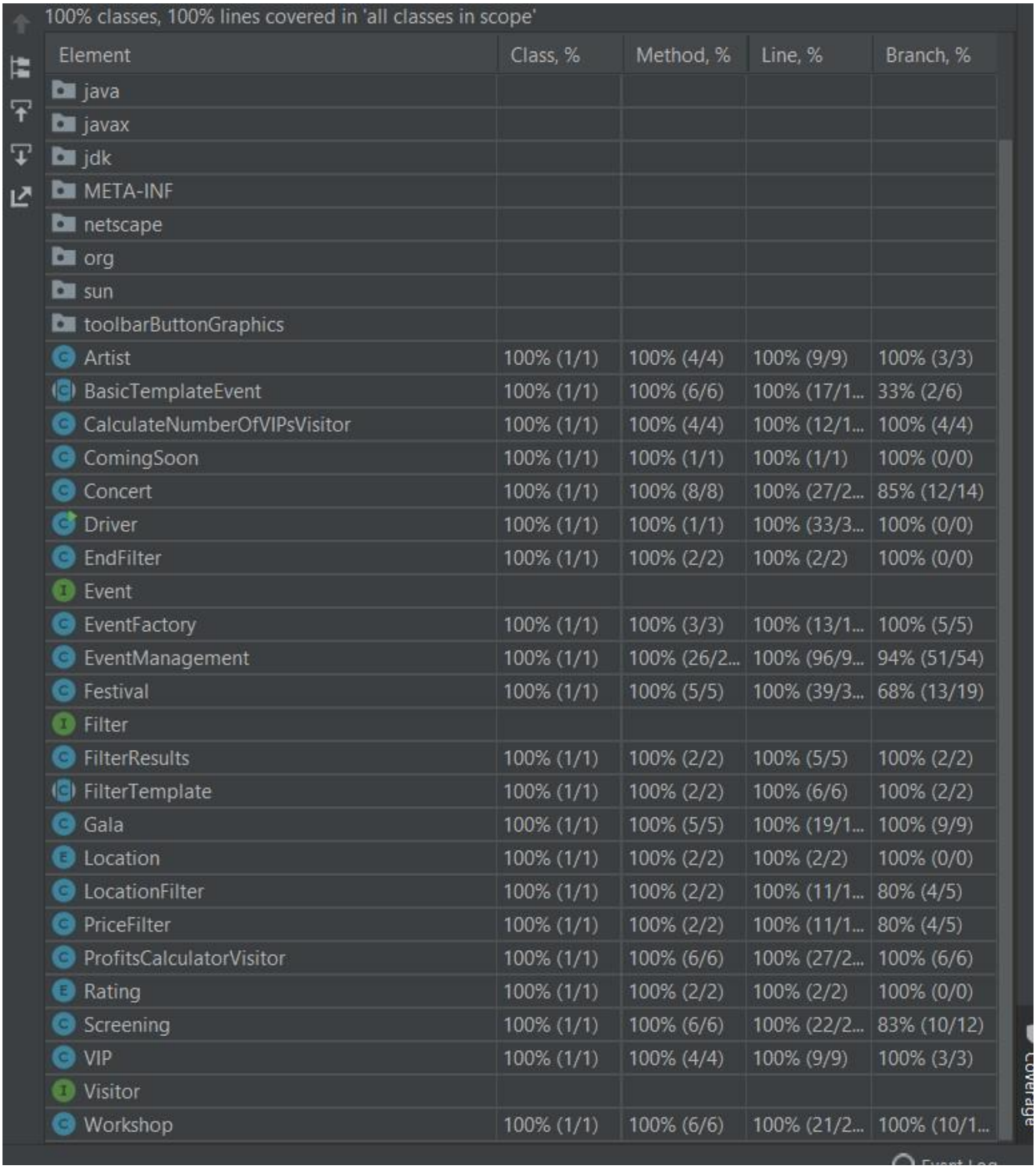
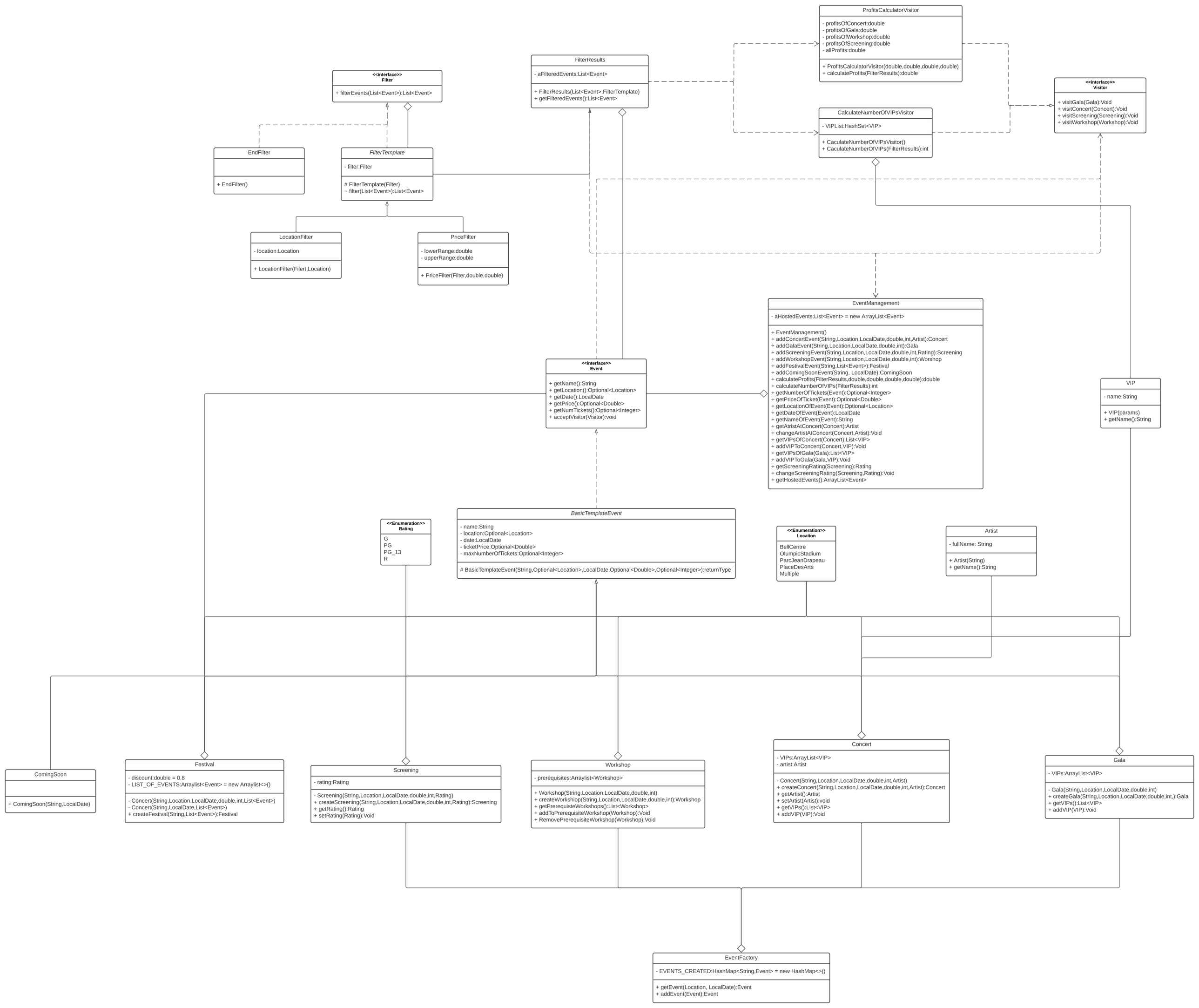
Major design Decisions:

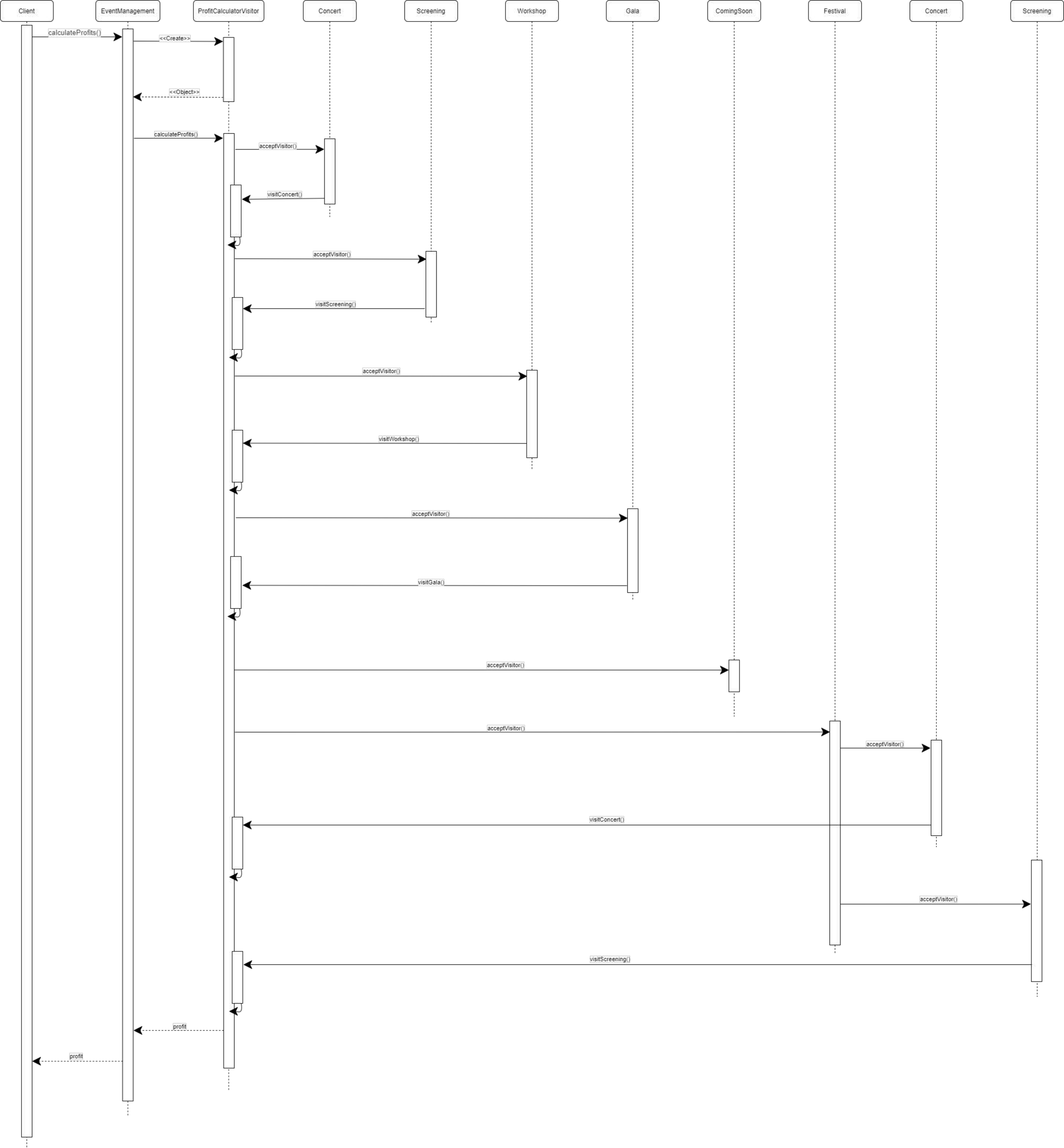
- Created an abstract class that has all the getters for the events. It also has the date, location, price of the ticket, number of tickets, and the name of the event in it. It also has a constructor to assign values to these fields. The accept visitor is made abstract so that each child of this abstract class has to implement it. This is so that we don't repeat code in all the classes.
- Used Enums for the ratings.
- Made classes for both the VIP and the Artist since they might want to add extra functionality to them.
- Added the requirements of each event in number on 1 in their own class. With setters and getters there.
- The Festival I found all the requirements in the createFestival function before creating the Festival object.
- The Festival class acted as/similar to the composite design pattern since it can have either other festivals or concrete Events inside it. Also, the acceptVisitor method calls the acceptVisitor method in the 4 concrete Event Objects in requirement 1.
- Used the Decorator design pattern to add filters dynamically and as many as needed. This is also very scalable since they just have to extend the FilterTemplate abstract class and then the class would be functional.
- Used the Visitor design pattern for calculating the profit and to calculate the number of VIPs. This makes it easier to join concrete algorithms with concrete Events. One downside of using this pattern is that if we were to add other concrete functional classes like the ones in question 1 we would have to go back and add extra methods to all the visitors. However, we don't then this pattern allows us to add extra functionality very easily and quickly.
- To make it possible to have ComingSoon events I made the location, price of the ticket, and the number of tickets of type Optional. However, there are checks (that throw an Illegal argument exception) in each concrete class so that we are sure to get all the required information.
- Used another class to manage the requirement of having only one event for each combination of date and location (flyweight factory). This is only applied to the 4 concrete Event classes in requirement 1.
