บทที่ 9
การเขียนโปรแกรมเชิงวัตถุ
(Object-Oriented Programming)

วัตถุประสงค์

1. เพื่อให้นักศึกษาเข้าใจหลักการเขียนโปรแกรมเชิงวัตถุและคลาส
2. เพื่อให้นักศึกษาสามารถสร้างและใช้งานโปรแกรมเชิงวัตถุและคลาสได้อย่างมีประสิทธิภาพ

การเขียนโปรแกรมเชิงวัตถุ

จากที่ผ่านมาจะเป็นการเขียนโปรแกรมแบบโครงสร้างเป็นหลัก และถือเป็นพื้นฐานที่สำคัญก่อนการศึกษาการเขียนโปรแกรมแบบเชิงวัตถุ (Object-Oriented Programming : OOP) การเขียนโปรแกรมเชิงวัตถุจะมีองค์ประกอบต่างๆ ได้แก่ การทำให้ข้อมูลอยู่ในขอบเขตจำกัด(encapsulation) และการซ่อนข้อมูล (Data hiding) การสืบทอด (Inheritance) การพ้องรูปแบบ(Polymorphism) การกำหนดข้อมูลนามธรรม(Abstraction) และการนำโค้ดกลับมาใช้(Reusable Code)

การกำหนดข้อมูลนามธรรม ซึ่งใช้ชนิดข้อมูลทั่วๆไปไม่สามารถกำหนดได้ ในภาษา C++ จะใช้คำสงวน class เป็นตัวกำหนดชนิดข้อมูลใหม่ ในรูปแบบของการสร้างคลาส ซึ่งจะประกอบด้วยสมาชิกต่างๆ เช่น ดาต้าเมมเบอร์ และเมมเบอร์ฟังก์ชั่น เป็นต้น รวมทั้งการประกาศสมาชิกต่างๆ ซึ่งสามารถประกาศการเข้าถึงได้ 3 แบบ คือ private, protected และ public

หลักการโปรแกรมเชิงวัตถุ

ภาษา C++ เป็นภาษาเชิงวัตถุเพราะสามารถสนับสนุนหลักการเขียนโปรแกรมเชิงวัตถุ ซึ่งประกอบด้วยรายละเอียดต่อไปนี้

- **การกำหนดข้อมูลนามธรรม (Abstraction)**

การแก้ปัญหาโจทย์ตามหลักกการเขียนโปรแกรมเชิงวัตถุ จะต้องตีความหมายโจทย์หรือคำตอบในรูปของการกำหนดข้อมูลนามธรรม จากนั้นนำมาสร้างเป็นออบเจกต์(Objects) หรือวัตถุในโปรแกรม จากหลักแนวความคิดการมองปัญหาให้เป็นแอ็บสแตรคชั่น จึงมีความยืดหยุ่นมาก

การกำหนดข้อมูลนามธรรมนั้น จะเป็นการรวบรวมข้อมูลและฟังก์ชั่นประเภทต่างๆเข้าด้วยกัน โดยใช้คำสงวน class บางครั้งอาจมีการใช้ Abstract Data Type : ADT หรือ User-Defined Type ต่างก็มีความหมายเช่นเดียวกับ Abstraction เพราะว่าใช้คำสงวน class สร้างเป็นข้อมูลชนิดใหม่

- **การซ่อนข้อมูล (Data Hiding)**

เมื่อใช้คลาสสร้างข้อมูลใหม่ขึ้นมา จะต้องปกปิดซอร์สโค้ดส่วนหนึ่งไว้ เพื่อไม่ให้ผู้ที่ไม่เกี่ยวข้องหรือไม่มีสิทธิในการใช้ข้อมูล สามารถเข้าถึงหรือแก้ไขข้อมูล ซึ่งจะช่วยให้โปรแกรมมีความปลอดภัยมากขึ้น ในภาษา C++ จะมีคำสงวน private, protected และ public เพื่อควบคุมการเข้าถึงข้อมูล โดยปกติแล้วจะใช้คำสงวน private กับข้อมูลเป็นหลัก ซึ่งจะทำให้ข้อมูลถูกปกป้อง หรือถูกซ่อนไว้

