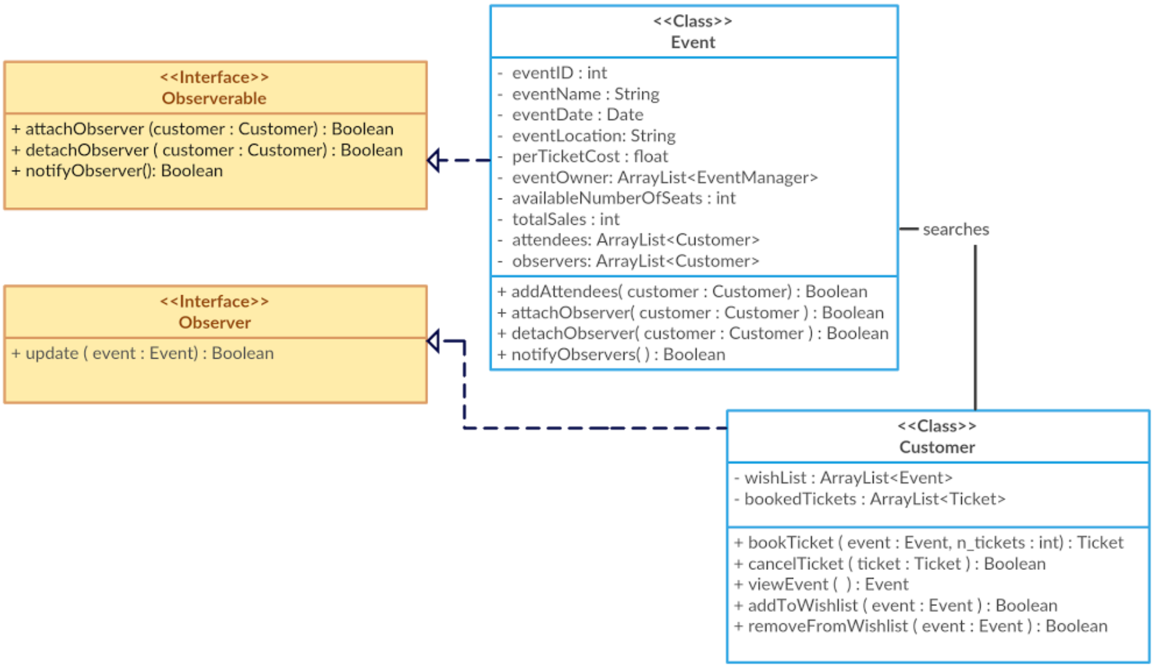
Project Part - 3: Refactoring

- **Team :**
  Ashwin Sankaralingam
  Hari Vignesh
  Ilamvazhuthy Subbiah
- **Title: BookIT**
- **Project Summary:**

  BookIT is a one stop destination for all your entertainment ticketing needs from concert ticket to NFL game tickets. You can also signup as an event Manager to host your own events.
- **Refactoring:**

  Design Patterns:

