UEE1303(1070) S12: Object-Oriented Programming

Inheritance (II)

What you will learn from Lab 10

In this laboratory, you will learn how to use multiple inheritance.

TASK 10-1 ACCESS TO BASE CLASSES

✓ In the following example, B is a public base for X. Please fix the compiler error here.

```
//lab10-1.cpp
#include <iostream>

class B
{
private:
    int i;
protected:
    float f;
public:
    B() { i =0; f = 0.0; d =0.0; }
    double d;
    void g1(B b){f = b.f;}
};

class X: public B
{
protected:
    short s;
public:
    X() {s=0;}
    void g2(X x) {f = x.f;}
    void g3(B b) {f = b.f;} //comment
};                                  [Error] 'float B::f' is protected

int main()
{
    B b1;
    X x1;
    x1.g1(b1);  [Error] 'void B::g1(B)' is inaccessible
                [Error] within this context
                [Error] 'B' is not an accessible base of 'X'
    return 0;
}
```

➢ Please modify B as a protected base and compiler the program again.

➢ Here provide guidelines for access control:

- ✧ If B is a ***private*** base, its public and protected members become private members of derived class.
- ✧ If B is a ***protected*** base, its public and protected members become protected members of derived class.
- ✧ If B is a ***public*** base, its public members become members of derived class and its protected members become protected members of derived class.

➢ The access control for protected member, similar to private member, is that only its member and friend can access it. However, the protected member can become private, protected or public members of derived class but private member cannot. Therefore, protected members of a class are designed for use by derived classes and are not intended for general use.

TASK 10-2 MULTIPLE INHERITANCE

✓ A class can be directly derived from two or more base classes. This is called multiple inheritance. The class `Circle_in_Triangle` is derived from classes `Circle` and `Triangle`.

```
// lab10-2
#include <iostream>
using std::cout;  using std::endl;

class Point2D
{
private:
   int x;
   int y;
public:
   Point2D(){x = 0;y=0;}
   void display() const;
   // …
};

class Circle
{
private:
   Point2D center;
   double radius;
public:
   void draw();
   //…
};


class Triangle
{
```

```
private:
   Point2D *vertices;
public:
   // …
   ~Triangle(){delete [] vertices;}
   void draw();
};


class Circle_in_Triangle: public Circle, public Triangle
{
public:
   // …
   void draw()
   {
      Circle::draw();
      Triangle::draw();
   }
};

int main()
{
   Point2D p;
   Point2D *vec = new Point2D [3];

   Circle_in_Triangle ct(p,0,vec);
   ct.draw();

   return 0;
}
```

TASK 10-3 AMBIGUITY RESOLUTION

✓ When two base classes have members with the same name, they can be resolved by using the scope resolution operator.

```
// lab10-3.cpp

/*  add area() for class Circle   */
/*  add area() for class Triangle */

int main()
{
   Point2D p;
   Point2D *vec = new Point2D [3];

   Circle_in_Triangle ct(p,0,vec);
   ct.draw();

   cout << "Area of Circle: " << ct.Circle::area() << endl;
```

```
    cout << "Area of Triangle: " << ct.Triangle::area() << endl;
    cout << "Area of Circle_in_Triangle: " << ct.area() << endl;
    return 0;
}
```

- The compiler shows the error message "`request for member `area' is ambiguous`" on screen.
- A `using`-declaration can bring different functions from base classes to a derived class and then overload resolution can be applied. You can add "`using Triangle::area;`" in `Circle_in_Triangle` and compiler the program again.

```
class Circle_in_Triangle: public Circle, public Triangle {
  public:
    using Triangle::area;
...
  }
};
```

TASK 10-4 REPLICATED BASE CLASSES

✓ With the possibility of derivation from two bases, a class can be a base twice.

```
// lab10-4.cpp

class Shape
{
protected:
    int color;
};
class Circle: public Shape
{
    // definition in lab10-3
}
class Triangle: public Shape
{
    // definition in lab10-3
}
class Circle_in_Triangle: public Circle, public Triangle
{
public:
    // …
    void draw()
    {
        cout << "Circle's color: " << Circle::color << endl;
        cout << "Triangle's color: " << Triangle::color << endl;
        Circle::draw();
        Triangle::draw();
    }
```

```
};
```

- In this example, the colors of a `Circle` and a `Triangle` for an object of `Circle_in_Triangle` can be different.

