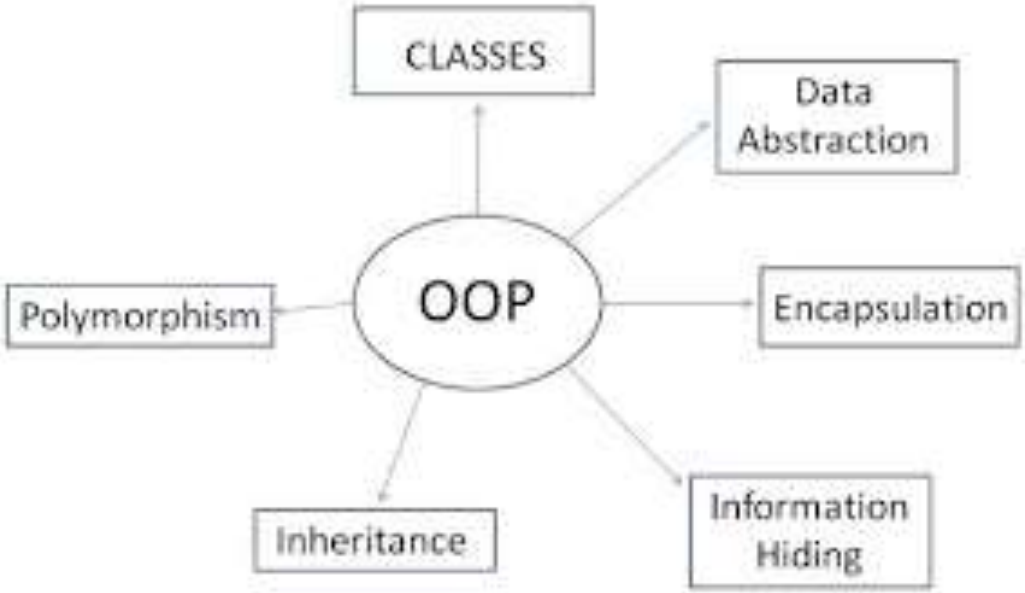
หน่วยที่ 5

การสืบทอดคุณสมบัติ และการพ้องรูป (Inheritance and Polymorphism)



หัวข้อเรื่อง

5.1 การสืบทอดคุณสมบัติ (Inheritance)

5.2 การสืบทอดคุณสมบัติคอนสตรัคเตอร์ (Constructor inherited)

