OOP & data struct

4. (1) Class composition

   (2) Multiple level

   (3) Inheritance assignment

   (4) virtual function/methods

BY SOMSIN THONGKRAIRAT

ATTASIT LASAKUL (EDITION)

(1) Class composition

Class composition คือ การที่คลาสหนึ่งเอาคลาสอื่นมาสร้าง Object ใช้งานในคลาสตนเอง (ไม่ใช่ สืบทอดมา)

โครงร่างโปรแกรม 4_1..CPP

((1) class composition)

class

constructor

main

```
#include <iostream>
using namespace std;
class point {
        int x, y;    }
class triangle {
        point p1, p2, p3;   }
class triangle_parism {
        triangle base;   }
class point3d : public point {
        int z;        }
class tetraheron {
        point3d p, q, r;        }
triangle t1;
        t1.p1.x = 5;
```

โครงร่างโปรแกรม 4_2..CPP

((2)Muti-level & muti-Inheritance)

```
#include <iostream>
using namespace std;
class creature {
        string name; }
class animal : public creature {
        int leg_count;  }
class cat : public animal {
        string cuteness;        }
class fying_things {
        int max_attitude; }
class brid : public animal,public  fying_thing
    {  string original;  }
brid seagull;
Seagull.name = “seagull”;
```

โครงร่างโปรแกรม 4_3..CPP

((3)Inheritance assignment)

```
#include <iostream>
using namespace std;
class creature {...}
class animal : public creature {...}
class cat : public animal {...}
class fying_things {...}
class brid : public animal, public fying_things {...}
int main( )
{   brid seagull;
    seagull.name =“seagull”;
   animal a1 = seagull;
cout<<seagull.name<<endl;
cout<<a1.name <<endl;
animal* p1;
p1 = &brid;
((brid*)p1)->name = “eagle”;
```

โครงร่างโปรแกรม 4_4.CPP, 4_5.CPP

((4) Virtual function/methods)

```
#include <iostream>
using namespace std;
class creature {
  virtual void move( ) {..}  }
class animal : public creature {
   virtual void move( ){..}    }
class cat : public animal {
     void move( ) {...}      }
class fying_things {
     int max_attitude; }
class brid : public animal,public  fying_thing
   {  void move( ) {.........}   }
brid seagull;
Seagull.name = “seagull”;
```

4_4.cpp ไม่มี
4_5.cpp มี

4_4.cpp ไม่มี
4_5.cpp มี

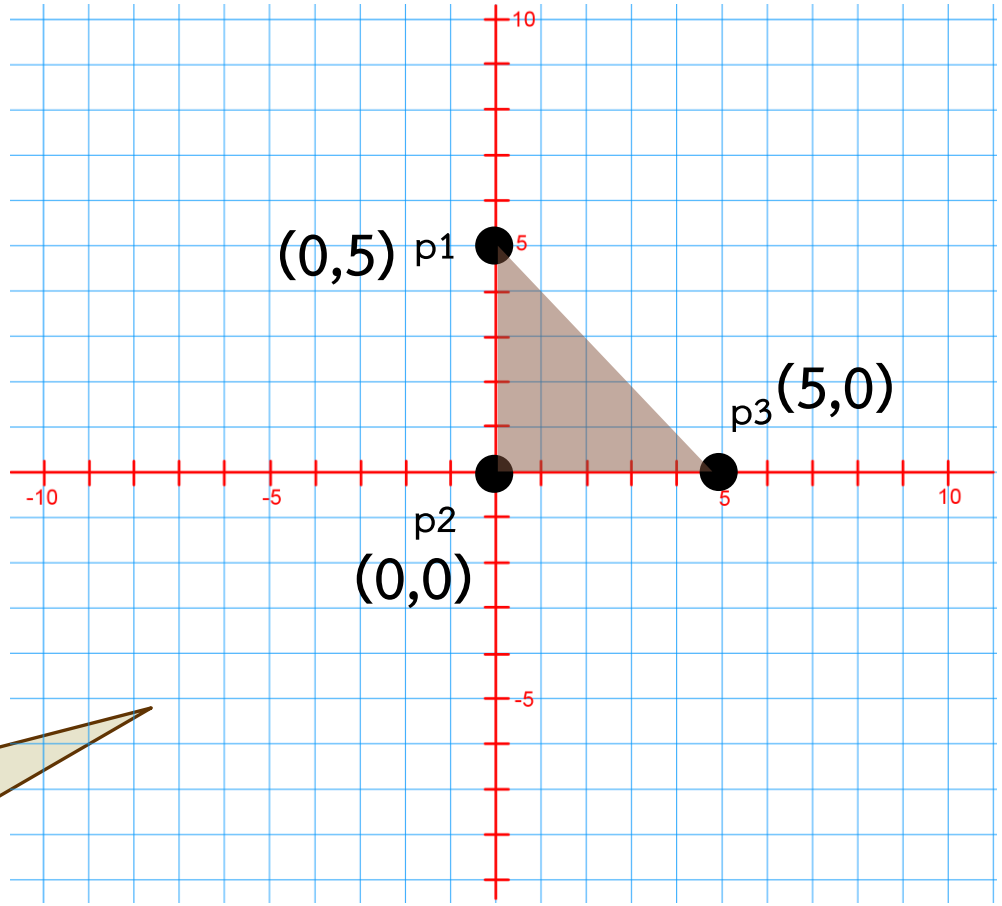
โครงร่างโปรแกรม 4_6.CPP

LAB

```
#include <iostream>
using namespace std;
class processor {
  string name, brand;
                          }
class input_device {
 string name; int number_button;    }
class computer_device {
  processor cpu; float ram_capacity;
                          }
class telephone {
  string network; int max_attitude;
  input_device input;
  virtual void print( ) {....}  }
processor i3 (“i3”,”intel”,4.5,8);
input_device keyboard(“key”,200,0);
```

(1) Class composition (4_1.cpp)

คลาสเราสามารถสร้างโดยสืบทอดคลาสแม่ได้ และในคลาสเราก็ยังสามารถสร้าง Object ใช้งานจากคลาสอื่นๆได้เช่นเดียวกัน ดังตัวอย่างไฟล์ 4_1 ... .CPP ซึ่งจะเป็นการสร้างรูปทรงเรขาคณิต ที่มีการสืบทอด และ/หรือ มีสร้าง Object ในคลาสไว้ใช้งานได้



Cartesian plane

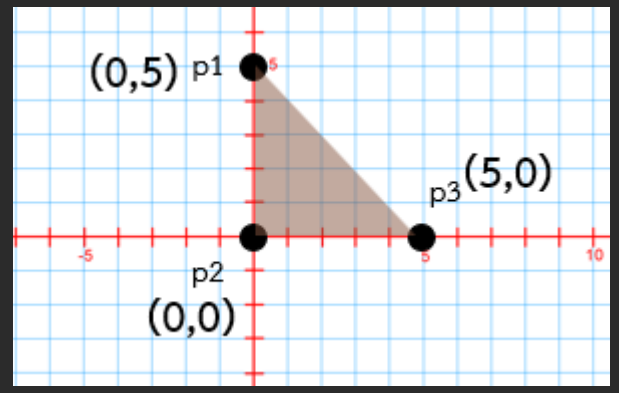
- มีการกำหนดจุด 2d point
- มีคำนวณ calculate area of triangle

ตัวอย่างไฟล์ 4_1 ... .CPP มีคลาสต่างๆดังนี้

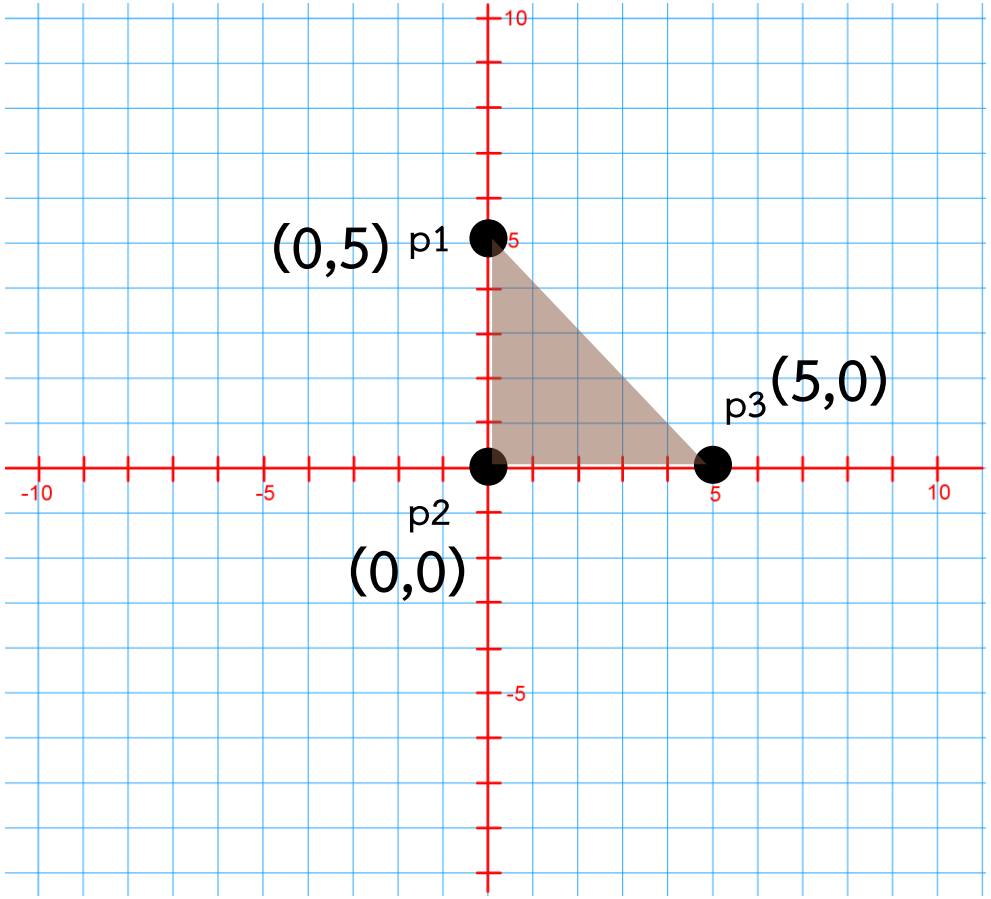
```cpp
class point
{
public:
    float x, y;
    point()
    {
        x = 0.00;
        y = 0.00;
    }
    point(float _x, float _y)
    {
        x = _x;
        y = _y;
    }
};
```

```cpp
class triangle
{
public:
    point p1;
    point p2;
    point p3;
    float get_area()
    {
        return 0.5 * abs((p1.x * (p2.y - p3.y)) +
        (p2.x * (p3.y - p1.y)) + (p3.x * (p1.y - p2.y)));
    }
};
```

