GP03 – Program Description Language

Table of Contents

1. Borders

1.1 Agent Border

```
public class AgentBorder {

   void LogIn(){
      AgentControl.LogIn(agentId, password)
   }

   void CreateNewCustomer(){
      AgentControl.CreateCustomerAccount(name, address, emailAddress,
phoneNumber, creditCard)
   }

   void ManageCustomer(){
      AgentControl.UpdateCustomer(customerID, name, address, emailAddress,
phoneNumber, creditCard)
   }

   void CreateNewFlightItinerary(){
      AgentControl.CreateNewItinerary(customerID, departureCity, arrivalCity,
departureDate, returnDate, preference)
   }

   void SearchFlights(){
      AgentControl.SearchFlights(itinerary)
   }

   void ReserveFlight(){
      AgentControl.ReserveFlight(flight)
   }

   void ModifyReservation(){
      AgentControl.ModifyReservation(customerID, departureCity, arrivalCity,
departureDate, returnDate, preference)
   }

   void CancelReservation(){
      AgentControl.CancelReservation(itinerary)
   }

   void CreatePriceWatch(){
      AgentControl.CreatePriceWatch(priceDesired, itinerary, notify)
   }

   void CancelWatch(){
      AgentControl.CancelWatch(itineary)
   }

   void ProvideMetWatches(){
      AgentControl.ProvideMetWatches()
   }

   void ProduceFlightList(){
      AgentControl.ProduceFlightList(itineraries)
   }

  void AddCredit (){
          AgentControl.AddCredit()
   }

   protected void UpdateFields(){
          //Updates Border Fields with Latest Entity Values
   }
}
```

1.2 Manager Border

```
public class ManagementBorder extends AgentBorder{

   void CreateNewAgent(){
      ManagerControl.CreateNewAgent(name, address, phoneNumber, emailAddress,
agentID, password)
   }

   void ModifyAgentAccount(){
      ManagerControl.UpdateAgent(agentID, name, address, emailAddress,
phoneNumber)
   }

   void UpdateFromCRATD(){
      ManagerControl.GetUpdatesFromCRATD()
   }

   void ManageFeeStructure(){
      ManagerControl.ManageFeeStructure(eflDatabaseInterface)
   }

   void ProduceContactReport(){
      ManagerControl.ProduceContactReport()
   }

   void ProduceFinancialReport(){
      ManagerControl.ProduceFinancialReport()
   }

   void AddCredit(){
      ManagerControl.AddCredit(creditAmount, customerEntity)
   }
}
```