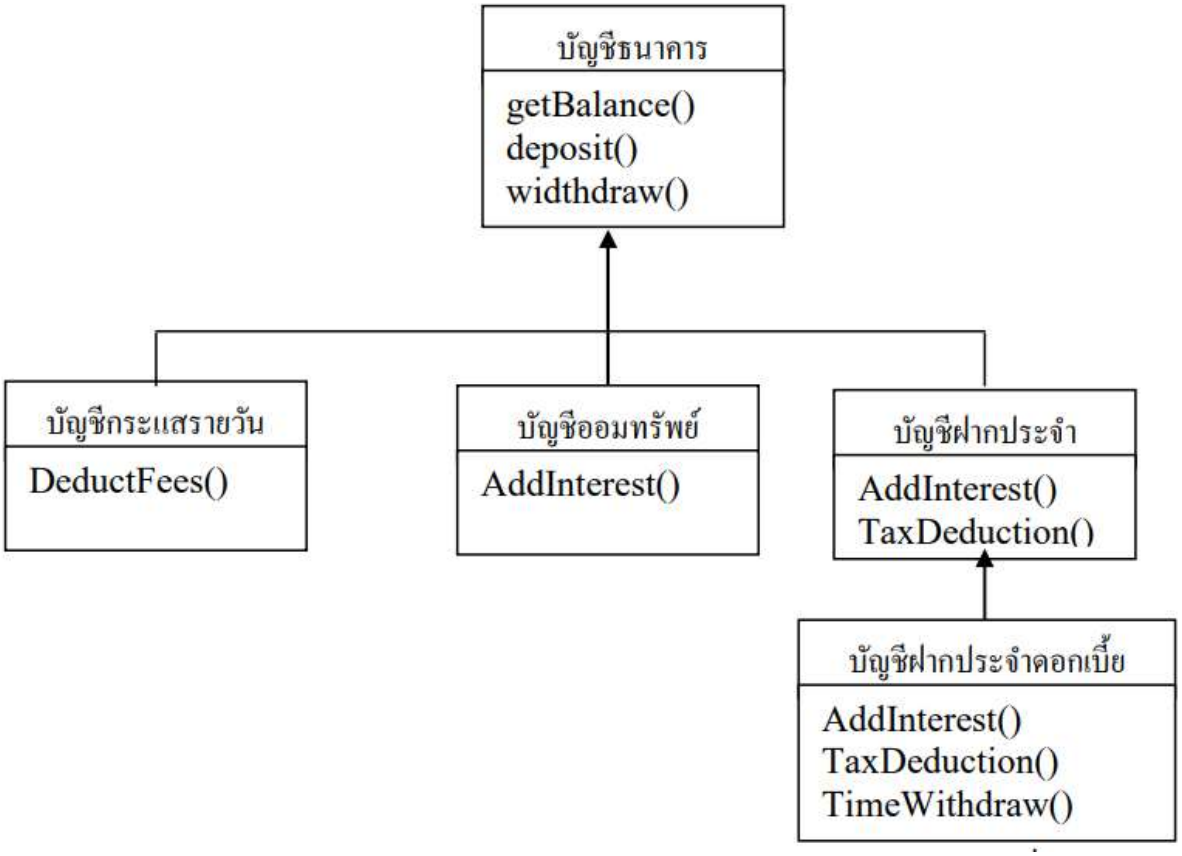
5.3 การสืบทอดคุณสมบัติแบบมัลติเบิ้ล (Multiple inheritance)

5.4 การพ้องรูป (Polymorphism)

ความสัมพันธ์ระหว่างอ็อบเจ็กต์ต่าง ๆ ในหลักของการเขียนโปรแกรมเชิงวัตถุคือ การสืบทอดคุณสมบัติ ทั้งยังมองความสัมพันธ์ระหว่างอ็อบเจ็กต์ต่าง ๆ ในรูปแบบที่หลากหลายที่เรียกว่า การพ้องรูป

5.1 การสืบทอดคุณสมบัติ (Inheritance)

หลักการ “Inheritance” คือการที่ “Class” ใด ๆ ยอมให้การสืบทอด (Inherit) ทั้งคุณสมบัติ (Data member) และการทำงาน (Member function) ไปยัง “Class” อื่น การสืบทอดเป็นกระบวนการในการนำคุณลักษณะ (Data member) ของ class มาใช้ใหม่ (Reuse) โดยการสืบทอดจาก Class หนึ่ง ไปยังอีก class หนึ่ง เช่น class A ซึ่งเรียกว่า Base Class หรือ Class ต้นแบบ สืบทอดไปยัง class B ซึ่งเรียกว่า Derived class นั้นหมายถึงว่า class B จะใช้คุณลักษณะทั่วไปของ class A ร่วมกัน ขณะเดียวกัน class B อาจมีคุณสมบัติเฉพาะตัวที่ class A ไม่มี การสืบทอดสามารถช่วยลดขนาด ของโปรแกรมและเวลาที่ใช้ในการพัฒนาโปรแกรม เนื่องมาจากการใช้ข้อมูลร่วมกัน ตัวอย่างเช่น Class บัญชีธนาคาร ซึ่งเป็น Base Class โดยคุณลักษณะของ Class บัญชีธนาคาร คือ ยอดคงเหลือ, ฝาก และถอน หากต้องการแยกบัญชีออกเป็นบัญชีประเภทต่าง ซึ่งก็คือการแยกออกเป็น Derived class เช่น บัญชี=กระแสรายวัน บัญชีออมทรัพย์ และบัญชีเงินฝากประจำ โดยบัญชีเงินฝากประจำยังสามารถออกเป็น Derived class ย่อยได้อีกเช่น ฝากประจำดอกเบี้ยพิเศษ เป็นต้น โดย Derived class สามารถเรียกใช้งานคุณลักษณะของ Base Class ได้ทุกประการ แต่ Base Class ไม่สามารถเรียกใช้งานคุณลักษณะพิเศษของ Derived class ได้ ดังรูป แสดงตัวอย่างการ Inheritance ระหว่าง Class



