following details:

- Title Bar:** Shows "pythonProject" and "Version control".
- Toolbars:** Standard PyCharm toolbars for file operations, search, and navigation.
- Code Editor:** The "Task3.py" tab is active. The code is identical to the first screenshot but includes additional print statements to show intermediate values.

```
14 print(2. oulu J
15 print("3. Diamond")
16 total_cost = 0
...
18 while True:
19     ticket_type_input = input("Enter the ticket type (Silver/Gold/Diamond), or type 'Exit' to stop: ")
20
21     if ticket_type_input.lower() == 'exit':
22         break
23
24     no_of_tickets = int(input("Enter the number of tickets needed: "))
25
26     current_cost = calculate_ticket_cost(ticket_type_input, no_of_tickets)
27
28     if current_cost is not None:
29         total_cost += current_cost
30         print(f"Total cost for {no_of_tickets} {ticket_type_input} tickets: ${current_cost}")
31     else:
32         print("Invalid ticket type. Please choose from Silver, Gold, or Diamond.")
33
34 print(f"Overall total cost for all booked tickets: ${total_cost}")
```

- Status Bar:** Shows "pythonProject > Assingment5 > Tasks > Task3.py", "35:1", "CRLF", "UTF-8", "4 spaces", and "Python 3.8 (pythonProject)".

Output:-

```
*C:\Users\SHREYASI\OneDrive\Desktop\Hexaware technical training code\pythonProject\.venv\Scripts\python.exe* "C:\Users\SHREYASI\OneDrive\Desktop\Hexaware technical training code\pythonProject\Task3.py"
Ticket Options:
1. Silver
2. Gold
3. Diamond
Enter the ticket type (Silver/Gold/Diamond), or type 'Exit' to stop: 1
Enter the number of tickets needed: 2
Invalid ticket type. Please choose from Silver, Gold, or Diamond.
Enter the ticket type (Silver/Gold/Diamond), or type 'Exit' to stop: exit
Overall total cost for all booked tickets: $0

Process finished with exit code 0
```

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')

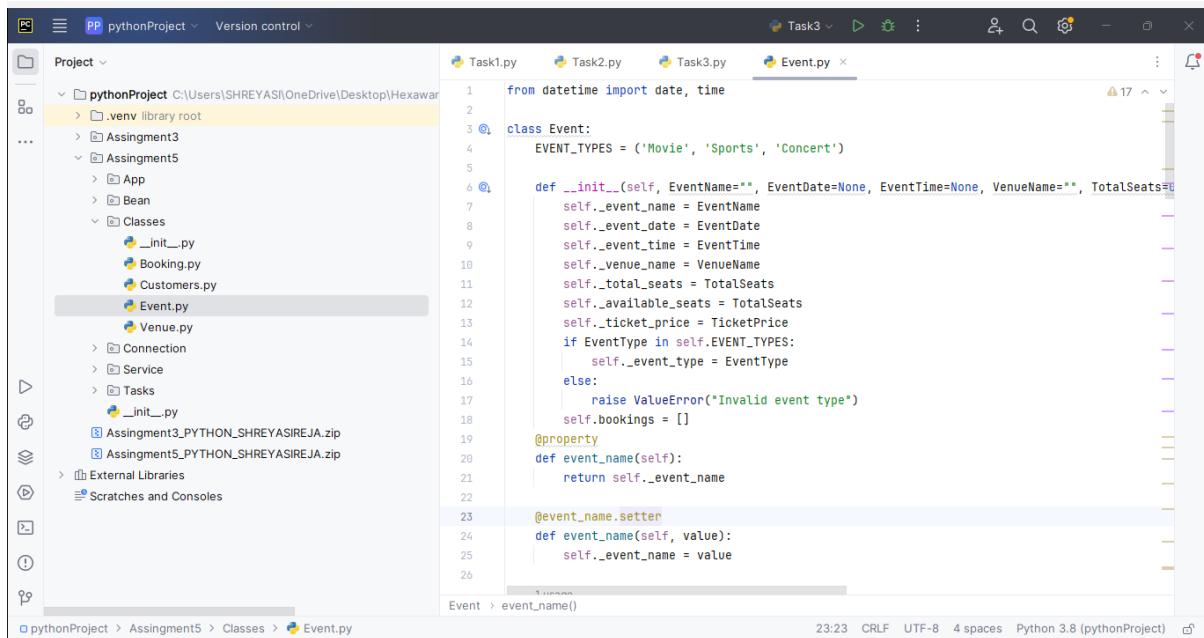


Methods and Constructors:

- 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.

EVENT:-



```

from datetime import date, time

class Event:
    EVENT_TYPES = ('Movie', 'Sports', 'Concert')

    def __init__(self, EventName="", EventDate=None, EventTime=None, VenueName="", TotalSeats=0):
        self._event_name = EventName
        self._event_date = EventDate
        self._event_time = EventTime
        self._venue_name = VenueName
        self._total_seats = TotalSeats
        self._available_seats = TotalSeats
        self._ticket_price = TicketPrice
        if EventType in self.EVENT_TYPES:
            self._event_type = EventType
        else:
            raise ValueError("Invalid event type")
        self.bookings = []

    @property
    def event_name(self):
        return self._event_name

    @event_name.setter
    def event_name(self, value):
        self._event_name = value

```

The screenshot shows the PyCharm IDE interface with the 'Event.py' file open in the code editor. The project structure on the left shows a 'pythonProject' folder containing 'Assingment3' and 'Assingment5' subfolders, with 'Event.py' selected. The code editor displays the provided Python class definition for 'Event'. The status bar at the bottom indicates the file is saved at 23:23, uses CRLF line endings, is in UTF-8 encoding, has 4 spaces for indentation, and was created with Python 3.8 (pythonProject).

PyCharm IDE interface showing the `Event.py` file. The code defines properties for `event_date`, `event_time`, and `venue_name`. Annotations in the gutter indicate usage counts: 17 for `event_date`, 1 for `event_time`, and 1 for `venue_name`.

```
1 usage
@property
def event_date(self):
    return self._event_date

@event_date.setter
def event_date(self, value):
    self._event_date = value

1 usage
@property
def event_time(self):
    return self._event_time

@event_time.setter
def event_time(self, value):
    self._event_time = value

1 usage
@property
def venue_name(self):
    return self._venue_name

@venue_name.setter
def venue_name(self, value):
    self._venue_name = value
```

PyCharm IDE interface showing the `Event.py` file. The code defines properties for `total_seats`, `available_seats`, and `ticket_price`. Annotations in the gutter indicate usage counts: 4 usages (3 dynamic) for `total_seats`, 3 usages (3 dynamic) for `available_seats`, and 1 for `ticket_price`.

```
@property
def total_seats(self):
    return self._total_seats

@total_seats.setter
def total_seats(self, value):
    self._total_seats = value

4 usages (3 dynamic)
@property
def available_seats(self):
    return self._available_seats

3 usages (3 dynamic)
@available_seats.setter
def available_seats(self, value):
    self._available_seats = value

@property
def ticket_price(self):
    return self._ticket_price

@ticket_price.setter
def ticket_price(self, value):
    self._ticket_price = value
```

```

1 usage
@usage
75     @property
76         def event_type(self):
77             return self._event_type
78
79     @event_type.setter
80         def event_type(self, value):
81             if value in self.EVENT_TYPES:
82                 self._event_type = value
83             else:
84                 raise ValueError("Invalid event type")
85
86     1 usage
87     def calculate_total_revenue(self):
88         return self._total_seats * self._ticket_price
89
90     1 usage
91     def get_booked_no_of_tickets(self):
92         return self._total_seats - self._available_seats
93
94     3 usages (2 dynamic)
95     def book_tickets(self, num_tickets):
96         if num_tickets <= self._available_seats:
97             self._available_seats -= num_tickets
98             print(f"Successfully booked {num_tickets} tickets for {self._event_name}")
99         else:
100             print("Insufficient available seats")
101
102     3 usages (2 dynamic)
103     def cancel_booking(self, num_tickets):
104         if num_tickets <= self._total_seats - self._available_seats:
105             self._available_seats += num_tickets
106             print(f"Successfully canceled booking for {num_tickets} tickets for {self._event_name}")
107         else:
108             print("Invalid number of tickets to cancel")
109
110     6 usages (2 dynamic)
111     def display_event_details(self):
112         print(f"Event Name: {self._event_name}")
113         print(f"Event Date: {self._event_date}")
114         print(f"Event Time: {self._event_time}")
115         print(f"Venue Name: {self._venue_name}")
116         print(f"Total Seats: {self._total_seats}")
117         print(f"Available Seats: {self._available_seats}")
118         print(f"Ticket Price: {self._ticket_price}")
119         print(f"Event Type: {self._event_type}")
120
121     event = Event(EventName="Movie Night", date(year=2024, month=2, day=10), time(hour=18, minute=30), VenueName="Cinema Hall", TotalSeats=100, TicketPrice=10)
122     event.display_event_details()
123     event.book_tickets(5)
124     event.cancel_booking(2)
125     print(f"Total Revenue: ${event.calculate_total_revenue()}")
126     print(f"Total Booked Tickets: {event.get_booked_no_of_tickets()}")

```

```

127
128     event = Event(EventName="Movie Night", date(year=2024, month=2, day=10), time(hour=18, minute=30), VenueName="Cinema Hall", TotalSeats=100, TicketPrice=10)
129     event.display_event_details()
130     event.book_tickets(5)
131     event.cancel_booking(2)
132     print(f"Total Revenue: ${event.calculate_total_revenue()}")
133     print(f"Total Booked Tickets: {event.get_booked_no_of_tickets()}")

```

2. Venue Class

Attributes:

- o venue_name,

- o address

Methods and Constructors:

- o display_venue_details(): Display venue details.

- o Implement default constructors and overload the constructor with Customer attributes, generate getter and setter methods.

The screenshot shows the PyCharm IDE interface with the following details:

- Project Bar:** Shows "pythonProject" as the current project.
- Toolbars:** Version control, Task3, and other standard icons.
- File List:** Shows files: Task1.py, Task2.py, Task3.py, Event.py, and the currently selected file, Venue.py.
- Code Editor:** Displays the Python code for the Venue class. The code includes:
 - A class `Event` with three types: Movie, Sports, and Concert.
 - A class `Venue` with an `__init__` method that takes `Venueame` and `Address` parameters, and sets them to `_venue_name` and `_address` respectively.
 - A `@property` decorator for `venue_name` which returns `_venue_name`.
 - A `@venue_name.setter` decorator for `venue_name` which sets `_venue_name` to the provided value.
 - A `@property` decorator for `address` which returns `_address`.
 - A `@address.setter` decorator for `address` which sets `_address` to the provided value.
- Status Bar:** Shows the current time as 7:32, and file format details as CRLF, UTF-8, 4 spaces, and Python 3.8 (pythonProject).

The screenshot shows the PyCharm IDE interface with the following details:

- Project Bar:** Shows "pythonProject" as the current project.
- File Bar:** Lists files: Task1.py, Task2.py, Task3.py, Event.py, and Venue.py (the active file).
- Status Bar:** Shows "pythonProject > Assingment5 > Classes > Venue.py".
- Code Editor:** Displays the Python code for the `Venue` class. The code includes usage notes, property definitions for `venue_name` and `address`, and a method `display_venue_details`. The code editor has syntax highlighting and a status bar indicating 4 changes (green triangle) and 1 warning (yellow exclamation mark).
- Sidebar:** Contains icons for file operations like new, open, save, and delete, along with other project-related icons.
- Bottom Status Bar:** Shows the current time (7:32), encoding (CRLF), character set (UTF-8), and code style (4 spaces). It also indicates the Python version used (Python 3.8) and the project name (pythonProject).