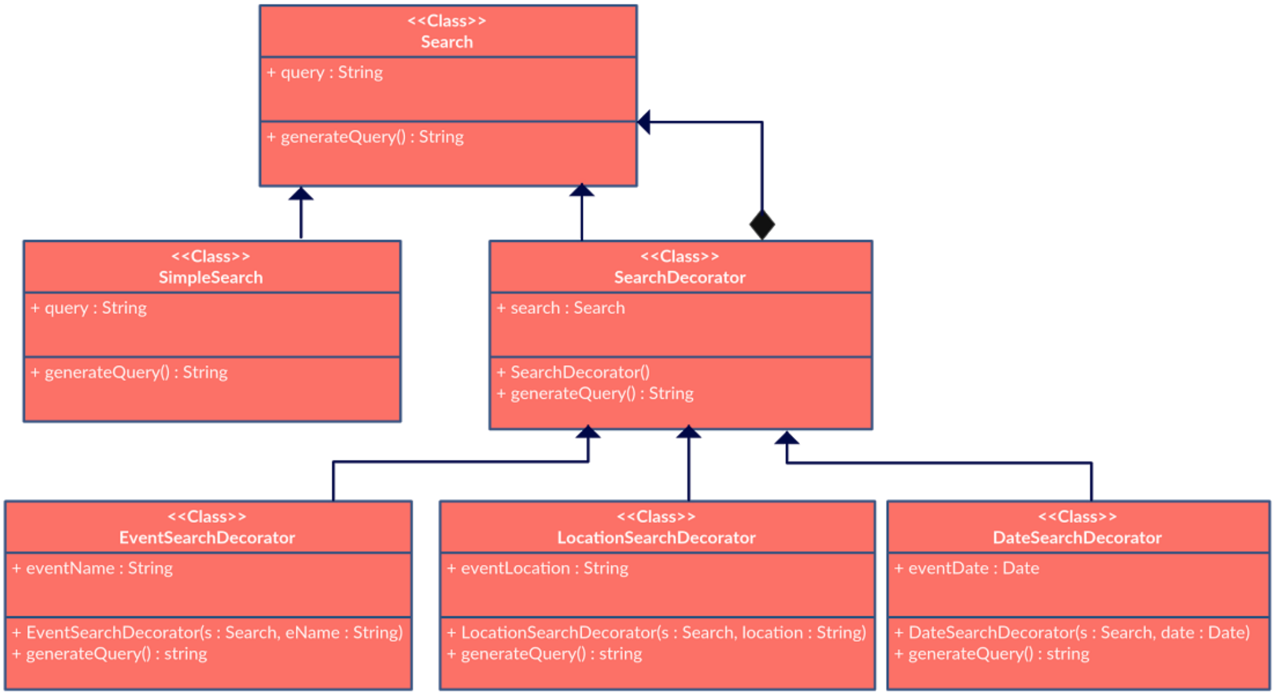
  1) Observer Pattern:
  Implementing the Observer Pattern lets the interested customer be notified whenever there is any change in the event details like change in location or time, etc.
  The Customer can either add an event to his wishlist or book tickets for the event. In either of the cases, he is an interested party regarding the event and hence should be notified of any changes. The Event class is implemented as the Subject class which knows its list of observers (The customers) and provides functionalities for attaching or detaching an observer. The Event class then notifies all the observers if there is any change in the event details.



<<Interface>>
Observerable
+ attachObserver (customer : Customer) : Boolean
+ detachObserver (customer : Customer) : Boolean
+ notifyObserver(): Boolean
<<Interface>>
Observer
+ update (event : Event) : Boolean
<<Class>>
Event
- eventID : int
- eventName : String
- eventDate : Date
- eventLocation: String
- perTicketCost : float
- eventOwner: ArrayList<EventManager>
- availableNumberOfSeats : int
- totalSales : int
- attendees: ArrayList<Customer>
- observers: ArrayList<Customer>
+ addAttendees(customer : Customer) : Boolean
+ attachObserver(customer : Customer) : Boolean
+ detachObserver(customer : Customer) : Boolean
+ notifyObservers( ) : Boolean
searches
<<Class>>
Customer
- wishList : ArrayList<Event>
- bookedTickets : ArrayList<Ticket>
+ bookTicket (event : Event, n_tickets : int) : Ticket
+ cancelTicket (ticket : Ticket) : Boolean
+ viewEvent ( ) : Event
+ addToWishlist (event : Event) : Boolean
+ removeFromWishlist (event : Event) : Boolean