มีสร้าง Object จากคลาสอื่น (Class composition)



ดังนั้นใน main( ) เราก็สามารถเรียกใช้งาน triangle ได้ปกติ



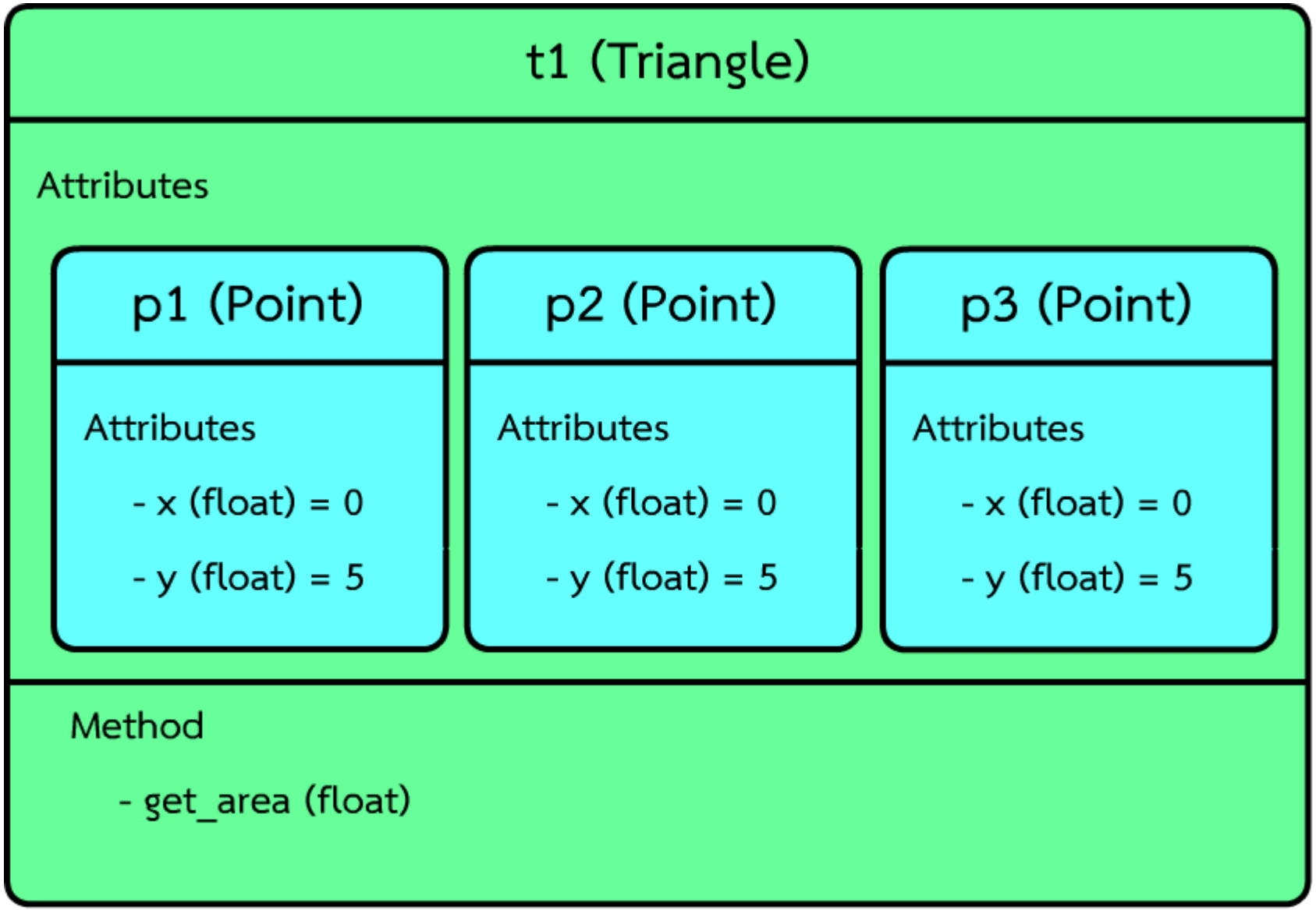

```
49  int main(){
50      
51      triangle t1;
52      t1.p1.x = 0; t1.p1.y = 5;
53      t1.p2.x = 0; t1.p2.y = 0;
54      t1.p3.x = 5; t1.p3.y = 0;
55  
56      cout << t1.get_area() << " sq.unit" << endl;
57  
```

Result :

12.5 sq.unit

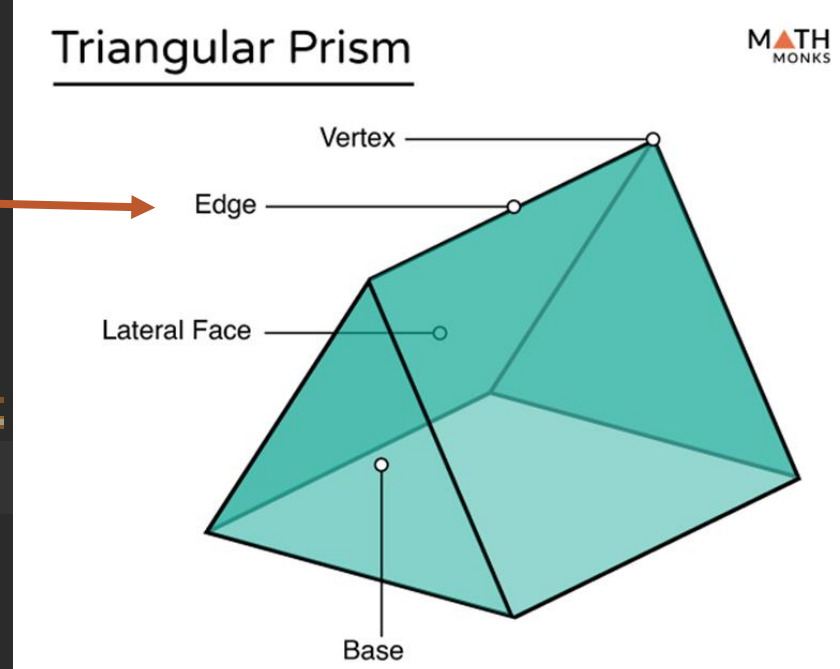
In main :

```
triangle t1;
t1.p1.x = 0;
t1.p1.y = 5;
t1.p2.x = 0;
t1.p2.y = 0;
t1.p3.x = 5;
t1.p3.y = 0;
```



สร้างคลาส ปริซึม โดยทำคลาสก่อนหน้ามาเป็นคลาสมแม่เพื่อสืบทอดต่อเนื่อง

```
36  class triangular_prism
37  {
38  public:
39      triangle base;
40      float edge_length;
41      float get_volume()
42      {
43          return base.get_area() * edge_length;
44      }
45      float get_surface_area()
46      {
47          return 0;
48      }
49  };
```



p1 (triangular_prism)

Attributes

- base (triangle)
- edge_length (float)

Method

- get_volume (float)
- get_surface_area (float)

```
int main()
{
    triangle t1;
    t1.p1.y = 5;
    t1.p2.x = 0;
    t1.p2.y = 0;
    t1.p3.x = 5;
    t1.p3.y = 0;
    cout << t1.get_area() << " sq.unit" << endl;

    triangular_prism p1;
    p1.base = t1;
    p1.edge_length = 20;
    cout << p1.get_volume() << " cubic unit" << endl;
```

Result :

12.5 sq.unit

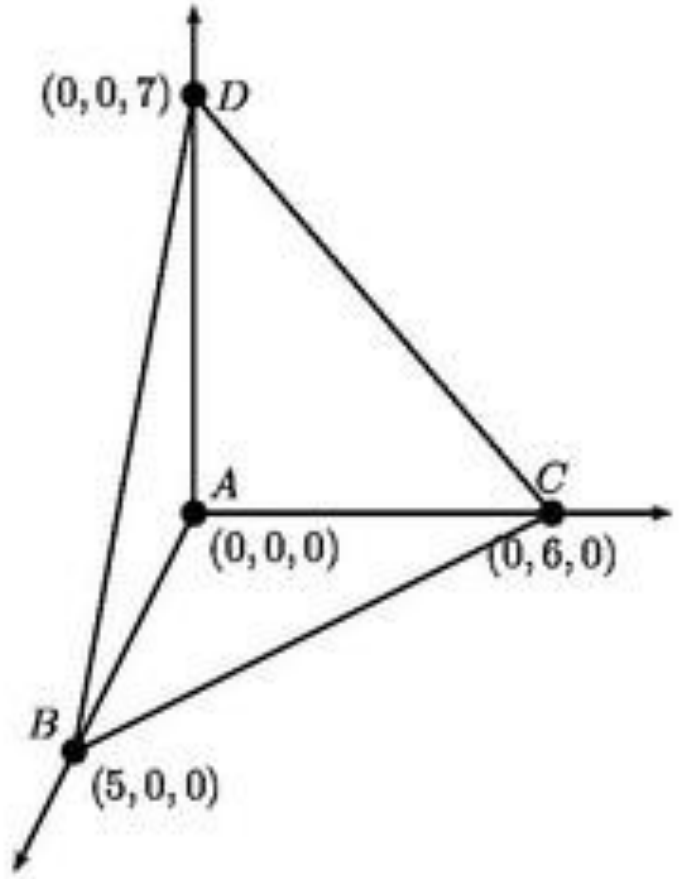
250 cubic unit

คลาสสร้างจุด point3d ก็สร้างได้ง่ายโดยสืบทอดจากคลาสจุดปกติ (point) ที่เรามีแล้ว โดยเพิ่มจุด Z เข้าไป

```
class point3d : public point {
public :
    float z;

    point3d(float _x, float _y, float _z) : point(_x,_y){
        z = _z;
    }
}
```