3. Customer Class

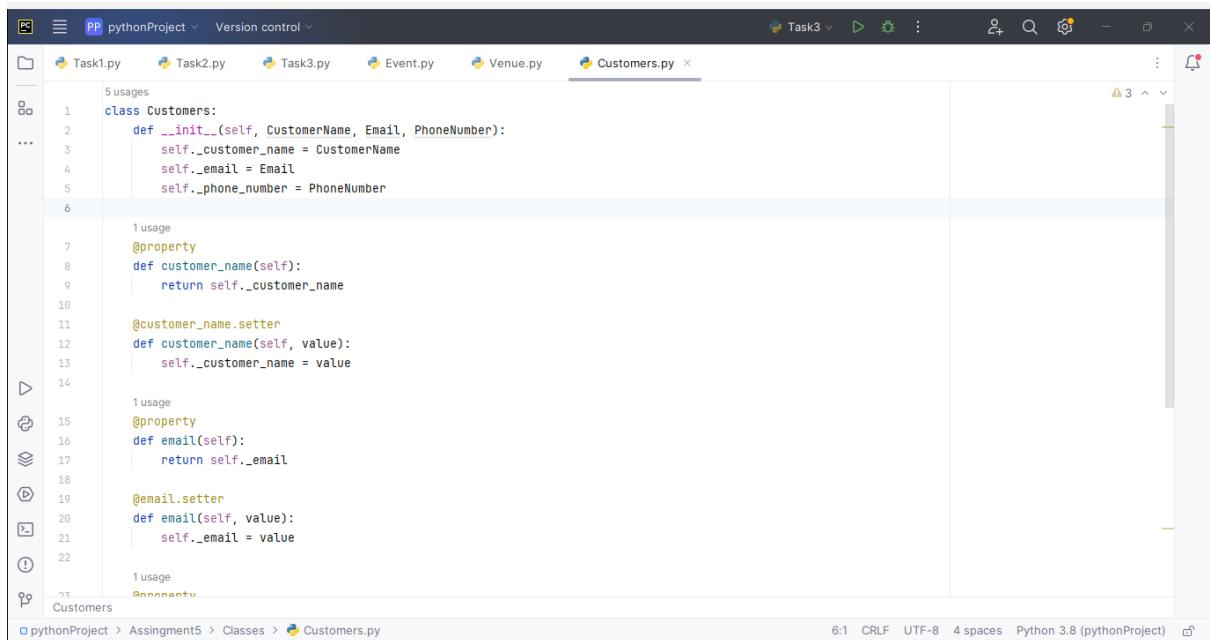
Attributes:

- o customer_name,
 - o email,
 - o phone_number.

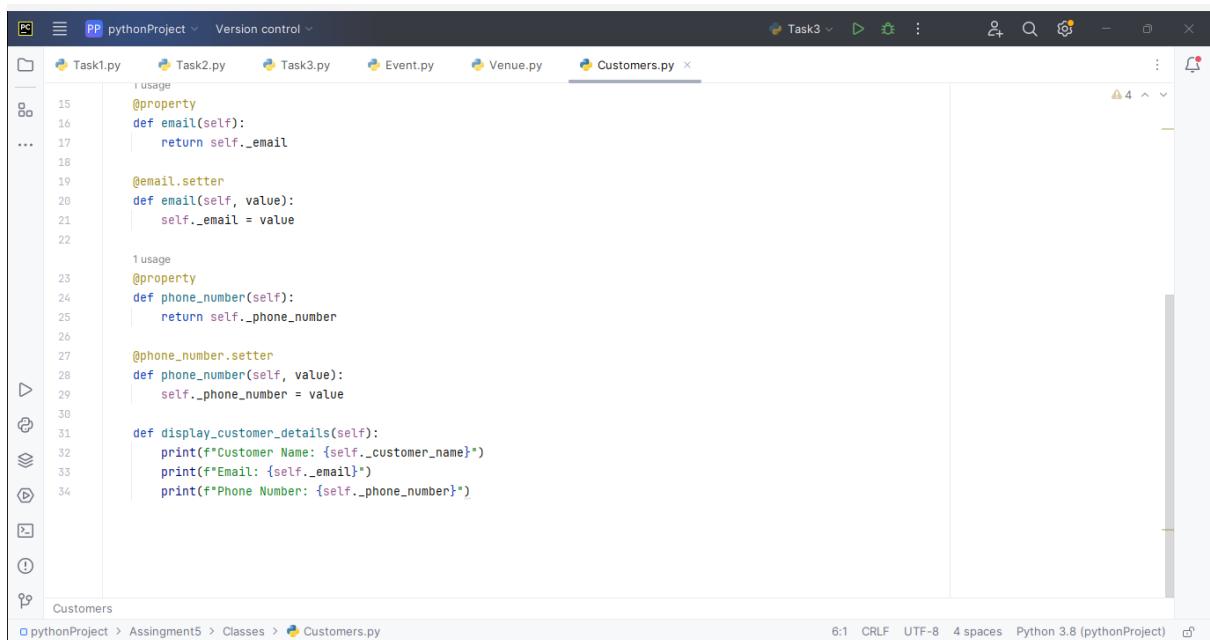
Methods and Constructors:

- o Implement default constructors and overload the constructor with Customer attributes, generate getter and setter methods.

o `display_customer_details()`: Display customer details.



```
5 usages
1 class Customers:
2     def __init__(self, CustomerName, Email, PhoneNumber):
3         self._customer_name = CustomerName
4         self._email = Email
5         self._phone_number = PhoneNumber
6
7     1 usage
8     @property
9     def customer_name(self):
10        return self._customer_name
11
12     @customer_name.setter
13     def customer_name(self, value):
14        self._customer_name = value
15
16     1 usage
17     @property
18     def email(self):
19        return self._email
20
21     @email.setter
22     def email(self, value):
23        self._email = value
24
25     1 usage
26     @property
27     def phone_number(self):
28        return self._phone_number
29
30     @phone_number.setter
31     def phone_number(self, value):
32        self._phone_number = value
33
34     def display_customer_details(self):
35         print(f"Customer Name: {self._customer_name}")
36         print(f"Email: {self._email}")
37         print(f"Phone Number: {self._phone_number}")
```



```
15     1 usage
16     @property
17     def email(self):
18        return self._email
19
20     @email.setter
21     def email(self, value):
22        self._email = value
23
24     1 usage
25     @property
26     def phone_number(self):
27        return self._phone_number
28
29     @phone_number.setter
30     def phone_number(self, value):
31        self._phone_number = value
32
33     def display_customer_details(self):
34         print(f"Customer Name: {self._customer_name}")
35         print(f"Email: {self._email}")
36         print(f"Phone Number: {self._phone_number}")
```

4. 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 Constructors:

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

The screenshot shows the PyCharm IDE interface with the 'pythonProject' selected in the top bar. The 'Booking.py' file is open in the center editor window. The code defines a class `Event` with three types: 'Movie', 'Sports', and 'Concert'. It then defines a class `Booking` that takes an `Event`, `NumTickets`, and `num_tickets` as parameters. Inside the constructor, it initializes `self.event` to the passed `Event` object, sets `self.num_tickets` to `NumTickets`, and calls `self.calculate_booking_cost()`. The `calculate_booking_cost` method calculates the total cost as `self.num_tickets * self.event.ticket_price`. The `Booking` class also has methods for booking tickets, canceling bookings, getting available seats, and displaying event details. The bottom status bar shows the file path as 'pythonProject > Assingment5 > Classes > Booking.py' and the Python version as 'Python 3.8 (pythonProject)'.

```
1 class Event:
2     EVENT_TYPES = ('Movie', 'Sports', 'Concert')
3 ...
4     2 usages
5     class Booking:
6         def __init__(self, Event, NumTickets, num_tickets):
7             self.event = Event
8             self.num_tickets = NumTickets
9             self.calculate_booking_cost()
10            1 usage
11            def calculate_booking_cost(self):
12                self.total_cost = self.num_tickets * self.event.ticket_price
13            2 usages (2 dynamic)
14            def book_tickets(self):
15                self.event.book_tickets(self.num_tickets)
16            2 usages (2 dynamic)
17            def cancel_booking(self):
18                self.event.cancel_booking(self.num_tickets)
19            def get_available_no_of_tickets(self):
20                return self.event.available_seats
21            def get_event_details(self):
22                return self.event.display_event_details()
23 ...
24            2 usages (1 dynamic)
25            def display_booking_details(self):
26                pass
27 ...
28 Booking > __init__()
29 pythonProject > Assingment5 > Classes > Booking.py
```

Task 5: Inheritance and polymorphism

1. Inheritance

Create a subclass `Movie` that inherits from `Event`. Add the following attributes and methods:

o Attributes:

1. `genre`: Genre of the movie (e.g., Action, Comedy, Horror).

2. `ActorName`

3. `ActresName`

o Methods:

1. Implement default constructors and overload the constructor with Customer attributes, generate getter and setter methods.

2. `display_event_details()`: Display movie details, including genre.

MOVIE:-

py pythonProject Version control

Task1.py Task2.py Task3.py Event.py Venue.py Customers.py Booking.py Task5Movie.py

```
from datetime import date, time
from Assingment5.Classes.Event import Event

3 usages
class Movie(Event):
    def __init__(self, event_name="", event_date=None, event_time=None, venue_name="", total_seats=0, ticket_price=0.0, event_type="", genre="", actor_name=""):
        super().__init__(event_name, event_date, event_time, venue_name, total_seats, ticket_price, event_type)
        self._genre = genre
        self._actor_name = actor_name
        self._actress_name = actress_name

    1 usage
    @property
    def genre(self):
        return self._genre

    @genre.setter
    def genre(self, value):
        self._genre = value

    1 usage
    @property
    def actor_name(self):
        return self._actor_name

    @actor_name.setter
    def actor_name(self, value):
        self._actor_name = value
```

pythonProject > Assingment5 > Tasks > Task5Movie.py

1:1 CRLF UTF-8 4 spaces Python 3.8 (pythonProject)

py pythonProject Version control

Task1.py Task2.py Task3.py Event.py Venue.py Customers.py Booking.py Task5Movie.py

```
@actor_name.setter
def actor_name(self, value):
    self._actor_name = value

1 usage
@property
def actress_name(self):
    return self._actress_name

@actress_name.setter
def actress_name(self, value):
    self._actress_name = value
3 usages (2 dynamic)
def display_event_details(self):
    super().display_event_details()
    print(f"Genre: {self._genre}")
    print(f"Actor: {self._actor_name}")
    print(f"Actress: {self._actress_name}")

if __name__ == "__main__":
    movie = Movie(event_name="Movie Night", date(2024, 2, 10), time(18, 30), venue_name="PVR", total_seats=100, ticket_price=10.5)
    movie.display_event_details()
```

pythonProject > Assingment5 > Tasks > Task5Movie.py

1:1 CRLF UTF-8 4 spaces Python 3.8 (pythonProject)

Output:-

The screenshot shows the PyCharm IDE interface. The top navigation bar includes 'pythonProject' and 'Version control'. Below the bar are tabs for 'Task1.py', 'Task2.py', 'Task3.py', 'Event.py', 'Venue.py', 'Customers.py', 'Booking.py', and 'Task5Movie.py'. The main code editor window displays Python code for a 'Movie' class with methods like 'display_event_details()'. The bottom terminal window shows the output of running 'Task5Movie.py', which lists event details such as Event Name, Date, Time, Venue, Seats, Price, Type, Genre, Actor, and Actress, followed by 'Process finished with exit code 0'.

```
24     @actor_name.setter
25         def actor_name(self, value):
26             self._actor_name = value
27
28     1 usage
29     @property
30     def actress_name(self):
31         return self._actress_name
32
33 Movie > display_event_details()
34
35 Run Task5Movie
36
37 Event Name: Movie Night
38 Event Date: 2024-02-10
39 Event Time: 18:30:00
40 Venue Name: PVR
41 Total Seats: 100
42 Available Seats: 100
43 Ticket Price: 10.5
44 Event Type: Movie
45 Genre: Action
46 Actor: Hrithik Roshan
47 Actress: Katrina Kaif
48
49 Process finished with exit code 0
```

Create another subclass Concert that inherits from Event. Add the following attributes and methods:

o Attributes:

1. artist: Name of the performing artist or band.
2. type: (Theatrical, Classical, Rock, Recital)

o Methods:

1. Implement default constructors and overload the constructor with Customer attributes, generate getter and setter methods.
2. display_concert_details(): Display concert details, including the artist.

