Class: CSE 3320
Semester: Spring 2016
Student Name: Craig Lautenslager
Student ID: 1000873407
Assignment: Project 4

Program Design

The program I have designed is written in a single class that creates an instance of itself and through which, it's methods are run. Overall the program has 4 methods: the main method where the program starts, a menu method that handles the program's main menu, a departures method that handles the task of creating a departure and an arrivals method that handles the task of creating an arrival time for a flight instance. Both the departure and arrival methods open and close a connection to the mysql server separately.

The **main()** method, is the starting point for the program and it is here that an instance of the class is created. The method then starts a while loop that executes the menu method of the newly created object and only exits when the menu returns false.

The **menu()** method, prints the different options then waits for the user to select either departures, arrivals or to exit the program. Using a switch statement, if the user types a '1' the departures method runs and upon completion, returns true. If a '2' is typed the arrivals method is run and true is returned. If 3 is typed, the method returns false which essentially stops the main method's while loop, ending the program. If something other than these three option is input, the method returns true and the main method continues to loop.

The **departures()** method starts by loading the driver and establishing the connection to the database. It then prompts the user for and stores the desired flight number, leg sequence and date of flight. A query then attempts to select the flight legs with matching flight number and leg sequence to determine if at least one exists. If not, an error is printed and the program returns to the menu. If this is the first leg sequence, the program runs a query for a flight leg instance looking for a tuple that matches all three user specifications. If so, the flight leg instance already exists, an error is printed and the program returns to the menu. If this is not the first leg, the program queries for a flight leg instance that has a leg sequence number one lower than the one specified. The program stores the arrival time of the previous leg. If the leg does not exist or the leg does not have an actual arrival time, an error is printed and the program returns to the menu.

If the method has not already returned, it prompts and stores a departure time from the user and assembles it into the required mysql time format. It also prompts and stores a specified pilot ID, before querying the Pilot table for a match. If it returns false, indicating no such pilot exists, an error is printed and the program returns to the menu. Then the method takes the previously stored arrival time and compares the time to the requested departure time. If it is equal or earlier an error is printed and the program returns to the menu. Next, if this is the first leg, an update statement is run, adding a row to the Flight Instance table with the flight number and date. Finally, if the program had not previously returned for an error: an update is run that adds a row to the flight leg instance table and containing all the previously specified information minus the actual arrival time which is set to null. The Result Set, Statement feed and connection are all then closed.

The **arrivals()** method starts by loading the driver and establishing the connection to the database. It then prompts the user for and stores the desired flight number, leg sequence and date of flight. A query is then run that selects the flight leg instance that matches the user specification. If this result is returned as false, an error is printed and the program returns to the menu. From the previous results, the arrival time of the instance is retrieved; if the time isn't null, an error is printed and the program returns to the menu. The departure time of the tuple is then stored. If the method has not already returned, it prompts and stores an arrival time from the user and assembles it into the required mysql time format. Then the method takes the previously stored departure time and compares the time to the requested arrival time. If it is equal or later, an error is printed and the program returns to the menu. Finally, an update statement is run that sets the previously null 'Actual Arrival' time of the tuple to the specified time. The Result Set, Statement feed and connection are all then closed.