P1 (x, y, z);

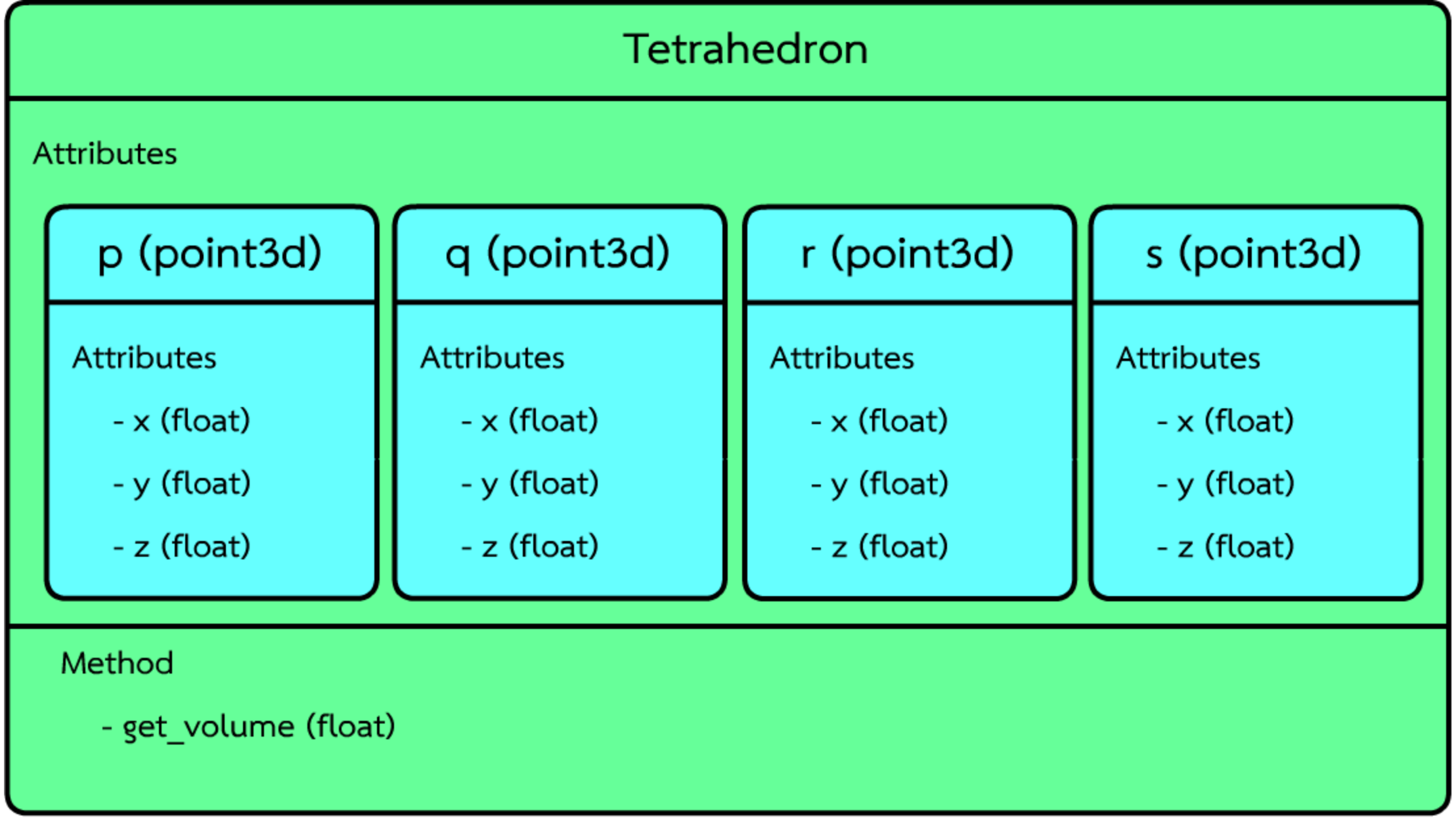
หรือจะสร้างแบบรูปทรงสี่หน้า (Tetrahedron)




```
class Tetrahedron{
public :
    point3d p,q,r,s;

}
```

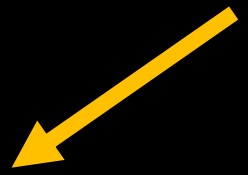
แล้วก็เพิ่ม methods ที่ต้องการได้เลยในคลาส

```
class Tetrahedron{
public :
    point3d p,q,r,s;

    float get_volume(){
        float p_volume =    ((s.x-p.x)*(((q.y-p.y)*(r.z-p.z))-((q.z-p.z)*(r.y-p.y)))) +
                            ((s.y-p.y)*(((q.z-p.z)*(r.x-p.x))-((q.x-p.x)*(r.z-p.z)))) +
                            ((s.z-p.z)*(((q.x-p.x)*(r.y-p.y))-((q.y-p.y)*(r.x-p.x))));
        return p_volume / 6;
    }
}
```


Tetrahedron
Attributes
p (point3d)
Attributes
- x (float)
- y (float)
- z (float)
q (point3d)
Attributes
- x (float)
- y (float)
- z (float)
r (point3d)
Attributes
- x (float)
- y (float)
- z (float)
s (point3d)
Attributes
- x (float)
- y (float)
- z (float)
Method
- get_volume (float)

ดังนั้นที่ main( ) เราก็จะสามารถเรียกใช้ได้ปกติ แต่ต้องเรียงลำดับคำสั่งให้ถูกต้องดังตัวอย่าง

```
  point3d a(0,0,0) ,b(5,0,0) ,c(0,6,0) ,d(0,0,7);
    tetrahedron tthd1;
    tthd1.p = a;
    tthd1.q = b;
    tthd1.r = c;
    tthd1.s = d;

    cout << tthd1.get_volume() << " cubic unit" <<
endl;
```



Result :

```
35 cubic unit
```

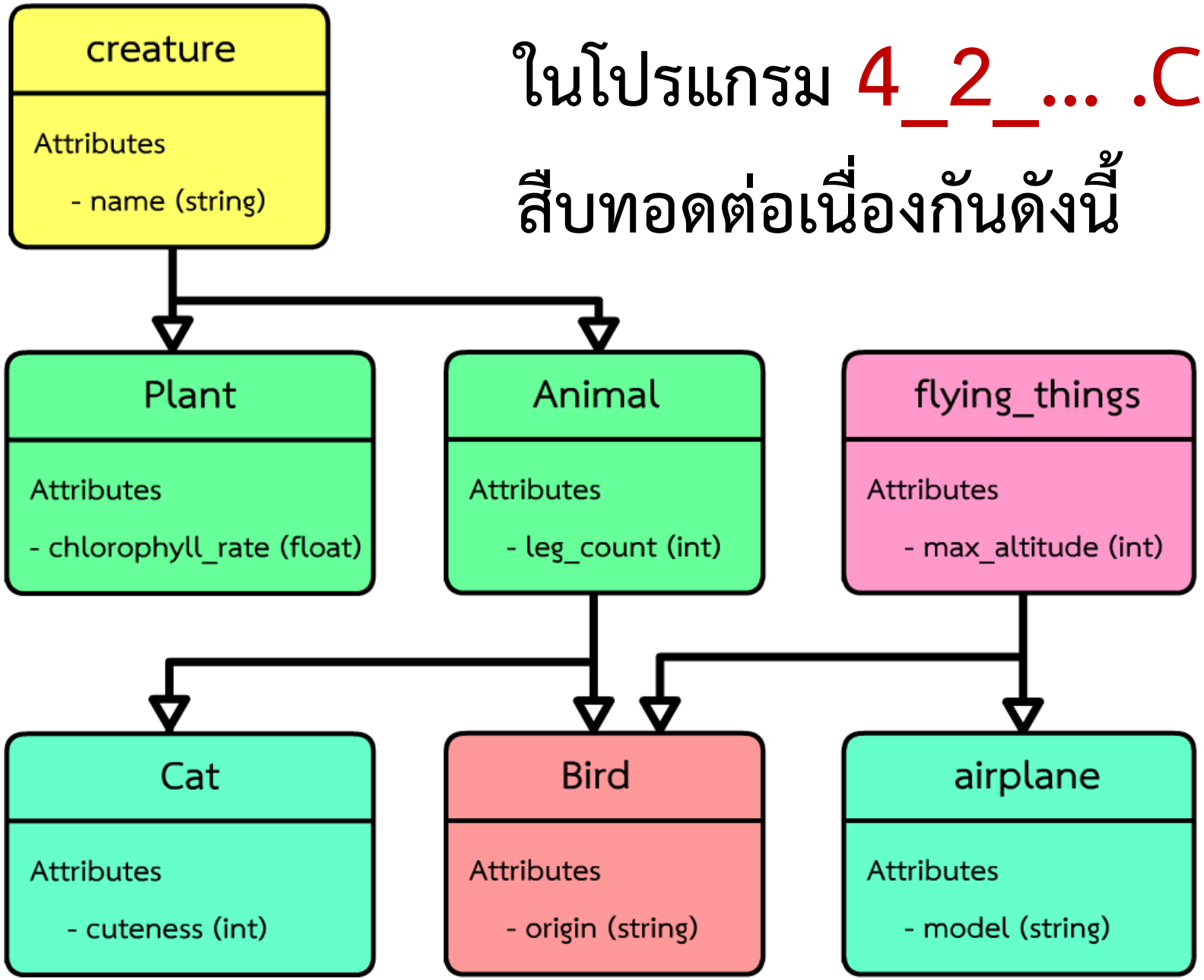
(2) Multiple level & Multiple inheritance