2) Decorator Pattern

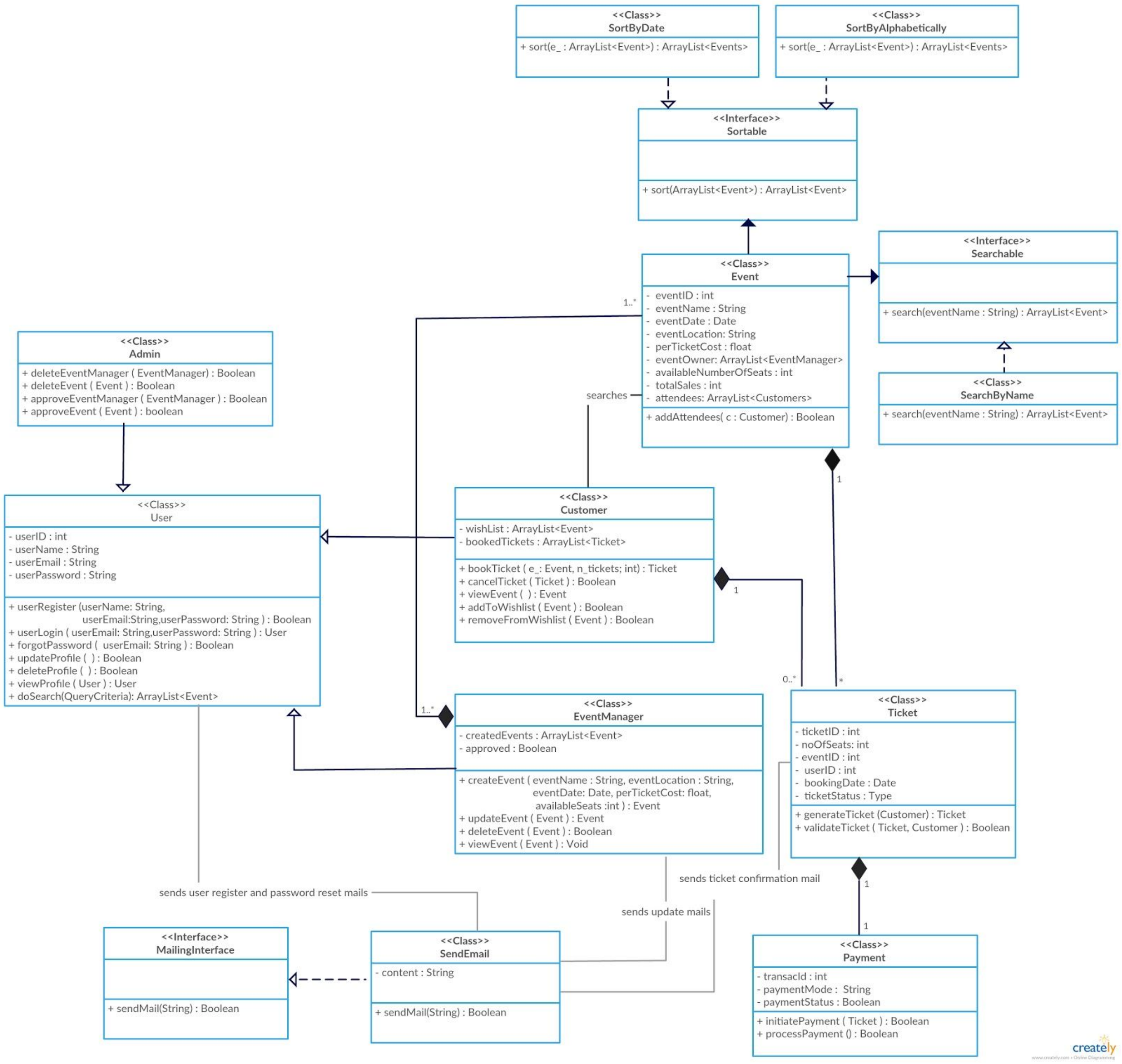
The decorator pattern can be used when a user searches for events based on a search criteria. The user can couple many search criteria like location, date or the event name when he/she performs the search. The decorator pattern can be employed to handle these couplings in an effective manner.


