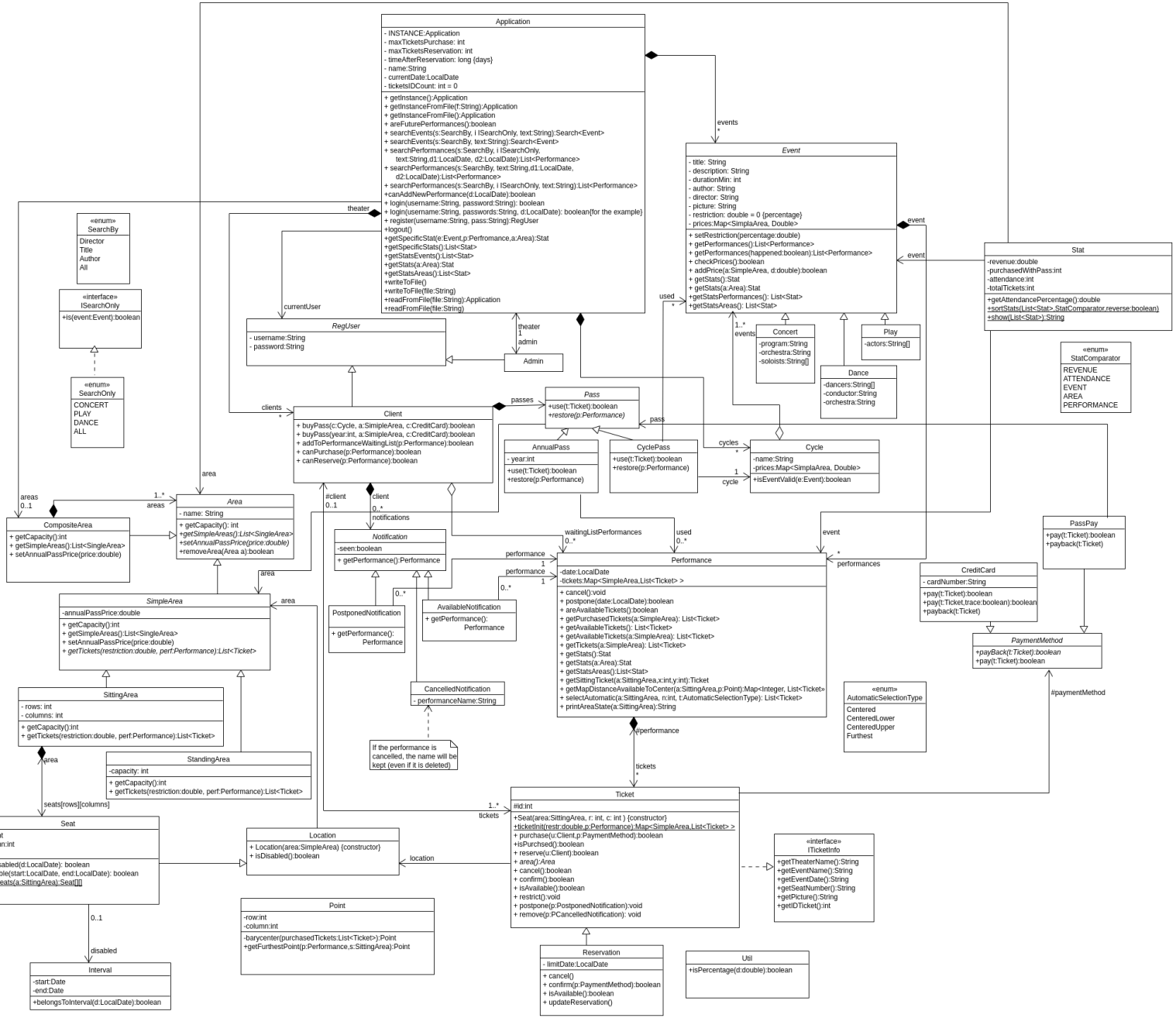
Application
- INSTANCE:Application
- maxTicketsPurchase: int
- maxTicketsReservation: int
- timeAfterReservation: long {days}
- name:String
- currentDate:LocalDate
- ticketsIDCount: int = 0
+ getInstance():Application
+ getInstanceFromFile(f:String):Application
+ getInstanceFromFile():Application
+ areFuturePerformances():boolean
+ searchEvents(s:SearchBy, i ISearchOnly, text:String):Search<Event>
+ searchEvents(s:SearchBy, text:String):Search<Event>
+ searchPerformances(s:SearchBy, i ISearchOnly, text:String,d1:LocalDate, d2:LocalDate):List<Performance>
+ searchPerformances(s:SearchBy, text:String,d1:LocalDate, d2:LocalDate):List<Performance>
+ searchPerformances(s:SearchBy, i ISearchOnly, text:String):List<Performance>
+canAddNewPerformance(d:LocalDate):boolean
+ login(username:String, password:String): boolean
+ login(username:String, passwords:String, d:LocalDate): boolean{for the example}
+ register(username:String, pass:String):RegUser
+logout()
+getSpecificStat(e:Event,p:Perfromance,a:Area):Stat
+getSpecificStats():List<Stat>
+getStatsEvents():List<Stat>
+getStats(a:Area):Stat
+getStatsAreas():List<Stat>
+writeToFile()
+writeToFile(file:String)
+readFromFile(file:String):Application
+readFromFile(file:String)
events
*
Event
- title: String
- description: String
- durationMin: int
- author: String
- director: String
- picture: String
- restriction: double = 0 {percentage}
- prices:Map<SimplaArea, Double>
+ setRestriction(percentage:double)
+ getPerformances():List<Performance>
+ getPerformances(happened:boolean):List<Performance>
+ checkPrices():boolean
+ addPrice(a:SimpleArea, d:double):boolean
+ getStats():Stat
+ getStats(a:Area):Stat
+getStatsPerformances(): List<Stat>
+getStatsAreas(): List<Stat>
theater
event
event
Stat
-revenue:double
-purchasedWithPass:int
-attendance:int
-totalTickets:int
+getAttendancePercentage():double
+sortStats(List<Stat>,StatComparator,reverse:boolean)
+show(List<Stat>):String
«enum»
SearchBy
Director
Title
Author
All
«interface»
ISearchOnly
+is(event:Event):boolean
«enum»
SearchOnly
CONCERT
