Flight Scheduler

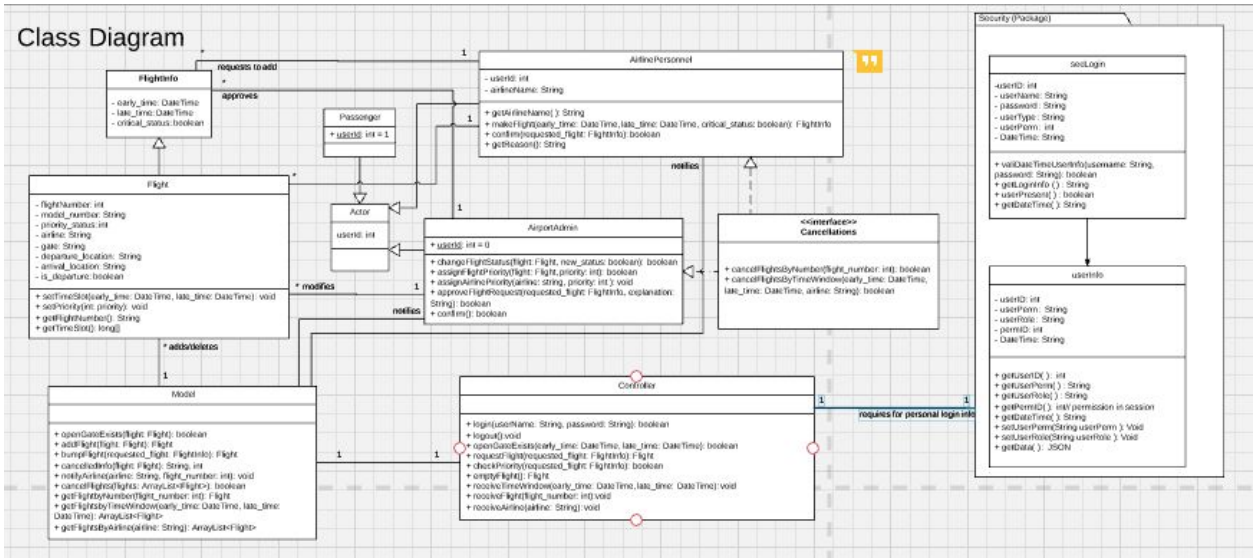
OOP Project Fall 2017

Team:

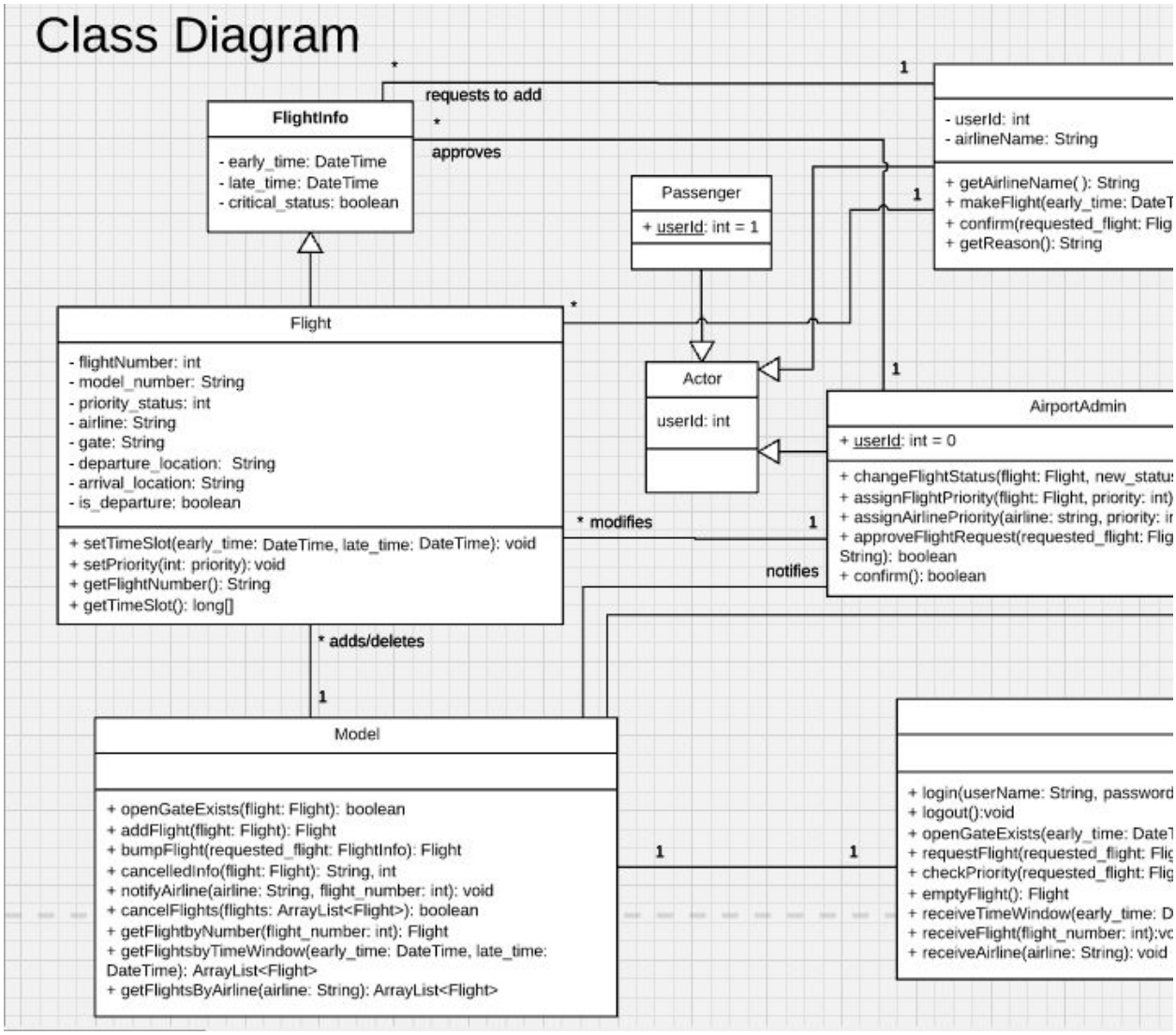
- Byron Becker,
- Johnathan Kruse
- Raghav Sharma
- Brian Chung
- Cory Morales

