Ticket (entity)

- Store Ticket's flight number
- Store Ticket's departure and arrival city
- Store Ticket's departure and arrival time
- Store Ticket's passenger's name
- Store Ticket's passenger's username
- Store Ticket's passenger's seat class
- Store Ticket's seat number
- Store Ticket's price
- Store Ticket's id
- Store Ticket's boarding gate
- Store Ticket's Meal_choice
- Store Ticket's Luggage id.
- Getters/setters for everything above
- Generate a ticket by using all information above with toString method.

TicketManager
PHManager
PurchaseHistory
Meal_choice
Luggage

Flight (entity)

• *Store flight info:* * *flight number* * *origin of the flight* * *destination of the flight* * *departure and arrival time of fight* * *distance traveled from departure to destination* * ***total seat numbers and seat type*** * *the number of available seat* * *boarding gate* * *the array list of array list of seat numbers and seat class (*the inner seat number will be changed to"X +seatNum" tto indicate that the seat is no longer available.) ● Getters for attributes ● Overrided toString method ● ReserveOneSeat method ● **BuildTime method(helper method for constructor)** ● **BuildSeatArray method(helper method for constructor)**	FlightManager PHManager PriceCalculator **SmallFlight (its subclass)** **MediumFlight (its subclass)** **LargeFlight (its subclass)**

SmallFlight (entity)

- *Store flight info:*
- ** flight number*
  ** origin of the flight*
  ** destination of the flight*
  ** departure and arrival time of fight*
  ** distance traveled from departure to destination*
  ** total seat numbers*
  ** the number of available seat*
  ** boarding gate*
  ** the array list of array list of seat numbers and seat class (*the inner seat number will be changed to"X +seatNum" t to indicate that the seat is no longer available.) (t*his type of flight only contains**10 first class seats***)
- Getters for attributes
- Overrided toString method
- ReserveOneSeat method
- BuildTime method(helper method for constructor)
- BuildSeatArray method(helper method for constructor)
- Contains static variable smallSeatArray which predefined the seat array for this type of flight.

FlightManager

Flight(its parent class)

MediumFlight (entity)

- *Store flight info:*
- ** flight number*
  ** origin of the flight*
  ** destination of the flight*
  ** departure and arrival time of fight*
  ** distance traveled from departure to destination*
  ** total seat numbers*
  ** the number of available seat*
  ** boarding gate*
  ** the array list of array list of seat numbers and seat class (*the inner seat number will be changed to"X +seatNum" tto indicate that the seat is no longer available.) (t*his type of flight only contains* ***6 first class,14 business*****)**
- Getters for attributes
- Overrided toString method
- ReserveOneSeat method
- BuildTime method(helper method for constructor)
- BuildSeatArray method(helper method for constructor)
- Contains static variable mediumSeatArray which predefined the seat array for this type of flight.

FlightManager

Flight(its parent class)

LargeFlight (entity)

- *Store flight info:*
- ** flight number*
  ** origin of the flight*
  ** destination of the flight*
  ** departure and arrival time of fight*
  ** distance traveled from departure to destination*
  ** total seat numbers*
  ** the number of available seat*
  ** boarding gate*
  ** the array list of array list of seat numbers and seat class (*the inner seat number will be changed to"X +seatNum" to indicate that the seat is no longer available.) (t*his type of flight only contains* ***6 first,8 business,16 economy***)
- Getters for attributes
- Overrided toString method
- ReserveOneSeat method
- BuildTime method(helper method for constructor)
- BuildSeatArray method(helper method for constructor) ->build seatArray for the 3types of flights
- Contains static variable largeSeatArray which predefined the seat array for this type of flight.

FlightManager

Flight(its parent class)

Luggage(entity)

- *Store luggage info:*
- *Luggage weight*
- *Luggage ID (flight number + seat number)*

- Getters for attributes
- Overrided toString method
- BuildTime method(helper method for constructor)

LuggageManager

LuggageManager(entity)

- Contains a HashMap<String, Luggage> (default to be empty) maps luggage id to luggage
- Generate and add Luggage to this hashmap
- Access luggage by weight, flight number and seatnember
- Return luggage by luggage ID
- Cancel generated luggage
- To string method that returns all luggage information in luggage manager

Luggage

RewardsItem

- Store isRedeemed, name of item and points needed to redeem;
- Getters for all attributes
- Setter for isRedeemed
- RedeemItem method(*redeem a rewards item if the customer has earned the points needed for the item.)*
- *Overrided toString method*

Its subclasses:
Mug
Backpack
GiftCrad
BookstoreCoupon
TVoucher

Redeemable interface

Redeemable(interface)

- *Has method* **redeemItem** which inputs customer, if user has enough redeem points then add a redeemed RewardsItem to customer's purchaseHistory, and minus the points redeemed.

