UEE1303 S18: Object-Oriented Programming

LAB #10: Inheritance (I)

What you will learn from Lab 10

In this laboratory session you will: learn the concept of inheritance and its usage.

LAB10-1: Example of Inheritance

✓ Please compiler and execute the program lab10-1-1, where Point4D is a derived class from the base class Point2D.

```
// lab10-1-1.cpp
#include <iostream>
using std::cout; using std::endl;
class Point2D
{
    public:
        Point2D(int n1 = 0, int n2 = 0):x(n1), y(n2){}
        void display() const;
    private:
        int x;
        int y;
};
void Point2D::display() const
{
    cout << x << "," << y;
}
class Point4D : public Point2D
{
    public:
    Point4D(int n1 = 0,int n2 = 0,int n3 = 0,int n4 = 0):Point2D(n1,n2),z(n3), t(n4){}
    void display() const;
    private:
    int z;
    int t;
};
void Point4D::display() const
{
    Point2D::display();
    cout << "," << z << "," << t << endl;
}
int main()
{
    Point4D pt(1,2,3,4);
    pt.display();
    return 0;
}
```

- Note that Point4D has member of class Point2D in addition to its own members.
- You can put the constructor of the base class in the initialization list for the derived class.
- Note that member function of a derived class cannot access the private part of a base

class. For example, the function Point4D::display() cannot be defined as

– void Point4D::display() const

```
{
      cout << x << "," << y; // x and y are inaccessible
      cout << "," << z << "," << t;
}
```

– The hidden member x and y of the derived class Point4D is accessible through the public member function Point2D::display();.
– You can define accessor and mutator functions in Point2D to access private members.

✓ Please compiler and execute the program lab10-1-2

```
// lab10-1-2.cpp
/* The Point2D and Point4D class defined in lab10-1-1 */
int main()
{
      Point2D pt2(3,4);
      Point4D pt4(1,2,3,4);
      pt4.display(); cout << endl;
      pt2 = pt4; // OK, every Point2D is a Point4D
      pt2.display(); cout << endl;
      pt4 = pt2; // Error, not every Ponint4D is a Point2D
      pt4.display(); cout << endl;
      return 0;
}
```

– You can comment the incorrect lines to observe the results.
– If you require type conversion from a base class to derived class (eg. pt4 = pt2), you have to provide additional member functions of Point4D to achieve it.

✓ Please compiler and execute the program `lab10-1-3`

```
// lab10-1-3.cpp
/* The Point2D and Point4D class defined in lab9-1-1 */
void f(const Point2D &p1, const Point2D &p2)
{
      p1.display(); cout << endl;
      p2.display(); cout << endl;
}
int main()
{
      Point2D pt2(3,4);
      Point4D pt4(1,2,3,4);
      f(pt2,pt4);
      return 0;
}
```

- Note that the prototype of function f is void f(const Point2D &, const Point2D &).

Lab10-2: Class Hierarchy

- ✓ A derived class can be a base class of another derived class.

```
// lab10-1-3.cpp
/* The Point2D and Point4D class defined in lab9-1-1 */
void f(const Point2D &p1, const Point2D &p2)
{
      p1.display(); cout << endl;
      p2.display(); cout << endl;
}
int main()
{
      Point2D pt2(3,4);
      Point4D pt4(1,2,3,4);
      f(pt2,pt4);
      return 0;
}
```

  - Note that, to enable copy constructor of Car, you should also provide copy constructor for Point2D and Point4D.

Exercise 10-1:

- ✓ Please modify the class Point2D and Point4D defined in lab10-1. In Point2D, the member x and y become two pointers to integer, respectively. Similarly, the member z and t should be changed as pointers. The modified classes are shown as follows,

```
// Point2D.h
#ifndef POINT2D_H
#define POINT2D_H
class Point2D
{
public:

private:
int *x;
int *y;
};
#endif
```

- ✓ Please implement `Point2D` and `Point4D` in different files.

```
// Point4D.h
#ifndef POINT4D_H
#define POINT4D_H
class Point4D
{
public:

private:
int *z;
int *t;
};
#endif
```

- ✓ Please finish the remaining part to make the following main function work successfully

```
#include <iostream>
#include "Point2D.h"
#include "Point4D.h"
using std::cout; using std::endl;
int main()
{
	Point2D pt1(1,2);
	Point2D pt2(3,4);
	pt1.display(); cout << endl;
	pt2.display(); cout << endl;
	pt2 = pt1;
	pt2.display(); cout << endl;
	Point4D pt4(5,6,7,8);
	pt4.display(); cout << endl;
	pt2 = pt4;
	pt2.display(); cout << endl;
	pt4 = pt1;
//pt4 could be (1,2,7,8) or (1,2,0,0)
	pt4.display(); cout << endl;
return 0;
}
```