- **การทำให้ข้อมูลอยู่ในขอบเขตจำกัดและ(Encapsulation)**

การซ่อนข้อมูลไว้โดยการกำหนดการเข้าถึงด้วยคำสงวน private ถือว่าเป็นการทำให้ข้อมูลอยู่ในขอบเขตจำกัด หรือภายในขอบเขตคลาส (Class Scope) เท่านั้น ถ้ามีการนำข้อมูลนั้นไปใช้ภายนอกคลาส จะถือ

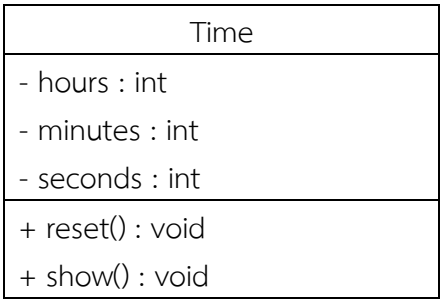
ว่าละเมิดกฎของหลักการเขียนโปรแกรมเชิงวัตถุ ส่วนฟังก์ชั่นของคลาสนั้นส่วนใหญ่จะใช้คำสงวน public ซึ่งจะทำให้สามารถเรียกใช้จากภายนอกขอบเขตของคลาสได้

โดยปกติเมื่อสร้างคลาสหนึ่งขึ้นมาใช้งาน จะมีการประกาศให้เป็นข้อมูลชนิดเดียวกับคลาส ตัวแปรเหล่านี้เรียกว่า ออบเจกต์(Object) หรือการสร้างตัวอย่างคลาส(Instance Class) ตัวอย่างเช่น คลาส Time เมื่อนำมาใช้จะถูกประกาศเป็นออบเจกต์ ดังนี้

```
Time  t;
```

ซึ่งจะเรียกตัวแปร t ว่า ออบเจกต์ t ดังนั้นออบเจกต์ t ก็คือชนิดข้อมูลหนึ่งที่สร้างมาจากคลาส Time ถ้าคลาส Time มีฟังก์ชั่นที่ถูกกำหนดการเข้าถึงแบบ public ออบเจดต์ t ก็สามารถเรียกใช้ โดยการส่งเมสเสจ(Message) ไป และเรียกออบเจกต์นั้นว่า มีอินเตอร์เฟส(Interface)

ในการเขียนโปรแกรมเชิงวัตถุ ทั้งข้อมูลและฟังก์ชั่นจะถูกเรียกใหม่เป็นดาต้าเมมเบอร์(Data Member) และเมมเบอร์ฟังก์ชั่น(Function Member) ตามลำดับ นอกจากนี้ยังมีภาษาที่ใช้บรรยายถึงคุณลักษณะของคลาส ซึ่งแสดงเป็นแผนภาพ คือภาษา UML (Unified Modeling Language) จากตัวอย่างของคลาส Time สามารถเขียนด้วย Class Diagram ของภาษา UML ดังรูป

