บทที่ 9
การเขียนโปรแกรมเชิงวัตถุ

วัตถุประสงค์

1. เพื่อให้นักศึกษาทราบถึงหลักการและแนวความคิดเชิงวัตถุ
2. เพื่อให้นักศึกษาเข้าใจรายละเอียดและคุณสมบัติต่างๆของการเขียนโปรแกรมเชิงวัตถุ
3. เพื่อให้นักศึกษาเข้าใจความสัมพันธ์ของวัตถุในรูปแบบต่างๆ
4. เพื่อให้นักศึกษาสามารถออกแบบและพัฒนาโปรแกรมเชิงวัตถุได้

แนวความคิดเชิงวัตถุ (Object-Oriented: O-O) เป็นการมองสิ่งต่างๆในโลกแห่งความเป็นจริงในลักษณะของรูปธรรม โดยมองระบบเป็นกลุ่มของวัตถุ ที่มีปฏิกิริยาต่อกันด้วยการนำข้อมูลและฟังก์ชันการทำงานรวมเข้ากันเป็นวัตถุ ซึ่งวัตถุสามารถอธิบายคุณสมบัติ รวมทั้งฟังก์ชันการทำงานในตัวเองได้ การติดต่อกันระหว่างวัตถุจะต้องติดต่อผ่านฟังก์ชันที่กำหนดไว้ให้เท่านั้น

ออบเจ็กต์(object) หมายถึงวัตถุที่อยู่ในโลกของความเป็นจริง ซึ่งมีทั้งจับต้องได้และไม่ได้ เช่น รถยนต์ เก้าอี้ คอมพิวเตอร์ บริษัท ลูกค้า พนักงาน คลังสินค้า เป็นต้น ในชีวิตประจำของมนุษย์จะต้องมีการเกี่ยวข้องกันหรือปฏิสัมพันธ์(Interactive)หรือมีความสัมพันธ์(Relationship)ระหว่างวัตถุต่างๆ เสมอ โดย

Reletionship คือความสัมพันธ์ระหว่างออบเจ็กต์ เช่น นายสมชาย เป็นเจ้าของรถยนต์

Interaction คือการปฏิสัมพันธ์หรือการกระทำที่เกิดขึ้นระหว่างออบเจ็กต์ เช่น นายสมชายขับรถยนต์ หรือ นายสมชายซ่อมรถยนต์

Concept คือแนวความคิดที่ให้กับวัตถุ ภายใต้กรอบที่กำหนด(Domain) เช่น แนวความคิดของรถยนต์ คือรถทุกคันต้องมีตัวถัง มีล้อ มีเครื่องยนต์ เหมือนกันทุกคัน ออบเจ็กต์ทุกตัวจะต้องอยู่ในคลาส(Class) โดยกลุ่มของออบเจ็กต์ที่มีโครงสร้างพื้นฐานพฤติกรรมเดียวกันหรือมีคุณลักษณะเหมือนกันจะรวมกลุ่มอยู่ในคลาสเดียวกัน ซึ่งคลาสคือต้นแบบข้อมูลที่มีไว้เพื่อสร้างออบเจ็กต์นั่นเอง

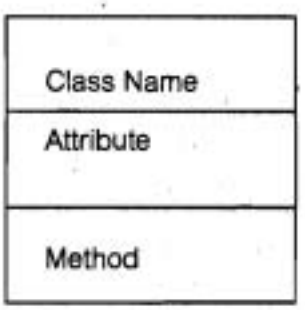
Abstraction คือกระบวนการในการให้แนวความคิดกับออบเจ็กต์ จนเกิดเป็นคลาส

Instance คือ ออบเจ็กต์ที่ถูกสร้างขึ้นมาจากคลาส

Attribute คือคุณสมบัติต่างๆของออบเจ็กต์ ซึ่งอยู่ภายในกรอบที่เรากำหนด

Function คือ พฤติกรรมหรือความสามารถในการทำกิจกรรมของออบเจ็กต์

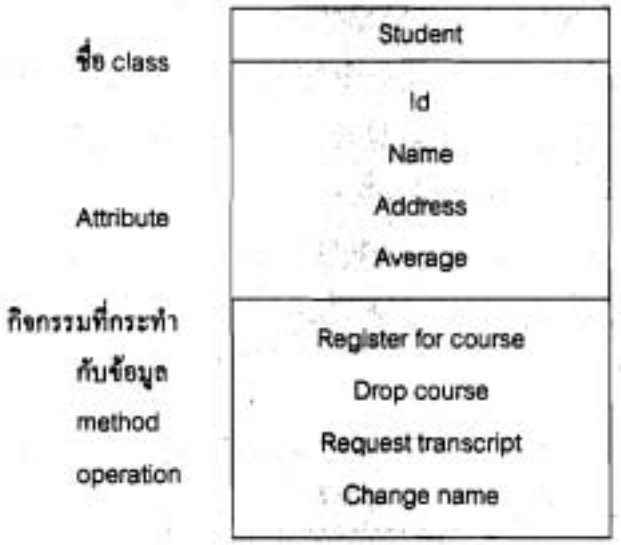
แผนภาพของคลาส



Object ทุกตัวต้องอยู่ใน class ใด class หนึ่ง และเราจะทราบคุณสมบัติรายละเอียดต่างๆ ของ Object ดูได้จากคลาส ประกอบไปด้วย

1. ชื่อคลาส (Name class)
2. คุณลักษณะ (Attribute) ของข้อมูล
3. กรรมวิธี (Method) หรือการกระทำ (Operation) และฟังก์ชั่น(Function)

ตัวอย่าง ต้นแบบของ class student



9.1 กระบวนการสร้างคลาส

กระบวนการสร้างคลาสเป็นการให้แนวความคิดกับออบเจ็กต์ในโลกของความเป็นจริงเพื่อสร้างคลาส สามารถแบ่งขั้นตอนการสร้างออกได้ 4 กระบวนการดังนี้

9.1.1 Classification Abstraction เป็นกระบวนการที่พิจารณาโดยแยกประเภท(classify) ของออบเจ็กต์ต่างๆที่อยู่ในกรอบหรือขอบเขตที่กำหนด เพื่อให้ได้คลาสที่ต้องการ มีขั้นตอนดังนี้

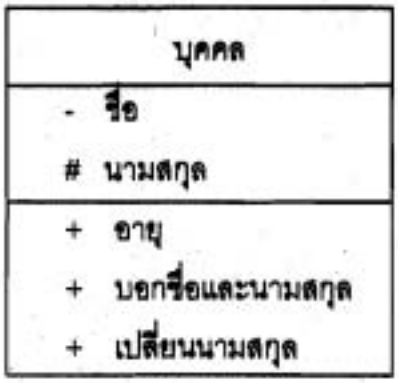
1) กำหนดขอบเขตของระบบที่กำลังพิจารณา จากการเก็บรวบรวมข้อมูลจากระบบ

2) พิจารณาหรือค้นหาออบเจ็กต์ภายในระบบนั้น

3) สร้างคลาสจากออบเจ็กต์ต่างๆ โดยพิจารณาถึง Attribute และ Function ต่างๆที่เกี่ยวข้องกัน

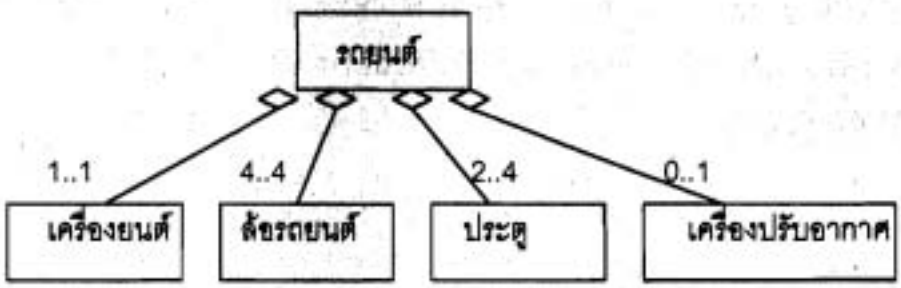
4) พิจารณาถึงประเภทของ Attribute และ Function ของแต่ละออบเจ็กต์ ว่ามีส่วนใดที่ต้องปกปิดไว้เป็นการส่วนตัว หรือสามารถมองเห็นหรือใช้ได้จากภายนอก ซึ่งประเภทของ Attribute และ Function สามารถแบ่งออกเป็น 3 ประเภทดังนี้