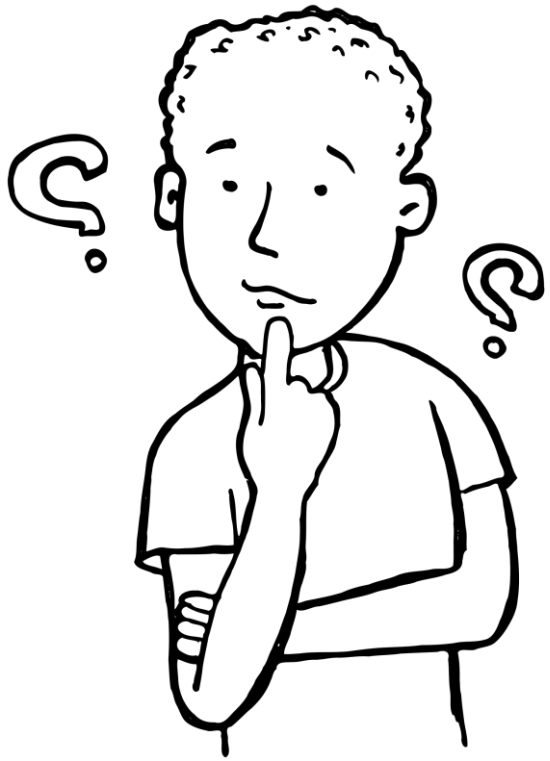
(โปรแกรม 4_2.CPP)

- class ที่รับการสืบทอดมาแล้วสามารถส่งต่อการสืบทอดไปยัง class ลูกได้อีก

- 1 class สามารถรับการสืบทอดได้มากกว่า 1 class

ในโปรแกรม 4_2_... .CPP จะมีการสร้างคลาสที่สืบทอดต่อเนื่องกันดังนี้





โปรแกรม 4_2_... .CPP

```cpp
class creature{
public :
    string name;
};

class plant : public creature{
public :
    float chlorophyll_rate;
};

class animal : public creature{
public :
    int leg_count;
};
```

```cpp
class flying_things{
public :
    int max_altitude;
};

class airplane : public flying_things{
public :
    string model;
};

class bird : public animal, public flying_things{
public :
    string origin;
};

class cat : public animal{
public :
    int cuteness;
};
```

คลาส bird เป็นแบบ Multiple inheritance

```
class bird : public animal, public flying_things{
public :
    string origin;
};
```

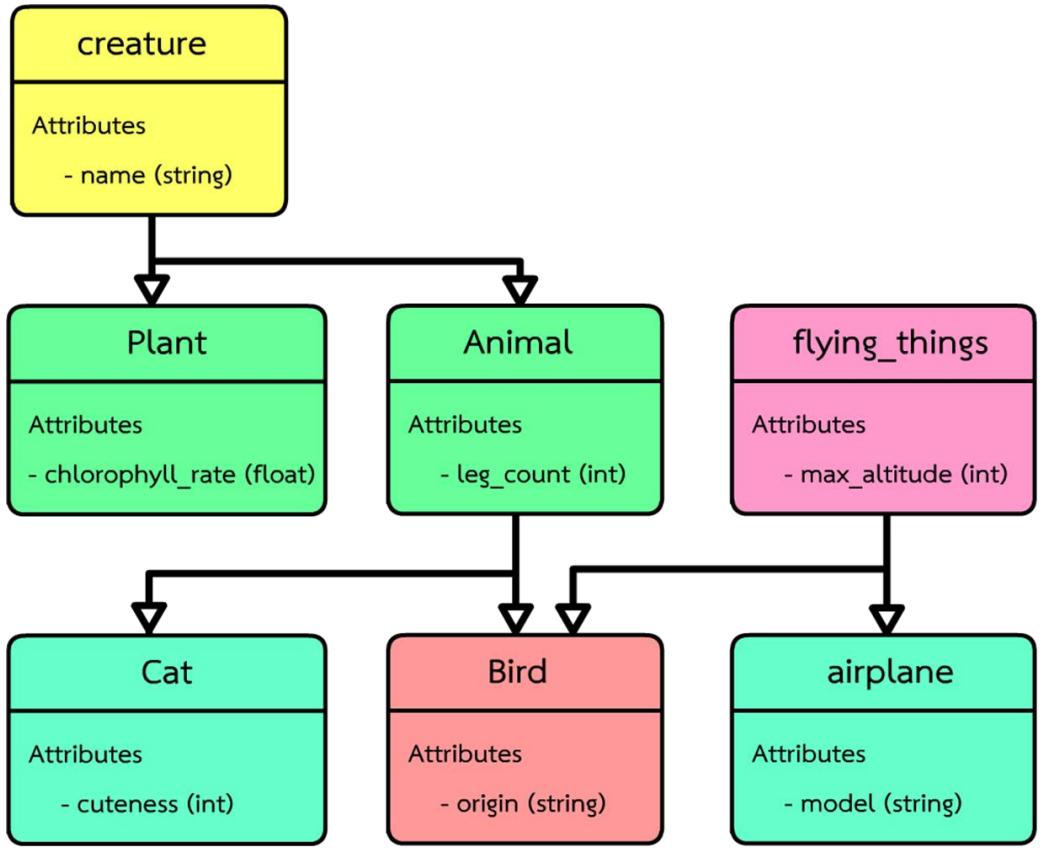
Class bird จะมีคุณสมบัติทั้ง animal (leg_count) และ flying_things (max_altitude)

ส่วนคลาสอื่นก็เป็นการสืบทอดต่อเนื่องเท่านั้น (Multiple level inheritance)

```
class creature{
public :
    string name;
};
class animal : public creature{
public :
    int leg_count;
};
class cat : public animal{
public :
    int cuteness;
};
```

class cat ก็จะมีคูณสมบัติของ class ปู่ (grand parent) นั้นก็คือ name

เมื่อนำมาใช้งานใน main( ) ก็สามารถกำหนดค่าเริ่มต้นของแต่ละ Object ได้ดังนี้




```
int main(){

    bird seagull;
    seagull.name = "seagull";
    seagull.leg_count = 2;
    seagull.max_altitude = 100; // 100 meter
    seagull.origin = "Australia";

    cat Sphynx;
    Sphynx.name = "Sphynx";
    Sphynx.leg_count = 4;
    Sphynx.cuteness = -10;
```

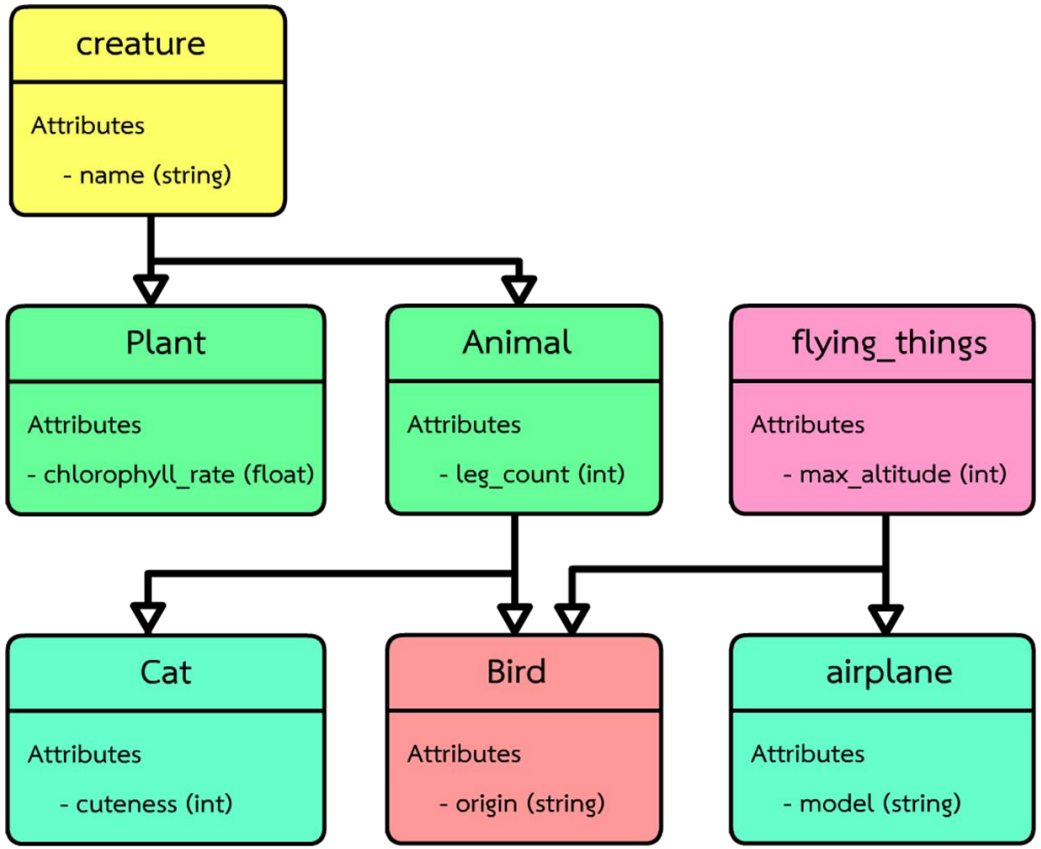
(3) Inheritance assignment

- object ที่เกิดจาก class ที่อยู่สูงกว่า (parent class) สามารถกำหนดให้เป็น object ที่เกิดจาก class ที่อยู่ต่ำกว่าได้ (child class) ไม่ว่าจะสืบทอดกันมากี่ชั้น เช่น เราสามารถกำหนด object ที่เกิดจากคลาส creature ให้เป็น object ที่เกิดจากคลาส bird ได้ หรือสามารถกำหนด flying_things เป็น airplane ได้