CONCERT:-

The screenshot shows the PyCharm IDE interface with the file `Task5Concert.py` open. The code defines a `Concert` class that inherits from the `Event` class. It includes properties for artist and concert type, and methods for displaying event details. A conditional block at the bottom creates a `Concert` instance and prints its details.

```
from datetime import date, time
from Assingment5.Classes.Event import Event

class Concert(Event):
    def __init__(self, event_name='', event_date=None, event_time=None, venue_name='', total_seats=0, ticket_price=0.0, event_type='', artist=''):
        super().__init__(event_name, event_date, event_time, venue_name, total_seats, ticket_price, event_type)
        self._artist = artist
        self._concert_type = concert_type

    @property
    def artist(self):
        return self._artist

    @artist.setter
    def artist(self, value):
        self._artist = value

    @property
    def concert_type(self):
        return self._concert_type

    @concert_type.setter
    def concert_type(self, value):
        self._concert_type = value

if __name__ == "__main__":
    concert = Concert(event_name="Concert Night", date=date(2024, 2, 15), time=time(20, 0), venue_name="Concert Hall", total_seats=200, ticket_price=50.0, event_type="Concert")
    concert.display_concert_details()
```

This screenshot shows the same `Task5Concert.py` file in the PyCharm IDE, but with a different scroll position. The code is identical to the one in the first screenshot, but the view is lower down, showing lines 14 through 32.

```
@artist.setter
def artist(self, value):
    self._artist = value

    1 usage
@property
def concert_type(self):
    return self._concert_type

    @concert_type.setter
    def concert_type(self, value):
        self._concert_type = value

    1 usage
    def display_concert_details(self):
        super().display_event_details()
        print(f"Artist: {self._artist}")
        print(f"Concert Type: {self._concert_type}")

if __name__ == "__main__":
    concert = Concert(event_name="Concert Night", date(date(2024, 2, 15)), time(time(20, 0)), venue_name="Concert Hall", total_seats=200, ticket_price=50.0, event_type="Concert")
    concert.display_concert_details()
```

Output:

The screenshot shows the PyCharm IDE interface. The top bar displays the project name "pythonProject" and the file "Task5Concert.py". The code editor contains Python code for a "Concert" class. The run output window shows the execution results:

```
Event Name: Concert Night
Event Date: 2024-02-15
Event Time: 20:00:00
Venue Name: Concert Hall
Total Seats: 200
Available Seats: 200
Ticket Price: 25000.0
Event Type: Concert
Artist: Bangtan Sonyeodan BTS
Concert Type: Kpop
```

Process finished with exit code 0

Create another subclass Sports that inherits from Event. Add the following attributes and methods:

o Attributes:

1. sportName: Name of the game.
2. teamsName: (India vs Pakistan)

o Methods:

1. Implement default constructors and overload the constructor with Customer attributes, generate getter and setter methods.
2. display_sport_details(): Display concert details, including the artist.

SPORTS:-

The screenshot shows the PyCharm IDE interface with the file `Task5Sports.py` open. The code defines a class `Sports` that inherits from `Event`. It includes properties for `sport_name` and `teams_name`, and methods for displaying sport details. A code completion tooltip is visible over the `sport_name` property.

```
from datetime import date, time
from Assingment5.Classes.Event import Event

class Sports(Event):
    def __init__(self, event_name='', event_date=None, event_time=None, venue_name='', total_seats=0, ticket_price=0.0, event_type='', sport_name='', teams_name=''):
        super().__init__(event_name, event_date, event_time, venue_name, total_seats, ticket_price, event_type)
        self._sport_name = sport_name
        self._teams_name = teams_name

    @property
    def sport_name(self):
        return self._sport_name

    @sport_name.setter
    def sport_name(self, value):
        self._sport_name = value

    @property
    def teams_name(self):
        return self._teams_name

    @teams_name.setter
    def teams_name(self, value):
        self._teams_name = value
```

The screenshot shows the PyCharm IDE interface with the file `Task5Sports.py` open. The code now includes a `display_sport_details` method that prints the sport and team names. An `if __name__ == "__main__":` block at the bottom creates an instance of `Sports` and calls its method. A code completion tooltip is visible over the `print` statement.

```
@sport_name.setter
def sport_name(self, value):
    self._sport_name = value

@teams_name.setter
def teams_name(self, value):
    self._teams_name = value

def display_sport_details(self):
    super().display_event_details()
    print(f"Sport Name: {self._sport_name}")
    print(f"Teams Name: {self._teams_name}")

if __name__ == "__main__":
    sports_event = Sports(event_name="Cricket Match", date( year: 2024, month: 3, day: 1), time( hour: 14, minute: 30), venue_name: "Stadium", total_seats: 500, ticket_price: 50.0)
```

Output:-

The screenshot shows the PyCharm IDE interface. The top navigation bar includes 'pythonProject' and 'Version control'. Below the bar are tabs for 'ask2.py', 'Task3.py', 'Event.py', 'Venue.py', 'Customers.py', 'Booking.py', 'Task5Movie.py', 'Task5Concert.py', and 'Task5Sports.py'. The 'Task5Sports.py' tab is active. The code in the editor is:

```

25     def display_sport_details(self):
26         super().display_event_details()
27         print(f"Sport Name: {self._sport_name}")
28         print(f"Teams Name: {self._teams_name}")
29 if __name__ == "__main__":
30     sports_event = Sports( event_name= "Cricket Match", date( year: 2024, month: 3, day: 1), time( hour: 14, minute: 30), venue_name: "Stadium", total_seats: 500,
31     sports_event.display_sport_details()
32

```

The 'Run' dropdown menu is open, showing 'Task5Sports' as the selected run configuration.

The bottom panel displays the output of the run:

```

Sports > display_sport_details()
Run Task5Sports

Event Name: Cricket Match
Event Date: 2024-03-01
Event Time: 14:30:00
Venue Name: Stadium
Total Seats: 500
Available Seats: 500
Ticket Price: 1500.0
Event Type: Sports
Sport Name: Cricket
Teams Name: India vs Pakistan

Process finished with exit code 0

```

The status bar at the bottom indicates the file path 'pythonProject > Assingment5 > Tasks > Task5Sports.py', encoding 'UTF-8', and Python version 'Python 3.8 (pythonProject)'.

Create a class `TicketBookingSystem` with the following methods:

- o `create_event(event_name: str, date:str, time:str, total_seats: int, ticket_price: float, event_type: str, venu_name:str)`: Create a new event with the specified details and event type (movie, sport or concert) and return event object.
- o `display_event_details(event: Event)`: Accepts an event object and calls its `display_event_details()` method to display event details.
- o `book_tickets(event: Event, num_tickets: int)`:
 1. Accepts an event object and the number of tickets to be booked.
 2. Checks if there are enough available seats for the booking.
 3. If seats are available, updates the available seats and returns the total cost of the booking.
 4. If seats are not available, displays a message indicating that the event is sold out.
- o `cancel_tickets(event: Event, num_tickets)`: cancel a specified number of tickets for an event.
- o `main()`: simulates the ticket booking system
 1. User can book tickets and view the event details as per their choice in menu (movies, sports, concerts).

2. Display event details using the `display_event_details()` method without knowing the specific event type (demonstrate polymorphism).

3. Make bookings using the book_tickets() and cancel tickets cancel_tickets() method.

TicketBookingSystem:-

The screenshot shows the PyCharm IDE interface with the following details:

- Title Bar:** pythonProject > Version control
- Toolbars:** Standard Java-like toolbar.
- Left Sidebar:** Shows project structure with files: main.py, Venue.py, Customers.py, Booking.py, Task5Movie.py, Task5Concert.py, Task5Sports.py, and Task5TicketBookingSystem.py.
- Code Editor:** The main window displays the content of `Task5TicketBookingSystem.py`. The code defines a class `TicketBookingSystem` with a method `create_event` that takes parameters like event name, date, time, seats, price, type, and venue. It then prompts the user for specific details based on the event type (Movie, Concert, or Sports) and returns a corresponding object.
- Status Bar:** Shows the file path as `pythonProject > Assingment5 > Tasks > Task5TicketBookingSystem.py`, and the bottom right corner indicates the code is 1:1 CRLF, UTF-8, 4 spaces, and Python 3.8 (pythonProject).

```
25     teams_name = input("Enter teams names: ")
26     return Sports(event_name, event_date, event_time, venue_name, total_seats, ticket_price, event_type,
27                   sport_name, teams_name)
28   else:
29     raise ValueError("Invalid event type. Please enter 'Movie', 'Sports', or 'Concert'.")
30
31   3 usages (2 dynamic)
32   def display_event_details(self, event):
33     event.display_event_details()
34
35   3 usages (2 dynamic)
36   def book_tickets(self, event, num_tickets):
37     return event.book_tickets(num_tickets)
38
39   1 usage
40   def cancel_tickets(self, event, num_tickets):
41     event.cancel_booking(num_tickets)
42
43   1 usage
44   def main(self):
45     print("Welcome to the Ticket Booking System!")
46
47     while True:
48       print("\nMenu:")
49       print("1. Create Event")
50       print("2. Display Event Details")
51       print("3. Book Tickets")
```

```
43     while True:
44         print("\nMenu:")
45         print("1. Create Event")
46         print("2. Display Event Details")
47         print("3. Book Tickets")
48         print("4. Cancel Tickets")
49         print("5. Exit")
50
51         choice = input("Enter your choice (1-5): ")
52
53         if choice == '1':
54             event_name = input("Enter event name: ")
55             event_date = input("Enter event date (YYYY-MM-DD): ")
56             event_time = input("Enter event time (HH:MM): ")
57             total_seats = int(input("Enter total seats: "))
58             ticket_price = float(input("Enter ticket price: "))
59             event_type = input("Enter event type (Movie/Sports/Concert): ")
60             venue_name = input("Enter venue name: ")
61
62             event = self.create_event(event_name, event_date, event_time, total_seats, ticket_price, event_type,
63                                     venue_name)
64             print(f"Event '{event_name}' created successfully!")
65
66         elif choice == '2':
67             event_type = input("Enter event type (Movie/Sports/Concert): ")
68             if event_type in Event.EVENT_TYPES:
69                 event = self.display_event_details(event_type)
70             else:
71                 print("Invalid event type")
72
73         elif choice == '3':
74             event_type = input("Enter event type (Movie/Sports/Concert): ")
75
76             if event_type in Event.EVENT_TYPES:
77                 event_name = input("Enter event name: ")
78                 event_date = input("Enter event date (YYYY-MM-DD): ")
79                 event_time = input("Enter event time (HH:MM): ")
80
81                 event = self.create_event(event_name, event_date, event_time, total_seats=0, ticket_price=0.0, event_type=event_type, venue_name="")
82                 self.display_event_details(event)
83             else:
84                 print("Invalid event type")
85
86         elif choice == '4':
87             event_type = input("Enter event type (Movie/Sports/Concert): ")
88
89             if event_type in Event.EVENT_TYPES:
90                 event_name = input("Enter event name: ")
91                 event_date = input("Enter event date (YYYY-MM-DD): ")
92                 event_time = input("Enter event time (HH:MM): ")
93
94                 event = self.create_event(event_name, event_date, event_time, total_seats=0, ticket_price=0.0, event_type=event_type, venue_name="")
95                 num_tickets = int(input("Enter the number of tickets to book: "))
96                 total_cost = self.book_tickets(event, num_tickets)
97                 print(f"Total cost of booking: ${total_cost}")
98             else:
99                 print("Invalid event type")
```