2. Controls

2.2 Agent Control

```
public class AgentControl {

   protected AgentBorder agentBorder;
   protected EFLDatabase eflDatabase;
   protected ArrayList<ArrayList<FlightEntity>> currentFlightsList;

   public AgentBorder getAgentBorder() {
      return agentBorder;
   }

   public void setAgentBorder(AgentBorder agentBorder) {
      this.agentBorder = agentBorder;
   }

   public EFLDatabase getEflDatabase() {
      return eflDatabase;
   }

   public void setEFLDatabase(EFLDatabase eflDatabase) {
      this.eflDatabase = eflDatabase;
   }

   protected boolean LogIn(String agentID, String password){
      //If the ID is an existing index into the array, check the entity's password
against the entered    (supplied) password
      return
(eflDatabase.getAgents().get(Integer.parseInt(agentID)).getPassword().equals(pass
word));
   }

   protected CustomerEntity CreateCustomerAccount(String name, String address,
String emailAddress, String phoneNumber, CreditCardEntity creditCard){
      CustomerEntity newCustomer = new CustomerEntity();
      String customerID = "";
      customerID += eflDatabase.getCustomers().size();
      newCustomer.setCustomerID(customerID);
      newCustomer.setName(name);
      newCustomer.setAddress(address);
      newCustomer.setEmailAddress(emailAddress);
      newCustomer.setPhoneNumber(phoneNumber);
```

```
        newCustomer.setCreditCard(creditCard);
        newCustomer.setItineraries(new ArrayList<ItineraryEntity>());
        eflDatabase.getCustomers().add(newCustomer);
        return newCustomer;
    }



protected int CustomerLookUp(String customerID){
        //Returns the index into an ArrayList of existing customers. This index will also
correspond
        //to the customer's customerID. If the customer doesn't exist, -1 will be
returned, signaling
        //that the customer specified is an invalid customer.
        int customerIndex = -1;
        if(Character.isDigit(customerID.charAt(0))){
            //This means the string is a customer's phone number.
            for(int i = 0; i < eflDatabase.getCustomers().size(); i++){
                if(eflDatabase.getCustomers().get(i).phoneNumber.equals(customerID)){
                    customerIndex =
Integer.parseInt(eflDatabase.getCustomers().get(i).getCustomerID());
                }
            }
        }
        else if(!Character.isDigit(customerID.charAt(0))){
            //This means the string is a customer's name.
            for (int i = 0; i < eflDatabase.getCustomers().size(); i++){
                if(eflDatabase.getCustomers().get(i).name.equals(customerID)){
                    customerIndex =
Integer.parseInt(eflDatabase.getCustomers().get(i).getCustomerID());
                }
            }
        }
        return customerIndex;
    }

    protected CustomerEntity UpdateCustomer(String customerID, String name,
String address, String emailAddress, String phoneNumber, CreditCardEntity
creditCard){
         CustomerEntity customer;
        if(CustomerLookUp(customerID) != -1){
            customer = eflDatabase.getCustomers().get(CustomerLookUp(customerID));
            customer.setName(name);
            customer.setAddress(address);
            customer.setEmailAddress(emailAddress);
            customer.setPhoneNumber(phoneNumber);
         customer.setCreditCard(creditCard);
      }
      else{
         customer = new CustomerEntity();
         customer.setName("Invalid");
         customer.setAddress("Invalid");
         customer.setEmailAddress("Invalid");
         customer.setPhoneNumber("Invalid");
         //Deal with an invalid customerID accordingly.
      }
       return customer;
   }


  protected ItineraryEntity CreateNewItinerary(String customerID, String
departureCity, String arrivalCity, GregorianCalendar depatureDate,
GregorianCalendar returnDate, String preference){
      ItineraryEntity newItinerary = new ItineraryEntity();
      if(CustomerLookUp(customerID) != -1){
         CustomerEntity customer =
eflDatabase.getCustomers().get(CustomerLookUp(customerID));
         newItinerary.setItineraryID(customer.getItineraries().size());
         newItinerary.setDepartureCity(departureCity);
         newItinerary.setArrivalCity(arrivalCity);
         newItinerary.setDepartureDate(depatureDate);
         newItinerary.setReturnDate(returnDate);
         newItinerary.setPreference(preference);
         customer.getItineraries().add(newItinerary);
      }
      else{
         //Return message that creation of a new itinerary failed.
      }
      return newItinerary;
   }

   protected ArrayList<FlightEntity> SearchFlights(ItineraryEntity itinerary){
      //     Traverse all flights comparing by preference:
      //     switch(itinerary.preference)
      //     CASE:cheapest fare -- refers to the overall cost of the trip.
      //     CASE:shortest time -- refers to the shortest flight time from the departure
until the last arrival.
      //     CASE:shortest number of flights -- refers to the least number of connecting
flights between the departure and arrival.

      ArrayList<FlightEntity> flightEntities = new ArrayList<>();
      switch(itinerary.getPreference()){
         case "Cheapest Fare":
            //Sort flights according to ASCENDING overall cost.
            break;

         case "Shortest Time":
            //Sort flights according to ASCENDING trip time.
            break;

         case "Shortest Number Of Flights":
            //Sort flights according to ASCENDING number of flights.
            break;
      }
      return flightEntities;
   }


 protected void ReserveFlight(ArrayList<FlightEntity> flights, String customerID, int
itineraryID){
      //flights will be a list of flights obtained from SearchFlights()
      //customerID will be provided by the GUI
      //itineraryID will be provided by the GUI
      if(CustomerLookUp(customerID) != -1){   //Checks that the customer is a valid
customer.
        CustomerEntity customer =
eflDatabase.getCustomers().get(CustomerLookUp(customerID));
         if (customer.getItineraries().get(itineraryID).getFlights().isEmpty()){
