Flex Project Submission 1

Jeremy Hernandez

Project Idea: Trip Planning Program

Purpose:

Create a program that will help the user plan a trip and calculate the total amount of expenses. The expenses will include flight tickets and hotel rates.

How it works:

The program will start by asking the user the name of the destination, the reason whether business or vacation, the start and return dates. It will then proceed to ask the departure city and state and give the user the option to choose from three different airlines each of which will have different ticket prices. This price will be random and therefore the prices will be different each time the program runs. The program will do this for departure flight and return flight. The program will also ask for number of tickets in order to make the adjustments to the trip total. The program will then give the user the option to choose from three hotels. Each hotel will have their own randomize price and the user will specify the number of rooms to book. If the number of the party is greater than 4 then the program will display a message stating that the user must book more than one room. Once the user has stated the number of rooms, the program will display the information for the trip. This information will include the destination, the start and end date, the depart and return airline, the price per tickets, hotel name, price per night, number of nights, rooms and the overall cost of the entire trip. Finally, the user will have the option to run the program again or exit.

Inheritance: Flight and Hotel subclasses inherit from the Trip superclass because they add functionality to the superclass, it simplifies the design of the program and reduces code duplication. The Trip, Flight, and Hotel objects are all part of a domain that is trying to provide a trip planning service. Also, flight and hotel objects are types of the Trip object.

Specifications:

This program will contain 3 classes:

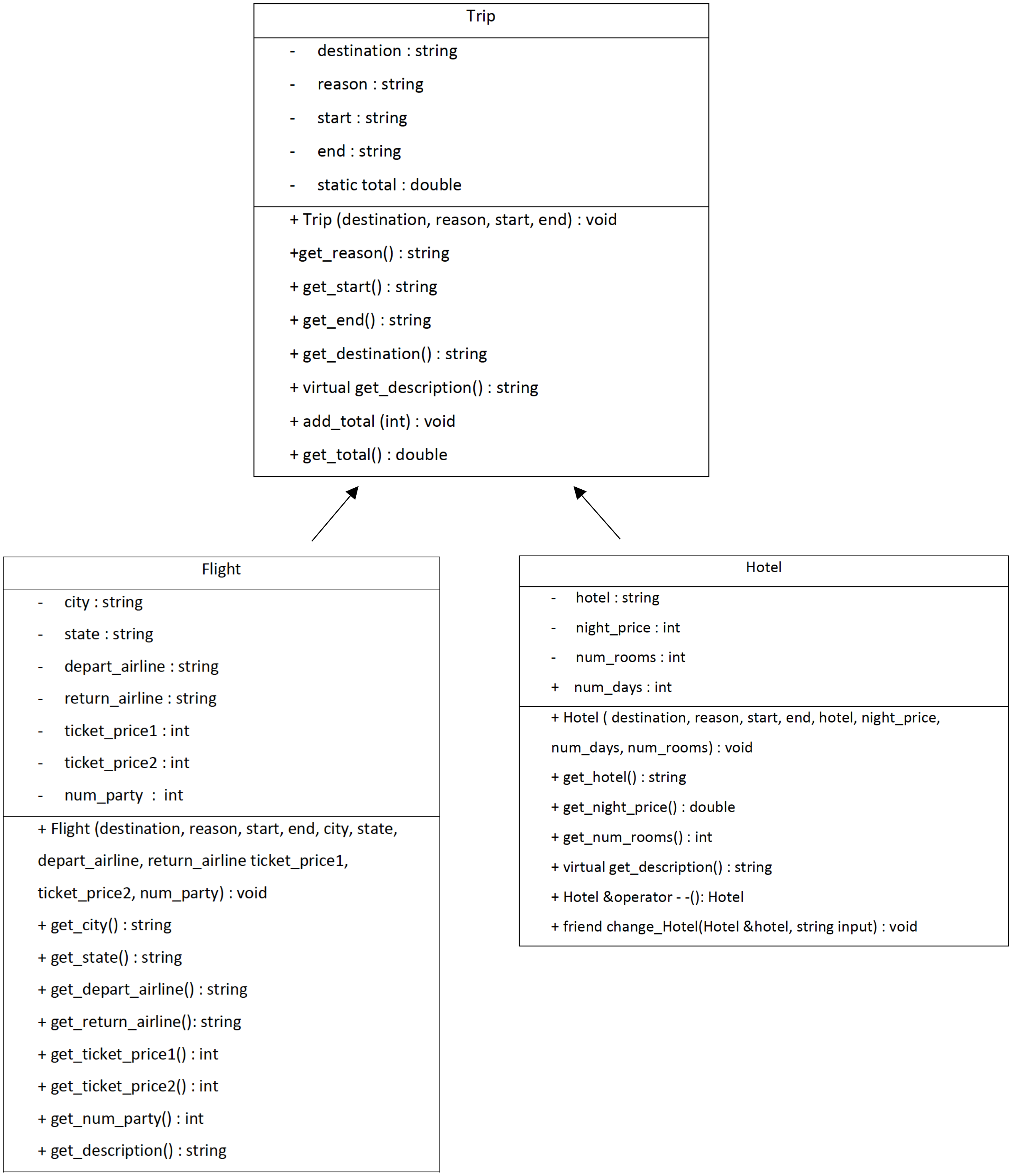
- **Trip** superclass will contain a destination, reason, start and end data members as well as a static member called total to keep tract of the total. The class will include a constructor to initialize the members and getter functions for get_start, get_end and get_description. The get_discription will be a pure virtual function that will be overridden by the subclasses. This class also includes a double function add_total to add prices to the total and double get_total to get the total cost.

