COMP1112 Object Oriented Programming
Spring 2020, Project #2, Design
Due: May 5, 2020 23:50

IŞIK UNIVERSITY
COMPUTER SCIENCE AND ENGINEERING

You are supposed to **design** an online ticket reservation and sales system for different types of events (a simpler version of Biletix.com).

Basically, there are events such as concerts, baseball games etc. There are customers who would like to buy/reserve tickets/seats for events; event holders who can add/modify/cancel events; and admins who are responsible for approving event requests sent by event holders.

`Event`: Stores

- name (for example: 39th Istanbul Film Festival), date, and venue. All can be changed by event holders.
- capacity: initially equal to the venue capacity. Increases or decreases by ticket sales.
- Unit price, can be changed
- One can buy ticket(s) for any kind of event.

There are four types of events:

- `Exhibition`: Reservation is not allowed.
- `Music`: tickets can be reserved.
- `Sport`: tickets can be reserved.
- `PerformingArts`: seats can be reserved/sold.

The difference between ticket and seat is that, when you buy/reserve a seat, you also buy/reserve a ticket. But when you reserve/buy a ticket you don't know where you will be sitting.

`Venue`:

- Name, address, contact person name, capacity are stored. None can be changed.

`Person`:

Any person can use your system to

- `displayEvents`(): displays all events. Should be **overloaded** to display events on a particular date, events in a particular venue, with a particular name, and type e.g., music, sports etc..
- `register`(): asks required info, and registers the person.

There are some other types of user:

- `RegisteredUser`: in addition to the capabilities of `Person`, registered users:
  - has username,
  - can `displayEmptySeats(), reserveTicketR`(), `reserveSeatR`() –for certain types of events –,`buyTicketR`(), `buySeatR`()–for certain types of events, `cancelReservationR`(), `returnTicketR`(). All methods take the event id as argument; reserve and buy methods takes also an integer representing the number of people, i.e., one can buy 8 tickets for example. Buy methods return the ticket issued.
- `EventHolders`: in addition to the `RegisteredUser` behavior, event holders can:
  - `addEventE`(): used to create events. Returns the created event instance.
  - `modifyEventE`(), `cancelEventE`(): both take event id as argument.
  - `displayGuestList`(): Event holders are able to display the list of guests who bought/reserved seats/tickets of events created by him/her.

- `Admin`: in addition to the `RegisteredUser` behavior, admin can see the event requests created by the event holders, and can perform following operations:
  - `approveEventRequestA()`, `rejectEventRequestA()`, `cancelEventA()`: all takes an `Event` instance as argument. In case of cancellation, all tickets issued should be refunded.

`reserveTicket()`: if the corresponding event has free spots, a number of tickets which is given as argument, are reserved.

`reserveSeat()`: same as `reserveTicket(),` the only difference is the reserved seat numbers are returned back.

`buyTicket():`if the corresponding event has free spots, a number of tickets which is given as argument, are sold.

`buySeat():` same as `buyTicket(),` except seat numbers are returned back.

`cancelReservation():` tickets and/or seats are freed.
`returnTicket():` tickets and/or seats are freed.

`Ticket` and `Seat`: you are expected to design these types. Note that none of the attributes can be changed for these types.

`TicketMaster`: This class represents the whole system. TestClass cannot have direct access to the above classes, but only the TicketMaster class can. Has the following attributes and behavior:

- list of events, list of venues, lists of registered users, admins, event holders, list of sold tickets, list of reserved tickets, **etc**.
- All methods mentioned in Person and its subclasses must have their corresponding methods to be able to invoke them.

Please note that, in all classes described above, some of the **attributes** to be introduced by the associations are **missing**. You are expected to **add** them. You are free to make any further assumptions.