            customer.getItineraries().get(itineraryID).setFlights(flights);
            //Send reservations to imaginary airline
         }
      }
      else{
         //Notify the agent that the customerID is invalid.
      }

   }

   protected void ModifyReservation(ArrayList<FlightEntity> flights, int itineraryID,
String customerID, String departureCity, String arrivalCity, GregorianCalendar
departureDate, GregorianCalendar returnDate, String preference){
      //Update a specific itinerary in the list of itineraries a customer has.
      CustomerEntity customer;
      ItineraryEntity itinerary;
      if(CancelReservation(customerID,itineraryID)){
         //CancelReservation checks that the customer is a valid customer
         //and that the itineraryID refers to a valid itinerary
         //and then removes the itinerary.
         customer = eflDatabase.getCustomers().get(CustomerLookUp(customerID));
         itinerary = customer.getItineraries().get(itineraryID);
         if (!itinerary.getFlights().isEmpty()){              //If the itinerary is empty,
            itinerary.setArrivalCity(arrivalCity);
            itinerary.setDepartureCity(departureCity);
            itinerary.setDepartureDate(departureDate);
            itinerary.setFlights(flights);
            itinerary.setItineraryID(itineraryID);
            itinerary.setPreference(preference);
            itinerary.setReturnDate(returnDate);
            //Send updated reservations to imaginary airline
         }
      }
      else{
         //Notify the agent that the provided customerID is invalid.
      }
   }


 protected boolean CancelReservation(String customerID, int itineraryID){
      boolean wasCancelled = false;
      CustomerEntity customer;
      if(CustomerLookUp(customerID) != -1){
         customer = eflDatabase.getCustomers().get(CustomerLookUp(customerID));
         if(itineraryID < customer.getItineraries().size()){
            customer.getItineraries().remove(itineraryID);
            wasCancelled = true;
         }
      }
      return wasCancelled;
   }

   protected PriceWatchEntity CreatePriceWatch(double priceDesired,
ItineraryEntity itinerary, boolean notify){
      //We assume at this point that we have a valid customer
      PriceWatchEntity priceWatch = new PriceWatchEntity();
      priceWatch.setPriceWatchExpiration();
      priceWatch.setPriceWatchPrice(priceDesired);
      priceWatch.setSendNotifyText(notify);
      itinerary.setPriceWatch(priceWatch);
      return priceWatch;
   }

    protected void CancelWatch(ItineraryEntity itinerary){
       //This will either get called when a watch expires,
       //or when a customer opts to cancel a watch.
```

```
        //It sets the price watch of a particular itinerary to null
        itinerary.setPriceWatch(null);
    }

     protected double ComputeCost(ItineraryEntity itinerary){
        double cost = 0.00;
        //Calculate Overall Cost//
        //1 Have CRATD provide all airline and airport cost information.
        //2 Have CRATD provide cost per mile per airline
        //3 Check to see if the trip is a multileg trip.
        //4 If the trip is a single leg trip,
        //5 cost = costPerMilePerAirline * distance;
        //6 If the trip is a multilegged trip,
        //7 Check to see if any subsequent legs are chartered by the same airline.
        //8 If there are subsequent legs chartered by the same airline,
        //9 costOfSubsequentLegs must be determined based on which legs are
chartered by the same              airline.
        //  consecutive legs being chartered by the same airline result in a 10%
reduction in costPerMilePerAirline Fee
        //10 cost = (costPerMilePerAirlineOfFirstLeg*distanceOfFirstLeg) +
costOfSubsequentLegs;
        return cost;
     }

    protected ArrayList<PriceWatchEntity> ProvideMetWatches(){
       ArrayList<PriceWatchEntity> metWatches = null;
       //Traverse watches, adding any "met" watches to the metWatches ArrayList.
       //while traversing check to see if currentDate==expiration date
       //if equal then call cancel watch
       return metWatches;
    }

    protected void ProduceFlightReceipt(ItineraryEntity itinerary){
       double totalCost = ComputeCost(itinerary);
       String flightReceipt = "";
       flightReceipt += "Departure City: ";
       flightReceipt += itinerary.getDepartureCity() + "\n";
       flightReceipt += "Arrival City: ";
       flightReceipt += itinerary.getArrivalCity() + "\n";
       flightReceipt += "Departure Date: ";
       flightReceipt += itinerary.getDepartureDate() + "\n";
       flightReceipt += "Arrival Date: ";
       flightReceipt += itinerary.getReturnDate() + "\n";
       flightReceipt += "Total Cost: ";
       flightReceipt += totalCost;
       //Call PrintReceipt
```

```
  }

  protected void ProduceFlightList(ItineraryEntity itinerary){
    String flightList = "";
    FlightEntity flight;
    ArrayList<FlightEntity> flights;
    flights = SearchFlights(itinerary);
    if(!flights.isEmpty()){                    //If at least one flights exists in the list
      for(int i = 0; i < flights.size(); i++){   //traverse the flights, adding info to a
string.
        flight = flights.get(i);
        flightList += flight.toString();       // .toString method returns all important
info on a flight.
      }
      //Call PrintFlightLlist
    }
  }


 protected double computeFlightTime(String originAirport, String
destinationAirport, GregorianCalendar departureTime, GregorianCalendar
arrivalTime){
    double flightTime = 0.0;
    //1 Search EFLDatabase for time offsets from each airport
    //2 Convert each local time to GMT time zone
    //3 Calculate difference in time
    //4 Convert to Double

    return flightTime;
  }

  protected void AddCredit(double creditAmount, String customerID){
    //If a customer has been referred by a preexisting customer,
    //credit will be added to the referring customer's account.
    double accountCredit;
    accountCredit =
eflDatabase.getCustomers().get(CustomerLookUp(customerID)).getAccountCredit()
;
    accountCredit += creditAmount;
  }
}
```