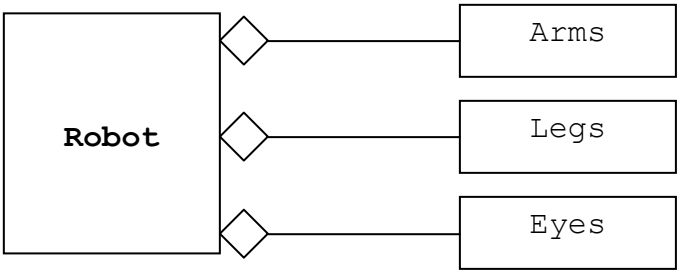


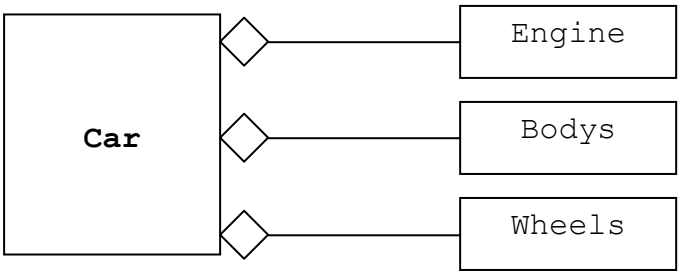
จากรูปคือ class diagram ของคลาส Time ประกอบด้วยสมาชิกดาต้าเมมเบอร์ ได้แก่ hours, minutes และ seconds ที่ถูกประกาศการเข้าถึงแบบ private และสมาชิกเมมเบอร์ฟังก์ชั่น ได้แก่ reset และ show ที่ถูกประกาศการเข้าถึงแบบ public

เครื่องหมาย – หมายถึงถูกประกาศการเข้าถึงแบบ private
เครื่องหมาย + หมายถึงถูกประกาศการเข้าถึงแบบ public
เครื่องหมาย # หมายถึงถูกประกาศการเข้าถึงแบบ protected

- **การนำคลาสมาเป็นออบเจกต์ของคลาสใหม่(Composition หรือ Aggregation)**

เป็นความสมพันธ์ระหว่างคลาสใหม่กับคลาสเดิม จะเรียกว่า มีคลาสหรือประกอบด้วย(has-a) ตัวอย่างเช่น มนุษย์ประกอบด้วย แขน ขา ตา ดังนั้นคลาสมนุษย์จึงมีคลาสแขน คลาสขา และคลาสตาประกอบกัน หรือรถยนต์ประกอบด้วย เครื่องยนต์ ตัวถัง ล้อ เป็นต้น คลาสรถยนต์จึงมีคลาสเครื่องยนต์ คลาสตัวถังและคลาสล้อ สามารถเขียนด้วยด้วย Class Diagram ของภาษา UML ได้ดังรูป





คลาส (class)

เมื่อแก้ปัญหาหรือวิเคราะห์โจทย์ได้แล้ว จะได้ข้อมูลและการกระทำประกอบกัน ข้อมูลจะบ่งบอกถึงคุณลักษณะของคลาสที่เรียกว่า แอตทริบิวต์ (Attributes) ภาษา C++ จะใช้ศัพท์เทคนิคว่า ดาต้าเมมเบอร์ ส่วนการกระทำต่างๆของคลาสจะบ่งบอกถึงพฤติกรรมของคลาส(Behavior) ภาษา C++ จะใช้ศัพท์เทคนิคว่า เมมเบอร์ฟังก์ชั่น

มีรูปแบบของการสร้างคลาสดังนี้

```
class  Class_name {
private:
        data member;
        member function();
protected:
        data member;
        member function();
public:
        data member;
        member function();
};
```

สำหรับ data member คือสมาชิกคลาส ซึ่งส่วนใหญ่จะเป็นข้อมูลชนิดพื้นฐาน เช่น double, int , float bool หรือ string สามารถใช้งานร่วมกับคำสงวนอื่นเช่น const , static เป็นต้น

ส่วน member function คือสมาชิกหนึ่งในคลาส แต่อยู่ในรูปฟังก์ชั่น ซึ่งจะประกอบด้วยฟังก์ชั่น ที่เรียกว่า คอนสตรัคเตอร์(Constructor) ดีสตรัคเตอร์(Destructor) ฟังก์ชั่นต่างๆ ที่ต้องใช้งานทั้งแบบ สแตติกและไม่ใช่สแตติก เป็นต้น