- Class ที่อยู่สูงกว่า (เช่น base class) สามารถมีตัวชี้เพื่อชี้ไปกระทำกับข้อมูลของคลาสลูกได้

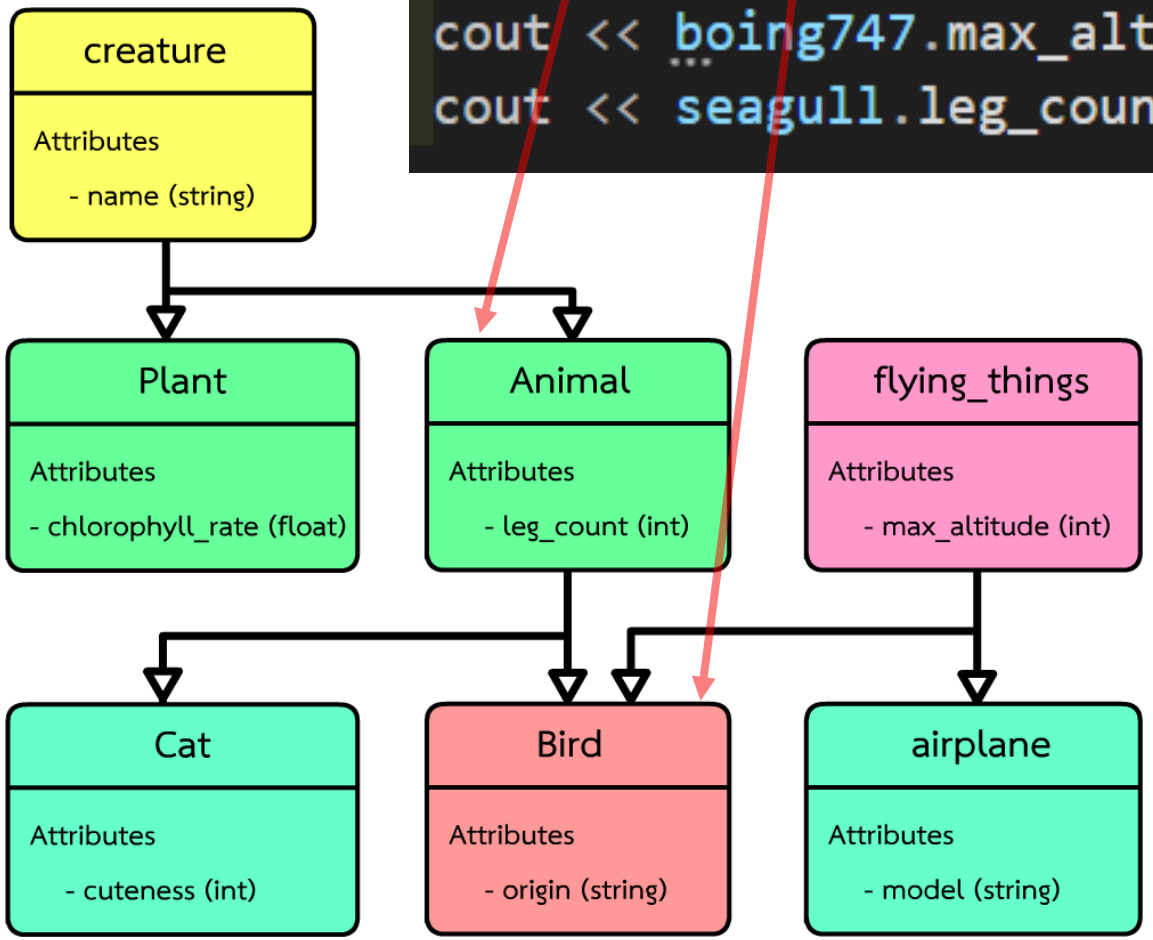
ตามตัวอย่างการใช้งานในโปรแกรม 4_3_ ... .CPP

Example :




```
43    bird seagull;
44    seagull.name = "seagull";
45    seagull.leg_count = 2;
46    seagull.max_altitude = 100; // 100 meter
47    seagull.origin = "Australia";
48
49    cat Sphynx;
50    Sphynx.name = "Sphynx";
51    Sphynx.leg_count = 4;
52    Sphynx.cuteness = -10;
53
54    airplane boing747;
55    boing747.model = "747";
56    boing747.max_altitude = 13610;
57
58    animal a1 = seagull; // no error
59    creature c1 = Sphynx; // no error
60    flying_things f1 = boing747; // no error
61
62    animal a2 = boing747; // error
63    flying_things f2 = Sphynx; // error
64    bird b1 = a1; // error
```

```
animal a1 = seagull; // no error
creature c1 = Sphynx; // no error
flying_things f1 = boing747; // no error

cout << Sphynx.name << " : " << c1.name << endl;
cout << boing747.max_altitude << " : " << f1.max_altitude << endl;
cout << seagull.leg_count << " : " << a1.leg_count << endl;
```

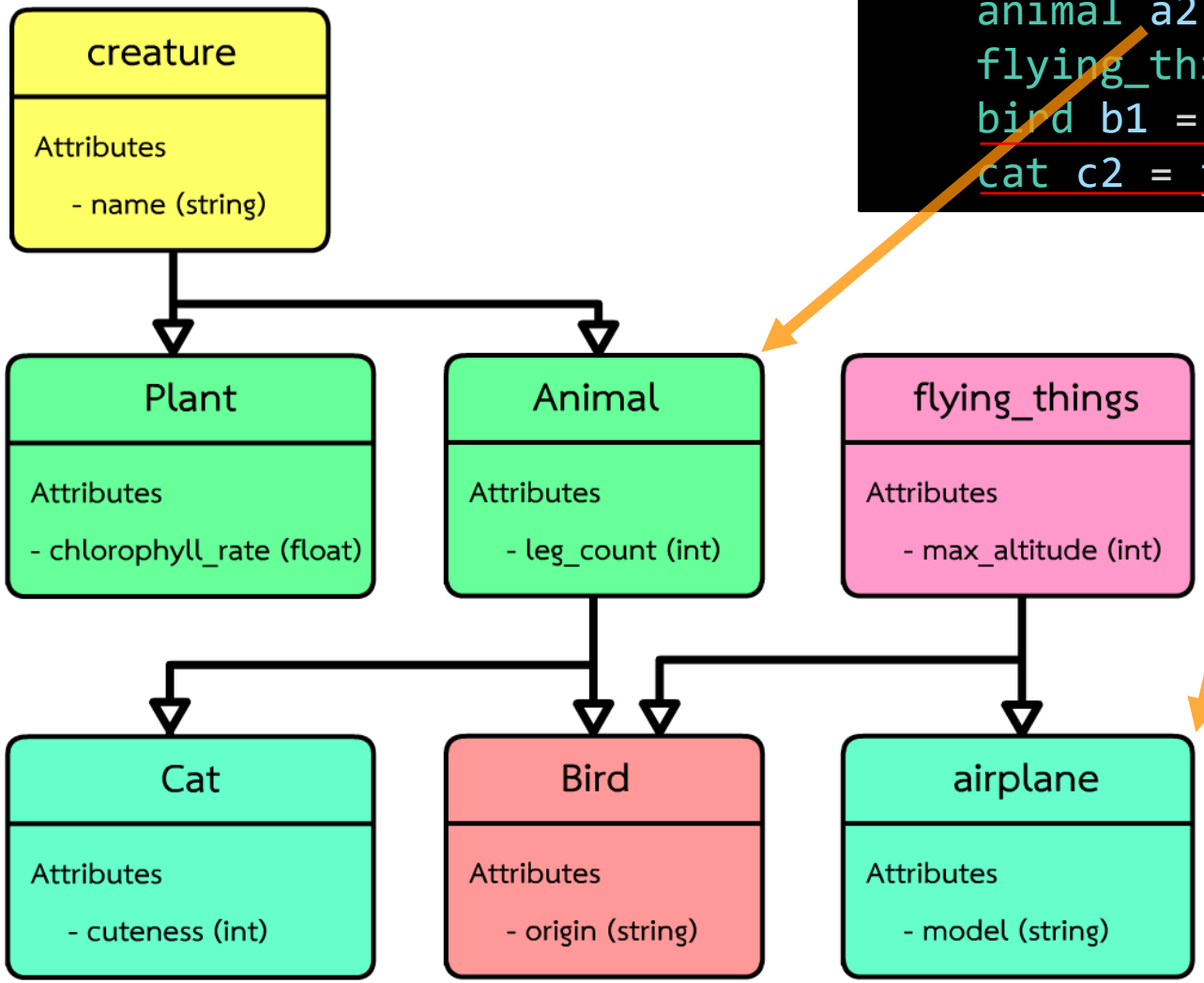


Result :