- **Plane** (subclass of Trip) will contain a departing city, state, ticket_price1 and ticket_price2, num_party, depart_airline and return_airline data members to store the departure and return location, price per ticket of each flight, size of the party (equal to number of tickets) and airline names. It will include a constructor that calls the Trip constructor and that initializes the added members as well add the ticket prices times the number of passengers to the total by using the add_total function. This class will have getter functions get_city, get_state, get_depart_airline, get_return_airline, get_ticket_price1, get_ticket_price2, get_num_party, and get_description. The get_description will override the superclass virtual function in Trip.

- **Hotel** (subclass of Trip) will contain hotel, night_price, num_rooms, and num_days data members. It will include a constructor that calls the Trip constructor and that initializes the new members as well as add the night_price times number of nights times the number of rooms to the total by using the add_total function. The construct also checks if the number of nights is zero and does the necessary adjustments to reflect in the results that a hotel was not booked. This class will have getter functions get_hotel, get_night_price, get_num_rooms, get_num_days and get_description. The get_description will override the virtual function in Trip. This subclass includes a friend function change_Hotel that updates the name of a hotel. Hotel will also have an overloaded --function that will be used to subtract one from num_days in order to get number of nights in the hotel.

- **Main**(). The main will have a function that calculates the total number of days from the start and return date since the start and end date will have the format (MM/DD/YYYY). It will have a function that checks if the date entered is valid date. It will include a template that will check input validation. A function that rounds a number and returns the number a string. Function that determines which flight the user choose. A friend function that changes the name of Hotel object's private member. A function that uses polymorphism to determine which function is executed based on the object's type. It will include a function that writes the results into a string and prints the result to the console and into a txt file. The main will also have a serious of while loops that make sure the user enters valid data for each question. It will also contain a random seed to create the random prices for hotels and flights.

UML



Trip
- destination : string
- reason : string
- start : string
- end : string
- static total : double
+ Trip (destination, reason, start, end) : void
+get_reason() : string
+ get_start() : string
+ get_end() : string
+ get_destination() : string
+ virtual get_description() : string
+ add_total (int) : void
+ get_total() : double

Flight
- city : string
- state : string
- depart_airline : string
- return_airline : string
- ticket_price1 : int
- ticket_price2 : int
- num_party : int
+ Flight (destination, reason, start, end, city, state, depart_airline, return_airline ticket_price1, ticket_price2, num_party) : void
+ get_city() : string
+ get_state() : string
+ get_depart_airline() : string
+ get_return_airline(): string
+ get_ticket_price1() : int
+ get_ticket_price2() : int
+ get_num_party() : int
+ get_description() : string

Hotel
- hotel : string
- night_price : int
- num_rooms : int
+ num_days : int
+ Hotel (destination, reason, start, end, hotel, night_price, num_days, num_rooms) : void
+ get_hotel() : string
+ get_night_price() : double
+ get_num_rooms() : int
+ virtual get_description() : string
+ Hotel &operator - -(): Hotel
+ friend change_Hotel(Hotel &hotel, string input) : void