**CIS 2275 C++Programming Part 2 NAME:**

**Program 6 – a class that has a class object as a class member**

**The C++ Express Train Tickets 100 points Due : Thursday, March 17, 2016**

**Turn In Requirements:**

1. **5 pts Name your Visual C++ 2015 project LastnameP6, such as NelsonP6.**
2. **5 pts Upload your project to Visual Studio Online.**
3. **5 pts If you upload to Blackboard, print out the \*.h and \*.cpp files, staple this page to the front of your printed source code when you turn it in for grading.**
4. **5 pts If you upload to Blackboard, remove BOTH debug folders and the .sdf file from your project before uploading it to Blackboard.**

**Program Requirements:**

1. **3 pts Write your name, email address and file name at the top of your source code in a comment.**
2. **5 pts Your main function should have cout statements that write “header” information to the screen. The header info includes your name, course and program information, as well as a 1-2 line description of the program.**
3. **5 pts. Use good C++ programming style and formatting for your program. Use appropriate comments to explain what you are doing.**

The C++ Express Train has a website that allows its customers to book seats on-line for exceptionally low fares. The train runs between Chicago and Albuquerque. For each train, there are 3 seat classifications and the One-way ticket prices and requirements are as follows:

BargainSeats $39 booked 30 days in advance

RegularSeat $79 booked 2 weeks in advance (14 days)

LastMinuteSeats $129 anything closer

We’re going to write a class that represents the reservation data for an individual or group travelling between Chicago and Albuquerque. Here is the CPPTrainTickets class declaration. The Date class here is the one from the beginning of the chapter. You may add more private members to the tickets class if you wish.

class CPPTrainTickets

{

private:

string reservation; //passenger name for this reservation

string errorString; //string containing error information

int numberOfPass; //number of passengers booked under this res

Date book, dep, ret; //pertinent date pbjects

bool bRoundTrip, bValidDate;

string DepartureCity, dateString;

float fare;

static int resNumber;  // number of the reservation made,declaration only

void CalculateFare();

void ValidateDate();

public:

CPPTrainTickets();

CPPTrainTickets(string resN, int nPass, Date &bk, Date &dpt, Date &rtn);

CPPTrainTickets(string resN, int nPass, Date &bk, Date &dpt);

void SetNumberOfPass(int n){numberOfPass = n;} //how many people?

void SetDepartingCity(string city){departureCity = city;}// ABQ or CHI

void SetDates(Date &bk, Date &dpt, Date &rtn);

void SetDates(Date &bk, Date &dpt);

void SetReservationName(string name){reservation = name;}

float GetTotalFare(){return fare;}

string GetReservationDescription();

};

Your job is to write the CPPTrainTickets class,and once it is written, test this class by using a Windows form application that declares one CPPTrainTickets object. Here is an example of the form:



When the RoundTrip check box is selected, a second set of nuds appears for the Date of Return.

Your Form should be titled using your name and the Program Name. You will have these controls on the form:

1. At the top of the form, present a menu of the fares and their limitations in a label
2. A textbox, for the Reservation Name
3. A numerical up/Down (nud) for the number in the party
4. Set of two radio buttons, to select the city of departure, either Albuquerque or Chicago
5. A checkbox for whether the trip is round-trip
6. Two sets of labels and nuds to allow the user to enter date of departure and date of return
7. Two buttons:
   1. One, titled “Make the Reservation. “ The event handler for this button contains the major part of the interface with the class,
   2. Two, titled “Make another Reservation? The event handler for this button contains a clearing function that resets all of the default values on the form and any that may be needed for the class.
8. A large, multi-line textbox that will house the reservation formatted string with all of the information for the user. It will display the string return from the class’s GetReservationDescription() method.

**On the form1.h, in the Make the Reservation Button\_Click Event Handler:**

Get the Reservation name, number in the party, and city of departure from the appropriate controls and set the values into the CPPTrainTickets object. Then get the date of departure from the nuds on the form. Do the same for the date of return if it is a round trip. You can get the booking date by simply making a default Date object, which will give you today’s date.

Finally, create an object by calling the one of the overloaded CPPTrainTickets constructors. Use the C++ 11 constructor initializer.

Then call GetReservationDescription which returns a string that contains the reservation name, number of passengers, the booking date, travel date(s), departure city, and total fare. Or if bValid is false, it will display the dateString with the error message. This string will be displayed in the multi-line textbox described above.

**The Date and CPPTrainTickets classes:**

You will also modify the Date class to add a ValidateThisDate function. This function will check that the day of the month does not exceed the number of days in that month. That is, you may not use the 31st of February. It will return true or false, depending whether the date is valid.

In your CPPTrainTickets class, the Overloaded Constructor and the SetDates methods set the Date object arguments into the class Date objects, then call ValidateDate( ). ValidateDate will check if the Date objects are valid, ie not the 34th of July, by calling the validateThisDate method of the Date class, and check that the Date of Departure is after the Date of Reservation and before the Date of Return (if it is a return ticket). We will allow the reservation to be made in the current year and in the next year only. It is permissible to book, leave and return on the same day. If the Dates are invalid, set a bool, bValidDate, to false and create a string stating that the program is unable to determine the fare due to ... whatever. The Set function will check bValidDate, and if true, call the private CalculateFare( ) function.

**Static Class Variables:** For the static class member variable ( static int resNumber;) , the variable definition must be made the class implementation file, otherwise known as HotelRes.cpp. At the same time, it can be initialized. This statement might look like:

int CPPTrainTickets::resNumber = 0;

This statement is the actual definition of the static variable and allocates the memory for it.

The purpose of this variable is to keep track of the number of reservation objects that are created.

The static variables is used to create the confirmation number. The confirmation numbers begin at 10001 and the reservation confirmation number is the 10000 number with the first 2 letters of the guest’s last name in upper case. For example, if the first guest were John Brown, the confirmation number would be 10001BR. The second guest Bob Smith, the confirmation number would be 10002SM. If the third customer were Nelson, the confirmation number would be: 10003NE.

In the Date class, we need to use the monName[12] array in more than one method. It is cumbersome to re-declare this array in each method. It can be declared static also, and accessed in any class method. Since it is fixed, it can also be declared const.

**Constructor Initializer:** We will also use a constructor initializer to initialize the date objects through the HotelRes overloaded constructor. First, get the user data, including the arrival and departure dates. Create Date objects to hold the date data. Finally, use one of the overloaded constructors to create the HotelRes object.