โดยกฎเกณฑ์ของการทำ Inheritance ดังนี้

1. Private Data member/Member function ของ Base class จะถ่ายทอดมาเป็น Private Data member/Member function ของ Derived class ในสวนที่สืบทอด มาจาก Base class มายัง Derived class จะไม่สามารถเข้าถึงได้โดย Function ที่มีอยู่ใน Derived class

2. Protected Data member/Member function ของ Base class จะถ่ายทอดมาเป็น Protected Data member/Member function ของ Derived class อนึ่ง การเข้าถึง Data member และ Member function Base class ของ Derived class ที่เกิดจากการสืบทอดจะทำได้โดยผ่าน Memberfunction ใด ๆ ของ Derived class โดยไม่คำนึงว่าจะเป็น Member function ที่ได้มาจากการสืบทอดหรือไม่

3. Public Data member/Member function จะถ่ายทอดมาเป็น Public Data member/ Member function ของ Derived class เสมอ

รูปแบบการสืบทอดคุณสมบัติ

รูปแบบทั่วไปสำหรับการสร้างคลาส y จากคลาส x คือ

```
class Y:public X {
        //code
};
```

ในที่นี้ X ได้รับการเรียกว่า คลาสต้นแบบ (Base Class) และ Y ถูกเรียกว่า คลาสสืบทอด (Derived Class) หรือคลาสย่อย มีเครื่องหมาย : แยกคลาสต้นแบบกับคลาสสืบทอด มีคำ public หน้าหลายต้นแบบ ระบุว่าเป็นการถ่ายทอดคุณสมบัติ public ซึ่งหมายถึง สมาชิก public ของคลาสต้นแบบกลายเป็นสมาชิก public ของกลาสสืบทอดด้วย

ตัวอย่างที่ 5.1 โปรแกรมคำนวณหาพื้นที่ของรูปเหลี่ยม

```
1    #include <iostream>
2    using namespace std;
3    class CPolygon {
4    protected:
5    int width, height;
6    public:
7    void set_values (int a, int b)
8    { width=a; height=b;}
9    };
10   class CRectangle: public CPolygon {
11   public:
12   int area ()
```

```
13    { return (width * height); }
14    };
15    class CTriangle: public CPolygon {
16    public:
17    int area ()
18    { return (width * height / 2); }
19    };
20    int main () {
21    CRectangle rect;
22    CTriangle trgl;
23    rect.set_values (4,5);
24    trgl.set_values (4,5);
25    cout << rect.area() << endl;
26    cout << trgl.area() << endl;
27    return 0;
28    }
```

ผลลัพธ์

```
20
10
```

อธิบายการทำงานของโปรแกรม

บรรทัดที่	คำอธิบาย
10	คลาสชื่อ CRectangle ได้รับการสืบทอดจากคลาส CPolygon

12	ฟังก์ชัน area อยู่ในคลาส CRectangle ใช้คืนค่าพื้นที่ให้ เมื่อมีการเรียกใช้ฟังก์ชัน
15	คลาสชื่อ CTriangle ได้รับการสืบทอดจากคลาส CPolygon
17	ฟังก์ชัน area อยู่ในคลาส CTriangle ใช้คืนค่าพื้นที่ให้ เมื่อมีการเรียกใช้ฟังก์ชัน
21	สร้างอ็อบเจ็กต์ rect จากคลาส CRectangle
22	สร้างอ็อบเจ็กต์ trgl จากคลาส CRectangle
23	อ็อบเจ็กต์ rect เรียกใช้ฟังก์ชัน set_values ในคลาส CPolygon
24	อ็อบเจ็กต์ trgl เรียกใช้ฟังก์ชัน set_values ในคลาส CPolygon
25	สร้างอ็อบเจ็กต์ rect เรียกใช้ฟังก์ชัน area เพื่อแสดงผล ในคลาส CRectangle
26	สร้างอ็อบเจ็กต์ trgl เรียกใช้ฟังก์ชัน area เพื่อแสดงผล ในคลาส CTriangle