- Made it so that Bob can use all the functionality of the events without ever having to deal with the Events directly. I also returned the objects that he tried to create back to him so that he can group them into lists and pass them as arguments or just use them as they are to change some of the Events mutable fields. I made this decision since we are told that "Bob cannot call methods of an event directly." Thus we can assume that he won't need and won't be able to change the fields in the events without ever needing to call them through the methods in their concrete Event classes.
- I used an EventStub when calling methods from the Event management to get information and used reflection when checking if the exact VIP is added to the Concert Object in the ConcertTest class in the test folder.

Coverage:




| Element | Class, % | Method, % | Line, % | Branch, % |
|---|---|---|---|---|
| java | | | | |
| javax | | | | |
| jdk | | | | |
| META-INF | | | | |
| netscape | | | | |
| org | | | | |
| sun | | | | |
| toolbarButtonGraphics | | | | |
| Artist | 100% (1/1) | 100% (4/4) | 100% (9/9) | 100% (3/3) |
| BasicTemplateEvent | 100% (1/1) | 100% (6/6) | 100% (17/1... | 33% (2/6) |
| CalculateNumberOfVIPsVisitor | 100% (1/1) | 100% (4/4) | 100% (12/1... | 100% (4/4) |
| ComingSoon | 100% (1/1) | 100% (1/1) | 100% (1/1) | 100% (0/0) |
| Concert | 100% (1/1) | 100% (8/8) | 100% (27/2... | 85% (12/14) |
| Driver | 100% (1/1) | 100% (1/1) | 100% (33/3... | 100% (0/0) |
| EndFilter | 100% (1/1) | 100% (2/2) | 100% (2/2) | 100% (0/0) |
| Event | | | | |
| EventFactory | 100% (1/1) | 100% (3/3) | 100% (13/1... | 100% (5/5) |
| EventManagement | 100% (1/1) | 100% (26/2... | 100% (96/9... | 94% (51/54) |
| Festival | 100% (1/1) | 100% (5/5) | 100% (39/3... | 68% (13/19) |
| Filter | | | | |
| FilterResults | 100% (1/1) | 100% (2/2) | 100% (5/5) | 100% (2/2) |
| FilterTemplate | 100% (1/1) | 100% (2/2) | 100% (6/6) | 100% (2/2) |
| Gala | 100% (1/1) | 100% (5/5) | 100% (19/1... | 100% (9/9) |
| Location | 100% (1/1) | 100% (2/2) | 100% (2/2) | 100% (0/0) |
| LocationFilter | 100% (1/1) | 100% (2/2) | 100% (11/1... | 80% (4/5) |
| PriceFilter | 100% (1/1) | 100% (2/2) | 100% (11/1... | 80% (4/5) |
| ProfitsCalculatorVisitor | 100% (1/1) | 100% (6/6) | 100% (27/2... | 100% (6/6) |
| Rating | 100% (1/1) | 100% (2/2) | 100% (2/2) | 100% (0/0) |
| Screening | 100% (1/1) | 100% (6/6) | 100% (22/2... | 83% (10/12) |
| VIP | 100% (1/1) | 100% (4/4) | 100% (9/9) | 100% (3/3) |
| Visitor | | | | |
| Workshop | 100% (1/1) | 100% (6/6) | 100% (21/2... | 100% (10/1... |


ProfitsCalculatorVisitor
- profitsOfConcert:double
- profitsOfGala:double
- profitsOfWorkshop:double
- profitsOfScreening:double
- allProfits:double
+ ProfitsCalculatorVisitor(double,double,double,double)
+ calculateProfits(FilterResults):double
FilterResults
- aFilteredEvents:List<Event>
+ FilterResults(List<Event>,FilterTemplate)
+ getFilteredEvents():List<Event>
<<interface>>
Filter
+ filterEvents(List<Event>):List<Event>
<<interface>>
Visitor
+ visitGala(Gala):Void
+ visitConcert(Concert):Void
+ visitScreening(Screening):Void
+ visitWorkshop(Workshop):Void
CalculateNumberOfVIPsVisitor
- VIPList:HashSet<VIP>
+ CaculateNumberOfVIPsVisitor()
+ CaculateNumberOfVIPs(FilterResults):int
EndFilter
+ EndFilter()
FilterTemplate
- filter:Filter
# FilterTemplate(Filter)
~ filter(List<Event>):List<Event>
LocationFilter
- location:Location
+ LocationFilter(Filert,Location)
PriceFilter
- lowerRange:double
- upperRange:double
+ PriceFilter(Filter,double,double)