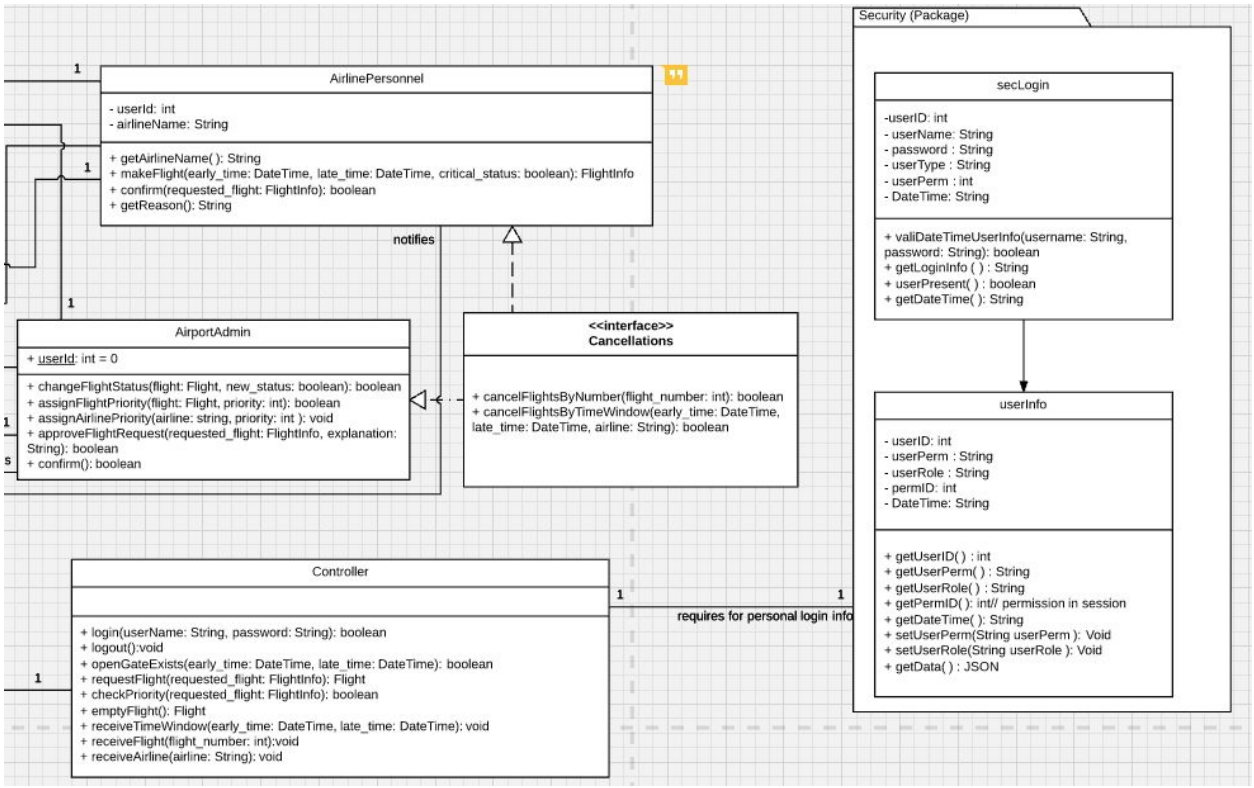
Part 2 Class Diagram

Full Class Diagram

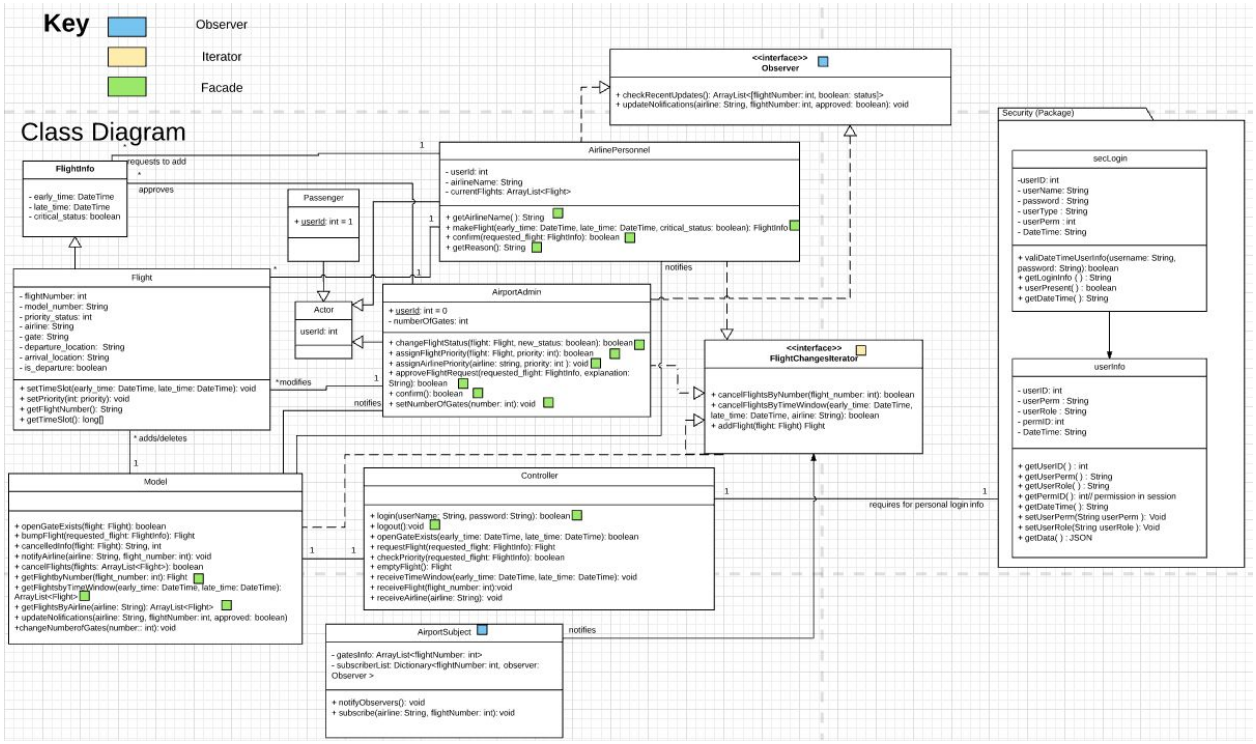


Split into images on next page for resolution purposes

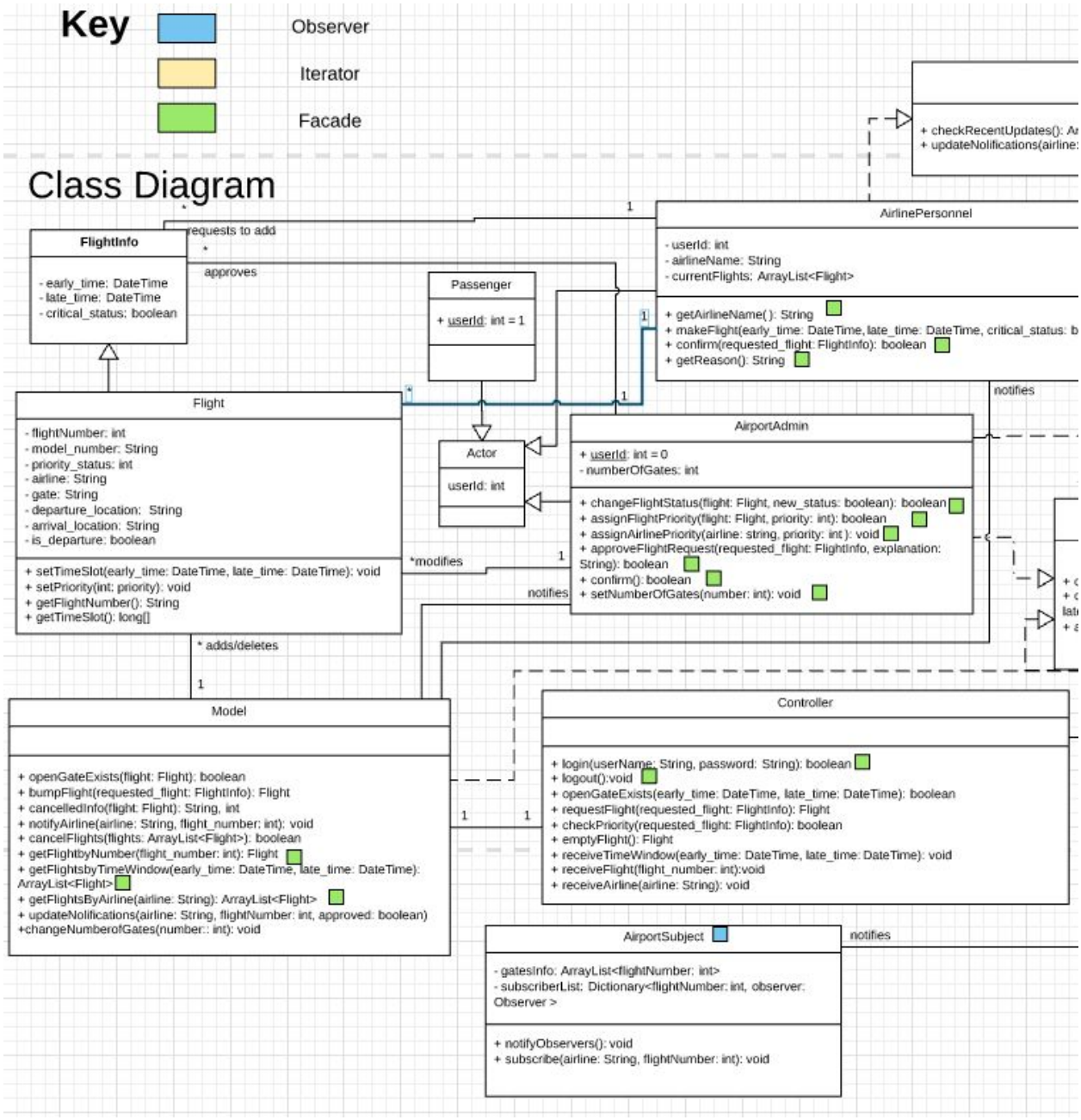

Class Diagram
FlightInfo
- early_time: DateTime
- late_time: DateTime
- critical_status: boolean
requests to add
approves
Passenger
+ userId: int = 1
- userId: int
- airlineName: String
+ getAirlineName( ): String
+ makeFlight(early_time: DateT
+ confirm(requested_flight: Flig
+ getReason(): String
Flight
- flightNumber: int
- model_number: String
- priority_status: int
- airline: String
- gate: String
- departure_location: String