- Private เป็นประเภทที่ปกปิดไว้เป็นส่วนตัว ไม่สามารถมองเห็นได้ การเข้าถึง Attribute ต้องกระทำผ่าน Function เสมอ โดยทั่วไปเราใช้เครื่องหมาย – กำกับไว้หน้า Attribute หรือ Function ของคลาสนั้นๆ
- Protected เป็นประเภทที่ไม่สามารถเห็นได้จากภายนอกเช่นกัน แต่ประเภทนี้สามารถส่งต่อให้ Inherited class ได้ เราใช้เครื่องหมาย # กำกับไว้หน้า Attribute หรือ Function ของคลาสนั้นๆ
- Public เป็นประเภทที่สามารถมองเห็นได้จากภายนอก เราใช้เครื่องหมาย + กำกับไว้หน้า Attribute หรือ Function นั้นๆ

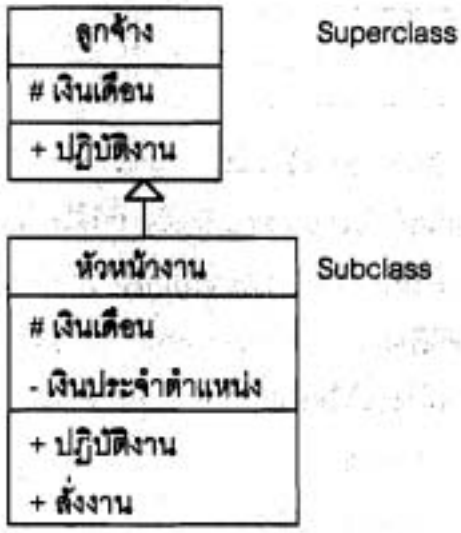


9.1.2 Aggregation Abstraction เป็นกระบวนการที่นำคลาสพื้นฐานที่สร้างขึ้นมาพิจารณาเพื่อรวมกันหรือประกอบกันให้เกิดเป็นคลาสใหม่มีแนวความคิดใหม่ เราจะใช้สัญลักษณ์ลูกศรสี่เหลี่ยมขนมเปียกปูนแสดงถึงการรวมกันของคลาสโดยมีการกำหนด

จำนวนความสัมพันธ์ในคลาสย่อย(Cardinality) ที่นำมาประกอบกัน เช่น 2..4 หมายความว่า จำนวนความสัมพันธ์น้อยที่สุดเท่ากับ 2 และมากที่สุดเท่ากับ 4 เป็นต้น

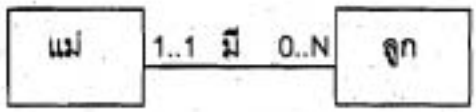


9.1.3 Generalization Abstraction เป็นกระบวนการพิจารณาคลาสที่มีลักษณะที่คล้ายคลึงกันหรือมีคุณลักษณะอย่างใดอย่างหนึ่งเหมือนกัน นำออกมาจัดเป็นหมวดหมู่เป็นคลาสที่เป็นสามัญ(General) และเมื่อเพิ่มคุณลักษณะพิเศษ(Specialization) ให้กับคลาสสามัญ จะทำให้ได้คลาสใหม่ ที่เราเรียกว่า คลาสย่อย(Subclass) โดยถ่ายทอดคุณลักษณะของคลาสสามัญมายังคลาสย่อยนั่นเองเราเรียกการที่ คลาสย่อยรับคุณสมบัติทุกอย่างมาจาก Superclass ว่า Inheritance สัญลักษณ์ ที่ใช้คือลูกศรหัวรูปสามเหลี่ยมชี้จาก Subclass ไปยัง Superclass



Inheritance คือการถ่ายทอดคุณสมบัติจากคลาสที่มีอยู่แล้วไปยังคลาสใหม่ ซึ่งทำให้เกิดข้อดีในหลายๆด้าน กล่าวคือ ทำให้มีโครงสร้างที่เป็นระบบ ปรับเปลี่ยนได้ง่าย รวมทั้งลดเวลาและค่าใช้จ่ายในการพัฒนาระบบ เพราะสามารถนำคลาสที่ออกแบบไว้แล้วนำกลับมาใช้ใหม่ได้(Reusable) และการถ่ายทอดที่เป็นลำดับขั้นทำให้ความสัมพันธ์ระหว่างออบเจ็กต์มีความชัดเจนมากยิ่งขึ้น แนวความคิดเชิงวัตถุถือว่าการสืบทอดเป็นสิ่งที่สำคัญ เพราะว่าไม่มีสิ่งใดในโลกที่เกิดขึ้นเอง

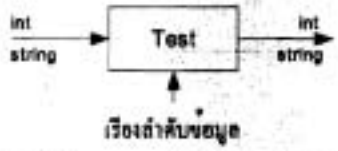
9.1.4 Association Abstraction เป็นกระบวนการสร้างความสัมพันธ์ระหว่างคลาสต่างๆในระบบที่เราสนใจ โดยเขียนความสัมพันธ์เป็นเส้นตรงเชื่อมระหว่างคลาสทั้งสอง โดยเขียนชื่อความสัมพันธ์และจำนวนความสัมพันธ์กำกับไว้เสมอ ที่ปลายเส้น



ในการพัฒนาโปรแกรมเชิงวัตถุนั้น เราสามารถสร้างรายละเอียดหรือหน้าที่ของแต่ละคลาส รวมทั้งความสัมพันธ์ระหว่างออบเจ็กต์ต่างๆได้โดยเน้นที่ผลลัพธ์ของการทำงานมากกว่ากระบวนการทำงาน ทำให้เราสามารถออกแบบฟังก์ชันหนึ่งๆให้มีการตอบสนองได้หลายรูปแบบได้ ซึ่งเป็นคุณสมบัติที่เรียกว่า Polymorphism หมายถึงการเรียกใช้ในชื่อฟังก์ชันเดียวกันแต่สามารถตอบสนองการทำงานได้หลายรูปแบบ เช่น เรียกใช้

```
circle();
circle(7);
```

จะเห็นได้ว่าเรียกใช้ฟังก์ชันชื่อ circle เหมือนกันแต่ตอบสนองการทำงานแตกต่างกัน ทำให้ลดคำสั่งในโปรแกรมได้มาก



จากรูปเป็นการสร้างฟังก์ชันชื่อ Test โดยมีหน้าที่เรียงลำดับกลุ่มของข้อมูล การทำงานของฟังก์ชันสามารถกำหนดชนิดของข้อมูลได้ โดยเรียงลำดับข้อมูลที่เป็นเลขจำนวนเต็มหรือเป็นกลุ่มของข้อความก็ได้ ประโยชน์คือทำให้ไม่ต้องสร้างฟังก์ชันที่มีการปฏิบัติงานคล้ายกันหรือเหมือนกันในระบบ ประโยชน์คือ ทำให้ลดความยุ่งยากในการจดจำคำสั่ง

รูปแบบของการกำหนด class

Class Definition

Form: class className
{
public:
- List of class attributes(variables, types, constants, and so on that may be accessed by name from outside the class
- List of prototype for each member function that may be accessed by name from outside the class
...
private:
- List of class attributes(variables, types, constants, and so on that intended to be hidden for reference from outside the class
- List of prototype for each member function intended to be hidden from outside of the class
...
}; // Don't forget the semicolon

```
Example: class checking Account
        {
            public:
                // Member functions ...
                // Make deposit into checking account
                void makeDeposit (int) ;        // IN : number of account to
                                                // receive deposit
                // Set service charge for account
                void setServiceCharge(int ,     // IN : number of account to
                                      float);   // IN : service charge
             private:
                // Data members ...
                char    initFirst , .
                        initmiddle ,
                        initLase ;      // initials
                int accountNum ;                        // account number
                float balance ;                         // account balance
                folat serviceCharge ;                   // service charge
        };
        // Don't forgot the semicolon
```

กรณีศึกษา 1 : การสร้าง class counter

เป็น class ที่ทำหน้าที่ในการนับ โดยมีการกำหนดคุณลักษณะและหน้าที่ที่กระทำดังนี้

Attribute

1. ตัวแปร ที่เก็บค่าตัวนับ int count;
2. ตัวแปร ที่เก็บค่าตัวนับที่มากที่สุด int maxValue

Member function กิจกรรมที่กระทำต่อข้อมูล

1. เพิ่มตัวนับ ก็ต้องสร้างฟังก์ชั่นขึ้นมา โดยสร้างฟังก์ชั่น increment
2. ลดค่าตัวนับ โดยสร้างฟังก์ชั่น decrement
3. กำหนดค่าตัวนับ โดยสร้างฟังก์ชั่น setCount
4. กำหนดค่าตัวนับที่มากที่สุด โดยสร้างฟังก์ชั่น setmaxValue
5. ส่งค่า count ไปยังจุดเรียกใช้ โดยสร้างฟังก์ชั่น getCount
6. ส่งค่า MaxValue กลับไปยังจุดเรียกใช้ โดยสร้างฟังก์ชั่น getmaxValue