All the classes which implement it.
Mug
Backpack
GiftCrad
BookstoreCoupon
TVoucher

Mug/Backpack/GiftCrad/BookstoreCoupon/TVoucher

• *Inhertied all attributes from RewardsItem.* • *But each has different name and points.* • (**UofT limited edition backpack:** 3.5k points **, Mug:** 0.5k points, **Starbucks $20 gift card:** 2k points, **T-card $20 voucher:**1k points, **UofT bookstore $20 coupon:**1k points)	RewardsItem(parent class) Redeemable(interface)

RewardsItemFactory	
• *Get an object of type RewardsItem based on inputted String itemType.*	RewardsItem and its subclasses

Meal_choice

- Store the Meal's name and selected status
- Get the meal_choice's name
- Get the current selected status for this meal_choice
- Change the meal_choice selected status to be true.
- Assign the meal_choice to the given Ticket.
- *Overrided toString method*

Its subclasses:
Diabetic
gluten_intolerance
low_calories
vegan

Meal_selection interface

Meal_selection(interface)

• *Has method* choosemeal which inputs ticket, if user select the meal_choice then assgined the selected Meal_choice to ticket's meal_choice.	All the classes which implement it. Diabetic gluten_intolerance low_calories vegan Meal_choiceFactory

Diabetic/gluten_intolerance/low_calories/vegan/regular_meal

- *Inhertied all attributes from Meal_choice.*
- *Each has different name,and they are "*Diabetic*",* "Gluten Intolerance", "Low Calories","Vegan" and "Regular Meal".

Meal_choice(parent class)
Meal_selection(interface)

Meal_choiceFactory	
• *A Getter method getmeal which could return an object of type* Meal_choice *based on inputted String* mealType*. If the String match the* Meal_choice subclass's name, then return the related Meal_choice class. If none of the Meal_choice subclass's name could match, then return null.	Meal_choice and its subclasses (Diabetic, gluten_intolerance, low_calories, vegan)

Customer (entity)

- Stores customer personal information(name, password and username).
- Stores account balance, millage earned.
- Store membership information (including membership status and membership level : level 0 (If not membership) , level 1, level 2, level 3.)
- Store current redeem points for this customer.
- Store purchase history for this customer.
- Methods: Getter methods to get this customer's personal information such as name, username, current balance amount, current mileage amount, membership status, membership levels, redeem points and purchase history.
- Verify if the given password in string form is correct or not for this customer.
- Replace the given password, name and username in string form and replace the original personal information.
- Upgrade the current balance amount for this customer (Increase or decrease).
- Upgrade the current Millage amount for this customer (Increase or decrease).
- Decrease the current Millage amount for this customer based on the given redeem points
- Upgrade the current membership status for this customer (Change this customer's membership status to true, membership level will become 1).
- Upgrade the current membership level for this customer (if the customer's total miles reached 1000, and 5000 miles, then upgrade to next level).
- Replace the current redeem points based on current mileage amount devide by 5.
- Minus the current redeem points with the given points which used to redeemed.
- Show customer's personal information with toString method.

CustomerManager
Membership
PriceCalculator
PurchaseHistory

PurchaseHistory (entity)

● Store all tickets purchased and items redeemed by a customer ● Add new purchased tickets ● Add new items redeemed ● Remove existing purchased tickets (if ticket is canceled) ● Remove existing items redeemed (if rewarded item is returned)	Customer Ticket RewardsItem PHManager

Membership (Use-case)

- Add the customer to the Membership HashMap.
- Check if the customer exist in Membership HashMap.
- Check if the customer is Membership or not by the given customer.
- Check the customer's current Membership level by the given customer.
- Change the customer's Membership status if this customer's membership status is false by the given customer.
- Change the customer's Membership levels if this customer's membership status is false by the given customer.
- Return this customer if this customer is membership by the given username.
- Return this customer's balance if this customer is membership by the given username.
- Increase this customer's balance if this customer is membership.
- Decrease this customer's balance if this customer is membership.
- Increase this customer's Millage if this customer is membership.
- Calculate this customer's Redeem Points if this customer is membership.
- Return this customer's Redeem Points if this customer is membership.
- Minus this customer's Redeem Points if this customer is membership bu the given amounts.
- Display this customer's related information if this customer is membership by the given amounts.