การทดลองที่ 9-1 การสร้างและใช้งาน class

ไฟล์ Lab09-01.cpp

```
1  #include <iostream>
2  #include "time.h"
3  using namespace std;
4  int main()
5  {
6          Time t1;
7          cout << "Enter hour : ";
8          cin >> t1.hours;
9          cout << "Enter minutes : ";
10         cin >> t1.minutes;
11         cout << "Enter second : ";
12         cin >> t1.second;
13
14         cout << "\nThis Time is : " << t1.hours << ":";
15         cout << t1.minutes << ":" << t1.second << endl;
16         cout << endl;
17         return(0);
18 }
```

ไฟล์ time.h

```
1  #ifndef TIME_H
2  #define TIME_H
3
4
5  class Time {
6  public:
7          int hours;
8
9          int minutes;
10         int second;
11 };
12
13
14 #endif
```

บันทึกผลการทดลอง

<u>การทดลองที่ 9-2</u> การสร้างและใช้งาน class และการเขียนโปรแกรมเชิงวัตถุ

ไฟล์ Lab09-02.cpp

```
#include <iostream>
#include <string>
#include "time.h"
using namespace std;
int main()
{
        Time t1;
        Time t2(11,30,40);

        cout << "\nThis time t1 : ";
        t1.showTime();
        cout << "\nThis time t2 : ";
        t2.showTime();
        cout << endl;
        t1.setHours(5);
        t1.setMinutes(25);
        t1.setSecond(50);
        t2.setMinutes(t2.getMinutes()+1);
        t2.setSecond(t2.getSecond()+1);
        cout << "\nNow time t1 : ";
        t1.showTime();
        cout << "\nNow time t2 : ";
        t2.showTime();
        cout << endl;
         return(0);
}
```

ไฟล์ time.h

```
#ifndef TIME_H
#define TIME_H

class Time {
private:
        int hours;
        int minutes;
        int second;
public:
        // constructor function
        Time();
        Time(int h, int m, int s);
        void setHours(int h);
        void setMinutes(int m);
        void setSecond(int s);
        int getHours();
        int getMinutes();
```

```
18          int getSecond();
19          void showTime();
20  };
21
22  #endif
```

ไฟล์ time.cpp

```
1   #include <iostream>
2   #include <string>
3   #include <iomanip>
4   #include "time.h"
5
6   using namespace std;
7
8   Time::Time()
9   {
10          hours = minutes = second = 0;
11  }
12
13  Time::Time(int h, int m, int s)
14  {
15          hours = h;  minutes = m;  second = s;
16  }
17
18  void Time::setHours(int h)
19  {
20          hours = h;
21  }
22
23  void Time::setMinutes(int m)
24  {
25          minutes = m;
26  }
27  void Time::setSecond(int s)
28  {
29          second = s;
30  }
31  int Time::getHours()
32  {
33          return(hours);
34  }
35
36  int Time::getMinutes()
37  {
38          return(minutes);
39  }
40
```

```
41  int Time::getSecond()
42  {
43          return(second);
44  }
45
46  void Time::showTime()
47  {
48          cout << setw(2) << getHours();
49          cout << ":" << setw(2) << getMinutes();
50          cout << ":" << setw(2) << getSecond();
51  }
```

บันทึกผลการทดลอง

ให้นักศึกษาเขียน Class Diagram ของ Class Time

การทดลองที่ 9-3 การออกแบบคลาส Employee สำหรับการเขียนโปรแกรมเชิงวัตถุ

ไฟล์ Lab09-03.cpp

```
1  #include <iostream>
2  #include <string>
3  #include "Employee.h"
4
5  using namespace std;
6
7  int main()
8  {
9          Employee E1;
10         Employee E2("Adom","Smith", "01","10","2542");
11
12         E1.setfirstName("Somchai");
13         E1.setlastName("Cheingpongpan");
14         E1.sethireDate( "10","10","2544");
15
16         cout << "Employee E1" << endl;
17         cout << E1.toString() << endl << endl;
18         cout << "Employee E2" << endl;
19         cout << E2.toString() << endl << endl;
20         return(0);
21 }
```