Specification for counter Class	
Attributes for counter Class	
int count	the counter value
int maxVaue	the maximum counter value
Member Functions for counter *Class*	
increment	Increments the counter
decrement	Decrements the counter
setCount	Sets the counter value
setmaxValue	Sets the maximum value
getcount	Returns the count
getmaxValue	Returns the maximum value

```
// Definition of class counter
// File: counter .h
// Counter class definition
#ifndef COUNTER_H          // used to avoid multiple definitions
#define COUNTER_H          //      not part of the class

class counter
{
    public:
        // Member Functions
        // Constructors
        counter () ;
        counter (int) ;

        // Increment counter
        void increment () ;

        // Decrement counter
        void decrement () ;

        // Set counter value
        void setcount (int) ;

        // Set maximum counter value.
        Void setmaxvalue (int) ;

        // Return current counter value
```

```
        // Return current counter value
        int getCount()  const ;

         //Return maximum counter value
         int getMaxValue() const;

private:
        // Data members (attributes)
        int count ;
        int maxvalue ;
} ; // Note – a class definition MUST end with a semicolon
#endif  // COUNTER_H
```

แฟ้มที่สร้างชื่อ counter.h เป็น header file จะบรรจุเพียงนิยามของ class ทำให้ทราบคุณลักษณะของ class เท่านั้น ส่วนการทำงานหรือฟังก์ชันในการปฏิบัติงานนั้นเราจะแยก สร้างในแฟ้มที่ชื่อว่า counter.cpp

Class Implementation

บรรจุคำสั่ง C++ สำหรับ member function โดยจะแยกเป็นแฟ้ม ชื่อ counter.cpp เพื่อเป็นการซ่อน จาก users เพื่อไม่ให้ทราบรายละเอียด โดยเพื่อเน้นผลลัพธ์มากกว่าการปฏิบัติงานจริง จะมีความแตกต่างจากฟังก์ชั่นในโปรแกรมตรงที่จะใช้ ตัวกระทำ :: เรียกว่า scope resolution operator เพื่อบอก compiler ให้ทราบว่าเป็นการกำหนดฟังก์ชั่นของ class ที่เราสนใจ

รูปแบบการกำหนดฟังก์ชันในคลาส

Class Member Function Definition
Form: type className: : fname (formal parameter list)

```
{
. . .
function body
. . .
}
```

Example:

```
// Constructor
CheckingAccount: : checkingAccount()
{
        accountNum = 0;
        balance = 0.0;
}
// Make deposit into checking account
void checkingAccount: :makeDeposit
        (int acNum)     // IN: number of account
                        //        to receive deposit
{
        // Local data . . .
        money amount;
        // Make deposit.
        If (accountNum = = acNum)
        {
        . . .
        }
        else
        cout << "Wrong account number specified."
             << endl;
        }
```

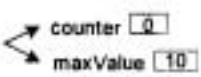
```
// Implementation file for counter class
// File: counter.cpp
// Counter class implementation

#include "counter.h"

#include <iostream>
#include <climits>              // For INT_MAX
using namespace std;

// Default constructor
counter : :counter ()
{
        count = 0 ;
        maxValue = INT_MAX ;        // Set maxValue to default maximum
}
// Constructor with argument
counter : : counter (int mVal)          // IN: maximum integer value
{
        count = 0 ;
        maxValue = mVal ;               // Set maxValue to mVal
}
//Increment counter
void count:: :increment()
{
        if (count < maxValue)
                count++;
```

```
        else
                Cerr << "Counter overflow. Increment ignored. "  << endl;
}
// Decrement counter
void counter: :decrement()
{
        if (count > 0)
          count--;

        else
           cerr  <<  "Counter underflow. Decrement ignored."  << endl ;
}
// Set counter value
void counter: :setCount (int val)
{
        if (val > = 0 &&  val <= maxValue)
                count = val:
        else
                cerr <<  "New value is out of range. Value not changed."
                     << endl;
}
// Set maximum counter value
void counter: : setMaxValue (int val)
{
        if (val >= 0 &&  val <= INT_MAX)
                maxValue = val;
        else
```

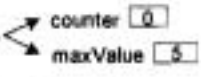
```
            cerr << "New maxValue is out of range – range – not changed."
                 << end;
}
// Return current counter value
int counter : :getCount () const
{
        return count;
}
// Return maximam counter value
int counter: : getMaxValue () const
{
        return maxValue;
}
```

ฟังก์ชันที่เป็น Constructer จะมีชื่อเดียวกับชื่อของคลาส ในที่นี้ชื่อฟังก์ชันคือ counter แต่การทำงานสามารถทำงานโดยอัตโนมัติ ถ้ามีการสร้างออบเจ็กต์ ใหม่เกิดขึ้น โดยต้องอยู่ในคลาสชื่อ counter การปฏิบัติงานขึ้นอยู่กับการกำหนดออบเจ็กต์ โดยผู้ใช้สามารถกำหนดได้ 2 แบบคือแบบมีอาร์กิวเมนต์ และแบบไม่มีอากิวเมนต์ ซึ่งจะส่งผลให้ทำงานในฟังก์ชันที่แตกต่างกันและมีค่า Attribute แตกต่างกันไปด้วยตามการปฏิบัติงานในฟังก์ชัน เช่น กำหนดออปเจกต์ชื่อ A และ B ดังนี้

counter A; //จะเรียกฟังก์ชันชื่อ counter ที่ไม่มี argument ทำงาน



counter B(5); //จะเรียกฟังก์ชันชื่อ counter ที่มี argument ทำงาน



เราสามารถสร้างโปรแกรมภาษา C++ เพื่อให้สามารถเรียกใช้ฟังก์ชันต่างๆที่บรรจุในคลาสชื่อ counter ได้ ดังนี้

```
// Test program and sample output for user of counter class
// File:  counterTest.cpp
// Test the counter class
#include "counter.h"
#include <iostream>
using namespace std;

int main ()
{
        counter c1;             // variable of type counter – maximum value   INT_MAX
        counter c2 (10) ;       // variable of type counter- maximum value    10

        // Test setcount, increment, decrement, and getCount   functions
        c1.setCount (50) ;      // SetValue of c1 to 50
        c1.decrement () ;
        c1.decrement () ;
        c1.increment () ;
        cout << "Final value of c1 is" << c1.getCount () << endl ;
        c2.increment () ;
        c2.increment () ;
        c2.increment () ;
        cout << "Final value of c2 is " << c2.getCount () << endl;
        return 0;
}
```

ผลจากการทำงานของโปรแกรมได้ผลลัพธ์ดังนี้

```
Final value of c1 is 49
Final value of c2 is 1
```

การทำงานของโปรแกรมนี้มีการใช้ฟังก์ชันชื่อเดียวกันแต่มีการทำงานที่แตกต่างกันซึ่งขึ้นอยู่กับอาร์กิวเมนต์ เราเรียกการทำงานของฟังก์ชันนี้ว่า Function Overloading นอกจากนี้ การทำงานของโปรแกรมนี้ยังมีการใช้คุณสมบัติของคลาสในข้อที่ว่า Polymorphism เป็นความสามารถของฟังก์ชั่น ที่ผู้ใช้สามารถใช้งานฟังก์ชั่นได้หลายๆรูปแบบ และมีการปฏิบัติงานเปลี่ยนไปตามอาร์กิวเมนต์ที่กำหนด

สำหรับฟังก์ชันที่สร้างขึ้นภายในคลาสนั้น เราสามารถออกแบบหรือสร้างฟังก์ชันให้อาร์กิวเมนต์ที่มีการส่งผ่านค่าระหว่างฟังก์ชันเป็นชนิดออบเจ็กต์ได้ โดยการอ้างอิงจะมีการใส่เครื่องหมาย & ไว้หลังชื่อของคลาสของออบเจ็กต์ และต้องใส่คำเฉพาะ const ไว้ข้างหน้าของคลาส ซึ่งจะเป็นการบ่งบอกให้คอมไพเลอร์ทราบว่า มีการส่งผ่านออบเจ็กมายังฟังก์ชันนี้ โดยเป็นการส่งผ่านค่าทางเดียว (pass by value)

```
int counter: : compareCounter (const counter& aCounter) const
{
        int result;
        if (count  <= aCounter.count)
                result  = -1;
        else if (count = = aCounter.count)
                result  = 0;
        else
                result   =  1;

        return result;
}
```

การทำงานของฟังก์ชันนี้ถ้ามีการเรียกใช้ดังนี้

```
I = c1.compareCounter(c2) ;
```

