MÔN HỌC LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG
BÀI THỰC HÀNH
CHƯƠNG 4: ĐA HÌNH

Bài 1:

Tạo một lớp mới **ColMbr** là lớp cơ sở cho các lớp dẫn xuất Student và Alumni. Lớp ColMbr có mô tả như sau:

- Có các thành viên dữ liệu private là **const unsigned** idNbr (ID number) và **string** name. Hai thành viên dữ liệu này được yêu cầu khi tạo đối tượng thuộc lớp ColMbr.
- Các hàm thành viên gồm:
    - Một constructor
    - Một phương thức **setName** để gán một chuỗi cho name
    - Một hàm ảo **display** để hiển thị các thành viên dữ liệu private
    - Một hàm thuần ảo **addClass**

Lớp Student được dẫn xuất từ lớp **ColMbr** có các thành viên dữ liệu private là **unsigned** credHrs (số giờ tính chỉ tích lũy được), **unsigned** qualPts (số điểm tích lũy được) và **string** degSought (thứ hạng đạt được) bao gồm các giá trị: "AA", "AS", "AAS", hoặc "Unspecified". Các phương thức public là:

- Một constructor với các tham số mặc định credHrs=0, qualPts=0, degSought="Unspecified".
- Một phương thức setDegree để đặt degSought là một trong các giá trị hợp lệ hoặc "Unspecified" trong trường hợp ngược lại
- Phương thức **addClass** nhận 2 tham số là số nguyên (**int**) và cộng chúng với credHrs và qualPts tương ứng
- Phương thức getGPA trả về một giá trị double biểu diễn điểm trung bình của của một student bằng cách lấy qualPts chia cho credHrs. Nếu credHrs=0 thì trả về 0
- Một phương thức **display** để hiển thị các thông tin về một student như: idNbr, name, GPA, and degree sought

Lớp Alumni được dẫn xuất từ lớp **ColMbr** có các thành viên dữ liệu private là **Date** dateGrad (ngày tốt nghiệp) và **string** degree (bằng cấp có được). Các phương thức public là:

- Một constructor yêu cầu các tham số: idNbr, name, 03 số nguyên biểu diễn tháng, ngày, năm tốt nghiệp và degree của một Alumni
- Phương thức **display** để hiển thị các thông tin về một alumni như: idNbr, name, date graduated và degree
- Phương thức **addClass** không làm gì cả (nhưng cần phải có để cho Alumni là một lớp cụ thể)

Đây là tập tin chương trình để test các lớp trên: colMbrTest.cpp có nội dung như sau:

```
#include <iostream>
#include <cstdlib>
#include "colMbrs.h"


int main()  //-------------- Driver ---------------//
{
  ColMbr *cMbr[4];     // array of base-class pointers
  int i;               // index to array

  // Create some college members (maybe read from a file?)

  cMbr[0] = new Student( 12345, "Steven DiFranco", 15, 33, "AA" );
  cMbr[1] = new Alumni( 98765, "Don Green", 12, 15, 1978, "AAS" );
  cMbr[2] = new Alumni( 24680, "Henry Thoreau", 5, 22, 1846, "AA" );
  cMbr[3] = new Student( 13579, "Millenia Best" );

  // display the array

  cout << "All college members:\n";
  for( i = 0; i < 4; ++i )
  {
    cMbr[i]->display();        // no need to check type field or cast
    cout << endl;
  }

  // test addClass for student
```

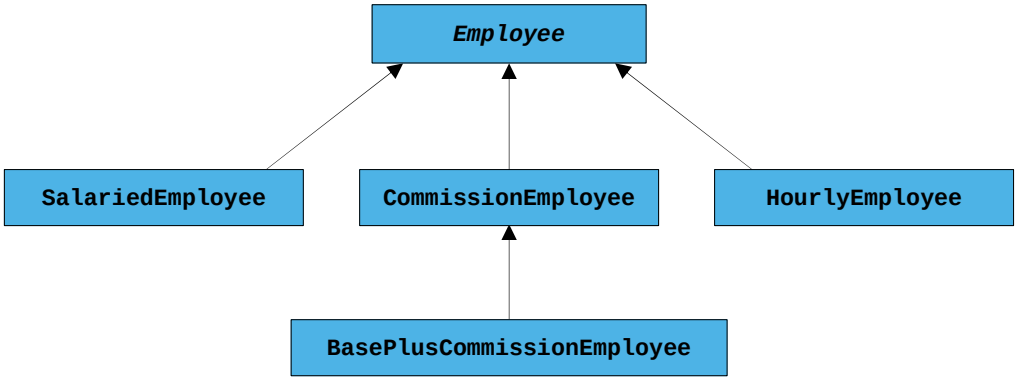
```
   cMbr[3]->addClass( 3, 12 );
   cMbr[3]->display();

   cout << endl;
   system("PAUSE");
   return 0;
}
```

Có thể lấy giao tiếp (date.h) và cài đặt (date.cpp) của lớp Date ở bài thực hành chương 1 để sử dụng.

Chú ý: Tất cả các giao tiếp và cài đặt của các lớp của các bạn đều nằm trong một tập tin duy nhất là **colMbrs.h**.