5.2 การสืบทอดคุณสมบัติคอนสตรัคเตอร์ (Constructor inherited)

Constructor ของ Base class จะถูกสืบทอดไปยัง Derived class ด้วย และจะถูกเรียกใช้ เมื่อ class ของ Derived class ทำการสร้างอ็อบเจ็กต์ใหม่ กรณี constructor ของ base class มีพารามิเตอร์ นั้นจะไม่ถูกสืบทอดไปยัง Derived class ถ้าต้องการให้สืบทอด ต้องกำหนดใน Derived class ในส่วนของ constructor ดังนี้

derived_constructor_name (parameters) : base_constructor_name (parameters) {...}

ตัวอย่างที่ 5.2 การสืบทอดคุณสมบัติคอนสตรัคเตอร์

```
1    #include <iostream>
2    using namespace std;
3    class mother {
4    public:
5    mother ()
6    { cout << "mother: no parameters\n"; }
7    mother (int a)
```

```
8    { cout << "mother: int parameter\n"; }
9    };
10   class daughter : public mother {
11   public:
12   daughter (int a)
13   { cout << "daughter: int parameter\n\n"; }
14   };
15   class son : public mother {
16   public:
17   son (int a) : mother (a)
18   { cout << "son: int parameter\n\n"; }
19   };
20   int main () {
21   daughter cynthia (0);
22   son daniel(0);
23   return 0;
24   }
```

ผลลัพธ์

```
mother: no parameters
daughter: int parameter

mother: int parameter
```

son: int parameter

อธิบายการทำงานของโปรแกรม

บรรทัดที่	คำอธิบาย
10	คลาสชื่อ daughter ได้รับการสืบทอดจากคลาส mother
12	daughter เป็นฟังก์ชันคอนสตรัคเตอร์อยู่ในคลาส daughter ต้องการพารามิเตอร์
15	คลาสชื่อ son ได้รับการสืบทอดจากคลาส mother
17	son เป็นฟังก์ชันคอนสตรัคเตอร์อยู่ในคลาส daughter ต้องการพารามิเตอร์และได้รับ การสืบทอด ฟังก์ชันคอนสตรัคเตอร์จากคลาส mother ที่มีพารามิเตอร์ด้วย
21	สร้างอ็อบเจ็กต์ cynthia จากคลาส daughter พร้อมส่งค่าอากิวเมนต์
22	สร้างอ็อบเจ็กต์ daniel จากคลาส son พร้อมส่งค่าอากิวเมนต์

5.3 การสืบทอดคุณสมบัติแบบมัลติเบิ้ล (Multiple inheritance)

ในภาษา C++ ในการสืบทอดคุณสมบัติ สามารถที่ได้รับการสืบทอดมากกว่าหนึ่งคลาสได้โดยการใช้เครื่องหมายคอมม่า (,) เป็นตัวแยกคลาสที่ได้สืบทอดมา ดังตัวอย่าง

ตัวอย่างที่ 7.3 โปรแกรมคำนวณหาพื้นที่ของรูปเหลี่ยม

```
1    #include <iostream>
2    using namespace std;
3    class CPolygon {
4    protected:
5    int width, height;
6    public:
7    void set_values (int a, int b)
8    { width=a; height=b;}
9    };
10   class COutput {
11   public:
12   void output (int i);
13   };
14   void COutput::output (int i) {
15   cout << i << endl;
16   }
17   class CRectangle: public CPolygon, public COutput {
```

```
18  public:
19  int area ()
20  { return (width * height); }
21  };
22  class CTriangle: public CPolygon, public COutput {
23  public:
24  int area ()
25  { return (width * height / 2); }
26  };
27  int main () {
28  CRectangle rect;
29  CTriangle trgl;
30  rect.set_values (4,5);
31  trgl.set_values (4,5);
32  rect.output (rect.area());
33  trgl.output (trgl.area());
34  return 0;
35  }
```

