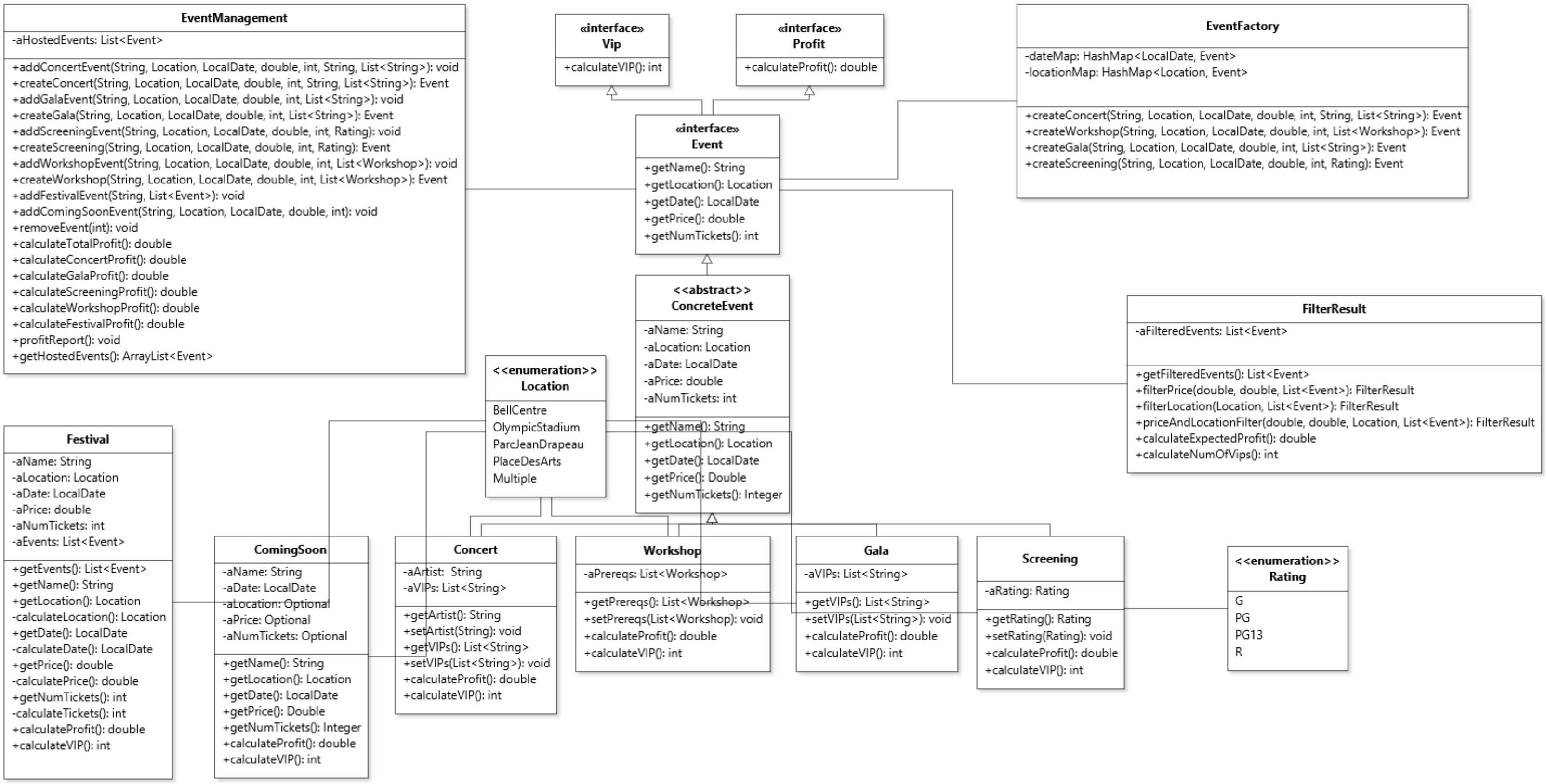
In order to allow for multiple events to exist, I created an abstract ConcreteEvent class which implements the Event interface and it is being extended by other specific event classes. The reasoning for this is that except for one or two attributes all events have the same attributes.

In order to implement the Festival event I created a Festival class which implements the event interface, and fields of the class are calculated through its own methods according to given requirements.

In order to implement FilterResult I used the filter design pattern. The expected profit and the total number of VIPs is calculated via polymorphism. I created two interfaces: Profit, and VIP. Profit interface only has one method which is calculateProfit and it is implemented by every event class. VIP interface too has only one method which is calculateVIP and it is implemented by classes that have VIPs. As a result during calculation of both profit and VIPS, I don't have to use instance of to calculate differently for each event.