TASK 10-5 VIRTUAL BASE CLASSES

- ✓ Often a base class need not be replicated. That is, only one copy of a replicated class need be inherited for a derived class object.

```
// lab10-5.cpp

class Shape
{
protected:
    int color;
};
class Circle: public virtual Shape
{
    // definition in lab10-3
}
class Triangle: public virtual Shape
{
    // definition in lab10-3
}
class Circle_in_Triangle: public Circle, public Triangle
{
public:
    // …
    void draw()
    {
        cout << "Circle's color: " << Circle::color << endl;
        cout << "Triangle's color: " << Triangle::color << endl;
        Circle::draw();
        Triangle::draw();
    }
};
```

- In this example, the colors of a `Circle` and a `Triangle` for an object of `Circle_in_Triangle` are the same since they are inherited for the same base.

TASK 10-6 EXERCISE

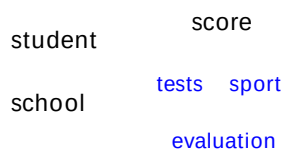
1. *CIRCLE AND TRIANGLE

✓ Please finish the program `lab10-3` and `lab10-5` show the area and color information on screen. Note that you need to define the center and radius of circle, and the vertices of triangle on the main function. Moreover, the area of `Circle_in_Triangle` is defined as *(area in Triangle – area of Circle)*

```
Circle's color: 255
Triangle's color: 255
Center: 2,2
Radius: 1
Vertices:
2,1
8,1
5,6
Area of Circle: 3.14
Area of Triangle: 15
Area of Circuit_in_Triangle: 11.86
```

2. EVALUATION SYSTEM FOR STUDENTS

✓ Please develop an evaluation system for `students`' performance. There are two different ways to evaluate the student's performance: `tests` and `sport`. You need to write a base class `score`, and two derived classes named `test` and `sport` which are inherited from `score`. Moreover, a derived class called `evaluation` is used to conclude the performance of the student and multiply inherited from `test` and `sport`. You may also need a class called `student` which contains three members, id, name, and final score (evaluation class), and a class called `school` to store all results.



✓ The command-line usage of the evaluation system is

```
>./ex10-2 performace.txt result.txt
```

✓ In the input file "`performance.txt`", the first line shows the total number of students in the school. Each row indicates a record for one student. As shown in the sample file, the first and second columns denote the student's ID and name, respectively. The later five numbers are the final scores for different subjects and the last five `1/0`s indicate win/loss on different sports games. If a student is a winner in one sport game, he/she can obtain extra five points on his/her evaluation report. For example, Tom's average score on five subjects is `70` and he won three games to obtain extra `15` points. Therefore, Tom's final score is `85`. The full score is `100`.

```
5
991001 Tom 50 60 80 90 70 1 0 0 1 1
991002 Jean 100 90 80 70 60 0 0 0 1 1
991003 Kevin 60 90 100 80 90 0 1 1 0 0
991004 John 100 80 90 70 80 0 0 0 0 1
991005 Marry 50 60 70 90 60 1 1 0 0 0
```

✓ The context in ‘result.txt’ is

```
991001 Tom 85
991002 Jean 90
991003 Kevin 94
991004 John 89
991005 Marry 76
Average: 86.8
```

✓ The main function of the program is shown as follows,

```
int main(int argc, char *argv[])
{
   school nctu(argv[1]);
   nctu.report(argv[2]);

   return 0;
}
```