pythonProject > Assingment5 > Tasks > Task5TicketBookingSystem.py 1:1 CRLF UTF-8 4 spaces Python 3.8 (pythonProject)

```
66         elif choice == '2':
67             event_type = input("Enter event type (Movie/Sports/Concert): ")
68             if event_type in Event.EVENT_TYPES:
69                 event_name = input("Enter event name: ")
70                 event_date = input("Enter event date (YYYY-MM-DD): ")
71                 event_time = input("Enter event time (HH:MM): ")
72
73                 event = self.create_event(event_name, event_date, event_time, total_seats=0, ticket_price=0.0, event_type=event_type, venue_name="")
74                 self.display_event_details(event)
75             else:
76                 print("Invalid event type")
77
78         elif choice == '3':
79             event_type = input("Enter event type (Movie/Sports/Concert): ")
80
81             if event_type in Event.EVENT_TYPES:
82                 event_name = input("Enter event name: ")
83                 event_date = input("Enter event date (YYYY-MM-DD): ")
84                 event_time = input("Enter event time (HH:MM): ")
85
86                 event = self.create_event(event_name, event_date, event_time, total_seats=0, ticket_price=0.0, event_type=event_type, venue_name="")
87                 num_tickets = int(input("Enter the number of tickets to book: "))
88                 total_cost = self.book_tickets(event, num_tickets)
89                 print(f"Total cost of booking: ${total_cost}")
90             else:
91                 print("Invalid event type")
```

pythonProject > Assingment5 > Tasks > Task5TicketBookingSystem.py 1:1 CRLF UTF-8 4 spaces Python 3.8 (pythonProject)

```

93     elif choice == '4':
94         event_type = input("Enter event type (Movie/Sports/Concert): ")
95         if event_type in Event.EVENT_TYPES:
96             event_name = input("Enter event name: ")
97             event_date = input("Enter event date (YYYY-MM-DD): ")
98             event_time = input("Enter event time (HH:MM): ")
99
100            event = self.create_event(event_name, event_date, event_time, total_seats=0, ticket_price=0.0, event_type, venue_name="")
101            num_tickets = int(input("Enter the number of tickets to cancel: "))
102            self.cancel_tickets(event, num_tickets)
103            print(f"Booking for {num_tickets} tickets canceled successfully!")
104        else:
105            print("Invalid event type")
106
107    elif choice == '5':
108        print("Thank you for using the Ticket Booking System!")
109        break
110
111    else:
112        print("Invalid choice. Please enter a valid option (1-5).")
113
114
115 if __name__ == "__main__":
116     ticket_booking_system = TicketBookingSystem()
117     ticket_booking_system.main()

```

OUTPUT:-

```

108     print("Thank you for using the Ticket Booking System!")
109     break
110
111     else:
112         print("Invalid choice. Please enter a valid option (1-5).")
113
114
115 if __name__ == "__main__":
116     ticket_booking_system = TicketBookingSystem()
TicketBookingSystem > main() > while True > else
Run Task5TicketBookingSystem
:
Enter your choice (1-5): 1
Enter event name: ABC
Enter event date (YYYY-MM-DD): 2024-04-04
Enter event time (HH:MM): 13:46
Enter total seats: 20
Enter ticket price: 10
Enter event type (Movie/Sports/Concert): Movie
Enter venue name: pvr
Enter movie genre: action
Enter actor name: Hritik
Enter actress name: Ktrina
Event 'ABC' created successfully!
Menu:

```

Task 6: Abstraction

Requirements:

1. Event Abstraction:

Create an abstract class `Event` that represents a generic event. It should include the following attributes and methods as mentioned in TASK 1:

2. Concrete Event Classes:

Create three concrete classes that inherit from Event abstract class and override abstract methods in concrete class should declare the variables as mentioned in above Task 2:

Movie.

Concert.

Sport.

ANS:-USE ABSTRACTION IN TASK 4 EVENT CLASS AND TASK 5 SUBCLASS MOVIE,CONCERT AND SPORT.

3. BookingSystem Abstraction:

Create an abstract class BookingSystem that represents the ticket booking system. It should include the methods of TASK 2 TicketBookingSystem:

4. Concrete TicketBookingSystem Class:

Create a concrete class TicketBookingSystem that inherits from BookingSystem:

TicketBookingSystem: Implement the abstract methods to create events, book tickets, and retrieve available seats. Maintain an array of events in this class.

Create a simple user interface in a main method that allows users to interact with the ticket booking system by entering commands such as "create_event", "book_tickets", "cancel_tickets", "get_available_seats," and "exit."

ANS:-USE ABSTRACTION IN TASK 5 TICKETBOOKINGSYSTEM.

Task 8: Interface/abstract class, and Single Inheritance, static variable

5. Create interface/abstract class IEventServiceProvider with following methods:

create_event(event_name: str, date:str, time:str, total_seats: int, ticket_price: float, event_type: str, venu: Venu): Create a new event with the specified details and event type (movie, sport or concert) and return event object.

getEventDetails(): return array of event details from the event class.

getAvailableNoOfTickets(): return the total available tickets.

```
Task5Concert.py Task5Sports.py Task5TicketBookingSystem.py IEventServiceProvider.py
1 from abc import ABC, abstractmethod
2 from Assingment5.Classes.Venue import Venue
3 from Assingment5.Classes.Event import Event
4 from typing import List
5
6 2 usages
7 @class IEventServiceProvider(ABC):
8
9     @abstractmethod
10        def create_event(self, event_name: str, date: str, time: str, total_seats: int, ticket_price:
11            event_type: str, venue: Venue) -> Event:
12            pass
13
14     @abstractmethod
15        def get_event_details(self) -> List[str]:
16            pass
17
18     @abstractmethod
19        def get_available_no_of_tickets(self):
20            pass
```

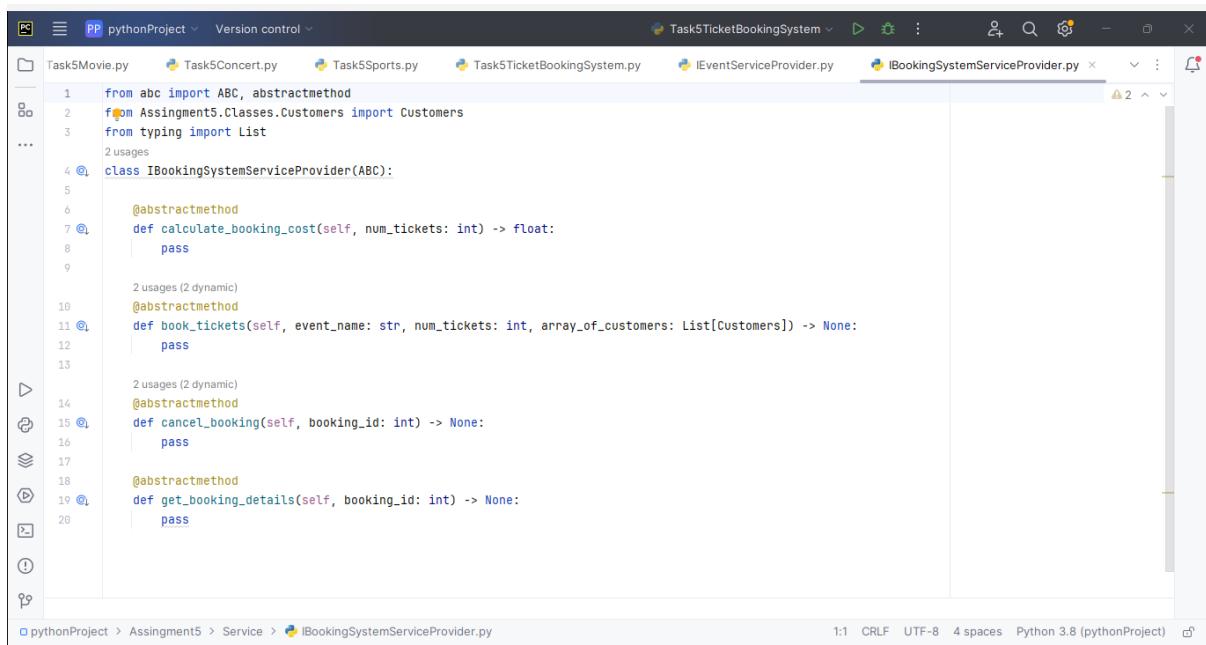
6. Create interface/abstract class IBookingSystemServiceProvider with following methods:

`calculate_booking_cost(num_tickets)`: Calculate and set the total cost of the booking.

`book_tickets(eventname:str, num_tickets, arrayOfCustomer)`: Book a specified number of tickets for an event. for each tickets customer object should be created and stored in array also should update the attributes of Booking class.

`cancel_booking(booking_id)`: Cancel the booking and update the available seats.

`get_booking_details(booking_id)`: get the booking details.



```
from abc import ABC, abstractmethod
from Assingment5.Classes.Customers import Customers
from typing import List

class IBookingSystemServiceProvider(ABC):

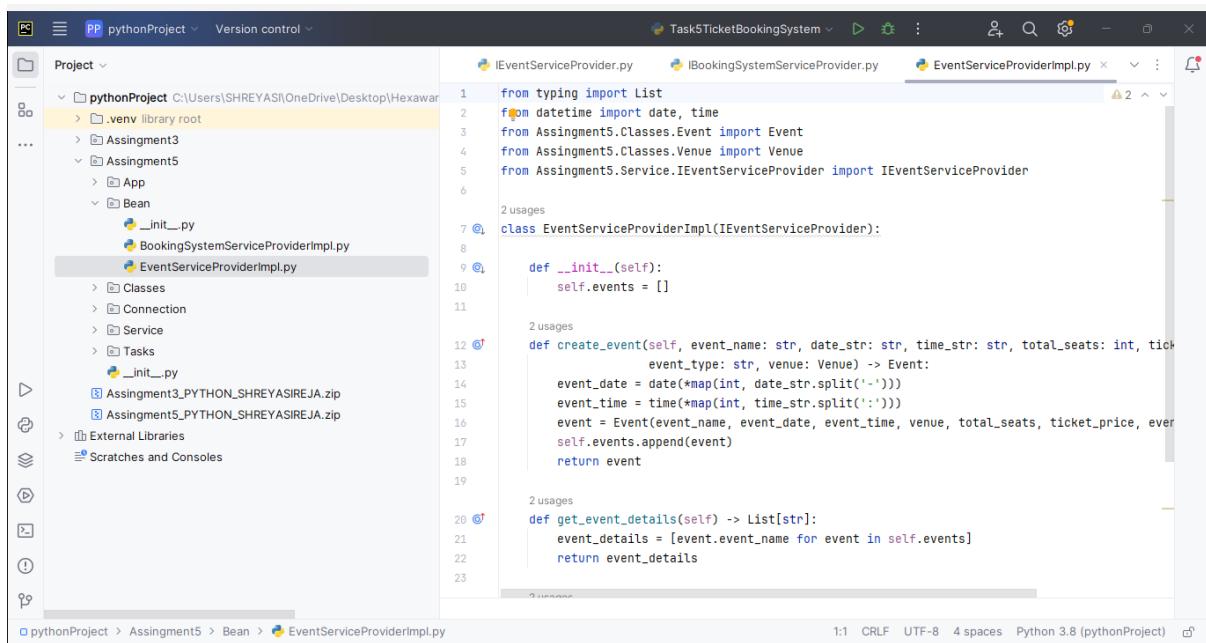
    @abstractmethod
    def calculate_booking_cost(self, num_tickets: int) -> float:
        pass

    2 usages (2 dynamic)
    @abstractmethod
    def book_tickets(self, event_name: str, num_tickets: int, array_of_customers: List[Customers]) -> None:
        pass

    2 usages (2 dynamic)
    @abstractmethod
    def cancel_booking(self, booking_id: int) -> None:
        pass

    @abstractmethod
    def get_booking_details(self, booking_id: int) -> None:
        pass
```