2.2 Manager Control

```
public class ManagerControl extends AgentControl {
   private ManagerBorder managerBorder;
   private CRATD cratd;

   private AgentEntity CreateNewAgent(String name, String address, String
phoneNumber, String emailAddress, String agentID, String password){
        AgentEntity agent = new AgentEntity();
        agent.setAddress(address);
        agent.setAgentID(agentID);
        agent.setEmailAddress(emailAddress);
        agent.setIsManager(false);
        agent.setName(name);
        agent.setPassword(password);
        agent.setPhoneNumber(phoneNumber);
        return agent;
   }


  private AgentEntity ModifyAgentAccount(String agentID, String name, String
address, String phoneNumber, String emailAddress, String password){
        AgentEntity agent = eflDatabase.getAgents().get(Integer.parseInt(agentID));
        agent.setAddress(address);
        agent.setAgentID(agentID);
        agent.setEmailAddress(emailAddress);
        agent.setIsManager(false);
        agent.setName(name);
        agent.setPassword(password);
        agent.setPhoneNumber(phoneNumber);
        return agent;
   }

   private void getUpdatesFromCRATD(){
      CRATD.getInstance();
      //Loads updates from CRATD
   }

   private void ManageFeeStructure(EFLDatabase eflDatabase, double newFee){
      //Manage Cougar Path Travel's Fee Structure.
      eflDatabase.setFee(newFee);
   }

   private void ProduceDailyReport(EFLDatabase eflDatabase){
      String dailyReport = "";
```

```
        dailyReport += ProduceContactReport(eflDatabase);
        dailyReport += ProduceFinancialReport(eflDatabase);
        //Print Daily Report
    }

    private String ProduceContactReport(EFLDatabase eflDatabase){
        String contactReport = "";
        contactReport += eflDatabase.getCustomers().toString();
        //This adds the new customers' contact information for the day to the report.
        return contactReport;
    }

    private String ProduceFinancialReport(EFLDatabase eflDatabase){
        String financialReport = "";
        financialReport += eflDatabase.getFlights().toString();
        //This adds the new customers' contact information for the day to the report.
        return financialReport;
    }

   protected void AddCredit(double creditAmount, String customerID){
        //A manager can add credit to a customer's account whenever they want.
        double accountCredit;
        accountCredit =
eflDatabase.getCustomers().get(CustomerLookUp(customerID)).getAccountCredit()
;
        accountCredit += creditAmount;
    }

    public ManagerBorder getManagerBorder() {
        return managerBorder;
    }

    public void setManagerBorder(ManagerBorder managerBorder) {
        this.managerBorder = managerBorder;
    }

    public CRATD getCratd() {
        return cratd;
    }


    public void setCratd(CRATD cratd) {
        this.cratd = cratd;
    }
}
```

