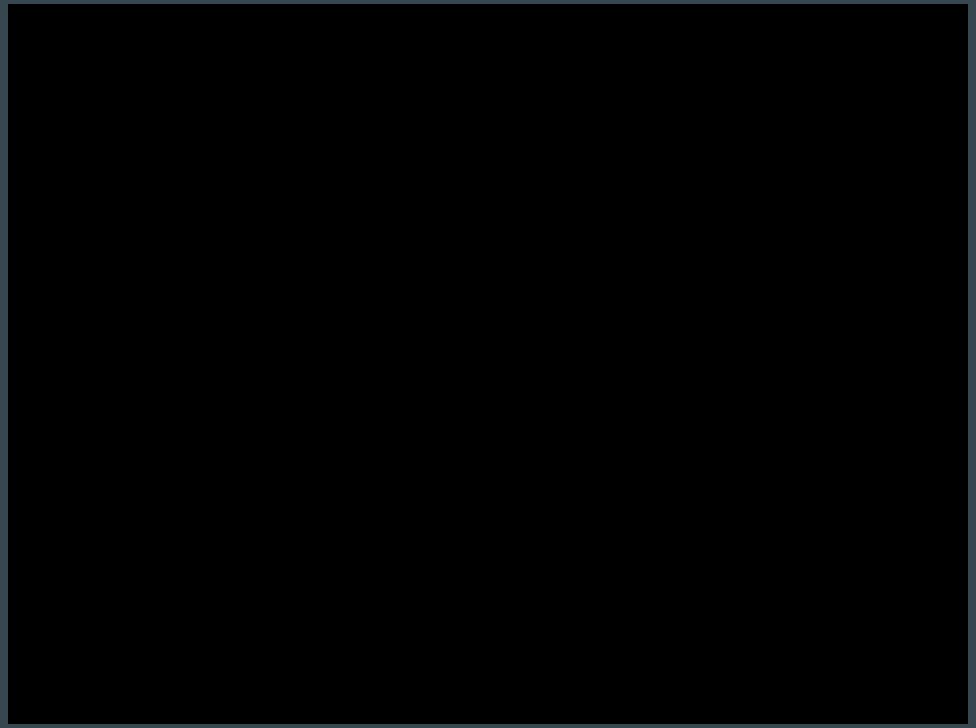
Flight Scheduler

Team #6

Raghav Sharma
Byron Becker
Johnathan Kruse
Brian Chung
Cory Morales

spring by Pivotal
MySQL
Java
HIBERNATE
Maven

Demo



Use Cases

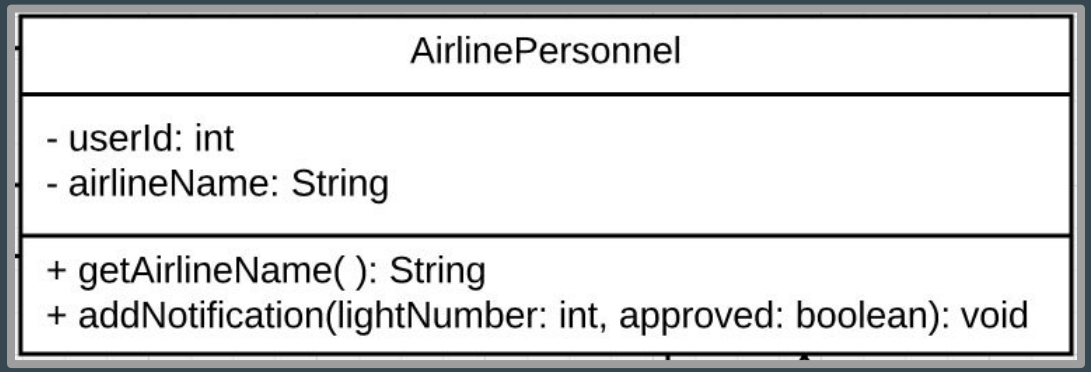
- Add Flight
- Cancel Flight
- “Bump” Flight (add priority flight)
- View Recent Changes

https://drive.google.com/a/colorado.edu/file/d/1zWBL4WwnUg0uujM21SXqrHaa_3EkvMKY/view?usp=sharing