EventManagement
- aHostedEvents:List<Event> = new ArrayList<Event>
+ EventManagement()
+ addConcertEvent(String,Location,LocalDate,double,int,Artist):Concert
+ addGalaEvent(String,Location,LocalDate,double,int):Gala
+ addScreeningEvent(String,Location,LocalDate,double,int,Rating):Screening
+ addWorkshopEvent(String,Location,LocalDate,double,int):Worshop
+ addFestivalEvent(String,List<Event>):Festival
+ addComingSoonEvent(String, LocalDate):ComingSoon
+ calculateProfits(FilterResults,double,double,double,double):double
+ calculateNumberOfVIPs(FilterResults):int
+ getNumberOfTickets(Event):Optional<Integer>
+ getPriceOfTicket(Event):Optional<Double>
+ getLocationOfEvent(Event):Optional<Location>
+ getDateOfEvent(Event):LocalDate
+ getNameOfEvent(Event):String
+ getAtristAtConcert(Concert):Artist
+ changeArtistAtConcert(Concert,Artist):Void
+ getVIPsOfConcert(Concert):List<VIP>
+ addVIPToConcert(Concert,VIP):Void
+ getVIPsOfGala(Gala):List<VIP>
+ addVIPToGala(Gala,VIP):Void
+ getScreeningRating(Screening):Rating
+ changeScreeningRating(Screening,Rating):Void
+ getHostedEvents():ArrayList<Event>
<<interface>>
Event
+ getName():String
+ getLocation():Optional<Location>
+ getDate():LocalDate
+ getPrice():Optional<Double>
+ getNumTickets():Optional<Integer>
+ acceptVisitor(Visitor):void
VIP
- name:String
+ VIP(params)
+ getName():String
BasicTemplateEvent
- name:String
- location:Optional<Location>
- date:LocalDate
- ticketPrice:Optional<Double>
- maxNumberOfTickets:Optional<Integer>
# BasicTemplateEvent(String,Optional<Location>,LocalDate,Optional<Double>,Optional<Integer>):returnType
<<Enumeration>>
Rating
G
PG
PG_13
R
<<Enumeration>>
Location
BellCentre
OlumpicStadium
ParcJeanDrapeau
PlaceDesArts
Multiple
Artist
- fullName: String
+ Artist(String)
+ getName():String
ComingSoon
+ ComingSoon(String,LocalDate)
Festival
- discount:double = 0.8
- LIST_OF_EVENTS:Arraylist<Event> = new Arraylist<>()
- Concert(String,Location,LocalDate,double,int,List<Event>)
- Concert(String,LocalDate,List<Event>)
+ createFestival(String,List<Event>):Festival
Screening
- rating:Rating
- Screening(String,Location,LocalDate,double,int,Rating)
+ createScreening(String,Location,LocalDate,double,int,Rating):Screening
+ getRating():Rating
+ setRating(Rating):Void
Workshop
- prerequisites:Arraylist<Workshop>
+ Workshop(String,Location,LocalDate,double,int)
+ createWorkshiop(String,Location,LocalDate,double,int):Workshop
+ getPrerequisteWorkshops():List<Workshop>
+ addToPrerequisiteWorkshop(Workshop):Void
+ RemovePrerequisiteWorkshop(Workshop):Void
Concert
- VIPs:ArrayList<VIP>
- artist:Artist
- Concert(String,Location,LocalDate,double,int,Artist)
+ createConcert(String,Location,LocalDate,double,int,Artist):Concert
+ getArtist():Artist
+ setArtist(Artist):void
+ getVIPs():List<VIP>
+ addVIP(VIP):Void
Gala
- VIPs:ArrayList<VIP>
- Gala(String,Location,LocalDate,double,int)
+ createGala(String,Location,LocalDate,double,int,):Gala
+ getVIPs():List<VIP>
+ addVIP(VIP):Void
EventFactory
- EVENTS_CREATED:HashMap<String,Event> = new HashMap<>()
+ getEvent(Location, LocalDate):Event
+ addEvent(Event):Event


Client
EventManagement
ProfitCalculatorVisitor
Concert
Screening
Workshop
Gala
ComingSoon
Festival
Concert
Screening
calculateProfits()
<<Create>>
<<Object>>
calculateProfits()
acceptVisitor()
visitConcert()
acceptVisitor()
visitScreening()
acceptVisitor()
acceptVisitor()
visitWorkshop()
acceptVisitor()
visitGala()
acceptVisitor()
acceptVisitor()
acceptVisitor()
acceptVisitor()
visitConcert()
acceptVisitor()
visitScreening()
profit
profit