Object Oriented Programing Lab

BSCS(Fall 2015)

Lab # 14

Friday, June 3, 2016

Task # 1:
Consider the two classes declared here:

Contents of **Date.h**

```
1 // Specification file for the Date class
2 #ifndef DATE_H
3 #define DATE_H
4
5 class Date
6 {
7 protected:
8       int day;
9       int month;
10      int year;
11 public:
12 // Default constructor
13      Date(int d, int m, int y)
14      { day = 1; month = 1; year = 1900; }
15
16 // Constructor
17      Date(int d, int m, int y)
18      { day = d; month = m; year = y; }
19
20 // Accessors
21      int getDay() const
22      { return day; }
23
24      int getMonth() const
25      { return month; }
26
27      int getYear() const
28      { return year; }
29 };
30 #endif
```

Contents of **Time.h**

```
1 // Specification file for the Time class
2 #ifndef TIME_H
3 #define TIME_H
4
5 class Time
6 {
7 protected:
8       int hour;
9       int min;
10      int sec;
11 public:
12 // Default constructor
13      Time()
14      { hour = 0; min = 0; sec = 0; }
15
16 // Constructor
17      Time(int h, int m, int s)
18      { hour = h; min = m; sec = s; }
19
20 // Accessor functions
21      int getHour() const
22      { return hour; }
23
24      int getMin() const
25      { return min; }
26
27      int getSec() const
28      { return sec; }
29 };
30 #endif
```

Both classes should be used as base classes for a third class we will call DateTime. The class has two constructors: a default constructor and a constructor that accepts arguments for each component of a date and time. The class has a **showDateTime()** function which gives output as following:

4/2/1960 5:32:27

Task # 2: Time Format
In **Task # 1,** we have **Time.h** contains a Time class. Design a class called **MilTime** that is derived from the Time class. The **MilTime** class should convert time in military (24-hour) format to the standard time format used by the Time class. The class should have the following member variables:

milHours: Contains the hour in 24-hour format. For example, 1:00 pm would be stored as 1300 hours, and 4:30 pm would be stored as 1630 hours.
milSeconds: Contains the seconds in standard format.

The class should have the following member functions:

Constructor: The constructor should accept arguments for the hour and seconds, in military format. The time should then be converted to standard time and stored in the hours, min, and sec variables of the Time class.

setTime: Accepts arguments to be stored in the milHours and milSeconds variables. The time should then be converted to standard time and stored in the hours, min, and sec variables of the Time class.

getHour: Returns the hour in military format.

getStandHr: Returns the hour in standard format.

Demonstrate the class in a program that asks the user to enter the time in military format. The program should then display the time in both military and standard format.

Task # 3: Time Format Exceptions

Modify the **MilTime** class you created in **Task # 2**. The class should implement the following exceptions:
BadHour Throw when an invalid hour (< 0 or > 2359) is passed to the class and also ask user for valid value.
BadSeconds Throw when an invalid number of seconds (< 0 or > 59) is passed to the class and also ask user for valid value.
Demonstrate the class in a driver program.

Task # 4: Date Exceptions

Modify the **Date** class in **Task # 1**. The class should implement the following exception classes:
InvalidDay Throw when an invalid day (< 1 or > 31) is passed to the class and also ask user for valid value.
InvalidMonth Throw when an invalid month (< 1 or > 12) is passed to the class and also ask user for valid value.
Demonstrate the class in a driver program.