ไฟล์ Employee.h

```
1  #ifndef EMPLOYEE_H
2  #define EMPLOYEE_H
3  #include <string>
4
5  using namespace std;
6
7  //  class of Employee
8  class Employee {
9  private:
10         string firstName;
11         string lastName;
12         string day;
13         string month;
14         string year;
15
16 public:
17         Employee();
```

```
18          Employee(string fName, string lName);
19          Employee(string fName, string lName, string d, string m , string y);
20          void  setfirstName(string s);
21          void  setlastName(string s);
22          void  sethireDate(string d, string m, string y);
23          void  setDay(string d);
24          void  setMonth(string m);
25          void  setYear(string y);
26          string getfirstName();
27          string getlastName();
28          string gethireDate();
29          string getDay();
30          string getMonth();
31          string getYear();
32          string toString();
33  };
34
35  #endif
```

ไฟล์ Employee.cpp

```
1   #include <string>
2   #include "Employee.h"
3
4   Employee::Employee()
5   {
6           setfirstName("");
7           setlastName("");
8           sethireDate("","","");
9   }
10
11  Employee::Employee(string fName, string lName)
12  {
13          setfirstName(fName);
14          setlastName(lName);
15          sethireDate("","","");
16  }
17
18  Employee::Employee(string fName, string lName, string d, string m, string y)
19  {
20          setfirstName(fName);
21          setlastName(lName);
22          sethireDate( d, m , y );
23  }
24
```

```
25  void Employee::setfirstName(string fname)
26  {
27          firstName = fname;
28  }
29
30  void Employee::setlastName(string lname)
31  {
32          lastName = lname;
33  }
34
35  void Employee::sethireDate(string d, string m, string y)
36  {
37          setDay( d );
38          setMonth( m );
39          setYear( y );
40  }
41
42  void Employee::setDay(string d)
43  {
44          day = d;
45  }
46
47  void Employee::setMonth(string m)
48  {
49          month = m;
50  }
51
52  void Employee::setYear(string y)
53  {
54          year = y;
55  }
56
57  string Employee::getfirstName()
58  {
59          return(firstName);
60  }
61
62  string Employee::getlastName()
63  {
64          return(lastName);
```

```
65  }
66
67  string Employee::gethireDate()
68  {
69          return ( getDay() + "/" + getMonth() + "/" + getYear());
70  }
71
72  string Employee::getDay()
73  {
74          return ( day );
75  }
76
77  string Employee::getMonth()
78  {
79          return ( month );
80  }
81
82  string Employee::getYear()
83  {
84          return ( year );
85  }
86
87  string Employee::toString()
88  {
89          string Str = getfirstName() + "  " + getlastName() + "  " +
90                                          gethireDate();
91          return(Str);
92  }
```

บันทึกผลการทดลอง

ให้นักศึกษาเขียน Class Diagram ของ Class Employee

คลาสเทมเพลต

คลาสเทมเพลตเป็นการสร้างคลาสเดียวที่สามารถรองรับการกำหนดพารามิเตอร์ที่มีชนิดข้อมูลได้ทุกประเภททำให้สะดวกต่อการใช้งาน และโปรแกรมที่เขียนมีประสิทธิภาพมากขึ้น

รูปแบบคำสั่ง

```
template<class identifier> class _declaration;
template<typename identifier> class _declaration;
```

สามารถเขียนโดยใช้คำว่า class หรือ typename ก็ได้

เดิมหากต้องการให้คลาสสามารถทำงานได้หลายชนิดข้อมูลก็ต้องเขียนคลาสแยกตามจำนวน ซึ่งทำให้เกิดการสิ้นเปลืองและยุ่งยาก

ดังตัวอย่างเป็นการสร้างคลาสเก็บคะแนนเป็นชนิด int , float

```
class ScoreInt
{
private:
      int data[10];
public:
      ScoreInt();
      void set(int n, int value);
      int get(int n);
      int toSum();
};
```

```
class ScoreFloat
{
private:
      float data[10];
public:
      ScoreFloat();
      void set(int n, float value);
      float get(int n);
      float toSum();
};
```

