# Lab 2 – Classes continued

Object Oriented Programming using C++

Scenario: You are continuing to work for the software firm that is creating flight scheduling software. You are extending the task from lab 1 to accommodate passengers. A teammate of yours has created an expected interface for your code. You are asked to implement the class definitions to support those operations.

## Task 1. The Date and DateOfBirth class:

Create a class called Date to store a with fields day, month , and year. A class called DateOfBirth is a derived class of Date, which supports one additional operation – getAge(). Your code should support the following set of statements:

```
Date d1; //d1 initialized to January 1st, 1900
Date d2 (3, 15, 2024); //d2 initialized to March 15th, 2024
DateOfBirth d3 (5, 10, 1970); //d3 initialized to May 10th, 1970
DateOfBirth d4 (d1) //d4 initialized to January 1st, 1900, copying from d1
cout << d1;
cout << d3 << d4;
cout << d3.getAge(); //You can make today's date a global or static variable
```

Note that in the Date class, the month field can only support numbers in the range 1 to 12, the day field can only be numbers between 1-31. For the year field, you can assume that the negative numbers are B.C.E. In the DateOfBirth class, you will need to make sure that the year field can only support numbers above 1900 and below 2024. Be sure to include getters and setters.

### Grading information:

[LP] Support all lines of the interface without only basic input validation.

[HP] Support complete input validation (February has only 28 days on all years, except leap years; April, June, September, and November have only 30 days).

## Task 2. The Passenger class:

Create another class called Passenger with the fields first_name (string type), last_name (string type), dateOfBirth (DateOfBirth type), and fare_discount (double type). Your code should support the following set of statements (Hint: Remember the "has-a" relationship):

```
string first_name = "John";
string last_name = "Smith";
DateOfBirth d(5, 10, 1970);
Passenger p1(first_name, last_name, d);
cout << p1; //Outputs flight information in a suitable format
p1.setFareDiscount(0.25); //Sets the fare_discount meeting to 0.25, i.e., 25%
```

### Grading information:

[LP] Support lines 1 – 6.

## Task 3. The Ticket class:

First, update your Flight class to include fields called baseFare (double type) and dateOfTravel (Date type). Next, create a class called Ticket with the fields passenger (Passenger type), flight (Flight type), and passengerFare (double type), and. Your code should support the following set of statements continuing from Lab 1 (lines 1-4 in Task 1 and lines 1-4 from Task 2), lines 1-7 in Task 1 and lines 1-6 in Task 2 (Hint: Remember the "has-a" relationship):

```
1 Ticket ticket(p1, f1);
//In f1, update 235.85 as the base fare and d2 as dateOfTravel
2 cout << ticket << endl; //Outputs flight information in a suitable format
3 p1.setFareDiscount(0.5); //Update fare discount to 0.5, i.e., 50%
4 ticket.updateFare(); //Updates the fare based on passenger discount
5 cout << ticket.getFare(); //Outputs flight information in a suitable format
6 Date d5(3, 18, 2024)
7 Flight f2("SFO", "LAS", d5);
8 ticket.updateFlight(f2); //Updates flight information
```

### Grading information:

[LP] Support line 1.

[HP] Support all lines 1-8

## Task 4. Thoughts and Documentation:

Discuss the following questions for Task 1:

1. [LP] What constructors did you need to use? Is one constructor enough?
2. [HP] What is the difference between lines 3 and 4?
3. [HP] Explain how you are performing input validation for the Date and the DateOfBirth class.
4. [HP] What would be a good way to implement getAge without hard-coding today's date?

Discuss the following questions for Task 2:

1. [LP] What constructors did you need to use? Is one constructor enough?
2. [HP] The fare_discount field is used to indicate if the passenger receives any discount on the fare. This could be used for children or infants. How did you interpret the fare_discount field? Is it necessary to use this field in the class? If not, what alternative would you use?

Discuss the following questions for Task 3:

1. [LP] What constructors did you need to use? Is one constructor enough?
2. [HP] Consider the role of classes Time, Flight, Date, DateOfBirth, and Passenger in the Ticket class. Was it necessary to use these many classes and inheritances? Briefly, in a short paragraph, comment on the need for object-oriented design for a project like this.

Include your discussions in a file called Notes.

**VERY IMPORTANT:** Also include ALL the references you used for this lab in the Notes file. Failure to cite your sources counts as an act of academic dishonesty and will be taken seriously without zero tolerance.

## Specifications

All tasks have components labeled [LP] and [HP]. If you complete ALL the LP components satisfactorily, you will receive a grade of "low pass" on the lab. If you complete ALL the LP components and the following HP components satisfactorily, you will receive a grade of "high pass":

- Task 1 HP
- Task 3 HP
- Task 3 answer 3/5 discussion questions satisfactorily

If you do not meet the criteria for a "low pass", the submission will be marked as "revision needed".

### What to submit:

Your final submission will need to have the files as follows:

- `DateOfBirth.h (include the Date class declaration here)`
- `DateOfBirth.cpp (include the Date class definitions here)`
- `Passenger.h`
- `Passenger.cpp`
- `Ticket.h`
- `Ticket.cpp`
- `lab2-cmpe126.cpp`
- `Notes`

The statements provided in the tasks should be in `lab2-cmpe126.cpp`. Feel free to add more lines to test your implementation. Beautify the output as you like it.

**NOTE: You can look for help on the Internet but refrain from referencing too much. Please cite all your sources in your `Notes` file.**

### When to submit:

Submit your lab before **Thursday, February 15th, 11:59pm**. You are strongly advised to submit before Friday, February 9th, 11:59pm.

When you submit your assignment, you automatically agree to the following statement. If you do not agree, it is your responsibility to provide the reason.

"*I affirm that I have neither given nor received unauthorized help in completing this homework. I am not aware of others receiving such help. I have cited all the sources in the solution file.*"