**Instructions:**

1. Create a MS VisualStudio solution named **Assignment15**.
2. Create projects according to the assigned problems.
3. Use techniques and experience that you have obtained from previous chapters and this chapter.
4. Code and test your programs. They **MUST** be completed without syntax, logic and run-time errors.
5. Include comments of your name, date, and brief descriptions in all source codes.
6. Compress **Assignment15**folder into **ONE** zipped file.
7. Submit or re-submit your zipped file before its due date&time.

int main()  
{  
   do  
   {  
       switch (menuOption())  
       {  
       case 0: exit(1); break;  
       case 3: Challenge3(); break;  
       case 4: Challenge4(); break;  
       case 6: Challenge6(); break;  
       case 7: Challenge7(); break;  
       case 8: Challenge8(); break;  
       case 12: Challenge12(); break; // 10pts extra  
       default: cout << "\t\tERROR - Invalid option. Please re-enter."; break;  
       }  
       cout << "\n";  
       system("pause");  
  
    } while (true);  
  
   return EXIT\_SUCCESS;  
}

3. TeamLeader Class

In a particular factory, a team leader is an hourly paid production worker who leads a small team. In addition to hourly pay, team leaders earn a fixed monthly bonus. Team leaders are required to attend a minimum number of hours of training per year. Design a TeamLeader class that extends the ProductionWorker class you designed in Programming Challenge 1 (Employee and Production Worker Classes). The TeamLeader class should have member variables for the monthly bonus amount, the required number of training hours, and the number of training hours that the team leader has attended. Write one or more constructors and the appropriate accessor and mutator functions for the class. Demonstrate the class by writing a program that uses a TeamLeader object.

4. Time Format

In Program 15-20, the file 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 p.m. would be stored as 1300 hours, and 4:30 p.m. 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.

Input Validation: The MilTime class should not accept hours greater than 2359, or less than 0. It should not accept seconds greater than 59 or less than 0.

6. Essay Class

Design an Essay class that is derived from the GradedActivity class presented in this chapter. The Essay class should determine the grade a student receives on an essay. The student’s essay score can be up to 100, and is determined in the following manner:

Grammar: 30 points

Spelling: 20 points

Correct length: 20 points

Content: 30 points

Demonstrate the class in a simple program.

7. PersonData and CustomerData Classes

Design a class named PersonData with the following member variables:

lastName

firstName

address

city

state

zip

phone

Write the appropriate accessor and mutator functions for these member variables.

Next, design a class named CustomerData, which is derived from the PersonData class. The CustomerData class should have the following member variables:

customerNumber

mailingList

The customerNumber variable will be used to hold a unique integer for each customer. The mailingList variable should be a bool. It will be set to true if the customer wishes to be on a mailing list, or false if the customer does not wish to be on a mailing list. Write appropriate accessor and mutator functions for these member variables. Demonstrate an object of the CustomerData class in a simple program.

1. PreferredCustomer Class

A retail store has a preferred customer plan where customers may earn discounts on all their purchases. The amount of a customer’s discount is determined by the amount of the customer’s cumulative purchases in the store.

When a preferred customer spends $500, he or she gets a 5 percent discount on all future purchases.

When a preferred customer spends $1,000, he or she gets a 6 percent discount on all future purchases.

When a preferred customer spends $1,500, he or she gets a 7 percent discount on all future purchases.

When a preferred customer spends $2,000 or more, he or she gets a 10 percent discount on all future purchases.

Design a class named PreferredCustomer, which is derived from the CustomerData class you created in Programming Challenge 7. The PreferredCustomer class should have the following member variables:

purchasesAmount (a double)

discountLevel (a double)

The purchasesAmount variable holds the total of a customer’s purchases to date. The discountLevel variable should be set to the correct discount percentage, according to the store’s preferred customer plan. Write appropriate member functions for this class and demonstrate it in a simple program.

Input Validation: Do not accept negative values for any sales figures.

12. Ship, CruiseShip, and CargoShip Classes

Design a Ship class that has the following members:

A member variable for the name of the ship (a string)

A member variable for the year that the ship was built (a string)

A constructor and appropriate accessors and mutators

A virtual print function that displays the ship’s name and the year it was built.

Design a CruiseShip class that is derived from the Ship class. The CruiseShip class should have the following members:

A member variable for the maximum number of passengers (an int)

A constructor and appropriate accessors and mutators

A print function that overrides the print function in the base class. The CruiseShip class’s print function should display only the ship’s name and the maximum number of passengers.

Design a CargoShip class that is derived from the Ship class. The CargoShip class should have the following members:

A member variable for the cargo capacity in tonnage (an int)

A constructor and appropriate accessors and mutators

A print function that overrides the print function in the base class. The CargoShip class’s print function should display only the ship’s name and the ship’s cargo capacity.

Demonstrate the classes in a program that has an array of Ship pointers. The array elements should be initialized with the addresses of dynamically allocated Ship, CruiseShip, and CargoShip objects. (See Program 15-14, lines 17 through 22, for an example of how to do this.) The program should then step through the array, calling each object’s print function.