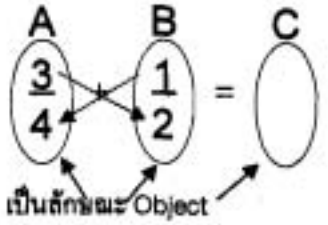
ผลของการทำงานค่าของตัวแปร I มีค่าเท่ากับ 1 โดยเมื่อมีการเรียกใช้ ฟังก์ชันนี้จะมีการส่งผ่านออบเจ็กต์ c2 เป็นข้อมูลเข้าไปยังฟังก์ชัน โดยจะนำ attrbiute ของ ออบเจ็กต์c2 ให้ค่าแก่ออบเจ็กต์ aCounter เป็นส่งผ่านค่าทางเดียว(pass by value) การทำงานภายในฟังก์ชันจะนำค่าของ count ของออบเจ็กต์ที่เรียกใช้ฟังก์ชันในที่นี้คือของออบเจ็กต์ c1 เปรียบเทียบกับ aCounter ในที่นี้คือ 49>1 ส่งผลให้ตัวแปร result เท่ากับ 1 และส่งผ่านค่ากลับไปยังจุดเรียกใช้

กรณีศึกษา 2 : การสร้าง class Fraction

เป็นการสร้าง class ที่ทำหน้าที่ในการคำนวณข้อมูลที่เป็นเศษและส่วน โดยจะจำลองการทำงานโดยกำหนดคลาสที่ชื่อ Fraction ที่มี Attribute ประกอบด้วย ค่าเศษ และ ค่าส่วน ที่แยกออกจากกัน ฟังก์ชันที่กระทำในคลาสนี้เป็นเรื่องของการคำนวณหาค่าบวก ลบ คูณ หาร ระหว่างออบเจ็กต่างๆ ก่อนอื่นต้องทำการพิจารณาการคำนวณหาค่าเศษส่วนเพื่อให้เห็นแนวความคิดก่อนดังนี้

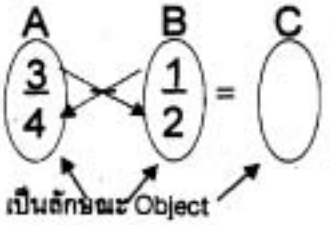
กำหนดให้ A ,B, C เป็นออบเจ็กต์ที่อยู่ในคลาสเดียวกัน เราสามารถสร้าง Member functions ของคลาสนี้ โดยระบุรายละเอียดของการทำงานดังนี้

1. การบวกกัน (add) เช่น



$$= \frac{(3*2)+(1*4)}{4*2} = \frac{6+4}{8} = \frac{10}{8}$$

2. การลบกัน (subtract) เช่น



$$= \frac{(3*2)-(1*4)}{4*2} = \frac{6-4}{8} = \frac{2}{8}$$

3. การคูณเศษส่วน (multiply) เช่น

$$\frac{3}{4} * \frac{1}{2} = \frac{3*1}{4*2} = \frac{3}{8}$$

4. การหารเศษส่วน (divide) เช่น

$$\frac{\frac{3}{4}}{\frac{1}{2}} = \frac{3}{4} * \frac{2}{1} = \frac{6}{4}$$

5. setnum กำหนดค่าให้กับเศษ
6. setdenom กำหนดค่าให้กับส่วน
7. displayFrac แสดงค่าที่เก็บ คือ เศษ และส่วน ออกทางจอภาพ
8. getNum เป็นลักษณะการส่งผ่านค่ากลับ (จะ return ค่า num กลับ)
9. getDenom จะ return ค่า denom กลับ
10. readFrac จะเป็นลักษณะอ่านค่าจากแป้นพิมพ์

กระบวนการสร้าง class

1. class definition เป็นการกำหนดต้นแบบของ class (Prototype) โดยจัดเก็บใน fraction.h
2. class Implementation เป็นการกำหนดการทำงานของ function ต่างๆของคลาสนี้ เป็นคำสั่งของภาษา C++ ที่กระทำใน class โดยจัดเก็บใน fraction.cpp

Specification for fraction Class	
Attributes for fraction Class	
int num	Numerator of the fraction
int denom	Denominator of the fraction
Member Functions for fraction Class	
fraction	A constructor

setNum	Sets the numerator
setDenom	Sets the denominator
multiply	Multiplies fractions
divide	Divides fractions
add	Adds fractions
subtract	Subtracts fractions
readFrac	Reads a fraction
displayFrac	Displays a fraction
getNum	Returns a numerator
getDenom	Returns the denominator

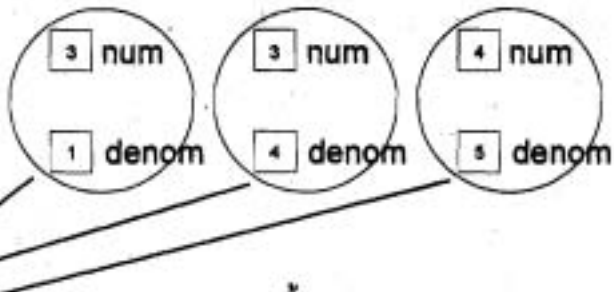
```
// Class definition for fraction
// File: fraction.h
//  Fraction class definition
#ifndef FRACTION_H
#define FRACTION_H

class fraction
{
        public:
                // Member functions
                // Constructors
                fraction () ;
                fraction (int);
                fraction (int, int) ;
```

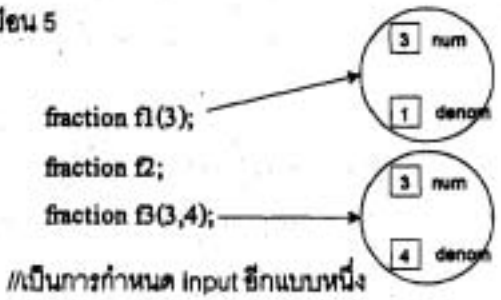
```
// Set numerator and denominator
void setNum (int) ;
void setDenom (int) ;

// Multiply fractions
fraction multiply (fraction f1) ;

// Divide fractions
fraction divide (fraction f1)

  //Add Fractions
  fraction add(fraction f1);

// Subtract Fractions
fraction subtract (fraction f1) ;

// Read a fraction
void readFrac () ;

// Display a fraction
void displayFrac ()  const;

// Accessors
int getNum ()  const;
int getDenom ()  const;

private:
```

```
        // Data members (attributes)
        int num;
        int denom;
};
#endif // FRACTION_H
```

สร้างโปรแกรมเพื่อเรียกใช้งาน

```
//test.cpp
#include <iostream>
#include "fraction.h"
using namespace std;
int main()
{
    fraction f1,f2,f3;          //Object นี้จะสร้าง num และ denom ของแต่ละตัว
    f1.setNum(3);   //เติม num เป็นอะไรอยู่ก็แล้วแต่ มันจะนำค่า argument (3) ไปไว้ที่ num
    f2.setDenom(4); //เติม denom เป็นอะไรอยู่ก็แล้วแต่ มันจะนำค่า argument (4) ไปไว้ที่ denom
    f2.readFrac();  //รับข้อมูลที่ป้อนจากแป้นพิมพ์ ครั้งแรกเป็น num ครั้งที่สองเป็น denom เช่น
                    // ครั้งแรกป้อน 4 ครั้งที่สองป้อน 5
    f3=f1.add(f2);
    f1.displayFrac(); cout<<"+";
    f2.displayFrac(); cout<<"=";
    f3.displayFrac();

    f3=f1.subtract(f2);
    f1.displayFrac(); cout<<"-";
    f2.displayFrac(); cout<<"=";
    f3.displayFrac();

    f3=f1.multiply(f2);
```





//เป็นการกำหนด input อีกแบบหนึ่ง

```
        f1.displayFrac(); cout<<"*";
        f2.displayFrac(); cout<<"=";
        f3.displayFrac();

        f3=f1.divide(f2);
        f1.displayFrac(); cout<<"/";
        f2.displayFrac(); cout<<"=";
        f3.displayFrac();
        return 0;
}
```

ทำงาน

```
Enter 1st   fraction:
Enter numerator / denominator:3/4
Enter 2 nd fraction:
Enter numerator / denominator: 5 / 6

3/4  * 5/6 = 15/24
3/4  / 3/6 = 18/24
3/4  + 5/6 = 38/24
3/4  - 5/6 = -2/24
```

Implementation file for class fraction

```
//File : fraction.cpp
#include "fraction.h"
#include <iostream>
using namespace std;
```

```
//Member function
//constructors
fraction :: fraction()
{
        num=0;
        denom=1;
}
fraction :: fraction(int n)
{
        num=n;
        denom=1;
}
fraction :: fraction(int n,int d)
{
        num=n;
        denom=d;
}
void fraction :: setNum(int n)
{
        num=n;
}
void fraction :: setDenom(int d)
{
        denom=d;
}
void fraction :: readFrac()
{
```

```
        cout <<"Enter numerator ";
        cin >>num;
        cout <<"Enter Denominitor ";
        cin >>denom;
}
fraction fraction :: multiply(fraction f)
{
        fraction temp (num*f.num,denom*f.denom);
        return temp;
}
fraction fraction :: divide(fraction f)
{
        fraction temp (num*f.denom,denom*f.num);
        return temp;
}
fraction fraction :: add(fraction f)
{
        fraction temp (num*f.denom+denom*f.num,denom*f.denom);
        return temp;
}
fraction fraction :: subtract(fraction f)
{
        fraction temp (num*f.denom-denom*f.num,denom*f.denom);
        return temp;
}
void fraction :: displayFrac() const
{
```