ผลลัพธ์

```
20
10
```

อธิบายการทำงานของโปรแกรม

เป็นตัวอย่างที่ดัดแปลงจากตัวอย่างที่ 5.1 โดยมีการเพิ่มการสืบทอดในลักษณะมัลติเบิ้ล

บรรทัดที่	คำอธิบาย
17	คลาสชื่อ CRectangle ได้รับการสืบทอดจากคลาส CPolygon และ คลาส COutput
19	ฟังก์ชัน area อยู่ในคลาส CRectangle ใช้คืนค่าพื้นที่ให้ เมื่อมีการเรียกใช้ฟังก์ชัน
22	คลาสชื่อ CTriangle ได้รับการสืบทอดจากคลาส CPolygon และ คลาส COutput
24	ฟังก์ชัน area อยู่ในคลาส CTriangle ใช้คืนค่าพื้นที่ให้ เมื่อมีการเรียกใช้ฟังก์ชัน
28	สร้างอ็อบเจ็กต์ rect จากคลาส CRectangle
29	สร้างอ็อบเจ็กต์ trgl จากคลาส CTriangle
30	อ็อบเจ็กต์ rect เรียกใช้ฟังก์ชัน set values ในคลาส CPolygon
31	อ็อบเจ็กต์ trgl เรียกใช้ฟังก์ชัน set values ในคลาส CPolygon
32	สร้างอ็อบเจ็กต์ rect เรียกใช้ฟังก์ชัน output ในคลาส COutput โดยส่งค่าอากิวเมนต์ ที่ได้รับจากการที่อ็อบเจ็กต์ rect เรียกใช้ฟังก์ชัน 1 area เพื่อแสดงผล
33	สร้างอ็อบเจ็กต์ trgl เรียกใช้ฟังก์ชัน output ในคลาส COutput โดยส่งค่าอากิวเมนต์ ที่ได้รับจากการที่อ็อบเจ็กต์ rect เรียกใช้ฟังก์ชัน area เพื่อแสดงผล

5.4 การพ้องรูป (Polymorphism)

การพ้องรูป (Polymorphism) เป็นคุณสมบัติประการหนึ่งของการเขียนโปรแกรมเชิงวัตถุที่ เอื้ออำนวยประโยชน์ต่อการนำคลาสกลับมาใช้งานอีก เป็นกลไกสำคัญสำหรับการเขียนโปรแกรม เชิงวัตถุ ที่เป็นผลพวงของการสืบทอดคุณสมบัติ ในการทำความเข้าใจกับเนื้อหาการพ้องรูป จะต้อง รู้ในเรื่อง Pointers และการสืบทอดกันระหว่างคลาส โดย คำสั่งที่มักพบ คือ

```
int a::b(c) {};          //function of class
a->b                     //pointers and objects
class a: public b;       //inheritance between classes
```

ความหมายของ Polymorphism

Polymorphism หรือภาษาไทยเรียกว่า “การพ้องรูป” เป็นคำที่มาจากภาษากรีก “Polymorphic Poly หมายถึง Many (หลาย) Morphs หมายถึง Forms (รูปแบบ) ดังนั้นจึงหมายถึง การสืบทอดรูปแบบได้หลากหลาย ซึ่งถือว่าเป็นหลักการที่แยกจากกันไม่ออกกับคุณสมบัติการ สืบทอด (inheritance) โดยปกติแล้ว

ภายใต้โครงร่างลำดับชั้นของการสืบทอดคุณสมบัติ นั้น Derived Class ที่ได้รับการสืบทอดมาจาก Base Class ของตัวเอง และในขณะเดียวกัน Derived Class แต่ละตัวนั้นต่างก็มีความเป็นอิสระจากกัน