3. Entities

3.1 Agent Entity

```
public class AgentEntity extends PersonEntity{

   private String agentID;
   private String password;
   private boolean isManager;

   public String getAgentID() {
      return agentID;
   }

   public void setAgentID(String agentID) {
      this.agentID = agentID;
   }

   public String getPassword() {
      return password;
   }

   public void setPassword(String password) {
      this.password = password;
   }
   public boolean isManager() {
      return isManager;
   }

   public void setIsManager(boolean isManager) {
      this.isManager = isManager;
   }

}
```

3.2 Airport Entity

```
public class AirportEntity {
   private String airlineAbbreviation;
   private int timeZoneOffset;
   private int xCoordinate;
   private int yCoordinate;
   private double airportFee;
   private String nameOfAirport;

   public String getAirlineAbbreviation() {
      return airlineAbbreviation;
   }

   public void setAirlineAbbreviation(String airlineAbbreviation) {
      this.airlineAbbreviation = airlineAbbreviation;
   }

   public int getTimeZoneOffset() {
      return timeZoneOffset;
   }

   public void setTimeZoneOffset(int timeZoneOffset) {
      this.timeZoneOffset = timeZoneOffset;
   }

   public int getxCoordinate() {
      return xCoordinate;
   }

   public void setxCoordinate(int xCoordinate) {
      this.xCoordinate = xCoordinate;
   }

   public int getyCoordinate() {
      return yCoordinate;
   }

   public void setyCoordinate(int yCoordinate) {
      this.yCoordinate = yCoordinate;
   }

   public double getAirportFee() {
      return airportFee;
   }
```

```
    public void setAirportFee(double airportFee) {
        this.airportFee = airportFee;
    }

    public String getNameOfAirport() {
        return nameOfAirport;
    }

    public void setNameOfAirport(String nameOfAirport) {
        this.nameOfAirport = nameOfAirport;
    }

}
```

3.3 Credit Card Entity

```
public class CreditCardEntity {
   private String holderName;
   private String cardType;
   private String cardNumber;
   private GregorianCalendar expirationDate;
   private String csvNumber;
   private String billingAddress;

   public String getHolderName() {
      return holderName;
   }

   public void setHolderName(String holderName) {
      this.holderName = holderName;
   }

   public String getCardType() {
      return cardType;
   }

   public void setCardType(String cardType) {
      this.cardType = cardType;
   }

   public String getCardNumber() {
      return cardNumber;
   }

   public void setCardNumber(String cardNumber) {
      this.cardNumber = cardNumber;
   }

   public GregorianCalendar getExpirationDate() {
      return expirationDate;
   }

   public void setExpirationDate(GregorianCalendar expirationDate) {
      this.expirationDate = expirationDate;
   }

   public String getCsvNumber() {
      return csvNumber;
   }
```

```
    public void setCsvNumber(String csvNumber) {
        this.csvNumber = csvNumber;
    }

    public String getBillingAddress() {
        return billingAddress;
    }

    public void setBillingAddress(String billingAddress) {
        this.billingAddress = billingAddress;
    }

}
```

3.4 Customer Entity

```
public class CustomerEntity extends PersonEntity{

   private String customerID;
   private CreditCardEntity creditCard;
   private double accountCredit;
   private ArrayList<ItineraryEntity> itineraries;

   //This ArrayList of integers refers to the flight numbers that a customer has
reserved.
   //private ArrayList<Integer> reservedFlights;

   public String getCustomerID() {
      return customerID;
   }

   public void setCustomerID(String customerID) {
      this.customerID = customerID;
   }

   public CreditCardEntity getCreditCard() {
      return creditCard;
   }

   public void setCreditCard(CreditCardEntity creditCard) {
      this.creditCard = creditCard;
   }

   public double getAccountCredit() {
      return accountCredit;
   }

   public void setAccountCredit(double accountCredit) {
      this.accountCredit = accountCredit;
   }

   public ArrayList<ItineraryEntity> getItineraries() {
      return itineraries;
   }

   public void setItineraries(ArrayList<ItineraryEntity> itineraries) {
      this.itineraries = itineraries;
   }
   }
```