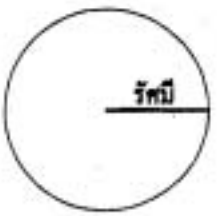
```
        cout <<num<<"/"<<denom;
}
int fraction :: getNum() const
{
        return num;
}
int fraction :: getDenom() const
{
        return denom;
}
```

สำหรับการสร้างฟังก์ชันเพื่อทำงานกับกลุ่มของออบเจ็กต์ของคลาสนี้ ในส่วนของ constructor นี้มีการเรียกใช้งานได้แตกต่างกันถึง 3 แบบด้วยกัน ขึ้นอยู่กับการกำหนดออบเจ็กต์ของผู้ใช้ ส่วนฟังก์ชันในการคำนวณนั้น เช่น fraction fraction :: subtract(fraction f) มีการส่งผ่านค่าเป็นออบเจ็กต์มายังฟังก์ชัน โดยนำออปเจ็กที่ส่งมาทำการลบจากออบเจ็กต์ตัวที่เรียกใช้งาน และส่งผ่านค่ากลับเป็นออบเจ็กต์ใหม่ไปยังจุดเรียกใช้

กรณีศึกษา 3 : การสร้าง class Circle

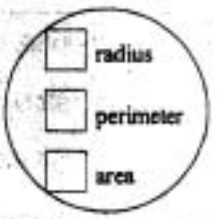
เป็นการสร้าง class Circle โดยสร้างกลุ่มของออบเจ็กต์ที่มี Attribute ประกอบด้วยรัศมี , พื้นที่ และความยาวของเส้นรอบวง โดยฟังก์ชันที่กระทำกับออบเจ็กต์ ต่างๆเหล่านี้ได้แก่ การหาพื้นที่ของวงกลม , ความยาวของเส้นรอบวง , การกำหนดค่ารัศมีให้แก่ออบเจ็กต์ และอื่นๆอีกดังนี้

วิเคราะห์



รัศมี ให้ใช้ตัวแปรชื่อ radius
พื้นที่ ให้ใช้ตัวแปรชื่อ area
เส้นรอบวง ให้ใช้ตัวแปรชื่อ perimeter

รูปแบบการสร้างออบเจ็กต์ต่างๆ เช่น



circle A;

circle B;

circle C, D ;

// ในแต่ละตัว ก็จะมี รัศมี, เส้นรอบวง, พื้นที่ เป็นของตนเอง

Member function

1. circle เป็นลักษณะของการกำหนดค่าเริ่มต้นให้กับ รัศมี, พื้นที่, ความยาวของเส้นรอบวง
2. setRadius เป็นลักษณะการกำหนดค่า รัศมี เป็นการส่งค่ารัศมีใหม่ไปให้
3. ComputeArea เป็นลักษณะการคำนวณหาพื้นที่ (สูตร πr^2) $\pi = \frac{22}{7}$, r คือ รัศมี
4. ComputePerimeter เป็นลักษณะการคำนวณหาเส้นรอบวง (สูตร $2\pi r$)
5. displayCircle เป็นลักษณะการพิมพ์ค่า รัศมี, พื้นที่, และเส้นรอบวง
6. getRadius() ส่งค่ารัศมีกลับไปยังจุดเรียกใช้
7. getArea() ส่งค่าพื้นที่กลับไปยังจุดเรียกใช้
8. getPerimeter() ส่งความยาวเส้นรอบวงกลับไปยังจุดเรียกใช้

Specification for circle Class		
Attributes for circle Class		
	float radius	Radius
	float area	Area
	float perimeter	Perimeter
Member Functions for circle Class		
	circle	A constructor
	setRadius	Sets the circle radius
	ComputeArea	Computes the area of the circle: area = □ x radius 2

Computeperimeter	Computes the perimeter of the circle:
	perimeter =2.0 x □ x radius
displayCircle	Displays circle attributes
getArea	Returns circle area
getperimeter	Returns circle perimeter

การสร้าง class circle

```
//File : circle.h
#ifndef CIRCLE_H
#define CIRCLE_H
class circle
{
public :
        circle();
        void setRadius(float);
        void ComputeArea();
        void ComputePerimeter();
        void displayCircle() const;
        float getRadius() const;
        float getArea() const;
        float getPerimeter() const;
private :
        float radius;
        float area;
        float perimeter;
};
#endif
```

สร้างโปรแกรมเพื่อเรียกใช้งาน

```
//Test.cpp
#include "circle.h"
#include <iostream>
using namespace std;

int main()
{
        circle C;
        C.setRadius(7);                 //จะเปลี่ยนค่า radius เดิมให้เป็น 7.0
        C.ComputeArea();        //นำ radius ปัจจุบันไปคำนวณหาค่าพื้นที่ แล้วเก็บใน area
        C.ComputePerimeter();           //นำ radius ปัจจุบันไปคำนวณหาค่าพื้นที่ แล้วเก็บใน
perimeter
        C.displayCircle();      //จะพิมพ์ค่า radius = .....
        return 0;               //              area = .....
}                               //         perimeter = .....
```

Implementation file for class circle

```
//File : circle.cpp
#include "circle"
#include <iostream>
using namespace std;
circle :: circle()
{
        radius = 0.0;
        area = 0.0;
        perimeter = 0.0;
}
```

```
void circle :: setRadius(float R)
{
        radius = R;
}
void circle :: ComputeArea()
{
        area = 22/7*radius*radius;
}
void circle :: ComputePerimeter()
{
        perimeter = 2*22/7*radius;
}
void circle :: displayCircle() const
{
        cout <<"radius = "<<radius<<endl;
        cout <<"area = "<<area<<endl;
        cout <<"perimeter = "<<perimeter<<endl;
}
float circle :: getRadius() const
{
        return radius;
}
float circle :: getArea() const
{
        return area;
}
```

```
float circle :: getPerimeter() const
{
        return perimeter;
}
```

กรณีศึกษา 4 : การสร้าง class simpleString

เป็นการสร้าง class simpleString เพื่อเลียนแบบการทำงานของ class string โดยระบุรายละเอียดของ class simpleString ดังนี้

Attributes for simpleString Class

capacity	Constant 255
contents	string data
length	string length

Member Functions for Class simpleString

simpleString	Constructor
readString	Reads the string data
writeString	Writes the string data
at	Retrieves a character at a specified position
getLength	Returns the string length
getCapacity	Returns the string capacity

เราสามารถนำรายละเอียดที่กำหนดมาสร้างเป็นต้นแบบของคลาสได้ดังนี้

```
// Class definition for simplestring
// File simpleString.h
// Simple string class definition
# ifndef SIMPLESTRING_H
```

```
# define SIMPLESSTRING_H
class simpleString
{
    public:
        // Member Functions
        // constructor
        simpleString( );

        // Read a simple string
        void readString( );

        // Display a simple string
        void writestring( ) const;

        // Retrieve the character at a specifled position
        // Returns the character \0 if position is out of bounds
        char at (int) const;

        // Return the string length
        int getLength( ) const;

        // Return the string capacity
        int getCcpacity ( ) const;

        // Return the string capacity
        int getCintents (char []) const;
```

```
    private:
        // Data members (attributes)
        enum {capacity = 255};      // Capacity of a string
        char contents [capacity];   // string data
        int length;                 // string length
};
# endif // SIMPLESTRING_H
```

ผู้ใช้สามารถเขียนโปรแกรมเพื่อเรียกใช้คลาส simpleString **ได้ดังนี้**

```
// Testing class simpleString
// File: simpleStringTest.cpp
// Tests the simple string class

# include "simpleString.h"
# inglude <iostream>
using mamespace std;

int main( )
{
    simpleString aString;   // input – data string

    // Read in a string.
    cout  << "Enter a string and press RETURN: ";
    Astring.readString( );

    // Display the string just read.
```

```
    cout  << "The string read was: ";
    aString.writeString( );
    cout  << endl;

    // Display each character on a separate line.
    cout << "The characters in the string follow:" << endl;
    for ( int pos = 0; pos < aString.getLength( ); pos++)
        cout << aString.at (pos) << endl;

    return 0;
}
```

เมื่อนำโปรแกรมนี้ไปทำงานได้ผลลัพธ์ดังนี้

```
Enter a string and press RETURN: Philly cheesesteak
The string read was : Philly cheesesteak
The characters in the string follow:
P
H
I
L
L
Y

C
H
E
E
```