ส่วนการพ้องรูป (Polymorphism) คลาสที่เป็น Derived Class จะมีฟังก์ชันที่มีชื่อเดียวกับ Base class เช่น การออกแบบระบบรูปทรงเรขาคณิตขึ้นมาใช้งาน โดยมีการสร้างมาตรฐานในการ สร้างรูปทรงว่ารูปทรงที่สร้างขึ้นต้องมาจากฟังก์ชัน “Draw” ที่กำหนดไว้ใน Base class และเมื่อใด ก็ตามที่ต้องการวาดรูปทรงเรขาคณิตใน Derived Class ก็เพียงกำหนด ฟังก์ชัน “Draw” ใน Derived class โดยกำหนดความแตกต่างของรูปทรงที่ต้องการนั้นจะวาดรูปทรงที่ต้องการด้วยตัวเอง โดยอยู่ บนพื้นฐานความรับผิดชอบแต่ละคลาสใน Derived class ที่ถูกเรียกใช้ นี่เป็นหลักการ Polymorphism นั่นเอง โดยภาษา C++ จะใช้คำสงวน virtual วางไว้หน้า ฟังก์ชันใน Base class ที่มี ชื่อเดียวกับฟังก์ชัน ใน Derive class

ฟังก์ชันเสมือน (Virtual Function)

คือฟังก์ชันที่มีคำสงวน virtual ไว้หน้าฟังก์ชันใน Base class และจะได้รับการคืนค่าด้วย ตัวชี้ (pointer) ของ base class ที่ชี้ไปที่ฟังก์ชัน derived class ใด ๆ ซึ่งเรียกว่า Polymorphism หากมี การเรียกใช้ฟังก์ชันใน Derived class มันจะไปเรียกฟังก์ชันที่แท้จริง

ตัวอย่างที่ 5.4 การสืบทอดในลักษณะที่หลากหลายโดยใช้ Virtual Function

```
1    #include <iostream>
2    using namespace std;
3    class Animal {
4    public:
5    virtual void Say() {
6    cout << "I am animal"<<endl;
7    }
8    };
9    class Dog:public Animal {
10   public:
11   void Say() {
12   cout <<"Box Box"<<endl;
13   }
```

```
14    };
15    class Beagle:public Dog {
16    public:
17    void Say() {
18    cout <<"I am Beagle "<<endl;
19    }
20    };
21    int main() {
22    Animal *Jeejee;
23    Jeejee =new Animal;
24    Jeejee ->Say();
25    Jeejee =new Dog;
26    Jeejee ->Say();
27    Jeejee =new Beagle;
28    Jeejee ->Say();
29    return 0;
30    }
```

ผลลัพธ์

```
I am animal
Box Box
I am Beagle
```

อธิบายการทำงานของโปรแกรม

บรรทัดที่	คำอธิบาย
3	คลาสชื่อ Animal เป็น base class
5	ฟังก์ชัน Say อยู่ในคลาส Animal มีคำ virtual เพื่อใช้เป็นตัวชี้
9	คลาสชื่อ Dog ได้รับการสืบทอดจากคลาส Animal
11	ฟังก์ชัน Say อยู่ในคลาส Dog
15	คลาสชื่อ Beagle ได้รับการสืบทอดจากคลาส Dog

17	ฟังก์ชัน Say อยู่ในคลาส Dog
22	สร้างอ็อบเจ็กต์ Jeejee จากคลาส Animal เป็นพอยน์เตอร์
23	จองหน่วยความจำให้กับคลาส Animal โดยนำตำแหน่งหน่วยความจำไปเก็บไว้ตัวแปรพอยน์เตอร์ Jeejee
24	ใช้ Jeejee เรียกฟังก์ชัน say ในคลาส Animal โดยใช้เครื่องหมาย ->
25	จองหน่วยความจำให้กับคลาส Day โดยนำตำแหน่งหน่วยความจำไปเก็บไว้ตัวแปรพอยน์เตอร์ Jeejee
26	ใช้ Jeejee เรียกฟังก์ชัน say ในคลาส Day โดยใช้เครื่องหมาย ->
27	จองหน่วยความจำให้กับคลาส Beagle โดยนำตำแหน่งหน่วยความจำไปเก็บไว้ตัวแปรพอยน์เตอร์ Jeejee
28	ใช้ Jeejee เรียกฟังก์ชันsayในคลาส Beagle โดยใช้เครื่องหมาย ->