<<Class>>
Search
+ query : String
+ generateQuery() : String
<<Class>>
SimpleSearch
+ query : String
+ generateQuery() : String
<<Class>>
SearchDecorator
+ search : Search
+ SearchDecorator()
+ generateQuery() : String
<<Class>>
EventSearchDecorator
+ eventName : String
+ EventSearchDecorator(s : Search, eName : String)
+ generateQuery() : string
<<Class>>
LocationSearchDecorator
+ eventLocation : String
+ LocationSearchDecorator(s : Search, location : String)
+ generateQuery() : string
<<Class>>
DateSearchDecorator
+ eventDate : Date
+ DateSearchDecorator(s : Search, date : Date)
+ generateQuery() : string

Other Changes:

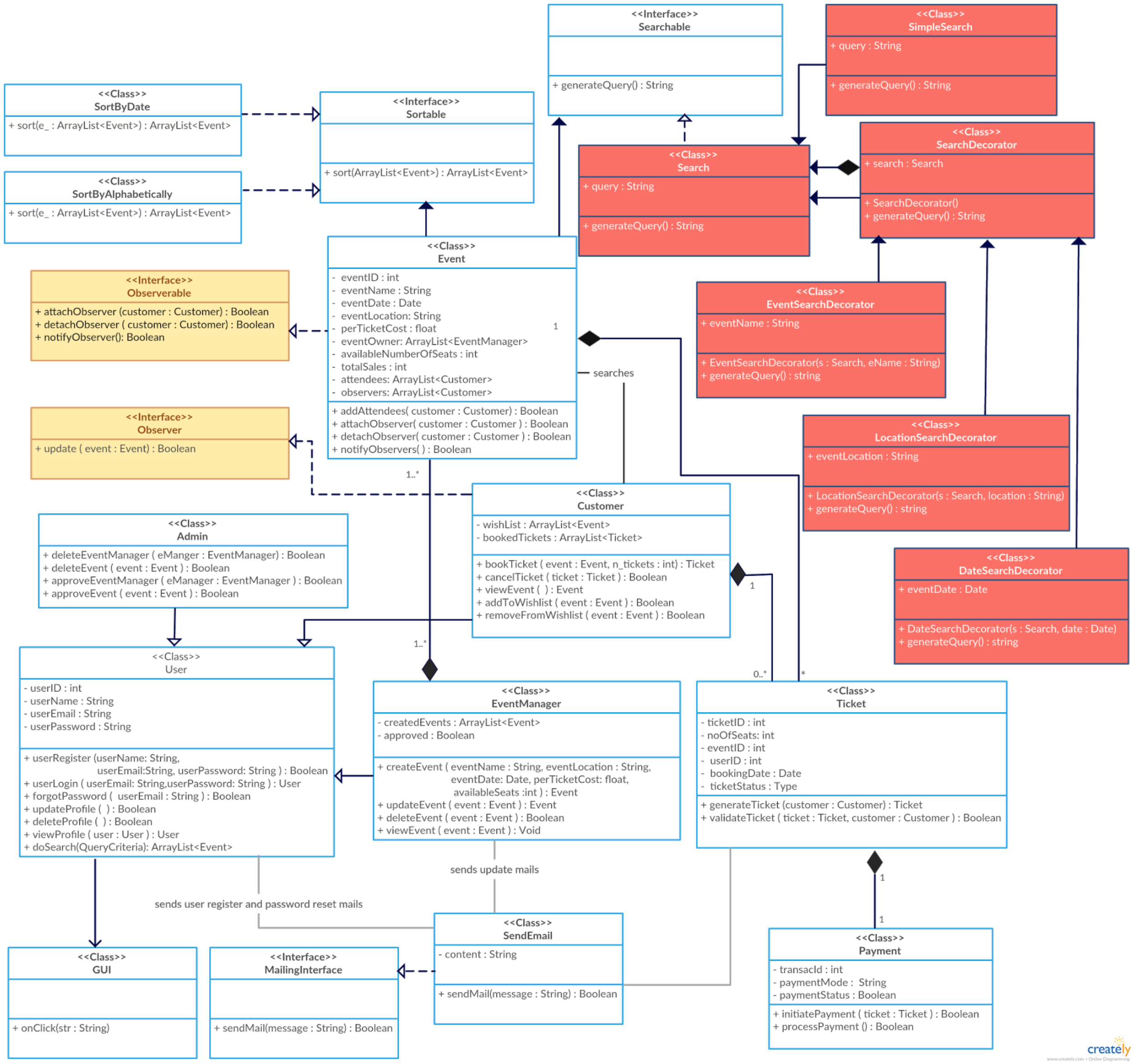
Class diagram inconsistencies with parameters as variable names vs data types have been corrected.

