Design Document

Requirements:

A plane reservation system(Name change to BookingSystem) shows the current reservation of seats on a plane, with function of booking, cancelling, modifying. Client (Name change to Person) can see the current updated reservation from the screen, and enter their choice to jump to book seat, cancel seat, or modify seat section. When booking a seat, ask user name and age, and then choose seat by telling row and column numbers, finally to choose food. After confirmation page which has name, age, seat, food, and time of reservation shows up, the current reservation will update on the screen. When cancelling a seat, users must verify their name and age as well as seat's position from the reservation history (Name change to BookingHistory) data. If matched, then they can cancel the seat, screen will update. Modifying seat reservation is to ask user enter the seat position, and then let them choose new seat and food. At the end it shows the confirmation page and updated seat reservation information. When client quit the system, data recorder (Name change to RecordingSystem) will save information into file. Next time, filereader (Name change to Plane) will update the screen for user.

Color codes: Nouns (either rejected, or name changes), Obvious Classes, Behaviors

Design:

How this program run:

- 1. main for reading file, file goes into plane
- 2. plane read the file and create seat array
- 3. array and some related information goes into BookingSystem
- 4. BookingSystem computing the logic part which are booking, cancelling and modify
- 5. BookingHistory accepts all information from system and store inside of array
- 6. After client quit, then transfers history into RecordingSystem to write into database and report

-Main: Call menu to let user enter file name to start program

-Person: Get the name and age to create a person and match if two persons are same

-Plane: Get the single data from the file, and send them to Seat class to make it like seat and install them on Plane.

-Seat: Accept data from file to make it to be a seat and match two seat is same type.

-Food: Create a food and match if food are same type

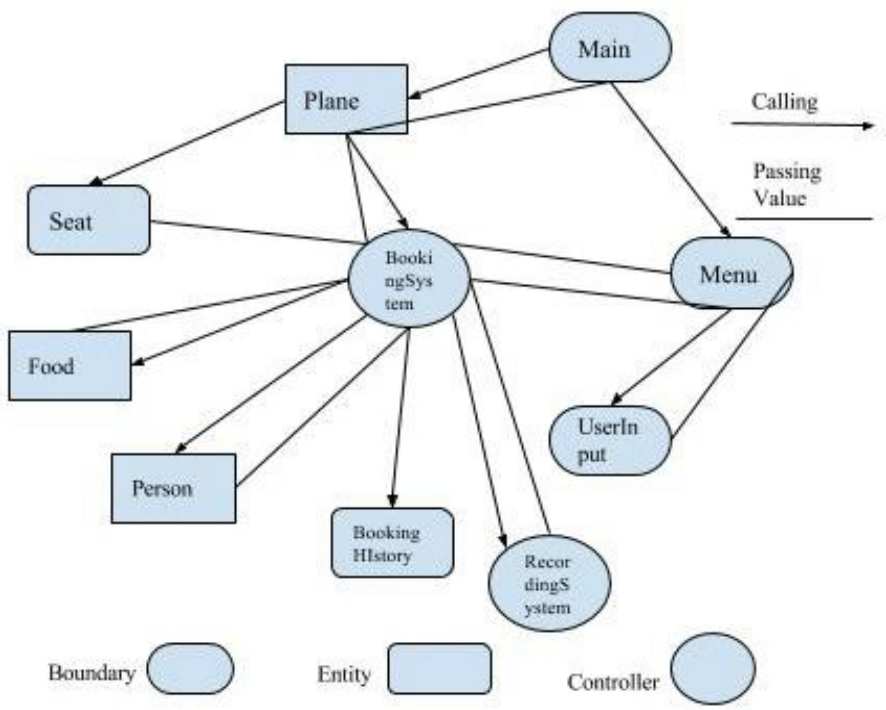
-BookingHistory: Get booking information and bind them to a history entity.

-Menu: interact with userInput ask user input and interact with BookingSystem to print menu and information on the screen.

-BookingSystem: Get seats and booking history on Plane, menu from Main, and other information related to seat and history. Run function book, cancel, modify, display seats, update history. Finally, passing history and filename into record system.

-UserInput: Ask user enter something

-RecordingSystem: get the history from BookingSystem and write them into file.



Testing:

1. User input: to check input mismatch
    a. they cannot enter the number does not show on the menu or seats position does not exist, and also they cannot enter non-numeric type when system requires number input.
    b. they cannot double book the seat being booked, cancel the seats never booked.
    c. to avoid user enter nothing, there is one method will catch that error.
    d. When user is required to enter y or n, check if their enter is valid.
2. Reading file: to check null pointer
    a. If there is no booking history, reading the file until last seat
    b. If there is booking history, reading booking history till the integer number following behind the last seat
    c. If all booking history has been cancelled, reading file will stop till the number 0 which represents the number of booking
3. BookingSystem: to check logical part
    a. Stop program if all seat has been booked
    b. Check the index of each loop, make sure it reset to 0 when it needs to be
4. Writing File; Do not let the program read the null inside of history array

Conclusions:

I would create Enum class for food instead of regular class if there were second chance to do it.

I learned the Object-Oriented design from this this assignment and how to organize the method in one class.