Sphynx : Sphynx

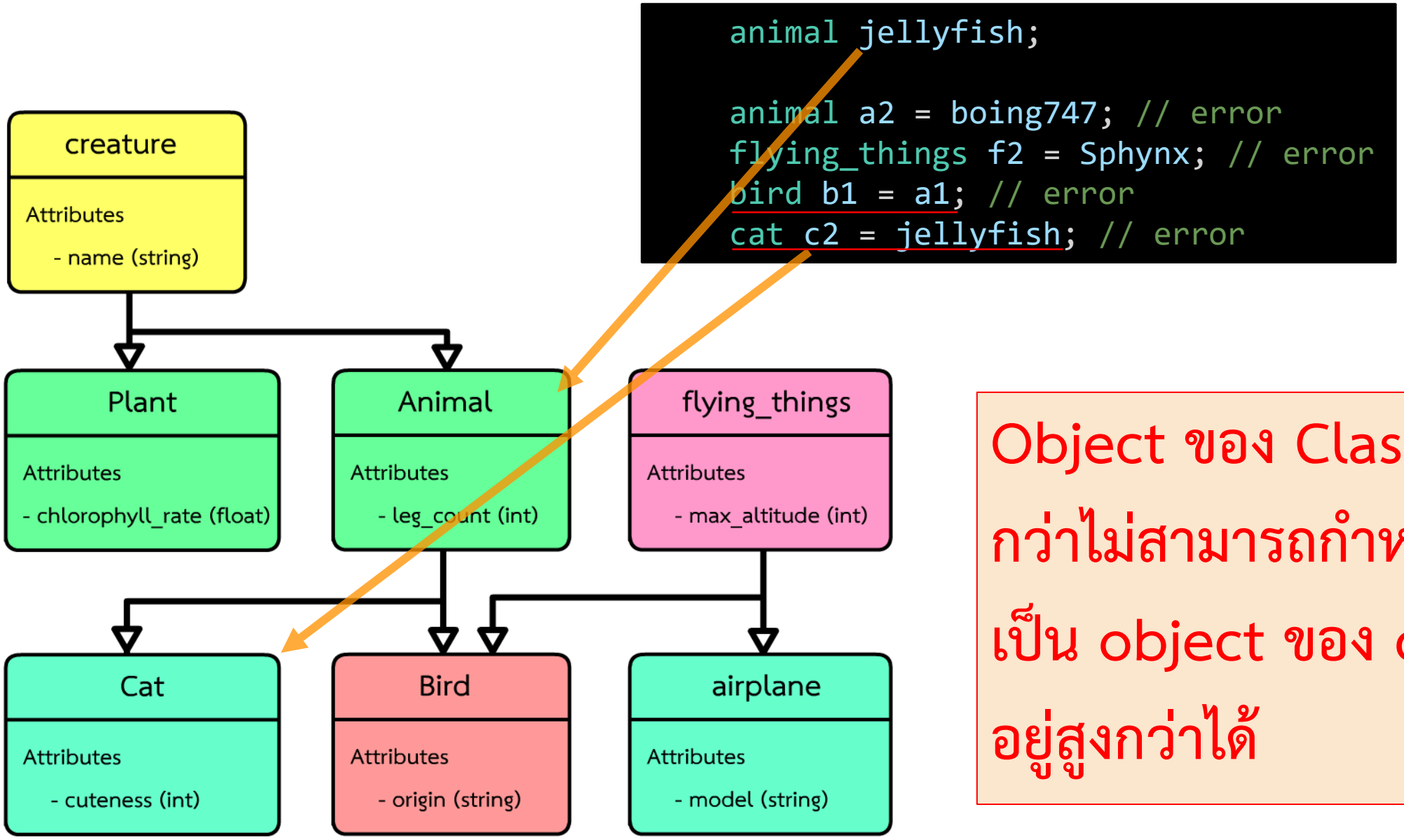
13610 : 13610

2 : 2




```
animal jellyfish;

animal a2 = boing747; // error
flying_things f2 = Sphynx; // error
bird b1 = a1; // error
cat c2 = jellyfish; // error
```

Class ที่ไม่เกี่ยวข้องกันไม่สามารถกำหนดค่าให้กันและกันได้



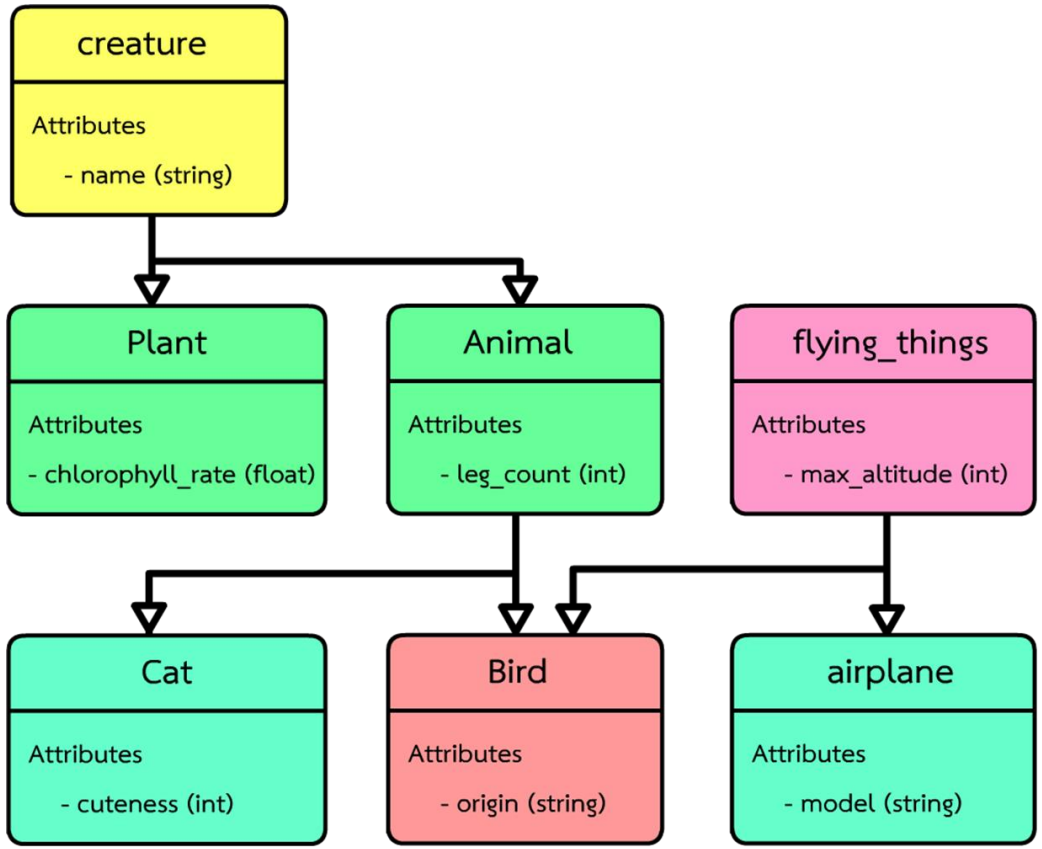
Object ของ Class ที่ต่ำกว่าไม่สามารถกำหนดให้เป็น object ของ class ที่อยู่สูงกว่าได้

ในบางครั้ง เราต้องการอ้างถึงหรือใช้งานตัวแปร, method ของคลาสลูก โดยที่เราไม่จำเป็นต้องสร้าง Object ของคลาสลูกนั้นก็ได้

(ไม่เปลืองหน่วยความจำที่ต้องสร้าง Object นั้นๆ เพื่อใช้งานถาวร หรือไม่มีความจำเป็นต้องใช้ คุณสมบัติทั้งหมดของ Object นั้นบ่อยๆ)

เราสามารถทำได้โดยการสร้างตัวชี้ (point) ของคลาสแม่ขึ้นมาแล้วนำ Address ของคลาสลูกไปใส่เพื่อชี้ไปส่วนประกอบต่างๆของคลาสลูกได้

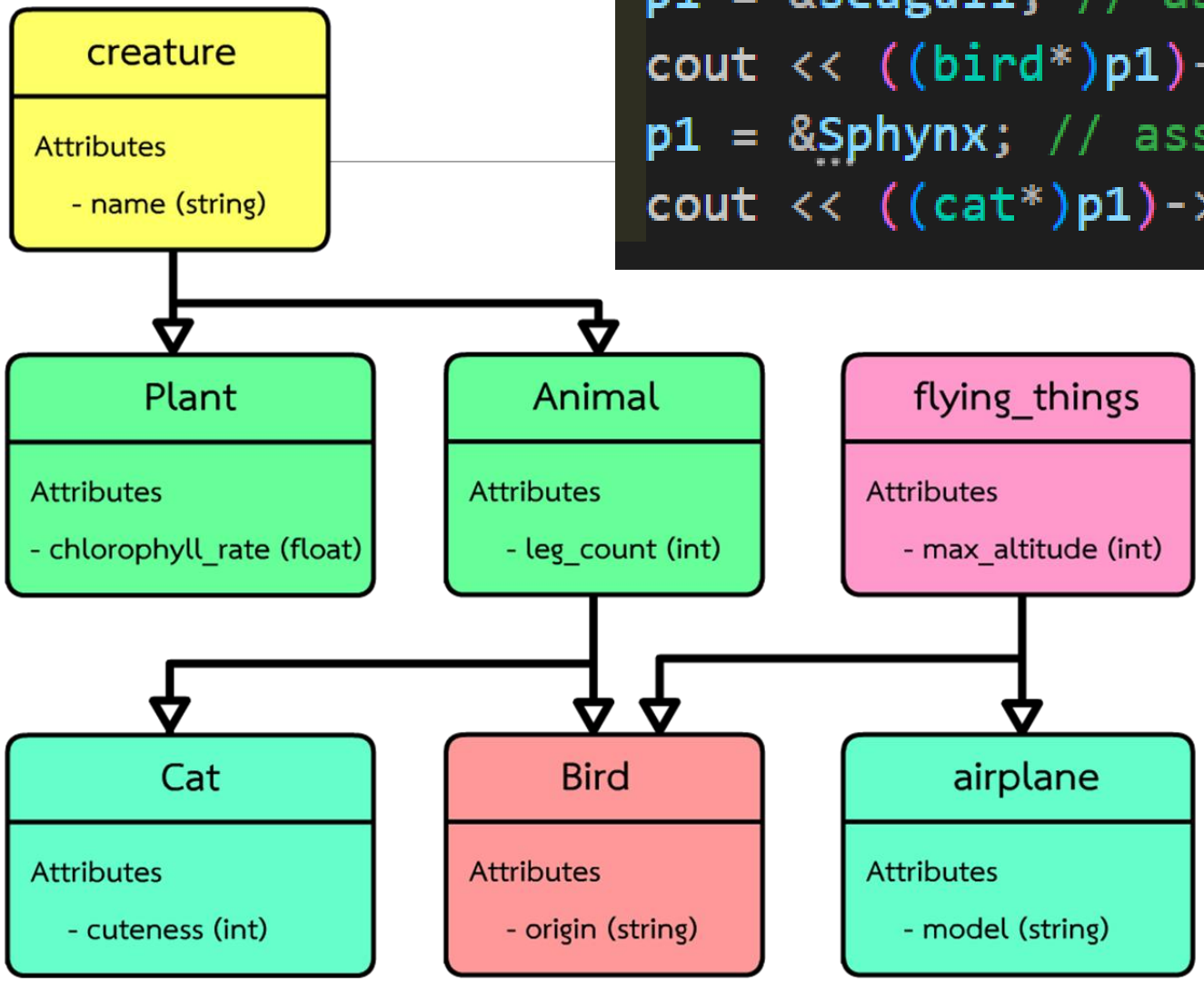
วิธีการใช้ pointer ของคลาสแม่ไปชี้คลาสลูก เช่น ต้องการ pointer ของคลาส creature ชี้ไปที่ ตัวแปร Origin ของ Object ที่ชื่อ Seagull ที่สืบทอดจากคลาส Brid ดังนี้




```
brid seagull;      //seaqull เป็น object ของคลาส Brid

creature *p1;   // p1 เป็นตัวชี้ข้อมูลแบบคลาส creature

p1 = &seagull;   // กำหนดให้ตัวชี้ p1 ชี้ไปที่ Object  seagull

cout << ((bird*)p1) -> origin << endl;
```