Existing Class Diagram:


<<Class>>
SortByDate
+ sort(e_ : ArrayList<Event>) : ArrayList<Events>
<<Class>>
SortByAlphabetically
+ sort(e_ : ArrayList<Event>) : ArrayList<Events>
<<Interface>>
Sortable
+ sort(ArrayList<Event>) : ArrayList<Event>
<<Interface>>
Searchable
+ search(eventName : String) : ArrayList<Event>
<<Class>>
Event
- eventID : int
- eventName : String
- eventDate : Date
- eventLocation: String
- perTicketCost : float
- eventOwner: ArrayList<EventManager>
- availableNumberOfSeats : int
- totalSales : int
- attendees: ArrayList<Customers>
+ addAttendees(c : Customer) : Boolean
1..*
searches
1
<<Class>>
SearchByName
+ search(eventName : String) : ArrayList<Event>
<<Class>>
Admin
+ deleteEventManager (EventManager) : Boolean
+ deleteEvent (Event) : Boolean
+ approveEventManager (EventManager) : Boolean
+ approveEvent (Event) : boolean
<<Class>>
User
- userID : int
- userName : String
- userEmail : String
- userPassword : String
+ userRegister (userName: String, userEmail:String,userPassword: String) : Boolean
+ userLogin (userEmail: String,userPassword: String) : User
+ forgotPassword (userEmail: String) : Boolean
+ updateProfile ( ) : Boolean
+ deleteProfile ( ) : Boolean
+ viewProfile (User) : User
+ doSearch(QueryCriteria): ArrayList<Event>
<<Class>>
Customer
- wishList : ArrayList<Event>
- bookedTickets : ArrayList<Ticket>
+ bookTicket (e_: Event, n_tickets; int) : Ticket
+ cancelTicket (Ticket) : Boolean
+ viewEvent ( ) : Event
+ addToWishlist (Event) : Boolean
+ removeFromWishlist (Event) : Boolean
1
0..*
*
1..*
<<Class>>
EventManager
- createdEvents : ArrayList<Event>
- approved : Boolean
+ createEvent (eventName : String, eventLocation : String, eventDate: Date, perTicketCost: float, availableSeats :int) : Event
+ updateEvent (Event) : Event
+ deleteEvent (Event) : Boolean
+ viewEvent (Event) : Void
<<Class>>
Ticket
- ticketID : int
- noOfSeats: int
- eventID : int
- userID : int
- bookingDate : Date
- ticketStatus : Type
+ generateTicket (Customer) : Ticket
+ validateTicket (Ticket, Customer) : Boolean
sends ticket confirmation mail
1
1
sends user register and password reset mails
sends update mails
<<Interface>>
MailingInterface
+ sendMail(String) : Boolean
<<Class>>
SendEmail
- content : String
+ sendMail(String) : Boolean
<<Class>>
Payment
- transacId : int
- paymentMode : String
- paymentStatus : Boolean
+ initiatePayment (Ticket) : Boolean
+ processPayment () : Boolean
creately

Refactored Class Diagram:

- Observer - Yellow
- Decorator - Red
- Classes from the previous class diagram are in blue.