PLAY
DANCE
ALL
currentUser
RegUser
- username:String
- password:String
theater
1
admin
Admin
used
*
1..*
events
Concert
-program:String
-orchestra:String
-soloists:String[]
Play
-actors:String[]
Dance
-dancers:String[]
-conductor:String
-orchestra:String
«enum»
StatComparator
REVENUE
ATTENDANCE
EVENT
AREA
PERFORMANCE
clients
*
Client
+ buyPass(c:Cycle, a:SimiplaArea, c:CreditCard):boolean
+ buyPass(year:int, a:SimiplaArea, c:CreditCard):boolean
+ addToPerformanceWaitingList(p:Performance):boolean
+ canPurchase(p:Performance):boolean
+ canReserve(p:Performance):boolean
passes
Pass
+use(t:Ticket):boolean
+restore(p:Performance)
pass
AnnualPass
- year:int
+use(t:Ticket):boolean
+restore(p:Performance)
CyclePass
+use(t:Ticket):boolean
+restore(p:Performance)
cycles
*
1
cycle
Cycle
-name:String
-prices:Map<SimplaArea, Double>
+isEventValid(e:Event):boolean
area
areas
0..1
1..*
areas
Area
- name: String
+ getCapacity(): int
+getSimpleAreas():List<SingleArea>
+setAnnualPassPrice(price:double)
+removeArea(Area a):boolean
CompositeArea
+ getCapacity():int
+ getSimpleAreas():List<SingleArea>
+ setAnnualPassPrice(price:double)
#client
0..1
client
0..*
notifications
Notification
-seen:boolean
+ getPerformance():Performance
waitingListPerformances
0..*
used
0..*
event
*
performances
PassPay
+pay(t:Ticket):boolean
+payback(t:Ticket)
performance
1
performance
1
0..*
Performance
-date:LocalDate
-tickets:Map<SimpleArea,List<Ticket> >
+ cancel():void
+ postpone(date:LocalDate):boolean
+ areAvailableTickets():boolean
+ getPurchasedTickets(a:SimpleArea): List<Ticket>
+ getAvailableTickets(): List<Ticket>
+ getAvailableTickets(a:SimpleArea): List<Ticket>
+ getTickets(a:SimpleArea): List<Ticket>
+ getStats():Stat
+ getStats(a:Area):Stat
+ getStatsAreas():List<Stat>
+ getSittingTicket(a:SittingArea,x:int,y:int):Ticket
+ getMapDistanceAvailableToCenter(a:SittingArea,p:Point):Map<Integer, List<Ticket»
+ selectAutomatic(a:SittingArea, n:int, t:AutomaticSelectionType): List<Ticket>
+ printAreaState(a:SittingArea):String
CreditCard
- cardNumber:String
+pay(t:Ticket):boolean
+pay(t:Ticket,trace:boolean):boolean