ชื่อตัวแปรของ Object เป้าหมาย

ชื่อคลาสของ Object เป้าหมาย




```
animal *p1;
p1 = &seagull; // assign parent to point child object
cout << ((bird*)p1)->origin << endl; // no error
p1 = &Sphynx; // assign parent to point child object
cout << ((cat*)p1)->cuteness << endl; // no error
```

Result :

```
Australia
-10
```

(4) Virtual function

คือ function (Methode) ที่ถูกประกาศไว้ใน base class (parent class) และสามารถถูกสร้างใหม่ทับ(override) ได้ใน class ลูก

ตัวอย่างในโปรแกรม 4_4_ ... .CPP

เป็นการอนุญาตให้คลาสลูกนำ method นี้ไปเขียนใช้เองได้
(เหมือนการ Override method)

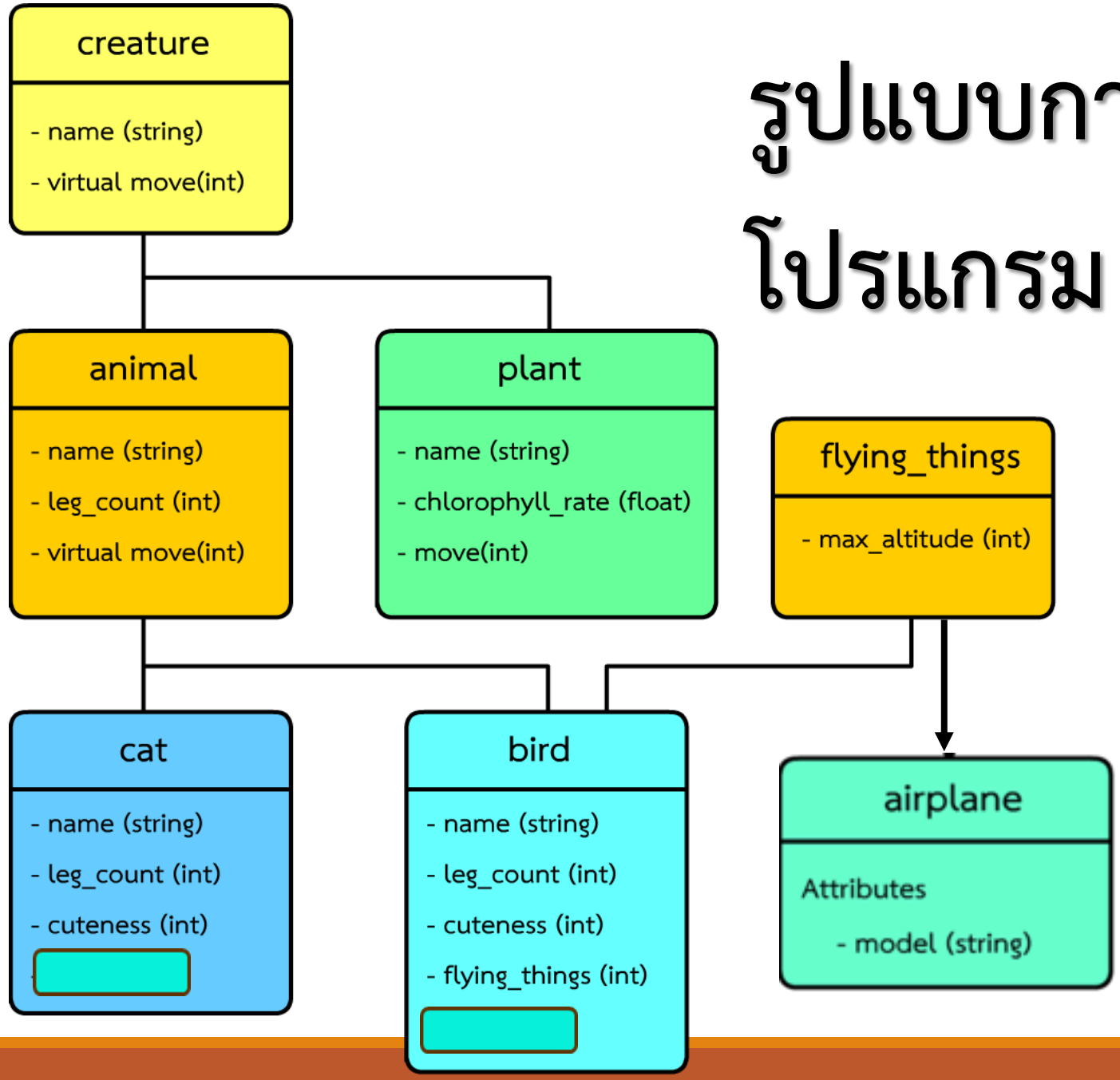
```
class creature
{
public:
    string name;
    int x;
    creature()
    {
        x = 0;
    }
   // void move(int _x)
    virtual void move(int _x)
    {
        x = _x;
        cout << name << " move to position : " << x << endl;
    }
};
```

```
class plant : public creature
{
public:
    float chlorophyll_rate;
    void move(int _x)
    {
        cout << name << " cannot move
    }
};

class animal : public creature
{
public:
    int leg_count;

    virtual void move(int _x)
    {
        x = _x;
        cout << name << " using " << leg_count << " leg(s) move to position : " << x << endl;
    }
};
```

คลาสลูกสืบทอดมาจากคลาสแม่ creature

คลาสลูกนำ method คลาสแม่มาเขียนใช้เอง

รูปแบบการสืบทอดของโปรแกรม 4_4 ... . CPP



ตัวคลาสแต่ละคลาสในโปรแกรม 4_4 .. .CPP

```cpp
class creature
{
public:
    string name;
    int x;
    creature()
    {
        x = 0;
    }
    virtual void move(int _x)
    {
        x = _x;
        cout << name << " move to position : " << x << endl;
    }
};
```

```cpp
class plant : public creature
{
public:
    float chlorophyll_rate;
    void move(int _x)
    {
        cout << name << " cannot move" << endl;
    }
};
```

```
class flying_things
{
public:
    int max_altitude;
};
```

```
class airplane : public flying_things
{
public:
    string model;
};
```

```
class animal : public creature
{
public:
    int leg_count;

    virtual void move(int _x)
    {
        x = _x;
        cout << name << " using " << leg_count << " leg(s) move to position : " << x << endl;
    }
};
```

```
class bird : public animal, public flying_things
{
public:
    string origin;
};
```

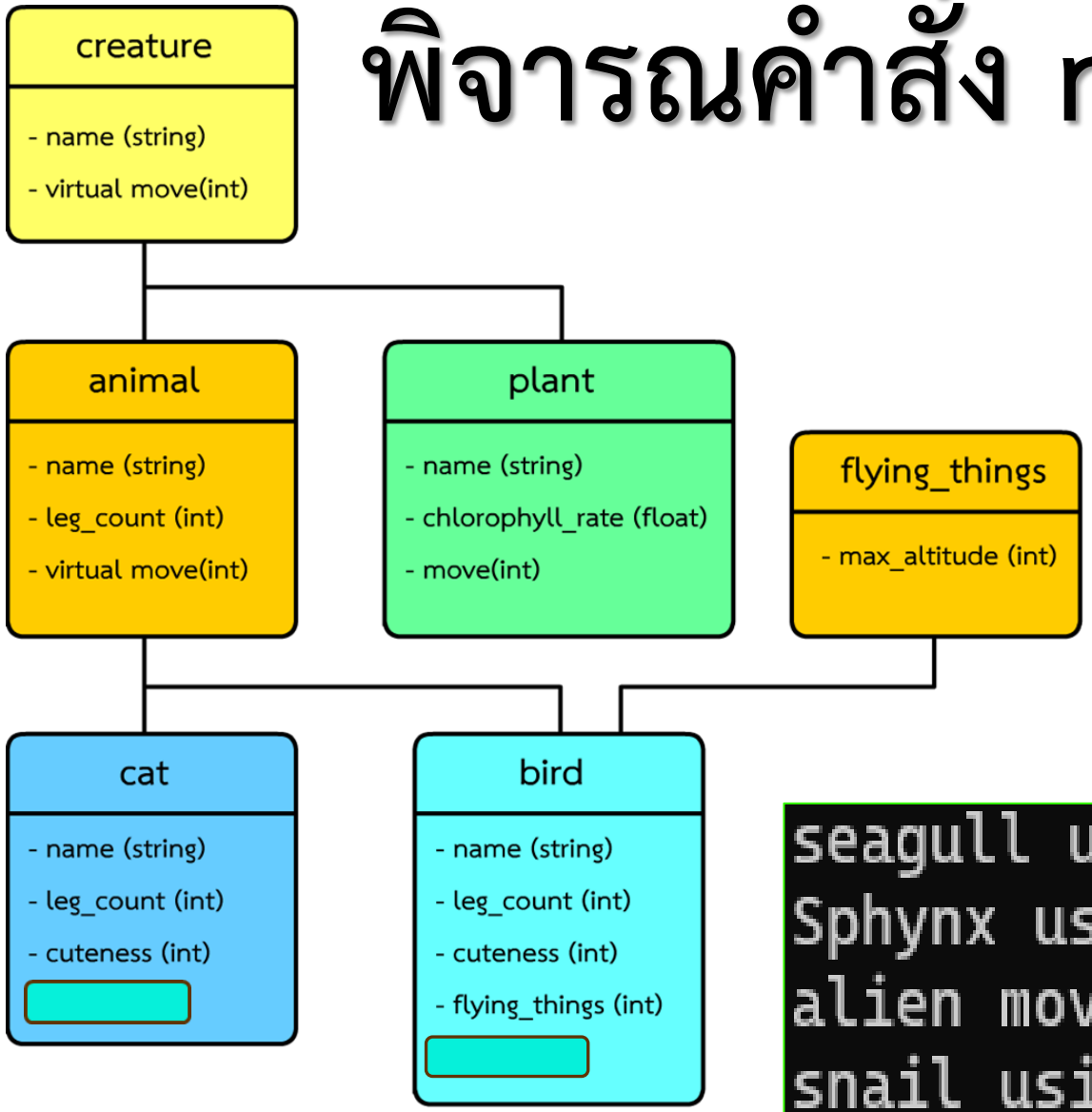
```
class cat : public animal
{
public:
    int cuteness;
};
```