In order to allow the creation of Coming Soon events I created a class ComingSoon with Optional type for fields that might be unknown. By using Optional type fields I avoid the null pointer exception.

In order to not allow Bob to create a new event with the same date and location I created a class called EventFactory which uses the flyweight design pattern. There are two hashmaps, one maps the event to a date, and the other maps the event to a location. Using these hashmaps we don't allow Bob to create new events with the same date and location.


EventManagement
-aHostedEvents: List<Event>
+addConcertEvent(String, Location, LocalDate, double, int, String, List<String>): void
+createConcert(String, Location, LocalDate, double, int, String, List<String>): Event
+addGalaEvent(String, Location, LocalDate, double, int, List<String>): void
+createGala(String, Location, LocalDate, double, int, List<String>): Event
+addScreeningEvent(String, Location, LocalDate, double, int, Rating): void
+createScreening(String, Location, LocalDate, double, int, Rating): Event
+addWorkshopEvent(String, Location, LocalDate, double, int, List<Workshop>): void
+createWorkshop(String, Location, LocalDate, double, int, List<Workshop>): Event
+addFestivalEvent(String, List<Event>): void
+addComingSoonEvent(String, Location, LocalDate, double, int): void
+removeEvent(int): void
+calculateTotalProfit(): double
+calculateConcertProfit(): double
+calculateGalaProfit(): double
+calculateScreeningProfit(): double
+calculateWorkshopProfit(): double
+calculateFestivalProfit(): double
+profitReport(): void
+getHostedEvents(): ArrayList<Event>
«interface»
Vip
+calculateVIP(): int
«interface»
Profit
+calculateProfit(): double
EventFactory
-dateMap: HashMap<LocalDate, Event>
-locationMap: HashMap<Location, Event>
+createConcert(String, Location, LocalDate, double, int, String, List<String>): Event
+createWorkshop(String, Location, LocalDate, double, int, List<Workshop>): Event
+createGala(String, Location, LocalDate, double, int, List<String>): Event
+createScreening(String, Location, LocalDate, double, int, Rating): Event
«interface»
Event
+getName(): String
+getLocation(): Location
+getDate(): LocalDate
+getPrice(): double
+getNumTickets(): int
<<abstract>>
ConcreteEvent
-aName: String
-aLocation: Location
-aDate: LocalDate
-aPrice: double
-aNumTickets: int
+getName(): String
+getLocation(): Location
+getDate(): LocalDate
+getPrice(): Double
+getNumTickets(): Integer
FilterResult
-aFilteredEvents: List<Event>
+getFilteredEvents(): List<Event>
+filterPrice(double, double, List<Event>): FilterResult
+filterLocation(Location, List<Event>): FilterResult
+priceAndLocationFilter(double, double, Location, List<Event>): FilterResult
+calculateExpectedProfit(): double
+calculateNumOfVips(): int
<<enumeration>>
Location
BellCentre
OlympicStadium
ParcJeanDrapeau
PlaceDesArts
Multiple
Festival
-aName: String
-aLocation: Location
-aDate: LocalDate
-aPrice: double
-aNumTickets: int
-aEvents: List<Event>
+getEvents(): List<Event>
+getName(): String
+getLocation(): Location
-calculateLocation(): Location
+getDate(): LocalDate
-calculateDate(): LocalDate
+getPrice(): double
-calculatePrice(): double
+getNumTickets(): int
-calculateTickets(): int
+calculateProfit(): double
+calculateVIP(): int
ComingSoon
-aName: String
-aDate: LocalDate
-aLocation: Optional
-aPrice: Optional
-aNumTickets: Optional
+getName(): String
+getLocation(): Location
+getDate(): LocalDate
+getPrice(): Double
+getNumTickets(): Integer
+calculateProfit(): double
+calculateVIP(): int
Concert
-aArtist: String
-aVIPs: List<String>
+getArtist(): String
+setArtist(String): void
+getVIPs(): List<String>
+setVIPs(List<String>): void
+calculateProfit(): double
+calculateVIP(): int
Workshop
-aPrereqs: List<Workshop>
+getPrereqs(): List<Workshop>
+setPrereqs(List<Workshop): void
+calculateProfit(): double
+calculateVIP(): int
Gala
-aVIPs: List<String>
+getVIPs(): List<String>
+setVIPs(List<String>): void
+calculateProfit(): double
+calculateVIP(): int
Screening
-aRating: Rating
+getRating(): Rating
+setRating(Rating): void
+calculateProfit(): double
+calculateVIP(): int
<<enumeration>>
Rating
G
PG
PG13
R