3.5 Flight Entity

```
public class FlightEntity {

  private String airlineAbbreviation;
  private int flightNumber;
  private String nameOfAirline;
  private double costPerMile;
  private GregorianCalendar departureTime;
  private String originAirport;
  private GregorianCalendar arrivalTime;
  private String destinationAirport;
  private int stopsDuringFlight;
  private double totalCost;
  private double travelTime;


  public String getAirlineAbbreviation() {
    return airlineAbbreviation;
  }

  public void setAirlineAbbreviation(String airlineAbbreviation) {
    this.airlineAbbreviation = airlineAbbreviation;
  }

  public int getFlightNumber() {
    return flightNumber;
  }

  public void setFlightNumber(int flightNumber) {
    this.flightNumber = flightNumber;
  }

  public String getNameOfAirline() {
    return nameOfAirline;
  }

  public void setNameOfAirline(String nameOfAirline) {
    this.nameOfAirline = nameOfAirline;
  }

  public double getCostPerMile() {
    return costPerMile;
  }
```

```
public void setCostPerMile(double costPerMile) {
   this.costPerMile = costPerMile;
}

public GregorianCalendar getDepartureTime() {
   return departureTime;
}

public void setDepartureTime(GregorianCalendar departureTime) {
   this.departureTime = departureTime;
}

public String getOriginAirport() {
   return originAirport;
}

public void setOriginAirport(String originAirport) {
   this.originAirport = originAirport;
}

public GregorianCalendar getArrivalTime() {
   return arrivalTime;
}

public void setArrivalTime(GregorianCalendar arrivalTime) {
   this.arrivalTime = arrivalTime;
}

public String getDestinationAirport() {
   return destinationAirport;
}

public void setDestinationAirport(String destinationAirport) {
   this.destinationAirport = destinationAirport;
}

public int getStopsDuringFlight() {
   return stopsDuringFlight;
}

public void setStopsDuringFlight(int stopsDuringFlight) {
   this.stopsDuringFlight = stopsDuringFlight;
}

public double getTotalCost() {
   return totalCost;
```

```
    }

    public void setTotalCost(double totalCost) {
        this.totalCost = totalCost;
    }

    public double getTravelTime() {
        return travelTime;
    }

    public void setTravelTime(double travelTime) {
        this.travelTime = travelTime;
    }


}
```

3.6 Itinerary Entity

```
public class ItineraryEntity {
   private int itineraryID;
   private String departureCity;
   private String arrivalCity;
   private GregorianCalendar departureDate;
   private GregorianCalendar returnDate;
   private int numberOfTravelers;
   private ArrayList<String> travelerNames;
   private String preference;
   private CreditCardEntity creditCard;
   private PriceWatchEntity priceWatch;
   private ArrayList<FlightEntity> flights;

   public int getItineraryID() {
      return itineraryID;
   }

   public void setItineraryID(int itineraryID) {
      this.itineraryID = itineraryID;
   }

   public String getDepartureCity() {
      return departureCity;
   }

   public void setDepartureCity(String departureCity) {
      this.departureCity = departureCity;
   }

   public String getArrivalCity() {
      return arrivalCity;
   }

   public void setArrivalCity(String arrivalCity) {
      this.arrivalCity = arrivalCity;
   }

   public GregorianCalendar getDepartureDate() {
      return departureDate;
   }

   public void setDepartureDate(GregorianCalendar departureDate) {
      this.departureDate = departureDate;
```

```
  }

  public GregorianCalendar getReturnDate() {
    return returnDate;
  }

  public void setReturnDate(GregorianCalendar returnDate) {
    this.returnDate = returnDate;
  }

  public int getNumberOfTravelers() {
    return numberOfTravelers;
  }

  public void setNumberOfTravelers(int numberOfTravelers) {
    this.numberOfTravelers = numberOfTravelers;
  }

  public ArrayList<String> getTravelerNames() {
    return travelerNames;
  }

  public void setTravelerNames(ArrayList<String> travelerNames) {
    this.travelerNames = travelerNames;
  }

  public String getPreference() {
    return preference;
  }

  public void setPreference(String preference) {
    this.preference = preference;
  }

  public CreditCardEntity getCreditCard() {
    return creditCard;
  }

  public void setCreditCard(CreditCardEntity creditCard) {
    this.creditCard = creditCard;
  }

  public PriceWatchEntity getPriceWatch() {
    return priceWatch;
  }
```

```
   public void setPriceWatch(PriceWatchEntity priceWatch) {
      this.priceWatch = priceWatch;
   }

   public ArrayList<FlightEntity> getFlights() {
      return flights;
   }

   public void setFlights(ArrayList<FlightEntity> flights) {
      this.flights = flights;
   }

}
```