<<Interface>>
Searchable
+ generateQuery() : String
<<Class>>
SimpleSearch
+ query : String
+ generateQuery() : String
<<Class>>
SortByDate
+ sort(e_ : ArrayList<Event>) : ArrayList<Event>
<<Interface>>
Sortable
+ sort(ArrayList<Event>) : ArrayList<Event>
<<Class>>
SortByAlphabetically
+ sort(e_ : ArrayList<Event>) : ArrayList<Event>
<<Class>>
Search
+ query : String
+ generateQuery() : String
<<Class>>
SearchDecorator
+ search : Search
+ SearchDecorator()
+ generateQuery() : String
<<Class>>
Event
- eventID : int
- eventName : String
- eventDate : Date
- eventLocation: String
- perTicketCost : float
- eventOwner: ArrayList<EventManager>
- availableNumberOfSeats : int
- totalSales : int
- attendees: ArrayList<Customer>
- observers: ArrayList<Customer>
+ addAttendees(customer : Customer) : Boolean
+ attachObserver(customer : Customer) : Boolean
+ detachObserver(customer : Customer) : Boolean
+ notifyObservers( ) : Boolean
1
searches
<<Interface>>
Observerable
+ attachObserver (customer : Customer) : Boolean
+ detachObserver (customer : Customer) : Boolean
+ notifyObserver(): Boolean
<<Class>>
EventSearchDecorator
+ eventName : String
+ EventSearchDecorator(s : Search, eName : String)
+ generateQuery() : string
<<Interface>>
Observer
+ update (event : Event) : Boolean
<<Class>>
LocationSearchDecorator
+ eventLocation : String
+ LocationSearchDecorator(s : Search, location : String)
+ generateQuery() : string
1..*
<<Class>>
Customer
- wishList : ArrayList<Event>
- bookedTickets : ArrayList<Ticket>
+ bookTicket (event : Event, n_tickets : int) : Ticket
+ cancelTicket (ticket : Ticket) : Boolean
+ viewEvent ( ) : Event
+ addToWishlist (event : Event) : Boolean
+ removeFromWishlist (event : Event) : Boolean
<<Class>>
Admin
+ deleteEventManager (eManger : EventManager) : Boolean
+ deleteEvent (event : Event) : Boolean
+ approveEventManager (eManager : EventManager) : Boolean
+ approveEvent (event : Event) : Boolean
<<Class>>
DateSearchDecorator
+ eventDate : Date
+ DateSearchDecorator(s : Search, date : Date)
+ generateQuery() : string
1
1..*
0..*
*
<<Class>>
User
- userID : int
- userName : String
- userEmail : String
- userPassword : String
+ userRegister (userName: String,
userEmail:String, userPassword: String) : Boolean
+ userLogin (userEmail: String,userPassword: String) : User
+ forgotPassword (userEmail : String) : Boolean
+ updateProfile ( ) : Boolean
+ deleteProfile ( ) : Boolean
+ viewProfile (user : User) : User
+ doSearch(QueryCriteria): ArrayList<Event>
<<Class>>
EventManager
- createdEvents : ArrayList<Event>
- approved : Boolean
+ createEvent (eventName : String, eventLocation : String,
eventDate: Date, perTicketCost: float,
availableSeats :int) : Event
+ updateEvent (event : Event) : Event
+ deleteEvent (event : Event) : Boolean
+ viewEvent (event : Event) : Void
<<Class>>
Ticket
- ticketID : int
- noOfSeats: int
- eventID : int
- userID : int
- bookingDate : Date
- ticketStatus : Type
+ generateTicket (customer : Customer) : Ticket
+ validateTicket (ticket : Ticket, customer : Customer) : Boolean
1
sends update mails
sends user register and password reset mails
1
<<Class>>
SendEmail
- content : String
+ sendMail(message : String) : Boolean
<<Class>>
Payment
- transacId : int
- paymentMode : String
- paymentStatus : Boolean
+ initiatePayment (ticket : Ticket) : Boolean
+ processPayment () : Boolean
<<Class>>
GUI
+ onClick(str : String)
<<Interface>>
MailingInterface
+ sendMail(message : String) : Boolean
creately
www.creately.com • Online Diagramming