- arrival_location: String
- is_departure: boolean
+ setTimeSlot(early_time: DateTime, late_time: DateTime): void
+ setPriority(int: priority): void
+ getFlightNumber(): String
+ getTimeSlot(): long[]
Actor
userId: int
AirportAdmin
+ userId: int = 0
+ changeFlightStatus(flight: Flight, new_status
+ assignFlightPriority(flight: Flight, priority: int)
+ assignAirlinePriority(airline: string, priority: in
+ approveFlightRequest(requested_flight: Flig
String): boolean
+ confirm(): boolean
* modifies
notifies
* adds/deletes
Model
+ openGateExists(flight: Flight): boolean
+ addFlight(flight: Flight): Flight
+ bumpFlight(requested_flight: FlightInfo): Flight
+ cancelledInfo(flight: Flight): String, int
+ notifyAirline(airline: String, flight_number: int): void
+ cancelFlights(flights: ArrayList<Flight>): boolean
+ getFlightbyNumber(flight_number: int): Flight
+ getFlightsbyTimeWindow(early_time: DateTime, late_time:
DateTime): ArrayList<Flight>
+ getFlightsByAirline(airline: String): ArrayList<Flight>
+ login(userName: String, password
+ logout():void
+ openGateExists(early_time: DateT
+ requestFlight(requested_flight: Flig
+ checkPriority(requested_flight: Flig
+ emptyFlight(): Flight
+ receiveTimeWindow(early_time: D
+ receiveFlight(flight_number: int):vo
+ receiveAirline(airline: String): void