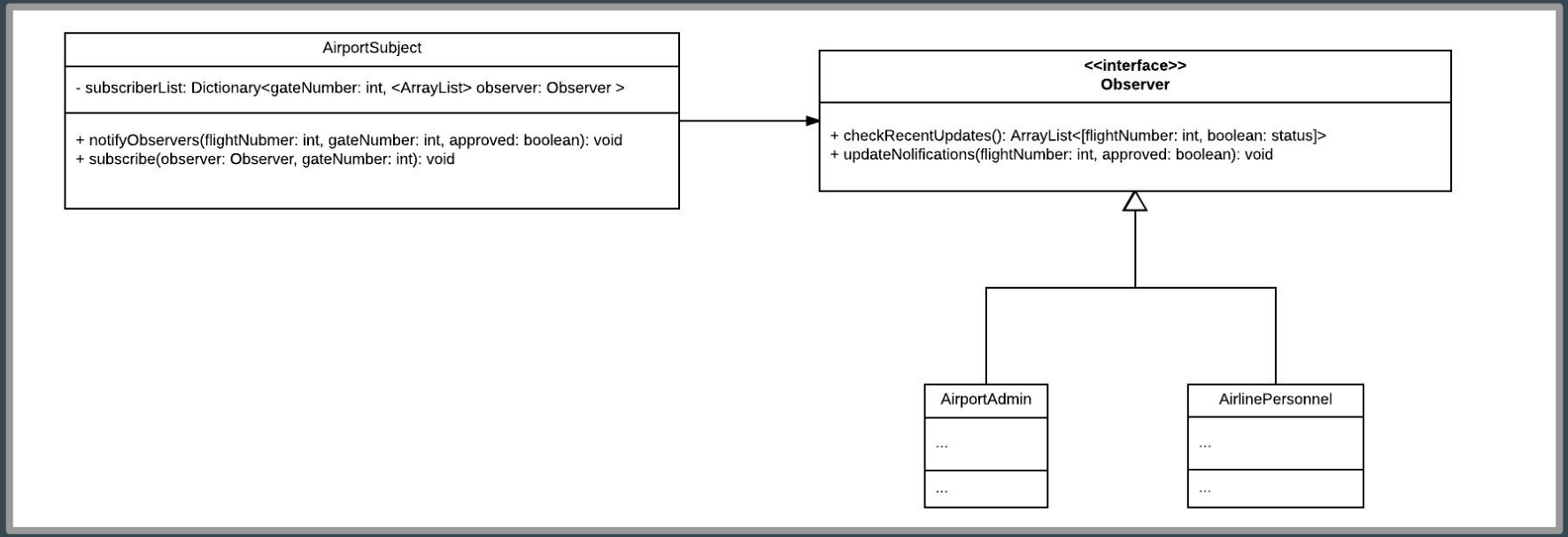
Design Patterns:

- Four main design patterns:
  - Observer
  - Facade
  - Singleton
  - Iterator

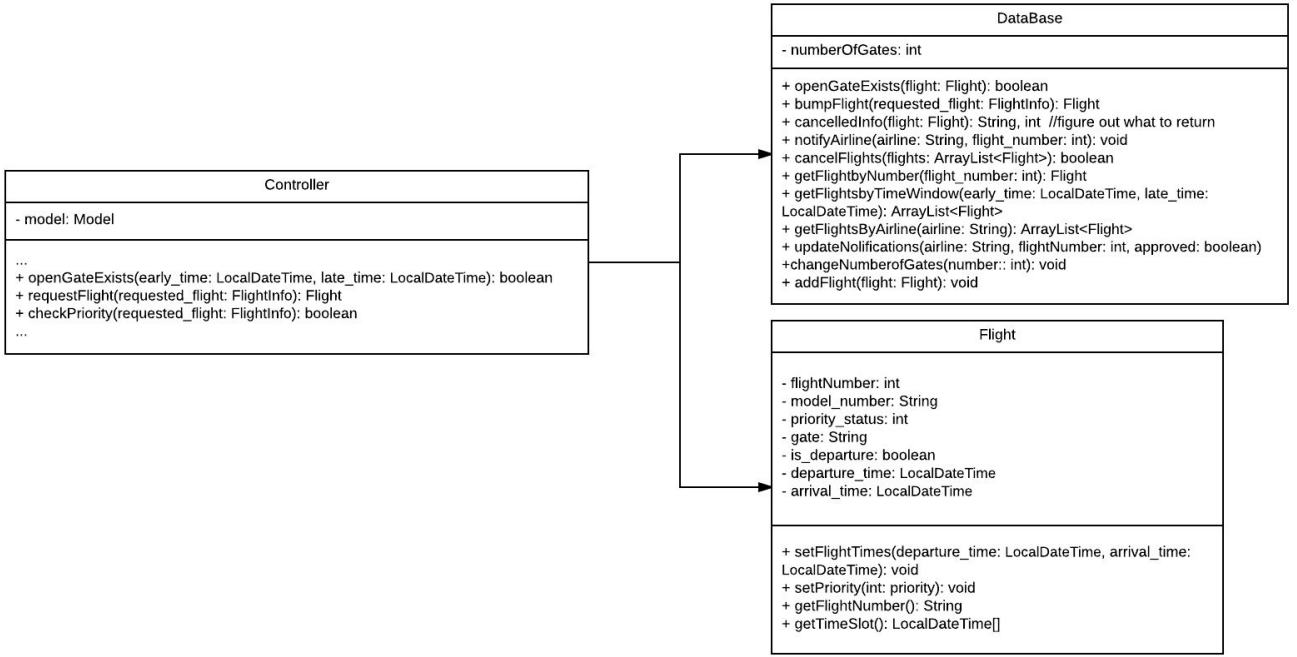
Observer Design Pattern - Before



Observer


AirportSubject
- subscriberList: Dictionary<gateNumber: int, <ArrayList> observer: Observer >
+ notifyObservers(flightNubmer: int, gateNumber: int, approved: boolean): void
+ subscribe(observer: Observer, gateNumber: int): void
<<interface>>
Observer
+ checkRecentUpdates(): ArrayList<[flightNumber: int, boolean: status]>
+ updateNolifications(flightNumber: int, approved: boolean): void
AirportAdmin
...
...
AirlinePersonnel
...
...

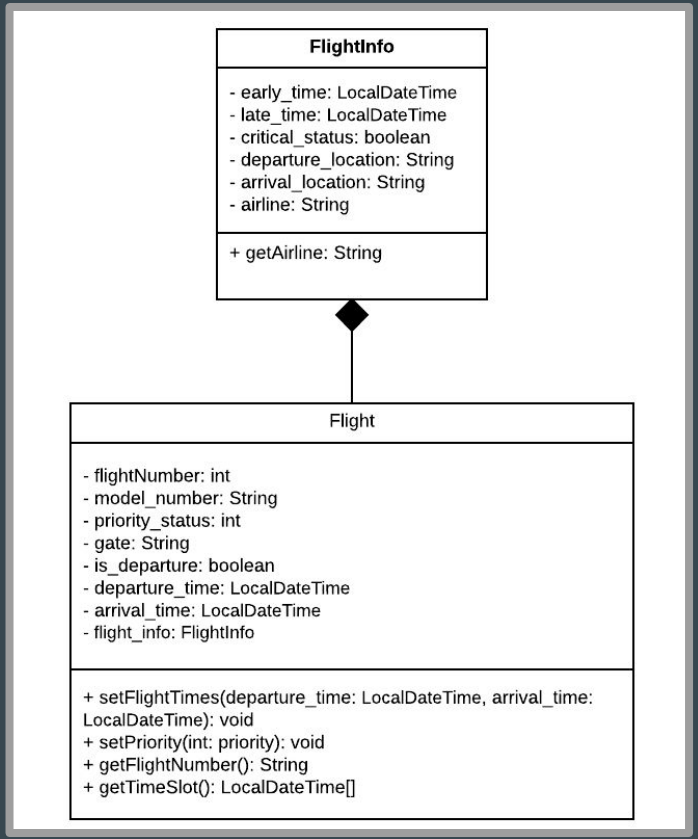
Facade


DataBase
- numberOfGates: int
+ openGateExists(flight: Flight): boolean
+ bumpFlight(requested_flight: FlightInfo): Flight
+ cancelledInfo(flight: Flight): String, int //figure out what to return
+ notifyAirline(airline: String, flight_number: int): void
+ cancelFlights(flights: ArrayList<Flight>): boolean
+ getFlightbyNumber(flight_number: int): Flight
+ getFlightsbyTimeWindow(early_time: LocalDateTime, late_time: LocalDateTime): ArrayList<Flight>
+ getFlightsByAirline(airline: String): ArrayList<Flight>
+ updateNolifications(airline: String, flightNumber: int, approved: boolean)
+changeNumberofGates(number:: int): void
+ addFlight(flight: Flight): void
Controller
- model: Model
...
+ openGateExists(early_time: LocalDateTime, late_time: LocalDateTime): boolean
+ requestFlight(requested_flight: FlightInfo): Flight
+ checkPriority(requested_flight: FlightInfo): boolean
...
Flight
- flightNumber: int
- model_number: String
- priority_status: int
- gate: String
- is_departure: boolean
- departure_time: LocalDateTime
- arrival_time: LocalDateTime
+ setFlightTimes(departure_time: LocalDateTime, arrival_time: LocalDateTime): void
+ setPriority(int: priority): void
+ getFlightNumber(): String
+ getTimeSlot(): LocalDateTime[]

What we would change:

- Remote Proxy
  - Airlines have a list of flights
  - Control time of access from DB

- Divide Classes up
  - Single purpose
  - Did change to composition



What we learned:

- Learned:
  - Process of designing and implementing a Design Pattern
  - Refactoring and the making the code pretty
  - Managing our time and expectation

The End

Singleton

```java
public class HibernateUtil {
    private static SessionFactory session_factory;

    public static SessionFactory getSession_factory(){
        if (session_factory == null){
            Configuration config = new Configuration();
            session_factory = config.configure("hibernate.cfg.xml").buildSessionFactory();
        }

        return session_factory;
    }
}
```