• flightScheduler() will schedule required number of flights e.g. 10, 20 randomly. addCustomerToFlight() will add the customer to the current flight customers list. • calculateFlightTime() calculates the flight time or the time airplane was in travel. • isUniqueData() returns true if the entered • fetchArrivalTime() calculates the date time when the airplane userName or email already exists in the list. touches the ground. • addFlightToCustomerList() will associate that flight with the current customer • deleteFlight() will delete the flight listed in the flight schedule.(Only Admin can do) • calculateDistance() calculates the distance between the airports of source and destination(selected randomly from the 2D array) cities in Miles and Kilometers. • distanceMeasurementInstructions() displays some common instructions, how the flight time is calculated, when to reach at airport etc... <persistent> Customer • displayFlightSchedule() displays the generated flight shcedule. RandomGenerator - userID:String Flight schedule changes every time the program runs - name:String • createNewFlightsAndTime() generates new flight with its - randomNum:String - email:String - destinations:String[][] required attributes and then it calls the - password:String - address:String getNearestHourQuarter() method to round off the minutes to customer has an ID - phone:String the nearest quarter of the hour... - randomIDGen():String - randomDestinations():String[][] - flightsRegisteredByUser:List<Flight> - randomNumOfSeats():int - numOfTIcketsBookedByUser:List<Integer> - randomFlightNumGen(int, int):String <u>- customerCollection:List<Customer></u> + Customer() + Customer(String, String, String, String, int) + addNewCustomer():void + isUniqueData(String, String):boolean User <persistent> + editUserInfo(String):void + deleteUser(String):void + searchUser(String):void adminUserNameAndPassword:String[][] + toString(int):String customersCollection:List<Customer> + addNewFlightToCustomerList(Flight):void + addExistingFlightToCustomerList(int, int):void -can be-+ displayCustomersData():void + displayMainMenu():void has flight <persistent> Flight + displayHeader():void + mannualInstructions():void + randomIDDisplay():String + welcomScreen(int):void + displayArtWork():void + printArtWork(int):void flightSchedule:String + settersAndGetters():<respectiveReturnTypes> + getCustomersCollection():List<Customer> flightNumber:String - fromWhichCity:String - gate:String - toWhichCity:String distanceInMiles:double is user registered - distanceInKm:double flightTime:String - numOfSeatsInTheFlight:int RolesAndPermissions - listOfRegisteredCustomersInTheFlight:List<Customer> has registered - customerIndex:int - flightList:List<Flight> - nextFlightDay:int + isPrivilligedUserOrNot(String, String):int + isPassengerRegistered(String, String):String 0...* + Flight() + Flight(String, String, int, String[][], String[], String) FlightReservation + flightScheduler():void + addNewCustomerToFlight(Customer):void <<Interface>> - c1:Customer + addTicketsToExistingCustomer(Customer, int):void DisplayClass - flight:Flight + isCustomerAlreadyRegistered(List<Customer>, Customer):boolean flightIndexInFlightList:int + calculateFlightTime(double):String + fetchArrivalTime():String + displayRegisteredUsersForAllFlights():void + deleteFlight():void + displayRegisteredUsersForASpecificFlight():void + calculateDistance(double, double, double, double):String[] + FlightReservation() + displayHeaderForUsers():void + radianToDegree():double + FlightReservation(FlightReservation) + displayFlightsRegisteredByOneUser():void + degreeToRadian(double):double + bookFlight(String, int, String):void + displayArtWork():void + displayFlightSchedule():void + cancelFlight(String, String):void + toString(int):String + addNumberOfTicketsToExistingBookedFlight(Customer, int):void + createNewFlightsAndTime():String + addNumberOfTicketsForNewFlight(Customer, int):void + getNearestHourQuarter(LocalDateTime):LocalDateTime + addNumberOfTicketsToUser(List<Flight>, Flight, Customer, int):void • bookFlight() books the user in that flight.. + settersAndGetters():<respectiveReturnTypes> + isFlightAlreadyAddedToCustomerList(List<Flight>,Flight):boolean • cancelFlight() cancels the registeration of the user in + flightStatus():boolean + equals():boolean that flight + toString(int, Flight, Customer):String flightStatus() will check if the flight is on schedule or + flightIndex(List<Flight>, Flight):int has been cancelled. + displayFlightsRegisteredByOneUser(String):void isFlightAlreadyAddedToCustomer() will check if the + toString(int, Customer):String current flight is already assoicaited with the user or + displayHeaderForUsers(Flight, List<Customer>):void <<abstract>> not. + displayRegisteredUsersForAllFlights():void **FlightDistance** displayFlightsRegisteredByOneUser() will display all + displayRegisteredUsersForASpecificFlight(String):void flights registered by a user. + displayArtWork(int):void + settersAndGetters():<respectiveReturnType> displayRegisteredUsersForAllFlights() will display all registered passenger for all flights displayRegisteredUsersForASpecificFlight() will + toString(int):String + calculateDistance(double, double, double, double):String[] display all registered passenger for a specific flights + distanceMeasurementInstructions():void