7. Create EventServiceProviderImpl class which implements IEventServiceProvider provide all implementation methods.



```
from typing import List
from datetime import date, time
from Assingment5.Classes.Event import Event
from Assingment5.Classes.Venue import Venue
from Assingment5.Service.IEventServiceProvider import IEventServiceProvider

class EventServiceProviderImpl(IEventServiceProvider):

    def __init__(self):
        self.events = []

    2 usages
    def create_event(self, event_name: str, date_str: str, time_str: str, total_seats: int, ticket_price: int,
                    event_type: str, venue: Venue) -> Event:
        event_date = date(*map(int, date_str.split('-')))
        event_time = time(*map(int, time_str.split(':')))
        event = Event(event_name, event_date, event_time, venue, total_seats, ticket_price, event_type)
        self.events.append(event)
        return event

    2 usages
    def get_event_details(self) -> List[str]:
        event_details = [event.event_name for event in self.events]
        return event_details
```

The screenshot shows the PyCharm IDE interface with the following details:

- Title Bar:** Shows "pythonProject" and "Version control".
- Toolbars:** Standard PyCharm toolbars for file operations, search, and navigation.
- Code Editor:** The main window displays the `EventServiceProviderImpl.py` file content. The code implements the `IEventServiceProvider` interface with methods for creating events, getting event details, and getting available tickets.
- Status Bar:** Shows "pythonProject > Assingment5 > Bean > EventServiceProviderImpl.py" and "1:1 CRLF UTF-8 4 spaces Python 3.8 (pythonProject)".

```
pythonProject pythonProject Version control Task5TicketBookingSystem EventServiceProviderImpl.py IEventServiceProvider.py IBookingSystemServiceProvider.py EventServiceProviderImpl.py .py Task5Sports.py Task5TicketBookingSystem.py IEventServiceProvider.py IBookingSystemServiceProvider.py EventServiceProviderImpl.py

10     self.events = []
11
12     2 usages
13     def create_event(self, event_name: str, date_str: str, time_str: str, total_seats: int, ticket_price: float,
14         event_type: str, venue: Venue) -> Event:
15         event_date = date(*map(int, date_str.split('-')))
16         event_time = time(*map(int, time_str.split(':')))
17         event = Event(event_name, event_date, event_time, venue, total_seats, ticket_price, event_type)
18         self.events.append(event)
19         return event
20
21     2 usages
22     def get_event_details(self) -> List[str]:
23         event_details = [event.event_name for event in self.events]
24         return event_details
25
26     2 usages
27     def get_available_no_of_tickets(self) -> int:
28         if not self.events:
29             return 0
30         return self.events[0].available_seats

pythonProject > Assingment5 > Bean > EventServiceProviderImpl.py 1:1 CRLF UTF-8 4 spaces Python 3.8 (pythonProject)
```

8. Create `BookingSystemServiceProviderImpl` class which implements `IBookingSystemServiceProvider` provide all implementation methods and inherits `EventServiceProviderImpl` class with following attributes.

Attributes

o array of events

The screenshot shows the PyCharm IDE interface with the following details:

- Title Bar:** Shows "pythonProject" and "Version control".
- Toolbars:** Standard PyCharm toolbars for file operations, search, and navigation.
- Code Editor:** The main window displays the `BookingSystemServiceProviderImpl.py` file content. It inherits from `EventServiceProviderImpl` and implements the `IBookingSystemServiceProvider` interface with methods for calculating booking cost and booking tickets.
- Status Bar:** Shows "pythonProject > Assingment5 > Bean > BookingSystemServiceProviderImpl.py" and "3:52 CRLF UTF-8 4 spaces Python 3.8 (pythonProject)".

```
pythonProject pythonProject Version control Task5TicketBookingSystem EventServiceProviderImpl.py IEventServiceProvider.py IBookingSystemServiceProvider.py BookingSystemServiceProviderImpl.py .py Task5Sports.py Task5TicketBookingSystem.py IEventServiceProvider.py IBookingSystemServiceProvider.py EventServiceProviderImpl.py

2     from Assingment5.Classes.Booking import Booking
3     from Assingment5.Classes.Customers import Customers
4     from Assingment5.Bean.EventServiceProviderImpl import EventServiceProviderImpl
5     from Assingment5.Service.IBookingSystemServiceProvider import IBookingSystemServiceProvider
6
7
8     4 usages
9     class BookingSystemServiceProviderImpl(EventServiceProviderImpl, IBookingSystemServiceProvider):
10
11         def __init__(self):
12             super().__init__()
13
14         1 usage
15         def calculate_booking_cost(self, num_tickets: int) -> float:
16             if num_tickets > 0:
17                 return num_tickets * self.selected_event.ticket_price
18             else:
19                 print("Invalid number of tickets.")
20                 return 0.0
21
22         4 usages (2 dynamic)
23         def book_tickets(self, event_name: str, num_tickets: int, array_of_customers: List[Customers]) -> None:
24             global booking
25             self.selected_event = None
26
27             for event in self.events:
28                 if event.event_name == event_name:
29                     self.selected_event = event
30
31             if self.selected_event is None:
32                 print("Event not found")
33             else:
34                 if num_tickets > 0:
35                     if num_tickets <= self.selected_event.available_seats:
36                         self.selected_event.available_seats -= num_tickets
37                         booking.append((self.selected_event.event_name, num_tickets))
38                     else:
39                         print("Not enough seats available")
40
41             return None

pythonProject > Assingment5 > Bean > BookingSystemServiceProviderImpl.py 3:52 CRLF UTF-8 4 spaces Python 3.8 (pythonProject)
```

The screenshot shows the PyCharm IDE interface with the 'pythonProject' selected in the top bar. The main editor window displays the `BookingSystemServiceProviderImpl.py` file. The code implements a service provider for a ticket booking system, handling events, customers, bookings, and their details.

```
24     for event in self.events:
25         if event.event_name == event_name:
26             self.selected_event = event
27             break
28
29         if self.selected_event is not None:
30             for _ in range(num_tickets):
31                 customer_name = input("Enter customer name: ")
32                 customer = Customers(customer_name)
33                 array_of_customers.append(customer)
34
35                 total_cost = self.calculate_booking_cost(num_tickets)
36                 booking = Booking(array_of_customers, self.selected_event, num_tickets)
37                 booking.total_cost = total_cost
38                 booking.display_booking_details()
39
40     else:
41         print(f"\nEvent with name '{event_name}' not found.")
42
43     4 usages (2 dynamic)
44     def cancel_booking(self, booking_id: int) -> None:
45         for event in self.events:
46             for booking in event.bookings:
47                 if booking.booking_id == booking_id:
48                     event.available_seats += booking.num_tickets
49                     event.bookings.remove(booking)
50                     print("\nBooking with ID {} canceled successfully.".format(booking_id))
51
52     def get_booking_details(self, booking_id: int) -> None:
53         for event in self.events:
54             for booking in event.bookings:
55                 if booking.booking_id == booking_id:
56                     print("\nBooking Details:")
57                     booking.display_booking_details()
58
59     print("\nBooking with ID {} not found.".format(booking_id))
```

The screenshot shows the PyCharm IDE interface with the 'pythonProject' selected in the top bar. The main editor window displays the `BookingSystemServiceProviderImpl.py` file, which has been modified to include a new method `get_booking_details` and a new line in the `cancel_booking` method.

```
39     else:
40         print(f"\nEvent with name '{event_name}' not found.")
41
42     4 usages (2 dynamic)
43     def cancel_booking(self, booking_id: int) -> None:
44         for event in self.events:
45             for booking in event.bookings:
46                 if booking.booking_id == booking_id:
47                     event.available_seats += booking.num_tickets
48                     event.bookings.remove(booking)
49                     print("\nBooking with ID {} canceled successfully.".format(booking_id))
50
51     def get_booking_details(self, booking_id: int) -> None:
52         for event in self.events:
53             for booking in event.bookings:
54                 if booking.booking_id == booking_id:
55                     print("\nBooking Details:")
56                     booking.display_booking_details()
57
58     print("\nBooking with ID {} not found.".format(booking_id))
```

9. Create TicketBookingSystem class and perform following operations:

Create a simple user interface in a main method that allows users to interact with the ticket booking system by entering commands such as "create_event", "book_tickets", "cancel_tickets", "get_available_seats", "get_event_details," and "exit."

10. Place the interface/abstract class in service package and interface/abstract class implementation class, all concrete class in bean package and TicketBookingSystem class in app package.

The screenshot shows the PyCharm IDE interface with the project navigation bar at the top. The left sidebar displays the project structure under 'pythonProject'. The main editor window contains the 'TicketBookingSystem.py' file. The code defines a class 'TicketBookingSystem' with methods for displaying a menu, handling user input, and performing various booking operations.

```
1   from Assingment5.Bean.BookingSystemServiceProviderImpl import BookingSystemService
2
3
4   usage
5   class TicketBookingSystem:
6
7       def __init__(self):
8           self.booking_system_provider = BookingSystemServiceProviderImpl()
9
10      usage
11      def display_menu(self):
12          print("\n===== Ticket Booking System Menu =====")
13          print("1. Create Event")
14          print("2. Get Event Details")
15          print("3. Book Tickets")
16          print("4. Cancel Booking")
17          print("5. Get Available Seats")
18          print("6. Exit")
19
20      usage
21      def main(self):
22          while True:
23              self.display_menu()
24              choice = input("Enter your choice (1-6): ")
25
26              if choice == "1":
27                  self.create_event()
28              elif choice == "2":
29                  self.get_event_details()
30              elif choice == "3":
31                  self.book_tickets()
32              elif choice == "4":
33                  self.cancel_booking()
34              elif choice == "5":
35                  self.get_available_seats()
36              elif choice == "6":
37                  print("Exiting the Ticket Booking System.")
38                  break
39              else:
40                  print("Invalid choice. Please enter a valid option.")
```

This screenshot shows the same PyCharm session with the 'TicketBookingSystem.py' code. The code has been modified to include a 'create_event' method and its implementation. The 'TicketBookingSystem' class now includes logic for creating events based on user input for event name, date, and time.

```
19
20
21      def main(self):
22          while True:
23              self.display_menu()
24              choice = input("Enter your choice (1-6): ")
25
26              if choice == "1":
27                  self.create_event()
28              elif choice == "2":
29                  self.get_event_details()
30              elif choice == "3":
31                  self.book_tickets()
32              elif choice == "4":
33                  self.cancel_booking()
34              elif choice == "5":
35                  self.get_available_seats()
36              elif choice == "6":
37                  print("Exiting the Ticket Booking System.")
38                  break
39              else:
40                  print("Invalid choice. Please enter a valid option.")
```