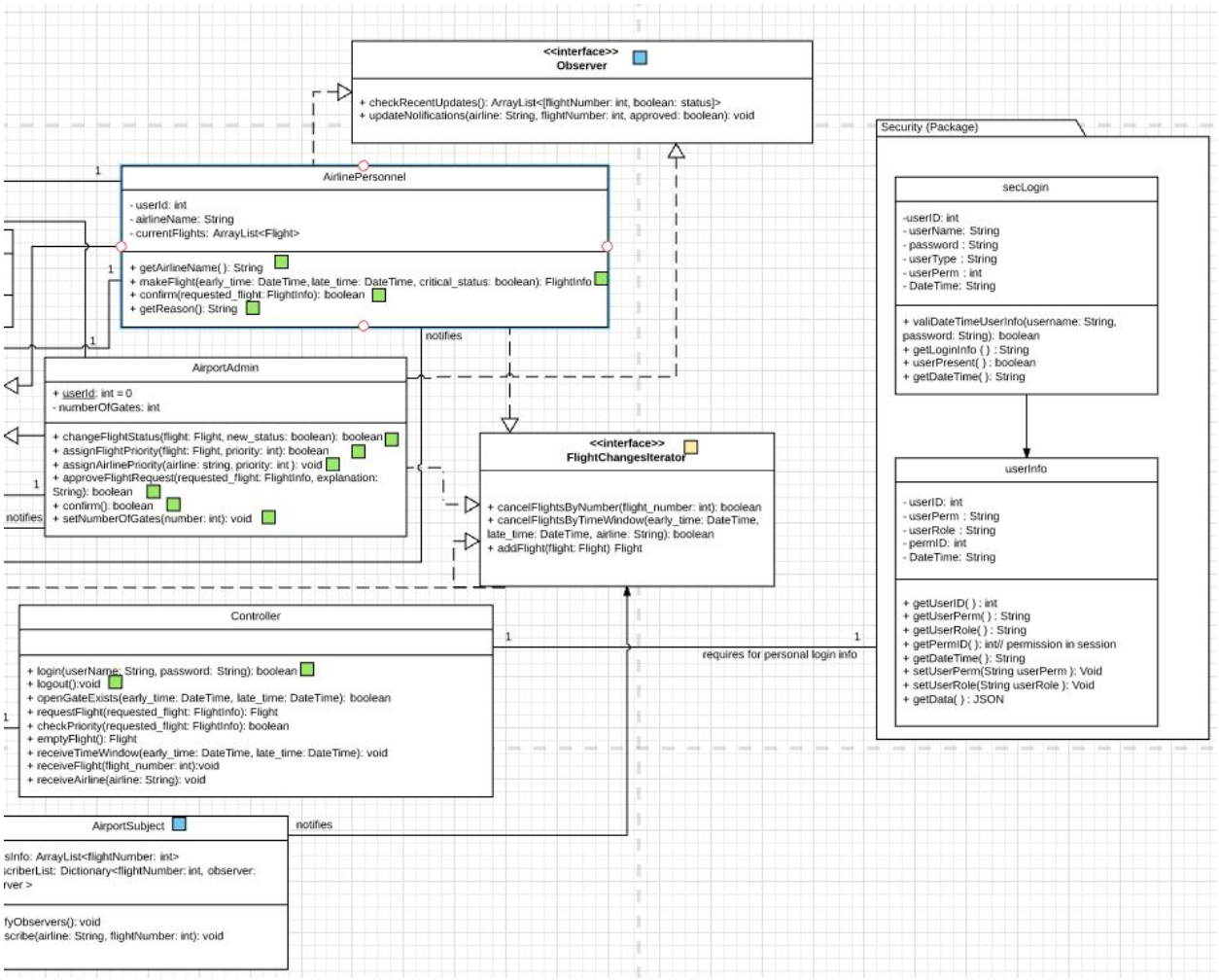

Security (Package)
AirlinePersonnel
- userId: int
- airlineName: String
+ getAirlineName( ): String
+ makeFlight(early_time: DateTime, late_time: DateTime, critical_status: boolean): FlightInfo
+ confirm(requested_flight: FlightInfo): boolean
+ getReason(): String
notifies
AirportAdmin
+ userId: int = 0
+ changeFlightStatus(flight: Flight, new_status: boolean): boolean
+ assignFlightPriority(flight: Flight, priority: int): boolean
+ assignAirlinePriority(airline: string, priority: int): void
+ approveFlightRequest(requested_flight: FlightInfo, explanation: String): boolean
+ confirm(): boolean
<<interface>>
Cancellations
+ cancelFlightsByNumber(flight_number: int): boolean
+ cancelFlightsByTimeWindow(early_time: DateTime, late_time: DateTime, airline: String): boolean
secLogin
-userID: int
- userName: String
- password : String
- userType : String
- userPerm : int
- DateTime: String
+ valiDateTimeUserInfo(username: String, password: String): boolean
+ getLoginInfo ( ) : String
+ userPresent( ) : boolean
+ getDateTime( ): String
userInfo
- userID: int
- userPerm : String
- userRole : String
- permID: int
- DateTime: String
+ getUserID( ) : int
+ getUserPerm( ) : String
+ getUserRole( ) : String
+ getPermID( ): int// permission in session
+ getDateTime( ): String
+ setUserPerm(String userPerm): Void
+ setUserRole(String userRole): Void
+ getData( ) : JSON
Controller
+ login(userName: String, password: String): boolean
+ logout():void
+ openGateExists(early_time: DateTime, late_time: DateTime): boolean
+ requestFlight(requested_flight: FlightInfo): Flight
+ checkPriority(requested_flight: FlightInfo): boolean
+ emptyFlight(): Flight
+ receiveTimeWindow(early_time: DateTime, late_time: DateTime): void
+ receiveFlight(flight_number: int):void
+ receiveAirline(airline: String): void
requires for personal login info
1