ตัว main() ได้กำหนดค่าเริ่มต้นตัวแปรและ Object ต่างๆไว้ดังนี้

```
int main()
{
    bird seagull;
    seagull.name = "seagull";
    seagull.leg_count = 2;
    seagull.max_altitude = 100; // 100 meter
    seagull.origin = "Australia";

    cat Sphynx;
    Sphynx.name = "Sphynx";
    Sphynx.leg_count = 4;
    Sphynx.cuteness = -10;

    airplane boing747;
    boing747.model = "747";
    boing747.max_altitude = 13610;
```

```
creature alien;
alien.name = "alien";

animal snail;
snail.name = "snail";
snail.leg_count = 0;

plant palm;
palm.name = "palm";
palm.chlorophyll_rate = 1.6;
```

พิจารณาคำสั่ง move ดูผลการทำงาน



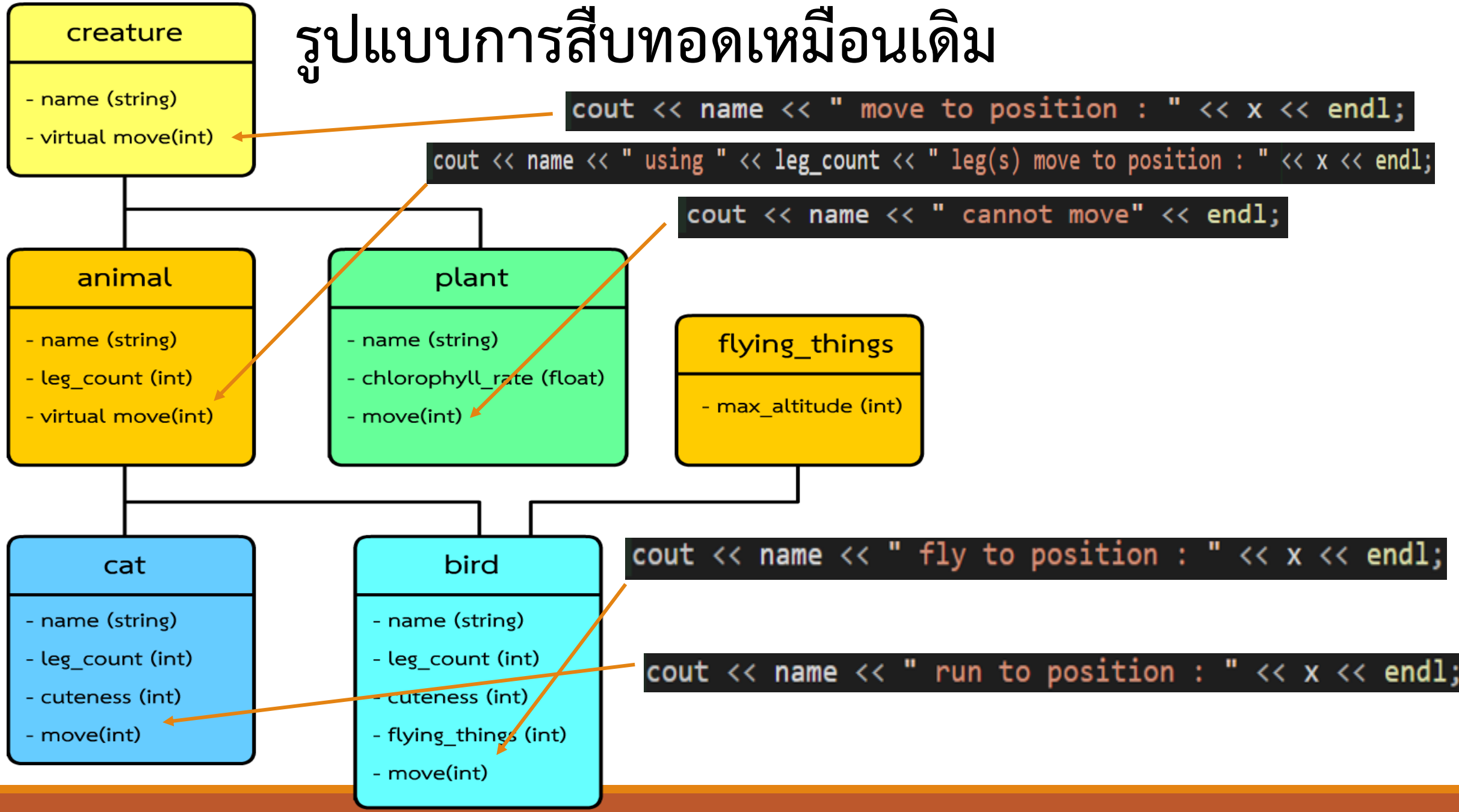

```
seagull.move(10);
Sphynx.move(10);
// boing747.move(10); error
alien.move(10);
snail.move(10);
palm.move(10);
```

```
seagull using 2 leg(s) move to position : 10
Sphynx using 4 leg(s) move to position : 10
alien move to position : 10
snail using 0 leg(s) move to position : 10
palm cannot move
```

โปรแกรม 4_5_ ... .CPP กำหนด move เพิ่มขึ้นในคลาส brid และ cat ดังนี้

```
class bird : public animal, public flying_things{
public :
    string origin;
    void move(int _x){
        x = _x;
        cout << name << " fly to position : " << x << endl;
    }
};

class cat : public animal{
public :
    int cuteness;
    void move(int _x){
        x = _x;
        cout << name << " run to position : " << x << endl;
    }
};
```

รูปแบบการสืบทอดเหมือนเดิม



เปรียบเทียบผลลัพธ์ก็มีเปลี่ยนแปลงตรง move( )

```
seagull.move(10);
Sphynx.move(10);
// boing747.move(10); error
alien.move(10);
snail.move(10);
palm.move(10);
```

```
seagull using 2 leg(s) move to position : 10
Sphynx using 4 leg(s) move to position : 10
alien move to position : 10
snail using 0 leg(s) move to position : 10
palm cannot move
```

4_4 ... CPP

```
seagull fly to position : 10
Sphynx run to position : 10
alien move to position : 10
snail using 0 leg(s) move to position : 10
palm cannot move
```

4_5 ... CPP

Conclusion

- composition vs inheritance
- Multiple level & Multiple inheritance
- Inheritance assignment
- Virtual function

LAB

แก้ไขโปรแกรมที่ 4_6_LAB.CPP ให้ได้คำตอบตามโจทย์ต้องการ

Class composition

processor

Attributes

- name (string)
- brand (string)
- max_speed(float)
- number_of_thread(int)

computing_device

Attributes

- cpu (processor)
- ram_capacity (float)
- storage_capacity (float)

Method

- float processing_power()

telephone

Attributes

- network (string)
- display_size (float)
- input (input_device)

Method

void virtual print()

Virtual function/methods

input_device

Attributes

- name (string)
- number_of_button (int)
- number_of_multitouch (int)

class processor

ที่อยู่ใน class computing_device *

class computing_device

float ram_capacity

class input_device

ที่อยู่ใน class telephone *

Object คลาส processor

Object คลาส input_device

```
labtop Hp_Envy("HP Envy 16-h0043TX", i5, 16, nb_keyboard, "windows");
```

สร้างใหม่ใน class labtop

string name

สร้างใหม่ใน class labtop

os_name

```
Hp_Envy.print();
```

void print( ){..}

ต้องสร้างใหม่ใน class labtop เพราะ Hp_Envy เป็น Object เกิดจาก class labtop

สร้างใหม่ใน class smartphone

class processor
ที่อยู่ใน class computing_device

class computing_device *

string name

Object คลาส processor

float ram_capacity

```
smartphone iphone4("iPhone 4", apple_a4, 0.512, cap_touch,"3G");
```

Object คลาส input_device

class telephone *

string network

```
iphone4.print();
```

void print( ){..}

class input_device
ที่อยู่ใน class telephone

ต้องสร้างใหม่ใน class smartphone เพราะ iPhone4 เป็น Object เกิดจาก class smartphone
และพิมพ์ให้ตรงกันกับผลลัพธ์ที่ต้องการ ** processor มีตัวแปร bard ที่ใช้แยก IOS และ Android ได้ **