The screenshot shows the PyCharm IDE interface with the 'TicketBookingSystem.py' file open. The code implements a ticket booking system with methods for creating events, getting event details, and booking tickets.

```
40     def create_event(self):
41         event_name = input("Enter event name: ")
42         date = input("Enter event date (YYYY-MM-DD): ")
43         time = input("Enter event time (HH:MM): ")
44         total_seats = int(input("Enter total seats: "))
45         ticket_price = float(input("Enter ticket price: "))
46         event_type = input("Enter event type (Movie/Concert/Sports): ")
47         venue_name = input("Enter venue name: ")
48
49         self.booking_system_provider.create_event(event_name, date, time, total_seats, ticket_price, event_type,
50                                                     venue_name)
51         print("Event created successfully!")
52
53     def get_event_details(self):
54         event_details = self.booking_system_provider.get_event_details()
55         print("Event Details:")
56         for detail in event_details:
57             print(detail)
58
59     def book_tickets(self):
60         event_name = input("Enter the name of the event to book tickets: ")
61         num_tickets = int(input("Enter the number of tickets to book: "))
62         array_of_customer_names = []
63
64         for _ in range(num_tickets):
65             customer_name = input("Enter customer name: ")
66             array_of_customer_names.append(customer_name)
67         self.booking_system_provider.book_tickets(event_name, num_tickets, array_of_customer_names)
68         print("Booking successful!")
69
70     def cancel_booking(self):
71         customer_name = input("Enter the customer name to cancel the booking: ")
72         self.booking_system_provider.cancel_booking(customer_name)
73         print("Booking canceled successfully!")
74
75     def get_available_seats(self):
76         available_seats = self.booking_system_provider.get_available_no_of_tickets()
77         print(f"Available Seats: {available_seats}")
78
79 if __name__ == "__main__":
80     ticket_booking_system = TicketBookingSystem()
81     ticket_booking_system.main()
```

The screenshot shows the PyCharm IDE interface with the 'TicketBookingSystem.py' file open. The code has been modified to include additional methods: book_tickets, cancel_booking, and get_available_seats.

```
58     def book_tickets(self):
59         event_name = input("Enter the name of the event to book tickets: ")
60         num_tickets = int(input("Enter the number of tickets to book: "))
61         array_of_customer_names = []
62
63         for _ in range(num_tickets):
64             customer_name = input("Enter customer name: ")
65             array_of_customer_names.append(customer_name)
66         self.booking_system_provider.book_tickets(event_name, num_tickets, array_of_customer_names)
67         print("Booking successful!")
68
69     def cancel_booking(self):
70         customer_name = input("Enter the customer name to cancel the booking: ")
71         self.booking_system_provider.cancel_booking(customer_name)
72         print("Booking canceled successfully!")
73
74     def get_available_seats(self):
75         available_seats = self.booking_system_provider.get_available_no_of_tickets()
76         print(f"Available Seats: {available_seats}")
77
78 if __name__ == "__main__":
79     ticket_booking_system = TicketBookingSystem()
80     ticket_booking_system.main()
```

Output:-

```
58     def book_tickets(self):
59         event_name = input("Enter the name of the event to book tickets: ")
60         num_tickets = int(input("Enter the number of tickets to book: "))
61         array_of_customer_names = []
62
63         for _ in range(num_tickets):
64             customer_name = input("Enter customer name: ")
65             array_of_customer_names.append(customer_name)
66         self.booking_system_provider.book_tickets(event_name, num_tickets, array_of_customer_names)
TicketBookingSystem > book_tickets() > for _ in range(num_tickets)
```

Run TicketBookingSystem

4. Cancel Booking
5. Get Available Seats
6. Exit

Enter your choice (1-6): 1

Enter event name: ABC

Enter event date (YYYY-MM-DD): 2024-02-03

Enter event time (HH:MM): 14:46

Enter total seats: 20

Enter ticket price: 10

Enter event type (Movie/Concert/Sports): Movie

Enter venue name: pvr

Event created successfully!

Task 9: Exception Handling

throw the exception whenever needed and Handle in main method,

1. EventNotFoundException throw this exception when user try to book the tickets for Event not listed in the menu.
2. InvalidBookingIDException throw this exception when user entered the invalid bookingId when he tries to view the booking or cancel the booking.
3. NullPointerException handle in main method.

Throw these exceptions from the methods in TicketBookingSystem class. Make necessary changes to accommodate exception in the source code. Handle all these exceptions from the main program.

py pythonProject Version control Task5TicketBookingSystem

```
from Assingment5.Bean.BookingSystemServiceProviderImpl import BookingSystemServiceProviderImpl
class EventNotFoundException(Exception):
    pass
class InvalidBookingIDException(Exception):
    pass
class CustomNullPointerException(Exception):
    pass
class TicketBookingExceptionHandler:
    def handle_event_not_found(self, event_name):
        raise EventNotFoundException(f"Event '{event_name}' not found in the menu.")
    def handle_invalid_booking_id(self):
        raise InvalidBookingIDException("Invalid Booking ID.")
    def handle_null_pointer(self):
        raise CustomNullPointerException("Null Pointer Exception.")

TicketBookingSystem > cancel_booking() > except Exception as e
```

pythonProject > Assingment5 > App > ExceptionHandling.py

py pythonProject Version control Task5TicketBookingSystem

```
def handle_null_pointer(self):
    raise CustomNullPointerException("Null Pointer Exception.")

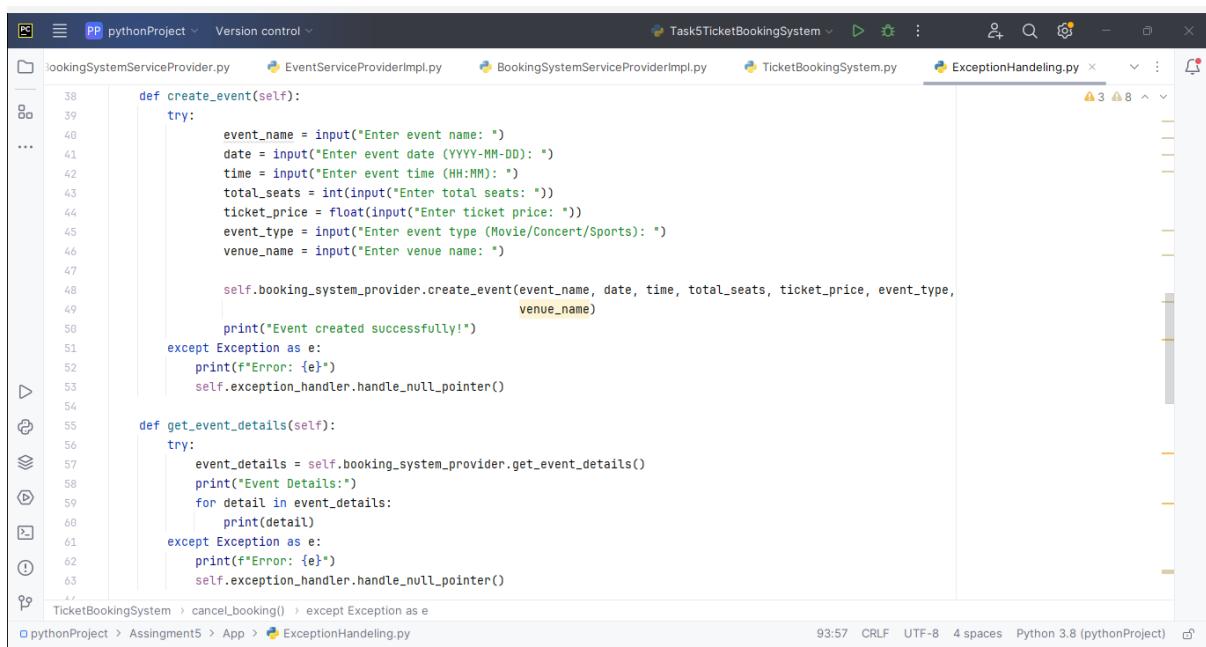
class TicketBookingSystem:
    def __init__(self):
        self.booking_system_provider = BookingSystemServiceProviderImpl()
        self.exception_handler = TicketBookingExceptionHandler()

    def display_menu(self):
        print("\n==== Ticket Booking System Menu ====")
        print("1. Create Event")
        print("2. Get Event Details")
        print("3. Book Tickets")
        print("4. Cancel Booking")
        print("5. Get Available Seats")
        print("6. Exit")

    def create_event(self):
        try:
            event_name = input("Enter event name: ")
            date = input("Enter event date (YYYY-MM-DD): ")
            time = input("Enter event time (HH:MM): ")
            total_seats = int(input("Enter total seats: "))
            ticket_price = float(input("Enter ticket price: "))
            event_type = input("Enter event type (Movie/Concert/Sports): ")
            venue_name = input("Enter venue name: ")
        except Exception as e:
            self.exception_handler.handle_null_pointer()

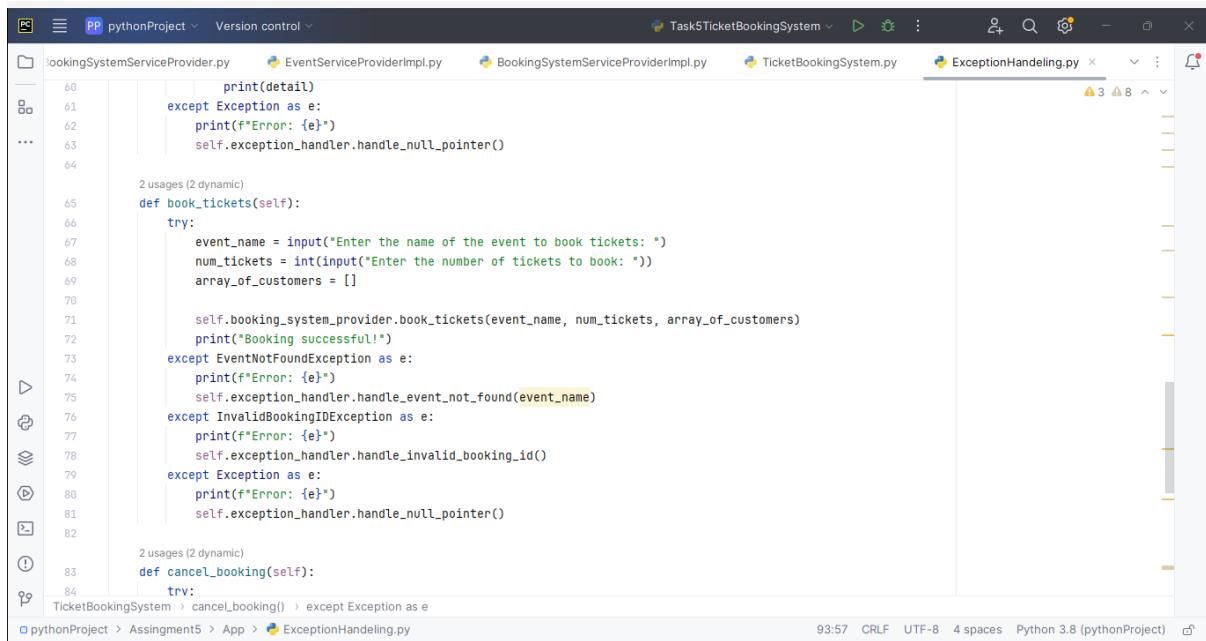
TicketBookingSystem > cancel_booking() > except Exception as e
```

pythonProject > Assingment5 > App > ExceptionHandling.py



```
38     def create_event(self):
39         try:
40             event_name = input("Enter event name: ")
41             date = input("Enter event date (YYYY-MM-DD): ")
42             time = input("Enter event time (HH:MM): ")
43             total_seats = int(input("Enter total seats: "))
44             ticket_price = float(input("Enter ticket price: "))
45             event_type = input("Enter event type (Movie/Concert/Sports): ")
46             venue_name = input("Enter venue name: ")
47
48             self.booking_system_provider.create_event(event_name, date, time, total_seats, ticket_price, event_type,
49                                                 venue_name)
49             print("Event created successfully!")
50         except Exception as e:
51             print(f"Error: {e}")
52             self.exception_handler.handle_null_pointer()
53
54     def get_event_details(self):
55         try:
56             event_details = self.booking_system_provider.get_event_details()
57             print("Event Details:")
58             for detail in event_details:
59                 print(detail)
60         except Exception as e:
61             print(f"Error: {e}")
62             self.exception_handler.handle_null_pointer()
63
64
TicketBookingSystem > cancel_booking() > except Exception as e
pythonProject > Assingment5 > App > ExceptionHandling.py
```