Part 3 Class Diagram (Refactored)



Split into images on next page for resolution purposes


Key
Observer
Iterator
Facade
Class Diagram
FlightInfo
- early_time: DateTime
- late_time: DateTime
- critical_status: boolean
requests to add
approves
Passenger
+ userId: int = 1
AirlinePersonnel
- userId: int
- airlineName: String
- currentFlights: ArrayList<Flight>
+ getAirlineName( ): String
+ makeFlight(early_time: DateTime, late_time: DateTime, critical_status: b
+ confirm(requested_flight: FlightInfo): boolean
+ getReason(): String
+ checkRecentUpdates(): Ar
+ updateNolifications(airline:
Flight
- flightNumber: int
- model_number: String
- priority_status: int
- airline: String
- gate: String
- departure_location: String
- arrival_location: String
- is_departure: boolean
+ setTimeSlot(early_time: DateTime, late_time: DateTime): void
+ setPriority(int: priority): void
+ getFlightNumber(): String
+ getTimeSlot(): long[]
Actor
userId: int
AirportAdmin
+ userId: int = 0
- numberOfGates: int
+ changeFlightStatus(flight: Flight, new_status: boolean): boolean
+ assignFlightPriority(flight: Flight, priority: int): boolean
+ assignAirlinePriority(airline: string, priority: int): void
+ approveFlightRequest(requested_flight: FlightInfo, explanation: String): boolean
+ confirm(): boolean
+ setNumberOfGates(number: int): void
notifies
*modifies
* adds/deletes
Model
+ openGateExists(flight: Flight): boolean
+ bumpFlight(requested_flight: FlightInfo): Flight
+ cancelledInfo(flight: Flight): String, int
+ notifyAirline(airline: String, flight_number: int): void
+ cancelFlights(flights: ArrayList<Flight>): boolean
+ getFlightbyNumber(flight_number: int): Flight
+ getFlightsbyTimeWindow(early_time: DateTime, late_time: DateTime): ArrayList<Flight>
+ getFlightsByAirline(airline: String): ArrayList<Flight>
+ updateNolifications(airline: String, flightNumber: int, approved: boolean)
+changeNumberofGates(number.: int): void
Controller
+ login(userName: String, password: String): boolean
+ logout():void
+ openGateExists(early_time: DateTime, late_time: DateTime): boolean
+ requestFlight(requested_flight: FlightInfo): Flight
+ checkPriority(requested_flight: FlightInfo): boolean
+ emptyFlight(): Flight
+ receiveTimeWindow(early_time: DateTime, late_time: DateTime): void
+ receiveFlight(flight_number: int):void
+ receiveAirline(airline: String): void
AirportSubject
- gatesInfo: ArrayList<flightNumber: int>
- subscriberList: Dictionary<flightNumber: int, observer: Observer >
+ notifyObservers(): void
+ subscribe(airline: String, flightNumber: int): void
notifies