Customer
Ticket
RewardsItem

TicketManager (Use-case)

● Store all sold tickets ● Generate tickets for customers ● Getter method ● Book tickets ● Get the Ticket's current meal selection ● Assign the user's select meal to this Ticket ● Return the mileage earned by purchasing a ticket so that it could be redeemed as points. ● **Cancel / Change tickets**	Ticket Flight Customer Luggage PurchaseHistory LuggageManager CustomerManager PHManager FlightManager

FlightManager (Use-case)

- Contains a HashMap<String, Flight> (default to be empty) maps flight number to flight.
- Add flight to this hashmap (create flight based on flight type and add it to the hashmap)
- Access flight by flight number
- Verify flight number
- Return flights by travel distance
- Provide a list that contains all flight numbers from the departure city and destination.
- **Provide a list that contains all flight numbers that satisfy the departure and arrival time.**
- **ToString method that returns all flight information in fm.**
- **Return ArrayList of seats and its' class information**
- *DisplayFlightInfoGUI:return a string of flight info and will call in GUI to display formatted flight information*

Flight
PHManager

CustomerManager (Use-case)

- Add new customers to customer manager system.
- Access each customer by the customer's username.
- Access each customer by the customer's username, and show the corresponding customer's personal information including: account balance)
- Access each customer by the customer, and show the corresponding customer's personal information including: purchase history)

- Check if the given username exist in the customer manager system or not.
- Check if the given password is correct for the given username in the customer manager system.
- Updates customer personal information(name, password, username).
- Updates account balance (increase or decrease) .
- Updates mile earned (increase or decrease) .
- Decrease the current Millage amount for this customer based on the given redeem points
- Check and Updates membership status.
- Check and Updates Membership levels.
- Get the current redeem point, calculate Redeem Points and minus Redeem Points
- Check and Updates current Millage amont.

Customer
Membership
PurchaseHistory

PHManager(Use-case)

- Contains a Hashmap <String, PurchaseHistory> that shows all information of the purchase history by customers' usernames.
- Update this map when a customer's purchase history is changed.
- Return the tickets in the purchase history of a customer stored in the hashmap
- Return the items redeemed by a customer from his/her purchase history stored in the hashmap

PurchaseHistory
Customer
Ticket
RewardsItem

LoginSystem (Use-case)

●Check if the user exists in users.csv ●Check if the username matches the password in users.csv ●Add user to users.csv ●Change a user's username, password, or full name in users.csv	GUI

PriceCalculator (Use-case)

● Apply discounts to ticket price if applicable. ● Apply penalty to late returning ticket. ● Apply penalty to overweight luggage. ● Apply penalty to the negative redeem points after returning a ticket.	Customer Flight Ticket

GUI **(**graphic user interface)	
● Have users login or create account ● Proceed user's request like booking tickets, managing accounts. ● For account management, users can see previously booked tickets or rewards items, see or update personal info, redeem rewards item(users have to join membership), load balance or join membership. ● For ticket booking, users can pick departure and arrival city to pick flights matched and seat, if users have sufficient balance, users will be able to book tickets, with balance subtracted, miles earned and redeem points earned. User could also select the meal for their Ticket.	Controller TicketManager FlightManager CustomerManager PHManager LuggageManager

Controller

- Deserialize all required managers.(so that previous state of these managers are restored)
- Execute GUI with these restored managers as inputs.

FlightManager
TicketManager
CustomerManager
PriceCalculator
PHManager
LuggageManager
GUI

FMSerialization/PHMSerialization/LMSerialization/ TMSerialization/CMSerialization	
● Has method saveFM/saveTM/savePHM/saveCM, which can deserialize the required manager, so that previous state of the manager is saved)	FlightManager TicketManager CustomerManager PHManager LuggageManager GUI Controller

FMDeserialization/PHMDeserialization/LMDeserialization/ TMDeserialization/CMDeserialization	
● Has method restoreFM/restoreTM/restorePHM/restoreCM, which can serialize the required manager, so that previous state of the manager is restored)	FlightManager TicketManager CustomerManager PHManager LuggageManager Controller