Bài 2: Hệ thống tính lương cho một công ty

Cho biểu đồ lớp sau:



Công ty có 04 loại nhân viên được trả lương hàng tuần. Trong đó:

Nhân viên lãnh lương cố định không tính đến giờ làm việc: SalariedEmployee.

Nhân viên lãnh lương theo giờ: Nếu số giờ làm việc trên 40 thì tính thêm gấp 1.5 so với giờ bình thường (≤40): (HourlyEmployee).

Nhân viên viên huê hồng: lãnh lương theo tỉ lệ bán được: (CommissionEmployee).

Nhân viên huê hồng có lương cơ bản: Lương được lãnh bằng lương cơ bản + tỉ lệ bán được: (BasePlusCommissionEmpoyee).

Lớp nhân **Employee** gồm:

- Các dữ liệu thành viên private là: **string firstName, string lastName** và **string socialSecurityNumber** (số bảo hiểm xã hội)
- Một constructor nhận 03 tham số là 03 chuỗi biểu diễn firstName, lastName và socialSecurityNumber tương ứng
- Các phương thức get và set để lấy/đặt các thành viên dữ liệu private
- Hàm thành viên thuần ảo **double earnings()** để cho lớp Employee là lớp cơ sở trừu tượng
- Phương thức ảo **void print() const** để thực hiện đa hình

Lớp **SalariedEmployee** gồm:

- Một constructor với tham số mặc định **weeklySalary=0**
- Thành viên dữ liệu private **weeklySalary** (lương tuần)
- Các phương thức set/get để đặt/lấy **weeklySalary**

Lớp **HourlyEmployee** gồm:

- Các thành viên dữ liệu private là: **double wage** (lương trên/1giờ) và **double hours** (số giờ làm việc trong tuần)
- Một constructor với 02 tham số mặc định là **wage=0** và **hours=0**
- Các phương thức set/get để đặt/lấy **wage** và **hours**

Lớp **CommissionEmployee** gồm:

- Các thành viên dữ liệu private là: **double grossSales** (tổng doanh số bán) và **double commissionRate** (tỉ lệ huê hồng)
- Một constructor với 02 tham số mặc định là **grossSales=0** và **commissionRate=0**
- Các phương thức set/get để đặt/lấy **grossSales** và **commissionRate**

Lớp **BasePlusCommissionEmployee** gồm:

- Thành viên dữ liệu private là **baseSalary** (lương cơ bản/tuần)
- Một constructor với tham số mặc định là **baseSalary=0**
- Các phương thức set/get để đặt/lấy **baseSalary**

Hãy viết giao tiếp và cài đặt cho các lớp trên.

Các tập tin chứa giao tiếp (.h) gồm:

employee.h

salaried.h

hourly.h

commission.h

baseplus.h

Các tập tin chứa cài đặt (.cpp) gồm:

employee.cpp

salaried.cpp

hourly.cpp

commission.cpp

baseplus.cpp

Tập tin chương trình để test các lớp trên được cho như sau:

```
// Driver for Employee hierarchy.
#include <iostream>

using std::cout;
using std::endl;
using std::fixed;

#include <iomanip>

using std::setprecision;

#include <vector>

using std::vector;

#include <typeinfo>
```

```
#include "employee.h"    // Employee base class
#include "salaried.h"    // SalariedEmployee class
#include "commission.h"  // CommissionEmployee class
#include "baseplus.h"    // BasePlusCommissionEmployee class
#include "hourly.h"      // HourlyEmployee class

int main()
{
  // set floating-point output formatting
  cout << fixed << setprecision( 2 );

  // create vector employees
  vector < Employee * > employees( 4 );

  // initialize vector with Employees
  employees[ 0 ] = new SalariedEmployee( "John", "Smith",
    "111-11-1111", 800.00 );
  employees[ 1 ] = new CommissionEmployee( "Sue", "Jones",
    "222-22-2222", 10000, .06 );
  employees[ 2 ] = new BasePlusCommissionEmployee( "Bob",
    "Lewis", "333-33-3333", 300, 5000, .04 );
  employees[ 3 ] = new HourlyEmployee( "Karen", "Price",
    "444-44-4444", 16.75, 40 );

  // generically process each element in vector employees
  for ( int i = 0; i < employees.size(); i++ ) {

    // output employee information
    employees[ i ]->print();
```

```
   // downcast pointer
   BasePlusCommissionEmployee *commissionPtr =
     dynamic_cast < BasePlusCommissionEmployee * >
       ( employees[ i ] );

   // determine whether element points to base-salaried
   // commission worker
   if ( commissionPtr != 0 ) {
     cout << "old base salary: $"
          << commissionPtr->getBaseSalary() << endl;
     commissionPtr->setBaseSalary(
       1.10 * commissionPtr->getBaseSalary() );
     cout << "new base salary with 10% increase is: $"
          << commissionPtr->getBaseSalary() << endl;

   } // end if

   cout << "earned $" << employees[ i ]->earnings() << endl;

} // end for

// release memory held by vector employees
for ( int j = 0; j < employees.size(); j++ ) {

   // output class name
   cout << "\ndeleting object of "
        << typeid( *employees[ j ] ).name();

   delete employees[ j ];
```

```
    } // end for

    cout << endl;

    return 0;

} // end main
```