3.7 Person Entity

```
public abstract class PersonEntity {

   protected String name;
   protected String address;
   protected String emailAddress;
   protected String phoneNumber;

   public String getName(){
      return name;
   }

   public void setName(String name){
      this.name=name;
   }

   public String getAddress(){
      return address;
   }

   public void setAddress(String address){
      this.address=address;
   }
   public String getEmailAddress(){
      return emailAddress;
   }

   public void setEmailAddress(String emailAddress){
      this.emailAddress=emailAddress;
   }

   public String getPhoneNumber(){
      return phoneNumber;
   }

   public void setPhoneNumber(String phoneNumber){
      this.phoneNumber=phoneNumber;
   }

}
```

3.8 Price Watch Entity

```
public class PriceWatchEntity {
   private double priceWatchPrice;
   private boolean sendNotifyText;
   private GregorianCalendar priceWatchExpiration;

   public double getPriceWatchPrice() {
      return priceWatchPrice;
   }

   public void setPriceWatchPrice(double priceWatchPrice) {
      this.priceWatchPrice = priceWatchPrice;
   }

   public boolean isSendNotifyText() {
      return sendNotifyText;
   }

   public void setSendNotifyText(boolean sendNotifyText) {
      this.sendNotifyText = sendNotifyText;
   }

   public GregorianCalendar getPriceWatchExpiration() {
      return priceWatchExpiration;
   }

   public void setPriceWatchExpiration() {
      //automatically set 30 days from today
      //this.priceWatchExpiration = TODAY + 30 DAYS;
   }


}
```

4. Databases

4.1 EFL Database

```
public class EFLDatabase {
   private EFLDatabase eflDatabase;
   private double fee;
   private ArrayList<AirportEntity> airports;
   private ArrayList<ArrayList<FlightEntity>> flights;
   private ArrayList<AgentEntity> managerEntities;    //A manager is a specialized
version of an agent and has more abilities.
   private ArrayList<AgentEntity> agentEntities;
   private ArrayList<CustomerEntity> customerEntities;

   private static EFLDatabase instance = null;

   private EFLDatabase EFLDatabase(){
      return new EFLDatabase();
   }
   public static EFLDatabase getInstance(){
      if(instance == null){
       instance = new EFLDatabase();
      }
      return instance;
   }

   public ArrayList<AirportEntity> getAirports() {
      return airports;
   }

   public void setAirports(ArrayList<AirportEntity> airports) {
      this.airports = airports;
   }

   public ArrayList<ArrayList<FlightEntity>> getFlights() {
      return flights;
   }

   public void setFlights(ArrayList<ArrayList<FlightEntity>> flights) {
      this.flights = flights;
   }

   public ArrayList<AgentEntity> getManagers() {
      return managerEntities;
```

```
  }

  public void setManagers(ArrayList<AgentEntity> managers) {
    this.managerEntities = managers;
  }


 public ArrayList<AgentEntity> getAgents() {
    return agentEntities;
  }

  public void setAgents(ArrayList<AgentEntity> agents) {
    this.agentEntities = agents;
  }

  public ArrayList<CustomerEntity> getCustomers() {
    return customerEntities;
  }

  public void setCustomers(ArrayList<CustomerEntity> customers) {
    this.customerEntities = customers;
  }

  public double getFee() {
    return fee;
  }

  public void setFee(double fee) {
    this.fee = fee;
  }
}
```

4.2 CRATD

```
public class CRATD {
   private static CRATD instance = null;
   private CRATD CRATD(){
      return new CRATD();
   }
   public static CRATD getInstance(){
      if(instance == null){
        instance = new CRATD();
      }
      return instance;
   }
   public void getCRATDUpdates(){
      //Can only be performed by a manager
      //Will essentially update the system with the most current info.
   }
}
```