รูปแบบการเรียกใช้งาน

```
ScoreInt si;
ScoreFloat sf;
```

แต่หากเขียนในรูปฟังก์ชั่นเทมเพลต จะได้ดังนี้

```
template<class T>
class Score
{
private:
      T data[10];
public:
      Score();
      void set(int n, T value);
      T get(int n);
      T toSum();
};
```

รูปแบบการเรียกใช้งาน

```
Score <int> DataInt;
Score <float> DataFloat;
Score <double> DataDouble;
```

<u>การทดลองที่ 9-4</u> การใช้งานคลาสเทมเพลต

ไฟล์ Lab09-04.cpp

```
#include<iostream>
#include<iomanip>
#include "Score.h"
using namespace std;

int main()
{
      Score<int> A;
      for(int n=0; n < 10;  n++)
            A.set(n, n);
      for(int n=0; n < 10;  n++)
            cout << A.get(n) << "  ";
      cout << endl;
      cout << A.toSum() << endl;

      Score<double> B;
      for(int n=0; n < 10;  n++)
            B.set(n, (double)n);
      for(int n=0; n < 10;  n++)
         cout << fixed<< setprecision(2) << B.get(n) << " ";
      cout << endl;
      cout << B.toSum() << endl;

      return(0);
}
```

ไฟล์ Score.h

```
#ifndef SCORE_H
#define SCORE_H

template<class T>
class Score
{
private:
      T data[10];
public:
      Score();
      void set(int n, T value);
      T get(int n);
      T toSum();
};

template<class T>
Score<T>::Score()
{
}

```

```
21 template<class T>
22 void Score<T>::set(int n, T value)
23 {
24       data[n] = value;
25 }
26
27 template<class T>
28 T Score<T>::get(int n)
29 {
30       if (n >= 10) n = 0;
31       return(data[n]);
32 }
33
34 template<class T>
35 T Score<T>::toSum( )
36 {
37       T total = 0;
38       for(int n = 0; n < 10 ; n++)
39             total += data[n];
40       return(total);
41 }
42
43 #endif
```

บันทึกผลการทดลอง

แบบฝึกหัดท้ายบท

1. ให้นักศึกษาเขียนโปรแกรมจัดการเกี่ยวกับข้อมูลนักศึกษา โดยเขียนโปรแกรมภาษา C++ แบบเชิงวัตถุ ให้นักศึกษาออกแบบคลาสไว้สำหรับเก็บข้อมูลนักศึกษาที่ประกอบด้วยรหัสนักศึกษา ชื่อนักศึกษา จำนวนวิชาที่เรียน ผลเกรดแต่ละวิชา และในคลาสออกรายงานเกรดเฉลี่ยของนักศึกษาได้ ให้นักศึกษาทำการออกแบบคลาสชื่อ Student โดยเขียน Class Diagram กำหนดรายละเอียดในส่วน data member และ member function ที่เกี่ยวข้องและเหมาะสมที่จะสามารถทำงานได้ครบ และเขียนโปรแกรมรียกใช้คลาสทำงานในการจัดเก็บข้อมูลนักศึกษาโดยสร้างการทำงานเป็นแบบเมนู

2. ให้นักศึกษาออกแบบคลาสที่เป็นแบบคลาสเทมเพลต สำหรับใช้เก็บข้อมูลแบบอาร์เรย์ 1 มิติ ตามจำนวนที่ต้องการได้ และสามารถรองรับการเก็บข้อมูลได้หลายชนิดข้อมูล โดยเขียน Class Diagram กำหนดรายละเอียดในส่วน data member และ member function ที่เกี่ยวข้องและเหมาะสมที่จะสามารถทำงานได้ครบ และเขียนโปรแกรมรียกใช้คลาสทำงานในการจัดเก็บข้อมูล