+payback(t:Ticket)
area
SimpleArea
-annualPassPrice:double
+ getCapacity():int
+ getSimpleAreas():List<SingleArea>
+ setAnnualPassPrice(price:double)
+ getTickets(restriction:double, perf:Performance):List<Ticket>
area
0..*
PostponedNotification
+ getPerformance(): Performance
AvailableNotification
+ getPerformance(): Performance
PaymentMethod
+payBack(t:Ticket):boolean
+pay(t:Ticket):boolean
SittingArea
- rows: int
- columns: int
+ getCapacity():int
+ getTickets(restriction:double, perf:Performance):List<Ticket>
CancelledNotification
- performanceName:String
If the performance is cancelled, the name will be kept (even if it is deleted)
«enum»
AutomaticSelectionType
Centered
CenteredLower
CenteredUpper
Furthest
#paymentMethod
#performance
tickets
*
area
StandingArea
-capacity: int
+ getCapacity():int
+ getTickets(restriction:double, perf:Performance):List<Ticket>
seats[rows][columns]
1..*
tickets
Ticket
#id:int
+Seat(area:SittingArea, r: int, c: int) {constructor}
+ticketInit(restr:double,p:Performance),Map<SimpleArea,List<Ticket> >
+ purchase(u:Client,p:PaymentMethod):boolean
+isPurchsed():boolean
+ reserve(u:Client):boolean
+ area():Area
+ cancel():boolean
+ confirm():boolean
+ isAvailable():boolean
+ restrict():void
+ postpone(p:PostponedNotification):void
+ remove(p:PCancelledNotification): void
Seat
-row:int
-column:int
+ isDisabled(d:LocalDate): boolean
+ disable(start:LocalDate, end:LocalDate): boolean
+ initSeats(a:SittingArea):Seat[][]
Location
+ Location(area:SimpleArea) {constructor}
+ isDisabled():boolean
location
«interface»
ITicketInfo
+getTheaterName():String
+getEventName():String
+getEventDate():String
+getSeatNumber():String
+getPicture():String
+getIDTicket():int
0..1
disabled
Point
-row:int
-column:int
-barycenter(purchasedTickets:List<Ticket>):Point
+getFurthestPoint(p:Performance,s:SittingArea):Point
Interval
-start:Date
-end:Date
+belongsToInterval(d:LocalDate):boolean
Reservation
- limitDate:LocalDate
+ cancel()
+ confirm(p:PaymentMethod):boolean
+ isAvailable():boolean
+ updateReservation()
Util
+isPercentage(d:double):boolean