จากตัวอย่างเป็นการเรียกใช้ฟังก์ชันใน derived class ผ่าน base class pointer ซึ่งมีความยุ่งยากในการเรียกใช้งานฟังก์ชัน ในภาษา C++ การเรียกใช้ฟังก์ชันสามารถที่เรียกใช้ฟังก์ชัน ผ่านbase class reference โดยสร้างฟังก์ชันที่มีการอ้างถึงฟังก์ชันใน derived class ขึ้นมา เมื่อมีการเรียกใช้ฟังก์ชันที่ต้องการ ก็จะเรียกผ่านฟังก์ชันที่สร้างขึ้นมาแทน

ตัวอย่าง 5.5 การใช้ Virtual Function ผ่าน base class reference

```
1    #include <iostream>
2    using namespace std;
3    class Animal {
4    public:
5    virtual void Say() {
6    cout << "I am animal"<<endl;
7    }
8    };
9    class Dog:public Animal {
10   public:
11   void Say() {
12   cout <<"Box Box"<<endl;
```

```
13    }
14    };
15    class Beagle:public Dog {
16    public:
17    void Say() {
18    cout <<"I am Beagle "<<endl;
19    }
20    };
21    void Jeejee(Animal &a) {
22    cout<<"Jeejee say";
23    a.Say(); }
24    int main() {
25    Animal A;
26    Dog D;
27    Beagle J;
28    Jeejee (A);
29    Jeejee (D);
30    Jeejee (J);
31    return 0;
32    }
```

ผลลัพธ์

```
Jeejee say I am animal
Jeejee say Box Box
Jeejee say I am Beagle
```

อธิบายการทำงานโปรแกรม

บรรทัดที่	คำอธิบาย
21	ฟังก์ชัน Jeejee เป็นฟังก์ชันที่สร้างขึ้น มีพารามิเตอร์ Animal ซึ่งเป็น base class reference และมีอ็อบเจ็กต์ &a ใช้ระบุตำแหน่ง address ของอ็อบเจ็กต์แต่ละคลาส

23	ใช้อ็อบเจ็กต์ a เรียกฟังก์ชัน Say ของแต่ละคลาสอาศัยการระบุตำแหน่ง address
25	สร้างอ็อบเจ็กต์ A จากคลาส Animal
26	สร้างอ็อบเจ็กต์ D จากคลาส Dog
27	สร้างอ็อบเจ็กต์ J จากคลาส Beagle
28	เรียกฟังก์ชัน Jeejee ส่งค่าอากิวเมนต์ อ็อบเจ็กต์ A
29	เรียกฟังก์ชัน Jeejee ส่งค่าอากิวเมนต์ อ็อบเจ็กต์ D
30	เรียกฟังก์ชัน Jeejee ส่งค่าอากิวเมนต์ อ็อบเจ็กต์ J

คลาสนามธรรม (Abstract base classes)

การพ้องรูปของคลาสที่สืบทอด จะสืบทอดฟังก์ชันจากคลาสหลัก โดยใช้คำสงวน virtualนำหน้าฟังก์ชันในคลาสหลัก กรณีคลาสหลักที่มีแต่ฟังก์ชันและไม่มีรายละเอียด จะเรียกว่า abstractซึ่งเป็นได้แต่เพียงนามธรรม เมื่อจะนำไปใช้ จะต้องไปกำหนดรายละเอียดอีกครั้ง

การสร้าง Abstract base classes

โดยประกาศฟังก์ชันของคลาสหลักเป็น pure virtual function คือการประกาศฟังก์ชันต้นแบบ พร้อมกำหนดค่าให้เท่ากับ 0 และไม่ต้องเขียนรายละเอียด

