Assignment 5

Use the design principles discussed in our class Software Design (COMP303) class over this term to complete this assignment.

Context

You are now in charge of creating a well-designed, well-documented software for EventBrite, the event management and ticketing website. The Event Manager from EventBrite, **Bob**, is the client who will be using your software to manage events and determine the revenue from events.

To start this assignment, download the baseline code that is attached. You may modify the base code appropriately, if needed.

Objective

Your software should allow Bob to use an object of the `EventManagement` class to schedule events, and **importantly, calculate the expected profit from different events** hosted by EventBrite. Bob has no knowledge of the Event class therefore he should **not** call any method of the Event classes directly.

EventBrite Code Base

The application in the base code so far has an Interface to represent an Event. The Event has getters for name, location, date, per-person ticket price, and the total number of tickets that can be issued, **all of which should be immutable fields for a concrete event**. The EventManagement class contains the list of events hosted on EventBrite.

Requirements

You are presented with the following requirements that must be fulfilled **using good design techniques and principles** discussed throughout the semester.

Hosting Different Events:

1. Modify the code to allow multiple types of Events to exist: Concert, Workshop, Gala, or Screening. Each type of event should be represented by a separate class with the following criteria:
    - Concerts have a field “artist” with the name of the artist who is performing. They also have a list of “VIPs” for the concert.
    - Workshops have a list of “prerequisites”, which is a list of existing Workshops that a participant must have attended before attending this one.
    - Galas have a list of “VIPs” for the party.
    - Screenings have a rating of “G”, “PG”, “PG-13”, or “R”.

Make sure to add methods to set and get these information appropriately. **[2 points]**

Festivals:

2. A Festival is a type of event in which multiple events can happen over a period of time. It has the following requirements:
    - The festival's ticket price is a subsidized price, equivalent to 20% of the price of all the events at the festival together.

- The total number of tickets available for a festival is equivalent to the number of tickets of the Event with the least number of tickets. For example, Bob should be able to create an event for the Montreal Jazz Festival during which three Concerts with 120, 100, and 150 tickets and two Galas with 50 and 100 tickets take place. Thus, the total number of tickets for the Jazz Festival is 50.
- The location of the Festival depends on the location of events within the Festival - if events are held at more than one location, the Festival's location should indicate that there are multiple. If all events are at the same location, then it should be set to the same location as the events.
- Set the date of the Festival to the date of the first event that will take place in the festival.
- Once created, the list of events in a Festival can not be changed. **[4]**

Calculating Revenue

3. Create a class called FilterResult that contains a list of events named 'aFilteredEvents'. Add functionality to the application that allows Bob to filter events based on either the price range of the event, or at a particular location. Bob should also be able to filter using both criteria or add future criteria himself. The call to filtering should return a FilterResult object with the list of events that match the criteria stored in 'aFilteredEvents'. **[6]**

4. Provide a way for Bob to calculate the total **expected profit** of a FilterResult. The profit is given by the sum of the profit of each event it contains. The profit of an individual event is given by the price per person for each event multiplied by the total number of tickets for that event multiplied by the profit per ticket (in percentage). The profit percentage per ticket for each event is provided by Bob when he wants to make the calculation of expected profit. For example, Bob may enter the following profit percentages per ticket for each event:
    - Concert: 60%
    - Workshop: 50%
    - Gala: 30%
    - Screening: 10%

The profit for a Festival is calculated based on the events it contains. Design this feature to avoid unnecessary coupling between the profit calculation and the events. Also provide a way for Bob to calculate the **total number of VIPs** attending events because they will be paid for their presence at the event. Make sure to use Polymorphism effectively. **[8]**

Other Requirements

5. Modify the code to allow the creation of events that are "coming soon". "Coming soon" events have a name, and date, but the location, ticket price, and total number of tickets are unknown. Ensure that calling the getters for unknown fields does not break the execution of the application. **[2]**

6. Do not allow Bob to create a new event with the same date and location as an existing event. **[2]**

7. Complete and update the methods in EventManagement allowing Bob to perform managing of the Events only via the EventManagement class. Assume Bob **can not** call methods of an Event directly. Make sure to check that input parameter values are valid. **[2]**

Supporting Requirements

8. **Tests:** You must create unit tests to test all aspects of your application. Be as thorough as possible so that you can guarantee to EventBrite that the entire application will work with no trouble. Demonstrate *at least* one test using a stub and one test using reflection, ensuring that they are used appropriately for meaningful test cases. Use and report the statement and branch coverage of your tests to see whether there were any parts of your code that you missed. Other developers and testers at EventBrite will look at your code and your unit tests to determine whether the application can be approved and is ready to be released **[5]**

9. **Documentation:** Document your code, so that developers can write client code for your software. Also, write a brief description of your software to explain the design techniques you used. Support this with two UML diagrams: (1) a class diagram representing the entire application, and (2) a sequence diagram that demonstrates the calculation of revenue for a FilterResult containing at least one of each type of Event (requirement 4). The EventBrite managers will look at this documentation of your application to give their approval for release. **[6]**

10. **Overall Design:** The overall solution of your assignment should be coherent. It should also demonstrate the proper application of all design techniques learnt in this course, where applicable. The evaluation of the overall design includes using proper Java and UML conventions, readability of the code, etc. **[3]**

Deliverables

The submission must include a zip file submitted to MyCourses -> Assignment -> Assignment5 named **'lastname_McGillID'** with the following:

1. The source code of your solution in a `src` folder;
2. A `Driver.java` file (located in the `src` folder) with a `main` method that exercises the main scenarios of the code;
3. The test files in a separate `test` folder
4. A pdf file of a short description of the design of your solution (major decisions, techniques used, trade-offs, etc.).

Good luck!