S

E

S

T

E

A

K

ผู้ใช้ต้องสร้างฟังก์ชันเพื่อให้สามารถทำงานได้ตามที่ต้องการ โดยในที่นี้สร้างเป็นแฟ้มที่มีชื่อเดียวกับชื่อของคลาส แต่มีนามสกุลเป็น cpp โดยภายในแฟ้มนี้จะบรรจุ Member function ทั้งหมดที่สามารถกระทำได้ในคลาสนี้

```
// File simpleString.cpp
// Simple String class implementation
#include "simpleString.h"
#include<iostream>
using namespace std;

//  Member Functions ...
// constructor
simpleString :: simpleString ()
{
        length = 0;      // empty string
}

// Read a simple string
void simpleString : : readString ()
{
```

```
    // Local data ...
    char next;          // input – next data character
    int pos = 10 ;      // subscript for array contents
    cin.get(next) ;     // Get first character from cin
    while ( (next != '\n')  && (pos < capacity) )
    {
    // Insert next character in array contents
    contents[pos} = next ;
    pos++ ;
    cin.get(next) ;     // Get next character from cin
    }
    length = pos ;      // Define length attribute
}

// Write a simple string
void simpleString : : writeString () const
{
    for  ( int pos = 0 ;  pos < length ; pos++)
        count  <<  contents[pos] ;
}

// Retrieve the character at a specified position
// Returns the character \0 if position is out of bounds
char simpleString : : at (int pos) const  // IN: position character to get
{
    // Local data
    const char NULL_CHARACTER = ' \0 ' ;
```

```
    if ( ( pos < 0 )   (pos >= length) )
    {
        cerr  << "Character at position " << pos
                << " not defind." << end  ;
      return NULL_CHARACTER ;
      }
      else
        return contents[pos];
}
// Return the string length
int simpleString : : getLength ()  const
{
      return length;
}

// Rerturn the string capacity
int simpleString : : getCapacity () const
{
      return capacity ;
}

// Return the string contents
void simpleString : : getContents (char str[] ) const
{
      for (int i = o; i < length; i ++)
            str[i] = contents[i];
}
```

กรณีศึกษา 5 : การสร้าง class saving

เป็นการสร้าง class saving เพื่อจำลองการฝาก-ถอนเงิน , คิดดอกเบี้ย ของลูกค้าธนาคารแห่งหนึ่งโดยกำหนด Attribute ประกอบด้วย ชื่อบัญชี , เลขที่บัญชี , ยอดคงเหลือ และอัตราดอกเบี้ย ในส่วนของ Member function จะประกอบด้วยฟังก์ชันในการเปิดบัญชี , ฝากเงิน , ถอนเงิน , การคิดดอกเบี้ย และอื่นๆ ดังระบุเป็นรายละเอียดดังนี้

Specification for Savings Account class

Attributes for Saving Account Class

name	The account holder's name
id	The annual interest rate (%)
balance	The account balance

Member Functions for Savings Account Class

savings	A constructor
openAccount	Opens an account
changeName	Changes the account name
addInterest	Adds quarterly interest
deposit	Processes a deposit
withdraw	Processes a withdrawal
closeAccount	Close the account
getBalance	Gets the account balance

Private Member Function for Saving Account Class

validId	Validates the account identification before performing an operation

จากการระบุรายละเอียดของคลาสข้างต้น เราสามารถสร้างต้นแบบของคลาสนี้ ในแฟ้ม savings.h ดังนี้

```
// File savings.h
// Savings account class definition

#include "money.h"    // access money class
#include <string>     // access string class
using namespace std ;

#ifndef SAVINGS_H
#defind SAVINGS_H

class savings
{
        public :
            // Member Functions...
             // constructor
              savings () ;

             // Open a new account
             void  openAccount () ;

             // Change account name
             void changeName (int,  string) ;
```

```
        // Add quarterly interetest
        void addInterest () ;

        // Process a deposit
        void  deposit (int, money) ;

        // Process a withdrawal
        void withdrawal (int, money) ;
        // Close an account
        void closeAccount (int) ;

        // Get account balance
        money getBalance ()  const ;

private :
        // Data menbers (attributes)
        int id ;
        string name ;
        money balance ;
        float interestRate;

    // Member functions ...
    // Validate user identification
        bool validId (int) const;
};
#endif // SAVING_H
```

สำหรับการทำงานของฟังก์ชันในคลาสนี้จะถูกจัดเก็บในแฟ้มที่มีนามสกุลเป็น cpp ดังนี้

```
// File savings.cpp
// Savings account implementation file
#include "saving.h"
#include "money.h"
#include <string>
#include <iostream>
using namespace std ;
// Member Functions ...
// constructor
savings : : savings ()
{
        name = " " ;
        id = 0 ;
        balance = 0.0 ;
}
// Open a new account
void savings : : openAccou ()
{
        cout << " Account name : " ;
        getline ( cin, name, '\n' ) ;
        cout << " Accoun ID : " ;
        cin >> id ;
        cout << "Initial balance : $ " ;
        cin >> balance ;
        cout << "Annual interest rate percentage: % " ;
        cin >> interestRate ;
}
```

```
// Validate user id
bool savings :: validId (int ident)  const
{
        if  (id == ident )
            return true ;
        else
        {
              cerr << "Error – ID's do not match!  " ;
        return false ;
        }
}
// Change account name
void savings : : changeName ( int  ident,  string na )
{
        if ( validId (ident) )
        {
              name = na ;
              cout << "Changing account name to " << na << endl ;
        }
        else
              cerr << "Reject name change request." << endl ;
}
// Add quarterly interest
void saving : : addInterest ()
{
        // Local data
        float quarterRateFrac;                    // quarterly rate as a decimal fraction
```

```
        quarterRateFrac = interestRate / 4000.00 ;
        balance += balance * quarterRateFrac ;
}
// Process a deposit
void savings : : deposit ( int ident, money amount)
{
        if ( validId(ident) )
        {
        balance += amount ;
        cout << " Depositing " << amount << endl ;
        }
        else
        cerr << " Reject deposit of " << amount << endl;
}
//Process a withdrawal
void savings : : withdrawal ( int ident, money amount)
{
        if ( ( validId (ident) ) && (amou <= balance) )
        {
                balande -= amount ;
                cout << "Withdrawing " " << amount < endl ;
        }
        else
                cerr << "Reject withdrawal of " << amount << endl ;
}
// Close an account
void savings : : closeAccount ( int ident)
```

```
{
        if (validId(ident) )
        {
                cout  << " Final balance for account number  " << id
                        << " is "  << balance  <<  endl;
                cout << "Account has been closed "  <<  endl;
                balane = 0.0 ;
                id = 0 ;
                name = "" ;
        }
        else
                cerr << "Account not closed " << endl ;
}
// Get account balance
money savings : : getBalance () const
{
        return balance ;
}
```

เราสามารถทดสอบการทำงานของ class savings ได้ดังนี้

```
// File savingTest..cpp
// Tests the savings class

#include <iostream>
#include "savings.h"
#include  "money.h"
using namespace std ;
```

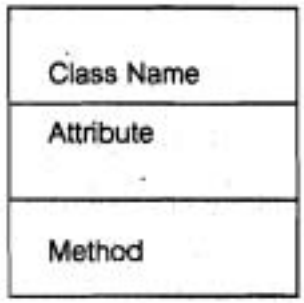
```
int main ()
{
        savings myAccount ;

        // Open a savings addount.
        myAccount.openAccount () ;
        cout << endl ;

        // Make valid and invalid deposit.
        myAccount.deposit (1234, 500.00) ;
        myAccount.deposit (1111,  300.00) ;

        // Get and display balance.
        cout  <<  endl << "Current balance is "
              <<  myAccount.getBalance ()  <<  endl ;

        // Make valid and invalid withdrawal.
        myAccount.withdraw (1234, 750.00) ;
        myAccount.withdraw (1234, 15000.00) :

        // Add interest.
        myAccount.addInterest () ;

        // Close the account.
        myAccount.closeAccount (1234) ;
        return 0 ;
}
```