93:57 CRLF UTF-8 4 spaces Python 3.8 (pythonProject)



```
60     print(detail)
61     except Exception as e:
62         print(f"Error: {e}")
63         self.exception_handler.handle_null_pointer()
64
65     2 usages (2 dynamic)
66     def book_tickets(self):
67         try:
68             event_name = input("Enter the name of the event to book tickets: ")
69             num_tickets = int(input("Enter the number of tickets to book: "))
70             array_of_customers = []
71
72             self.booking_system_provider.book_tickets(event_name, num_tickets, array_of_customers)
73             print("Booking successful!")
74         except EventNotFoundException as e:
75             print(f"Error: {e}")
76             self.exception_handler.handle_event_not_found(event_name)
77         except InvalidBookingIDException as e:
78             print(f"Error: {e}")
79             self.exception_handler.handle_invalid_booking_id()
80         except Exception as e:
81             print(f"Error: {e}")
82             self.exception_handler.handle_null_pointer()
83
84     2 usages (2 dynamic)
85     def cancel_booking(self):
86         try:
87
TicketBookingSystem > cancel_booking() > except Exception as e
pythonProject > Assingment5 > App > ExceptionHandling.py
```

93:57 CRLF UTF-8 4 spaces Python 3.8 (pythonProject)

```
def cancel_booking(self):
    try:
        customer_name = input("Enter the customer name to cancel the booking: ")
        self.booking_system_provider.cancel_booking(customer_name)
        print("Booking canceled successfully!")
    except InvalidBookingIDException as e:
        print(f"Error: {e}")
        self.exception_handler.handle_invalid_booking_id()
    except Exception as e:
        print(f"Error: {e}")
        self.exception_handler.handle_null_pointer()

def get_available_seats(self):
    try:
        available_seats = self.booking_system_provider.get_available_no_of_tickets()
        print(f"Available Seats: {available_seats}")
    except Exception as e:
        print(f"Error: {e}")
        self.exception_handler.handle_null_pointer()
```

Task 11: Database Connectivity.

6. Create DBUtil class and add the following method.

static getDBConn():Connection Establish a connection to the database and return Connection reference

```
from mysql.connector import connect

class DBUtil:
    def __init__(self, host, user, password, port, database):
        self.connection = connect(
            host=host,
            user=user,
            password=password,
            port=port,
            database=database
        )
        self.cursor = self.connection.cursor()

    def execute_query(self, query, values=None):
        try:
            self.cursor.execute(query, values)
            self.connection.commit()
            self.cursor.fetchall()
        except Exception as e:
            print(f"Query Execution Error! {e}")
            self.connection.rollback()

if __name__ == "__main__":
    pass
```

```
iceProvider.py EventServiceProviderImpl.py BookingSystemServiceProviderImpl.py TicketBookingSystem.py DBUtil.py ExceptionHandling.py
1 usage
def fetch_all(self, query, values=None):
    try:
        self.cursor.execute(query, values)
        result = self.cursor.fetchall()
        return result
    except Exception as e:
        print(f"FetchAll Error: {e}")
        self.connection.rollback()
        return None

1 usage
def fetch_one(self, query, values=None):
    try:
        self.cursor.execute(query, values)
        result = self.cursor.fetchone()
        return result
    except Exception as e:
        print(f"FetchOne Error: {e}")
        self.connection.rollback()
        return None

1 usage
def close_connection(self):
    self.cursor.fetchall()
    self.cursor.close()
    self.connection.close()

if __name__ == "__main__":

```

pythonProject > Assingment5 > Connection > DBUtil.py

```
iceProvider.py EventServiceProviderImpl.py BookingSystemServiceProviderImpl.py TicketBookingSystem.py DBUtil.py ExceptionHandling.py
44
45
46
47
48
49 D if __name__ == "__main__":
50     host = "localhost"
51     user = "root"
52     password = "Root"
53     port = 3306
54     database = "ticketbookingsystem"

55
56     db_util = DBUtil(host, user, password, port, database)

57
58     query = "SELECT * FROM Event"
59
60     db_util.execute_query(query)
61
62     result_all = db_util.fetch_all(query)
63     for result in result_all:
64         print(result)
65
66     result_one = db_util.fetch_one(query)
67     if result_one:
68         print(result_one[0])
69     else:
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
5510
5511
5512
5513
5514
5515
5516
5517
5518
5519
5520
5521
5522
5523
5524
5525
5526
5527
5528
5529
5530
5531
5532
5533
5534
5535
5536
5537
5538
5539
5540
5541
5542
5543
5544
5545
5546
5547
5548
5549
55410
55411
55412
55413
55414
55415
55416
55417
55418
55419
55420
55421
55422
55423
55424
55425
55426
55427
55428
55429
55430
55431
55432
55433
55434
55435
55436
55437
55438
55439
55440
55441
55442
55443
55444
55445
55446
55447
55448
55449
55450
55451
55452
55453
55454
55455
55456
55457
55458
55459
55460
55461
55462
55463
55464
55465
55466
55467
55468
55469
55470
55471
55472
55473
55474
55475
55476
55477
55478
55479
55480
55481
55482
55483
55484
55485
55486
55487
55488
55489
55490
55491
55492
55493
55494
55495
55496
55497
55498
55499
554100
554101
554102
554103
554104
554105
554106
554107
554108
554109
554110
554111
554112
554113
554114
554115
554116
554117
554118
554119
554120
554121
554122
554123
554124
554125
554126
554127
554128
554129
554130
554131
554132
554133
554134
554135
554136
554137
554138
554139
554140
554141
554142
554143
554144
554145
554146
554147
554148
554149
554150
554151
554152
554153
554154
554155
554156
554157
554158
554159
554160
554161
554162
554163
554164
554165
554166
554167
554168
554169
554170
554171
554172
554173
554174
554175
554176
554177
554178
554179
554180
554181
554182
554183
554184
554185
554186
554187
554188
554189
554190
554191
554192
554193
554194
554195
554196
554197
554198
554199
554200
554201
554202
554203
554204
554205
554206
554207
554208
554209
554210
554211
554212
554213
554214
554215
554216
554217
554218
554219
554220
554221
554222
554223
554224
554225
554226
554227
554228
554229
554230
554231
554232
554233
554234
554235
554236
554237
554238
554239
554240
554241
554242
554243
554244
554245
554246
554247
554248
554249
554250
554251
554252
554253
554254
554255
554256
554257
554258
554259
554260
554261
554262
554263
554264
554265
554266
554267
554268
554269
554270
554271
554272
554273
554274
554275
554276
554277
554278
554279
554280
554281
554282
554283
554284
554285
554286
554287
554288
554289
554290
554291
554292
554293
554294
554295
554296
554297
554298
554299
554300
554301
554302
554303
554304
554305
554306
554307
554308
554309
554310
554311
554312
554313
554314
554315
554316
554317
554318
554319
554320
554321
554322
554323
554324
554325
554326
554327
554328
554329
554330
554331
554332
554333
554334
554335
554336
554337
554338
554339
554340
554341
554342
554343
554344
554345
554346
554347
554348
554349
554350
554351
554352
554353
554354
554355
554356
554357
554358
554359
554360
554361
554362
554363
554364
554365
554366
554367
554368
554369
554370
554371
554372
554373
554374
554375
554376
554377
554378
554379
554380
554381
554382
554383
554384
554385
554386
554387
554388
554389
554390
554391
554392
554393
554394
554395
554396
554397
554398
554399
554400
554401
554402
554403
554404
554405
554406
554407
554408
554409
554410
554411
554412
554413
554414
554415
554416
554417
554418
554419
554420
554421
554422
554423
554424
554425
554426
554427
554428
554429
554430
554431
554432
554433
554434
554435
554436
554437
554438
554439
554440
554441
554442
554443
554444
554445
554446
554447
554448
554449
554450
554451
554452
554453
554454
554455
554456
554457
554458
554459
554460
554461
554462
554463
554464
554465
554466
554467
554468
554469
554470
554471
554472
554473
554474
554475
554476
554477
554478
554479
554480
554481
554482
554483
554484
554485
554486
554487
554488
554489
554490
554491
554492
554493
554494
554495
554496
554497
554498
554499
554500
554501
554502
554503
554504
554505
554506
554507
554508
554509
554510
554511
554512
554513
554514
554515
554516
554517
554518
554519
5545110
5545111
5545112
5545113
5545114
5545115
5545116
5545117
5545118
5545119
55451100
55451101
55451102
55451103
55451104
55451105
55451106
55451107
55451108
55451109
55451110
55451111
55451112
55451113
55451114
55451115
55451116
55451117
55451118
55451119
554511100
554511101
554511102
554511103
554511104
554511105
554511106
554511107
554511108
554511109
554511110
554511111
554511112
554511113
554511114
554511115
554511116
554511117
554511118
554511119
5545111100
5545111101
5545111102
5545111103
5545111104
5545111105
5545111106
5545111107
5545111108
5545111109
5545111110
5545111111
5545111112
5545111113
5545111114
5545111115
5545111116
5545111117
5545111118
5545111119
55451111100
55451111101
55451111102
55451111103
55451111104
55451111105
55451111106
55451111107
55451111108
55451111109
55451111110
55451111111
55451111112
55451111113
55451111114
55451111115
55451111116
55451111117
55451111118
55451111119
554511111100
554511111101
554511111102
554511111103
554511111104
554511111105
554511111106
554511111107
554511111108
554511111109
554511111110
554511111111
554511111112
554511111113
554511111114
554511111115
554511111116
554511111117
554511111118
554511111119
5545111111100
5545111111101
5545111111102
5545111111103
5545111111104
5545111111105
5545111111106
5545111111107
5545111111108
5545111111109
5545111111110
5545111111111
5545111111112
5545111111113
5545111111114
5545111111115
5545111111116
5545111111117
5545111111118
5545111111119
55451111111100
55451111111101
55451111111102
55451111111103
55451111111104
55451111111105
55451111111106
55451111111107
55451111111108
55451111111109
55451111111110
55451111111111
55451111111112
55451111111113
55451111111114
55451111111115
55451111111116
55451111111117
55451111111118
55451111111119
554511111111100
554511111111101
554511111111102
554511111111103
554511111111104
554511111111105
554511111111106
554511111111107
554511111111108
554511111111109
554511111111110
554511111111111
554511111111112
554511111111113
554511111111114
554511111111115
554511111111116
554511111111117
554511111111118
554511111111119
5545111111111100
5545111111111101
5545111111111102
5545111111111103
5545111111111104
5545111111111105
5545111111111106
5545111111111107
5545111111111108
5545111111111109
5545111111111110
5545111111111111
5545111111111112
5545111111111113
5545111111111114
5545111111111115
5545111111111116
5545111111111117
5545111111111118
5545111111111119
55451111111111100
55451111111111101
55451111111111102
55451111111111103
55451111111111104
55451111111111105
55451111111111106
55451111111111107
55451111111111108
55451111111111109
55451111111111110
55451111111111111
55451111111111112
55451111111111113
55451111111111114
55451111111111115
55451111111111116
55451111111111117
55451111111111118
55451111111111119
554511111111111100
554511111111111101
554511111111111102
554511111111111103
554511111111111104
554511111111111105
554511111111111106
554511111111111107
554511111111111108
554511111111111109
554511111111111110
554511111111111111
554511111111111112
554511111111111113
554511111111111114
554511111111111115
554511111111111116
554511111111111117
554511111111111118
554511111111111119
5545111111111111100
5545111111111111101
5545111111111111102
5545111111111111103
5545111111111111104
5545111111111111105
5545111111111111106
5545111111111111107
5545111111111111108
5545111111111111109
5545111111111111110
5545111111111111111
5545111111111111112
5545111111111111113
5545111111111111114
5545111111111111115
5545111111111111116
5545111111111111117
5545111111111111118
5545111111111111119
55451111111111111100
55451111111111111101
55451111111111111102
55451111111111111103
55451111111111111104
55451111111111111105
55451111111111111106
55451111111111111107
55451111111111111108
55451111111111111109
55451111111111111110
55451111111111111111
55451111111111111112
55451111111111111113
55451111111111111114
55451111111111111115
55451111111111111116
55451111111111111117
55451111111111111118
55451111111111111119
554511111111111111100
554511111111111111101
554511111111111111102
554511111111111111103
554511111111111111104
554511111111111111105
554511111111111111106
554511111111111111107
554511111111111111108
554511111111111111109
554511111111111111110
554511111111111111111
554511111111111111112
554511111111111111113
554511111111111111114
554511111111111111115
554511111111111111116
554511111111111111117
554511111111111111118
554511111111111111119
5545111111111111111100
5545111111111111111101
5545111111111111111102
5545111111111111111103
5545111111111111111104
5545111111111111111105
5545111111111111111106
5545111111111111111107
5545111111111111111108
5545111111111111111109
5545111111111111111110
5545111111111111111111
5545111111111111111112
5545111111111111111113
5545111111111111111114
5545111111111111111115
5545111111111111111116
5545111111111111111117
5545111111111111111118
5545111111111111111119
55451111111111111111100
55451111111111111111101
55451111111111111111102
55451111111111111111103
55451111111111111111104
55451111111111111111105
55451111111111111
```