<<interface>>
Observer
+ checkRecentUpdates(): ArrayList<[flightNumber: int, boolean: status]>
+ updateNolifications(airline: String, flightNumber: int, approved: boolean): void
Security (Package)
AirlinePersonnel
- userId: int
- airlineName: String
- currentFlights: ArrayList<Flight>
+ getAirlineName( ): String
+ makeFlight(early_time: DateTime, late_time: DateTime, critical_status: boolean): FlightInfo
+ confirm(requested_flight: FlightInfo): boolean
+ getReason(): String
notifies
secLogin
-userID: int
- userName: String
- password : String
- userType : String
- userPerm : int
- DateTime: String
+ valiDateTimeUserInfo(username: String, password: String): boolean
+ getLoginInfo ( ) : String
+ userPresent( ) : boolean
+ getDateTime( ): String
AirportAdmin
+ userId: int = 0
- numberOfGates: int
+ changeFlightStatus(flight: Flight, new_status: boolean): boolean
+ assignFlightPriority(flight: Flight, priority: int): boolean
+ assignAirlinePriority(airline: string, priority: int): void
+ approveFlightRequest(requested_flight: FlightInfo, explanation: String): boolean
+ confirm(): boolean
+ setNumberOfGates(number: int): void
notifies
<<interface>>
FlightChangesIterator
+ cancelFlightsByNumber(flight_number: int): boolean
+ cancelFlightsByTimeWindow(early_time: DateTime, late_time: DateTime, airline: String): boolean
+ addFlight(flight: Flight) Flight
userInfo
- userID: int
- userPerm : String
- userRole : String
- permID: int
- DateTime: String
+ getUserID( ) : int
+ getUserPerm( ) : String
+ getUserRole( ) : String
+ getPermID( ): int// permission in session
+ getDateTime( ): String
+ setUserPerm(String userPerm): Void
+ setUserRole(String userRole): Void
+ getData( ) : JSON
Controller
+ login(userName: String, password: String): boolean
+ logout():void
+ openGateExists(early_time: DateTime, late_time: DateTime): boolean
+ requestFlight(requested_flight: FlightInfo): Flight
+ checkPriority(requested_flight: FlightInfo): boolean
+ emptyFlight(): Flight
+ receiveTimeWindow(early_time: DateTime, late_time: DateTime): void
+ receiveFlight(flight_number: int):void
+ receiveAirline(airline: String): void
1
1
requires for personal login info
AirportSubject
notifies
sInfo: ArrayList<flightNumber: int>
icriberList: Dictionary<flightNumber:int, observer:
rver >
fyObservers(): void
scribe(airline: String, flightNumber: int): void

Patterns Added

1. **Observer** - we are using the Observer design pattern (blue square) to allow airline personnel and the airport admin to subscribe to various flights when they request to add a flight. The AirportSubject class has been added, which keeps a subscriber list of observers and their flight numbers, as well as a lookup of gate info for the airport. The AirportSubject will call notifyObservers() when a change is made to a flight that has been subscribed to, which will both send an immediate notification to the observer airline personnel and airport admin, as well as send this notification to the model, which through updateNotifications() will add the change information to a recently updated list in the database, that can be pinged by the observer with checkRecentUpdates() upon logging in (this way the observer will not miss out on a notification if it was logged out when the change happened.

2. **Iterator** - We are using the Iterator design pattern (yellow square) via the interface FlightChangesIterator which is implemented in AirportAdmin and AirlinePersonnel. This iterator will allow the ability for looping over a collection of flights for three different functions. Mainly being when the program is searching for flights to cancel either by flight number or to cancel flights within a time period and then additionally when adding a new flight to the existing collection of flights.

3. **Facade** - the Facade design pattern (green square) was already used in the previous iteration of our class diagram, and essentially encompasses the higher level functions that “automate” many of the necessary actions that each user needs to make. For example, when the airline personnel confirms information for adding a flight, the addFlight() function has been abstracted into the iterator interface, which is called by the confirm() function. Additionally, the approveFlightRequest() has abstracted the replacement flight out of the direct role of the airport admin by putting bumpFlight() into the model class. Therefore, we can see that these higher level functions provide a simplified interface for the clients to interact with.