ตัวอย่างที่ 7.6 การใช้ Virtual Function แบบ Abstract

```
1    #include <iostream>
2    using namespace std;
3    class convert {
4    protected:
5    double val1;
6    double val2;
7    public:
8    convert(double i) {
9    val1 = i;
10   }
11   double getconv() { return val2; }
12   double getinit() { return val1; }
```

```
13  virtual void compute() = 0;
14  };
15  class l_to_g : public convert {
16  public:
17  l_to_g(double i) : convert(i) { }
18  void compute() {
19  val2 = val1 / 3.7854;
20  }
21  };
22  class f_to_c : public convert {
23  public:
24  f_to_c(double i) : convert(i) { }
25  void compute() {
26  val2 = (val1-32) / 1.8;
27  }
28  };
29  int main()
30  {
31  convert *p;
32  p = new l_to_g (4);
33  cout << p->getinit() << " liters is ";
34  p->compute();
35  cout << p->getconv() << " gallons\n";
36  p = new f_to_c (70);
37  cout << p->getinit() << " in Fahrenheit is ";
38  p->compute();
39  cout << p->getconv() << " Celsius\n";
40  return 0;
```

```
41      }
```

ผลลัพธ์

```
4 liters is 1.05669 gallons
70 in Fahrenheit is 21.1111 Celsius
```

อธิบายการทำงานของโปรแกรม

บรรทัดที่	คำอธิบาย
3	คลาสชื่อ convert
8	convert เป็นฟังก์ชันคอนสตรัคเตอร์อยู่ในคลาส convert ต้องการพารามิเตอร์
11	ฟังก์ชัน getconv
12	ฟังก์ชัน getinit
13	ฟังก์ชัน compute อยู่ในคลาส convert มีคำ virtual เพื่อใช้เป็น Abstract มีค่าเท่ากับ 0
17	l_to_gเป็นฟังก์ชันคอนสตรัคเตอร์อยู่ในคลาส l_to_g ต้องการพารามิเตอร์ และได้รับการสืบทอด ฟังก์ชันคอนสตรัคเตอร์จากคลาส convert ที่มีพารามิเตอร์ด้วย
18	ฟังก์ชัน compute อยู่ในคลาส l_to_g
24	f_to_cเป็นฟังก์ชันคอนสตรัคเตอร์อยู่ในคลาส f_to_c ต้องการพารามิเตอร์ และได้รับการสืบทอด ฟังก์ชันคอนสตรัคเตอร์จากคลาส comment ที่มีพารามิเตอร์ด้วย
25	ฟังก์ชัน compute อยู่ในคลาส l_to_g
31	สร้างอ็อบเจ็กต์ p จากพารามิเตอร์คลาส convert เป็นพอยน์เตอร์
32	จองหน่วยความจำให้กับพารามิเตอร์คลาส l_to_g นำตำแหน่งหน่วยความจำไปเก็บไว้ตัวแปรพอยน์เตอร์ p มีอากิวเมนต์เป็นเลขจำวนเต็ม
33,34,35	เรียกใช้ฟังก์ชัน โดยใช้ p
36	จองหน่วยความจำให้กับพารามิเตอร์คลาส f_to_c นำตำแหน่งหน่วยความจำไปเก็บไว้ตัวแปรพอยน์เตอร์ p มีอากิวเมนต์เป็นเลขจำวนเต็ม
37,38,39	เรียกใช้ฟังก์ชัน โดยใช้ p

สรุป

การสืบทอดคุณสมบัติ (inheritance) คือคุณสมบัติหนึ่งของการเขียนโปรแกรมเชิงวัตถุ ที่ อนุญาตให้ผู้เขียนโปรแกรมสามารถสร้างคลาสใหม่ โดยการสืบทอดเอาคุณสมบัติต่าง ๆ มาจาก คลาสที่มีอยู่แล้วมาสร้าง

เป็นคลาสใหม่ โดยมีรูปแบบการเขียนโตใช้สัญลักษณ์ : (colon) โดยอยู่ หลังชื่อคลาสสืบทอด (คลาสลูก) ตามด้วยคลาสต้นแบบ (คลาสแม่) ดังนี้

```
class Myderivedclass : public Mybaseclass
```

การฟ้องรูป (polymorphism) คือการเพิ่มความสามารถของคลาสต้นแบบ โดยไม่จำเป็นต้อง แก้ไขคลาสต้นแบบ ที่สืบทอดคุณสมบัติมา สำหรับการทำงานแบบการฟ้องรูป จะต้องใช้คำสงวน virtual กำหนดไว้หน้าฟังก์ชัน คลาสต้นแบบ