The screenshot shows the PyCharm IDE interface with the DBUtil.py file open. The code is as follows:

```
34     db_util = DBUtil(host, user, password, port, database)
35
36     query = "SELECT * FROM Event"
37
38     db_util.execute_query(query)
39
40     result_all = db_util.fetch_all(query)
41     for result in result_all:
42         print(result)
43
44     result_one = db_util.fetch_one(query)
45     if result_one:
46         print(result_one[0])
47     else:
48         print("Error fetching one row.")
49         result_one[0]
50
51     db_util.close_connection()
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
279
280
281
282
283
284
285
286
287
288
289
289
290
291
292
293
294
295
296
297
298
299
299
300
301
302
303
304
305
306
307
308
309
309
310
311
312
313
314
315
316
317
318
319
319
320
321
322
323
324
325
326
327
328
329
329
330
331
332
333
334
335
336
337
338
339
339
340
341
342
343
344
345
346
347
348
349
349
350
351
352
353
354
355
356
357
358
359
359
360
361
362
363
364
365
366
367
368
369
369
370
371
372
373
374
375
376
377
378
379
379
380
381
382
383
384
385
386
387
388
389
389
390
391
392
393
394
395
396
397
398
399
399
400
401
402
403
404
405
406
407
408
409
409
410
411
412
413
414
415
416
417
418
419
419
420
421
422
423
424
425
426
427
428
429
429
430
431
432
433
434
435
436
437
438
439
439
440
441
442
443
444
445
446
447
448
449
449
450
451
452
453
454
455
456
457
458
459
459
460
461
462
463
464
465
466
467
468
469
469
470
471
472
473
474
475
476
477
478
479
479
480
481
482
483
484
485
486
487
488
489
489
490
491
492
493
494
495
496
497
498
499
499
500
501
502
503
504
505
506
507
508
509
509
510
511
512
513
514
515
516
517
518
519
519
520
521
522
523
524
525
526
527
528
529
529
530
531
532
533
534
535
536
537
538
539
539
540
541
542
543
544
545
546
547
548
549
549
550
551
552
553
554
555
556
557
558
559
559
560
561
562
563
564
565
566
567
568
569
569
570
571
572
573
574
575
576
577
578
579
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
598
599
599
600
601
602
603
604
605
606
607
608
609
609
610
611
612
613
614
615
616
617
618
619
619
620
621
622
623
624
625
626
627
628
629
629
630
631
632
633
634
635
636
637
638
639
639
640
641
642
643
644
645
646
647
648
649
649
650
651
652
653
654
655
656
657
658
659
659
660
661
662
663
664
665
666
667
668
669
669
670
671
672
673
674
675
676
677
678
679
679
680
681
682
683
684
685
686
687
687
688
689
689
690
691
692
693
694
695
696
697
697
698
699
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
717
718
719
719
720
721
721
722
723
724
725
726
727
728
729
729
730
731
732
733
734
735
736
737
738
739
739
740
741
742
743
744
745
746
747
748
749
749
750
751
752
753
754
755
756
757
758
759
759
760
761
762
763
764
765
766
767
768
769
769
770
771
772
773
774
775
776
777
778
779
779
780
781
782
783
784
785
786
787
787
788
789
789
790
791
792
793
794
795
796
797
797
798
799
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
816
817
817
818
819
819
820
821
822
823
824
825
826
827
828
829
829
830
831
832
833
834
835
836
837
838
838
839
840
841
842
843
844
845
846
847
848
848
849
850
851
852
853
854
855
856
857
858
859
859
860
861
862
863
864
865
866
867
868
869
869
870
871
872
873
874
875
876
877
878
879
879
880
881
882
883
884
885
886
887
887
888
889
889
890
891
892
893
894
895
896
897
897
898
899
899
900
901
902
903
904
905
906
907
908
909
909
910
911
912
913
914
915
916
917
917
918
919
919
920
921
922
923
924
925
926
927
928
929
929
930
931
932
933
934
935
936
937
938
938
939
940
941
942
943
944
945
946
947
947
948
949
949
950
951
952
953
954
955
956
957
958
959
959
960
961
962
963
964
965
966
967
968
968
969
969
970
971
972
973
974
975
976
977
978
978
979
980
981
982
983
984
985
986
987
987
988
989
989
990
991
992
993
994
995
996
997
997
998
999
999
1000
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1009
1010
1011
1012
1013
1014
1015
1016
1017
1017
1018
1019
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1047
1048
1049
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1068
1069
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1087
1088
1088
1089
1089
1090
1091
1092
1093
1094
1095
1096
1096
1097
1097
1098
1098
1099
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1109
1110
1111
1112
1113
1114
1115
1116
1117
1117
1118
1119
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1128
1129
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1138
1139
1139
1140
1141
1142
1143
1144
1145
1146
1147
1147
1148
1148
1149
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1168
1169
1169
1170
1171
1172
1173
1174
1175
1176
1177
1177
1178
1178
1179
1179
1180
1181
1182
1183
1184
1185
1186
1187
1187
1188
1188
1189
1189
1190
1191
1192
1193
1194
1195
1196
1197
1197
1198
1198
1199
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1209
1210
1211
1212
1213
1214
1215
1216
1217
1217
1218
1218
1219
1219
1220
1221
1222
1223
1224
1225
1226
1227
1227
1228
1228
1229
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1238
1239
1239
1240
1241
1242
1243
1244
1245
1246
1247
1247
1248
1248
1249
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1268
1269
1269
1270
1271
1272
1273
1274
1275
1276
1277
1277
1278
1278
1279
1279
1280
1281
1282
1283
1284
1285
1286
1287
1287
1288
1288
1289
1289
1290
1291
1292
1293
1294
1295
1296
1297
1297
1298
1298
1299
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1309
1310
1311
1312
1313
1314
1315
1316
1317
1317
1318
1318
1319
1319
1320
1321
1322
1323
1324
1325
1326
1327
1327
1328
1328
1329
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1338
1339
1339
1340
1341
1342
1343
1344
1345
1346
1347
1347
1348
1348
1349
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1368
1369
1369
1370
1371
1372
1373
1374
1375
1376
1377
1377
1378
1378
1379
1379
1380
1381
1382
1383
1384
1385
1386
1387
1387
1388
1388
1389
1389
1390
1391
1392
1393
1394
1395
1396
1397
1397
1398
1398
1399
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1409
1410
1411
1412
1413
1414
1415
1416
1417
1417
1418
1418
1419
1419
1420
1421
1422
1423
1424
1425
1426
1427
1427
1428
1428
1429
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1438
1439
1439
1440
1441
1442
1443
1444
1445
1446
1447
1447
1448
1448
1449
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1468
1469
1469
1470
1471
1472
1473
1474
1475
1476
1477
1477
1478
1478
1479
1479
1480
1481
1482
1483
1484
1485
1486
1487
1487
1488
1488
1489
1489
1490
1491
1492
1493
1494
1495
1496
1497
1497
1498
1498
1499
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1509
1510
1511
1512
1513
1514
1515
1516
1517
1517
1518
1518
1519
1519
1520
1521
1522
1523
1524
1525
1526
1527
1527
1528
1528
1529
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1538
1539
1539
1540
1541
1542
1543
1544
1545
1546
1547
1547
1548
1548
1549
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1568
1569
1569
1570
1571
1572
1573
1574
1575
1576
1577
1577
1578
1578
1579
1579
1580
1581
1582
1583
1584
1585
1586
1587
1587
1588
1588
1589
1589
1590
1591
1592
1593
1594
1595
1596
1597
1597
1598
1598
1599
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1609
1610
1611
1612
1613
1614
1615
1616
1617
1617
1618
1618
1619
1619
1620
1621
1622
1623
1624
1625
1626
1627
1627
1628
1628
1629
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1638
1639
1639
1640
1641
1642
1643
1644
1645
1646
1647
1647
1648
1648
1649
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1668
1669
1669
1670
1671
1672
1673
1674
1675
1676
1677
1677
1678
1678
1679
1679
1680
1681
1682
1683
1684
1685
1686
1687
1687
1688
1688
1689
1689
1690
1691
1692
1693
1694
1695
1696
1697
1697
1698
1698
1699
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1709
1710
1711
1712
1713
1714
1715
1716
1717
1717
1718
1718
1719
1719
1720
1721
1722
1723
1724
1725
1726
1727
1727
1728
1728
1729
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1738
1739
1739
1740
1741
1742
1743
1744
1745
1746
1747
1747
1748
1748
1749
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1768
1769
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1778
1779
1779
1780
1781
1782
1783
1784
1785
1786
1787
1787
1788
1788
1789
1789
1790
1791
1792
1793
1794
1795
1796
1797
1797
1798
1798
1799
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1809
1810
1811
1812
1813
1814
1815
1816
1817
1817
1818
1818
1819
1819
1820
1821
1822
1823
1824
1825
1826
1827
1827
1828
1828
1829
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1838
1839
1839
1840
1841
1842
1843
1844
1845
1846
1847
1847
1848
1848
1849
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1868
1869
1869
1870
1871
1872
1873
1874
1875
1876
1877
1877
1878
1878
1879
1879
1880
1881
1882
1883
1884
1885
1886
1887
1887
1888
1888
1889
1889
1890
1891
1892
1893
1894
1895
1896
1897
1897
1898
1898
1899
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1909
1910
1911
1912
1913
1914
1915
1916
1917
1917
1918
1918
1919
1919
1920
1921
1922
1923
1924
1925
1926
1927
1927
1928
1928
1929
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1938
1939
1939
1940
1941
1942
1943
1944
1945
1946
1947
1947
1948
1948
1949
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1968
1969
1969
1970
1971
1972
1973
1974
1975
1976
1977
1977
1978
1978
1979
1979
1980
1981
1982
1983
1984
1985
1986
1987
1987
1988
1988
1989
1989
1990
1991
1992
1993
1994
1995
1996
1997
1997
1998
1998
1999
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2009
2010
2011
2012
2013
2014
2015
2016
2017
2017
2018
2018
2019
2019
2020
2021
2022
2023
2024
2025
2026
2027
2027
2028
2028
2029
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2038
2039
2039
2040
2041
2042
2043
2044
2045
2046
2047
2047
2048
2048
2049
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2068
2069
2069
2070
2071
2072
2073
2074
2075
2076
2077
2077
2078
2078
2079
2079
2080
2081
2082
2083
2084
2085
2086
2087
2087
2088
2088
2089
2089
2090
2091
2092
2093
2094
2095
2096
2097
2097
2098
2098
2099
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2109
2110
2111
2112
2113
2114
2115
2116
2117

```