สรุป

1. แนวความคิดเชิงวัตถุ (Object-Oriented: O-O) เป็นการมองสิ่งต่างๆในโลกแห่งความเป็นจริงในลักษณะของรูปธรรม โดยมองระบบเป็นกลุ่มของวัตถุ ที่มีปฏิกิริยาต่อกันด้วยการนำข้อมูลและฟังก์ชันการทำงานรวมเข้ากันเป็นวัตถุ ซึ่งวัตถุสามารถอธิบายคุณสมบัติ รวมทั้งฟังก์ชันการทำงานในตัวเองได้ การติดต่อกันระหว่างวัตถุจะต้องติดต่อผ่านฟังก์ชันที่กำหนดไว้ให้เท่านั้น
2. ออบเจ็กต์(object) หมายถึงวัตถุที่อยู่ในโลกของความเป็นจริง ซึ่งมีทั้งจับต้องได้และไม่ได้ เช่น รถยนต์ เก้าอี้ คอมพิวเตอร์ บริษัท ลูกค้า พนักงาน คลังสินค้า เป็นต้น
3. Reletionship คือความสัมพันธ์ระหว่างออบเจ็กต์ เช่น นายสมชาย เป็นเจ้าของรถยนต์
4. Interaction คือการปฏิสัมพันธ์หรือการกระทำที่เกิดขึ้นระหว่างออบเจ็กต์ เช่น นายสมชายขับรถยนต์ หรือ นายสมชายซ่อมรถยนต์
5. Concept คือแนวความคิดที่ให้กับวัตถุ ภายใต้กรอกที่กำหนด(Domain) เช่น แนวความคิดของรถยนต์ คือรถทุกคันต้องมีตัวถัง มีล้อ มีเครื่องยนต์ เหมือนกันทุกคัน ออบเจ็กทุกตัวจะต้องอยู่ในคลาส(Class) โดยกลุ่มของออบเจ็กต์ที่มีโครงสร้างพื้นฐานพฤติกรรมเดียวกันหรือมีคุณลักษณะเหมือนกันจะรวมกลุ่มอยู่ในคลาสเดียวกัน
6. คลาส(class)คือต้นแบบข้อมูลที่มีไว้เพื่อสร้างออบเจ็กต์นั่นเอง
7. Abstraction คือกระบวนการในการให้แนวความคิดกับออบเจ็กต์ จนเกิดเป็นคลาส
8. Instance คือ ออบเจ็กต์ที่ถูกสร้างขึ้นมาจากคลาส
9. Attribute คือคุณสมบัติต่างๆของออบเจ็กต์ ซึ่งอยู่ภายในกรอบที่เรากำหนด
10. Function คือ พฤติกรรมหรือความสามารถในการทำกิจกรรมของออบเจ็กต์
11. Object ทุกตัวต้องอยู่ใน class ใด class หนึ่ง และเราจะทราบคุณสมบัติรายละเอียดต่างๆ ของ Object ดูได้จากคลาส ประกอบไปด้วย ชื่อคลาส (Name class) ,

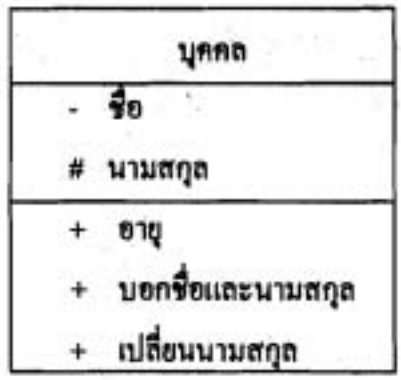
คุณลักษณะ (Attribute) ของข้อมูล , กรรมวิธี (Method) หรือการกระทำ (Operation) และฟังก์ชั่น(Function)

แผนภาพของคลาส



12. ประเภทของ Attribute และ Function สามารถแบ่งออกเป็น 3 ประเภทดังนี้

- Private เป็นประเภทที่ปกปิดไว้เป็นส่วนตัว ไม่สามารถมองเห็นได้ การเข้าถึง Attribute ต้องกระทำผ่าน Function เสมอ โดยทั่วไปเราใช้เครื่องหมาย – กำกับไว้หน้า Attribute หรือ Function ของคลาสนั้นๆ
- Protected เป็นประเภทที่ไม่สามารถเห็นได้จากภายนอกเช่นกัน แต่ประเภทนี้สามารถส่งต่อให้ Inherited class ได้ เราใช้เครื่องหมาย # กำกับไว้หน้า Attribute หรือ Function ของคลาสนั้นๆ
- Public เป็นประเภทที่สามารถมองเห็นได้จากภายนอก เราใช้เครื่องหมาย + กำกับไว้หน้า Attribute หรือ Function นั้นๆ



13. Inheritance คือการถ่ายทอดคุณสมบัติจากคลาสที่มีอยู่แล้วไปยังคลาสใหม่ ซึ่งทำให้สามารถนำออบเจ็กต์ที่ผู้อื่นออกแบบไว้แล้วนำกลับมาใช้ใหม่ได้(Reusable) และการถ่ายทอดที่เป็นลำดับขั้นทำให้ความสัมพันธ์ระหว่างออบเจ็กต์มีความชัดเจนมากยิ่งขึ้น แนวความคิดเชิงวัตถุถือว่าการสืบทอดเป็นสิ่งที่สำคัญ เพราะว่าไม่มีสิ่งใดในโลกที่เกิดขึ้นเอง

14. กระบวนการสร้างคลาส สามารถแบ่งขั้นตอนการสร้างออกได้ 4 กระบวนการดังนี้
    - Classification Abstraction เป็นกระบวนการที่พิจารณาโดยแยกประเภท (classify) ของออบเจ็กต์ต่างๆที่อยู่ในกรอบหรือขอบเขตที่กำหนด เพื่อให้ได้คลาสที่ต้องการ
    - Aggregation Abstraction เป็นกระบวนการที่นำคลาสพื้นฐานที่สร้างขึ้นมาพิจารณาเพื่อรวมกันหรือประกอบกันให้เกิดเป็นคลาสใหม่มีแนวความคิดใหม่
    - Generalization Abstraction เป็นกระบวนการพิจารณาคลาสที่มีลักษณะที่คล้ายคลึงกันหรือมีคุณลักษณะอย่างใดอย่างหนึ่งเหมือนกัน นำออกมาจัดเป็นหมวดหมู่เป็นคลาสที่เป็นสามัญ(General) และเมื่อเพิ่มคุณลักษณะพิเศษ (Specialization) ให้กับคลาสสามัญ จะทำให้ได้คลาสใหม่ ที่เราเรียกว่า คลาสย่อย(Subclass) โดยถ่ายทอดคุณลักษณะของคลาสสามัญมายังคลาสย่อยนั่นเองเราเรียกการที่ คลาสย่อยรับคุณสมบัติทุกอย่างมาจาก Superclass ว่า Inheritance
    - Association Abstraction เป็นกระบวนการสร้างความสัมพันธ์ระหว่างคลาสต่างๆในระบบที่เราสนใจ

15. Polymorphism หมายถึงการเรียกใช้ในชื่อฟังก์ชันเดียวกันแต่สามารถตอบสนองการทำงานได้หลายรูปแบบ ทำให้ลดคำสั่งในโปรแกรมได้มาก

16. **รูปแบบของการกำหนด class**

Class Definition

```
Form:  class className
       {
        public:
          - List of class attributes(variables, types, constants, and so
            on that may be accessed by name from outside the class
          - List of prototype for each member function that may be
            accessed by name from  outside the class
          ...
        private:
          - List of class attributes(variables, types, constants, and so
            on that intended to be hidden for reference from outside the class
          - List of prototype for each member function intended to be
            hidden  from  outside  of the class
          ...
       }; // Don't forget the semicolon
```

17. **รูปแบบการกำหนดฟังก์ชันในคลาส**

Class Member Function Definition

```
Form:  type className: : fname
       (formal parameter list)
{ .....
  function body
  . . .
}
```

18. การสร้างคลาสนั้นเราสามารถสร้างแฟ้มที่ชื่อเดียวกับชื่อของคลาสแต่มีนามสกุลเป็น .h ซึ่งหมายความว่าแฟ้มที่สร้างเป็น header file จะบรรจุเพียงนิยามของ class ทำให้ทราบคุณลักษณะของ class เท่านั้น ส่วนการทำงานหรือฟังก์ชันในการปฏิบัติงานนั้นเราจะแยกสร้างในแฟ้มที่ชื่อเดียวกับชื่อของคลาสแต่มีนามสกุลเป็น .cpp
19. Class Implementation จะบรรจุคำสั่ง C++ สำหรับ member function เพื่อซ่อน จากผู้ใช้เพื่อไม่ให้ทราบรายละเอียด โดยเพื่อเน้นผลลัพธ์มากกว่าการปฏิบัติงานจริง จะมีความแตกต่าง
จากฟังก์ชั่นในโปรแกรมตรงที่จะใช้ ตัวกระทำ :: เรียกว่า scope resolution operator เพื่อบอก compiler ให้ทราบว่าเป็นการกำหนดฟังก์ชั่นของ class ที่เราสนใจ
20. Member function ที่เป็น Constructor จะเป็นฟังก์ชันที่ชื่อเดียวกับชื่อของคลาส โดยจะมีการกำหนดค่าของ Attribute เริ่มต้น โดยอัตโนมัติให้แก่ออบเจ็กต์เมื่อผู้ใช้มีการประกาศออบเจ็กต์เพื่อใช้งานขึ้นใหม่ในโปรแกรม
21. Member function ที่เป็น Modifier จะเป็นฟังก์ชันที่เมื่อมีการเรียกใช้ฟังก์ชัน การปฏิบัติงานภายในฟังก์ชันจะส่งผลต่อการเปลี่ยนแปลงค่าของ Attribute หรือกล่าวง่ายๆว่าค่าต่างๆถูกปรับเปลี่ยนไป
22. Member function ที่เป็น Accessor จะเป็นฟังก์ชันที่เมื่อมีการเรียกใช้ฟังก์ชันนั้นๆ การทำงานของฟังก์ชันจะไม่มีผลกระทบต่อค่าที่เก็บอยู่ของออบเจ็กต์นั้นๆ เราสามารถสังเกตได้จากการกำหนดคำ const ไว้หลังสุดของต้นแบบฟังก์ชัน ดังตัวอย่าง

```
int  getCount() const ;
```

