UEE 1303(1068): Object-Oriented Programming

Lab #4: Basic Class

In this laboratory session you will:

- learn how to write a basic class

Lab 4-1: Basic Class

✓ We rewrite the structure `Point2D`, defined in program `lab3-1-3`, as a class object.

```
// lab4-1-1.cpp
#include <iostream>
using namespace std;
class Point2D
{
   void assignPoint2D(int n1, int n2, double v);
   void displayPoint2D();
   int x;
   int y;
   double value;
};
void Point2D::assignPoint2D(int n1, int n2, double v)
{
   x = n1;
   y = n2;
   value = v;
}
void Point2D::displayPoint2D()
{
   cout << "(" << x << "," << y << ") = ";
   cout << value << endl;
}
int main()
{
   Point2D ptArray[10];
   for (int i = 0; i < 10; i++)
```

```
    {
        ptArray[i].assignPoint2D(i, i+2, i*10);
        ptArray[i].displayPoint2D();
    }
    return 0;
}
```

- Please fix the compiler error in this example.
- If you do not specific the member access modifiers, the compiler will take as private member.

✓ We rewrite the above program and modify the class `Point2D` with member access modifiers.

```
// lab4-1-2.cpp
#include <iostream>
using namespace std;
class Point2D
{
public:
    void assignPoint2D(int n1, int n2, double v);
    void displayPoint2D();
private:
    int x;
    int y;
    double value;
};
void Point2D::assignPoint2D(int n1, int n2, double v)
{
    x = n1;
    y = n2;
    value = v;
}
void Point2D::displayPoint2D()
{
    cout << "(" << x << "," << y << ") = ";
    cout << value << endl;
}
int main()
```

```
{
    Point2D ptArray[10];
    for (int i = 0; i < 10; i++)
    {
        ptArray[i].assignPoint2D(i, i+2, i*10);
        ptArray[i].displayPoint2D();
    }
    return 0;
}
```

- The program demonstrates that a pointer is used to point the structure object.

✓ To access private data members, we can add get (accessor) and set (mutator) functions as public member functions.

```
// lab4-2-3.cpp
…
class Point2D
{
public:
    void setCoord(int n1, int n2);
    void setValue(double v);
    int getCoordX();
    int getCoordY();
    double getValue();
    void assignPoint2D(int n1, int n2, double v);
    void displayPoint2D();
private:
    int x;
    int y;
    double value;
};
// Please implement the definitions of five additional
member functions.
int main()
{
    Point2D a;
    a.setCoord(1,3);
    cout << "a(x,y) = " << a.getCoordX() << " "
                        << a.getCoordY() << endl;
    Point2D *b = new Point2D;
    b->setValue(5);
    cout << "value of b is " << b->getValue() << endl;

    return 0;
}
```

− Please fix the compiler error.

Exercise 4-1 (COMPLEX NUMBER)

✓ Create a `Complex` class to perform complex number arithmetic and write a program to test your class. The class provides four complex operations: addition, subtraction, multiplication and division. The sample output is shown as follows

```
(1.0, 7.0) + (9.0, 2.0) = (10.0, 9.0)
(1.0, 7.0) - (9.0, 2.0) = (-8.0, 5.0)
(1.0, 7.0) * (9.0, 2.0) = (-5.0, 65.0)
(1.0, 7.0) / (9.0, 2.0) = (0.3, 0.7)
(10.0, 7.0) - (9.0, -1.0) = (1.0, 8.0)
```

✓ The main structure of the program is like as

```
// Complex.h
#ifndef COMPLEX_H
#define COMPLEX_H

// Write class definition for Complex

#endif
```

```
// Complex.cpp
#include <iostream>
#include "Complex.h"
using namespace std;
// Member-function definitions for class Complex.
```

```
// ex4-1.cpp
#include <iostream>
```

```
#include "Complex.h"
using namespace std;

int main()
{
   Complex a, b, c; // create three Complex objects

   a.assign(1.0,7.0);
   b.assign(9.0,2.0);
   a.printComplex(); // output object a
   cout << " + ";
   b.printComplex(); // output object b
   cout << " = ";
   // invoke add function and assign to object c
   c = a.add(b);
   c.printComplex(); // output object c
   cout << endl;

   a.printComplex(); // output object a
   cout << " - ";
   b.printComplex(); // output object b
   cout << " = ";
   c = a.subtract(b); // invoke subtract function
   c.printComplex(); // output object c
   cout << endl;

   a.printComplex(); // output object a
   cout << " * ";
   b.printComplex(); // output object b
   cout << " = ";
   c = a.multiply(b); // invoke subtract function
   c.printComplex(); // output object c
   cout << endl;

   a.printComplex(); // output object a
   cout << " / ";
   b.printComplex(); // output object b
   cout << " = ";
```

```
    c = a.division(b); // invoke subtract function
    c.printComplex(); // output object c
    cout << endl;

    a.assignReal(10.0); // reset object a
    b.assignImage(-1.0); // reset object b
    a.printComplex(); // output object a
    cout << " - ";
    b.printComplex(); // output object b
    cout << " = ";
    // invoke subtract function and assign to object c
    c = a.subtract( b );
    c.printComplex(); // output object c
    cout << endl;

    return 0;
}
```