แบบฝึกหัด

1. แนวความคิดเชิงวัตถุ (Object Oriented Programming) แตกต่างจากแนวความคิดแบบ Modular อย่างไร จงอธิบายพอเข้าใจ
2. จงอธิบายถึงความหมายของคำศัพท์ต่อไปนี้พอเข้าใจ
   a. Inheritance
   b. Polymorphism
   c. Encapsulation
3. กระบวนการสร้างคลาสมีกี่ขั้นตอนอะไรบ้าง จงอธิบายพอเข้าใจ
4. Attribute และ Function ที่ผู้พัฒนาสามารถกำหนดขอบเขตของการทำงานแบ่งเป็นกี่ประเภทอะไรบ้าง จงอธิบายให้เข้าใจ
5. จงเขียนโปรแกรมภาษา C++ โดยเขียนแบบ OOP

   ปัญหา ต้องการพัฒนาโปรแกรมเพื่อให้สามารถเก็บประวัติของลูกค้าที่ซื้อคอนโดฝึกข้าวโพด ของนายสมัคร สุนทรเวช โดยข้อมูลของลูกค้าที่ต้องการเก็บประกอบด้วย ขนาดของห้อง และ ราคา โดยกำหนดให้ขนาดของห้องมี 3 ขนาด คือ40 ตารางเมตร ราคา 500,000 บาท, ขนาด 50 ตารางเมตร ราคา 800,000 บาท และ 60 ตารางเมตร ราคา 1,000,000 บาท กำหนด Attribute และ Mathod ของ Class ดังนี้

Condo	/ ชื่อ Class
int Size;	/ ขนาดของห้อง
float Cost ;	/ ราคาของห้องพัก
void Condo(int);	/ Constructor โดยกำหนดขนาดของห้องพักตามค่าที่ส่งมา
void FindCost();	/ เป็นฟังก์ชันที่นำขนาดของห้องพักมาคิดราคาของห้องพัก
float getCost() const	/ เป็นฟังก์ชันในการส่งผ่านค่าราคาห้องพักกลับไปยังโปรแกรมที่ เรียกใช้

โปรแกรมภายนอกสามารถทำงานโดยเรียกใช้ Class ที่ชื่อ Condo.h นี้ได้

ตัวอย่างการเรียกใช้

```
Condo  Ural(50) ;      / เป็นการกำหนดObject ที่ชื่อว่า Ural โดยส่ง
                       / ผ่านค่าขนาดของห้อง 50 ไปยัง Class Condo
Condo  Ty(60);         / เป็นการกำหนดObject ที่ชื่อว่า Ty โดยส่ง
                       / ผ่านค่าขนาดของห้อง 60 ไปยัง Class Condo
Ural.FindCost();       / หาราคาของห้อง เพื่อเก็บใน Object Ural
cout << Ural.getCost() << endl ; / ส่งผ่านราคาของห้องมายังภายนอก
Ty.FindCost();         / หาราคาของห้อง เพื่อเก็บใน Object Ural
Cout << Ty.getCost() << endl ;  / ส่งผ่านราคาของห้องมายังภายนอก
```

คำถาม

จงสร้าง Class ที่ชื่อว่า Condo เพื่อให้สามารถทำงานได้ตามที่กำหนด โดยสามารถออกแบบให้อยู่ในแฟ้มที่ชื่อ Condo.h เพียงแฟ้มเดียว หรือ จะแยกแฟ้มเป็น 2 แฟ้มประกอบด้วย Condo.h และ Condo.cpp ก็ได้

6. จงอธิบายถึงฟังก์ชันที่เป็น Constructor , Accessor และ Modifier ว่าเหมือนหรือแตกต่างกันอย่างไร จงอธิบายให้เข้าใจ
7. รายการ ' อาสาพาไปเที่ยว ' เป็นรายการหนึ่งทางทีวีที่มีการแจกรางวัลให้กับผู้ชมที่ชมรายการ โดยในแต่ละครั้งที่มีการออกอากาศได้มีช่วงเวลาเปิดโอกาสให้ผู้ชมสามารถโทรศัพท์เข้าไปในรายการเพื่อชิงรางวัล ทางรายการจะเก็บบันทึกประวัติของผู้ที่โทรศัพท์เข้ามาเป็นลำดับต่อจากนั้นจะถามคำถามให้ผู้โทรศัพท์เข้ามาตอบ และเก็บประวัติว่าตอบถูกหรือไม่ถูก เพื่อดำเนินการส่งของรางวัลให้แก่ผู้ชมต่อไป

   7.1 จงสร้างClass ดังนี้

```
Name        customer
Attribute   string  Name        // ชื่อผู้ชมรายการ
            string  Tel         // โทรศัพท์ผู้เข้าชมรายการ
            bool    question    // ตอบคำถามผิดหรือถูก
```

Operation

customer()	//constructor กำหนดค่าเริ่มต้น
readData()	//รับชื่อ และเบอร์โทรศัพท์ทางแป้นพิมพ์
setQuestion(bool)	//กำหนดค่าให้กับ question ว่าผู้ชมรายการตอบผิดหรือถูก
getName()	//ส่งกลับชื่อมายังจุดเรียกใช้
getTel()	//ส่งกลับโทรศัพท์มายังจุดเรียกใช้
getQuestion()	//ส่งกลับตอบมายังจุดเรียกใช้

7.2 ให้นักศึกษาเขียนโปรแกรมภาษา C++ เพื่อเรียกใช้คลาส customer โดยให้โปรแกรมสามารถเรียกใช้ฟังก์ชันต่างๆที่อยู่ภายในคลาสนี้ได้

8. จงเขียนโปรแกรมโดยแก้ปัญหาโดยใช้แนวความคิดเชิงวัตถุโดยกำหนดคลาสชื่อ student กำหนดให้มีฟังก์ชันต่างๆที่สามารถทำงานในคลาสดังนี้
   - ฟังก์ชัน InputData เป็นฟังก์ชันในการรับรหัสนักศึกษา(id), ชื่อ-นามสกุล(name) , คะแนนสอบปฏิบัติ(score1) , และคะแนนสอบทฤษฎี(score2)
   - ฟังก์ชัน ComputeSum เป็นฟังก์ชันในการนำคะแนนสอบทั้งทฤษฎีและปฏิบัติรวมกันเป็นผลรวมของคะแนนทั้งหมด
   - ฟังก์ชัน ComputeGrade เป็นฟังก์ชันในการนำคะแนนรวม(sum) ไปหาว่าได้เกรดเป็นเท่าใด P หรือ G หรือ F
   - ฟังก์ชัน GetData เป็นฟังก์ชันในการนำข้อมูลทั้งหมดของออบเจ็กต์ที่กำหนดพิมพ์ค่าต่างๆออกมาทางจอภาพ

9. กำหนดรายละเอียดของคลาสดังต่อไปนี้

Name : computeArea

Attributes for computeMul Class :

```
int   a       // ค่าของข้อมูลตัวแรก
int   b       // ค่าของข้อมูลตัวที่สอง
int   area    // ผลคูณของค่า a และ b นำมาหาร 2 เป็นพื้นที่ของสามเหลี่ยม
```

Member Functions for computeMul class :

SetA	// กำหนดค่าของ a
SetB	// กำหนดค่าของ b
AreaAB	// นำ (a * b)/2 เก็บผลลัพธ์ในค่าของ area
Getarea	// ส่งค่าของ area กลับ

คำสั่ง จงเขียนโปรแกรม เพื่อกำหนดคุณสมบัติของคลาสนี้ โดยกำหนดให้ attributeทั้งหมดซ่อนไว้ ส่วน function ทั้งหมดผู้ใช้ภายนอกสามารถเรียกใช้ได้ ให้ส่วนของโปรแกรมหลักมีการกำหนดอ๊อปเจกต์ชื่อ A เป็นสมาชิกของคลาส computeArea โดยมีการเรียกใช้ฟังก์ชันทั้งหมดของคลาสนี้เพื่อทำงานได้