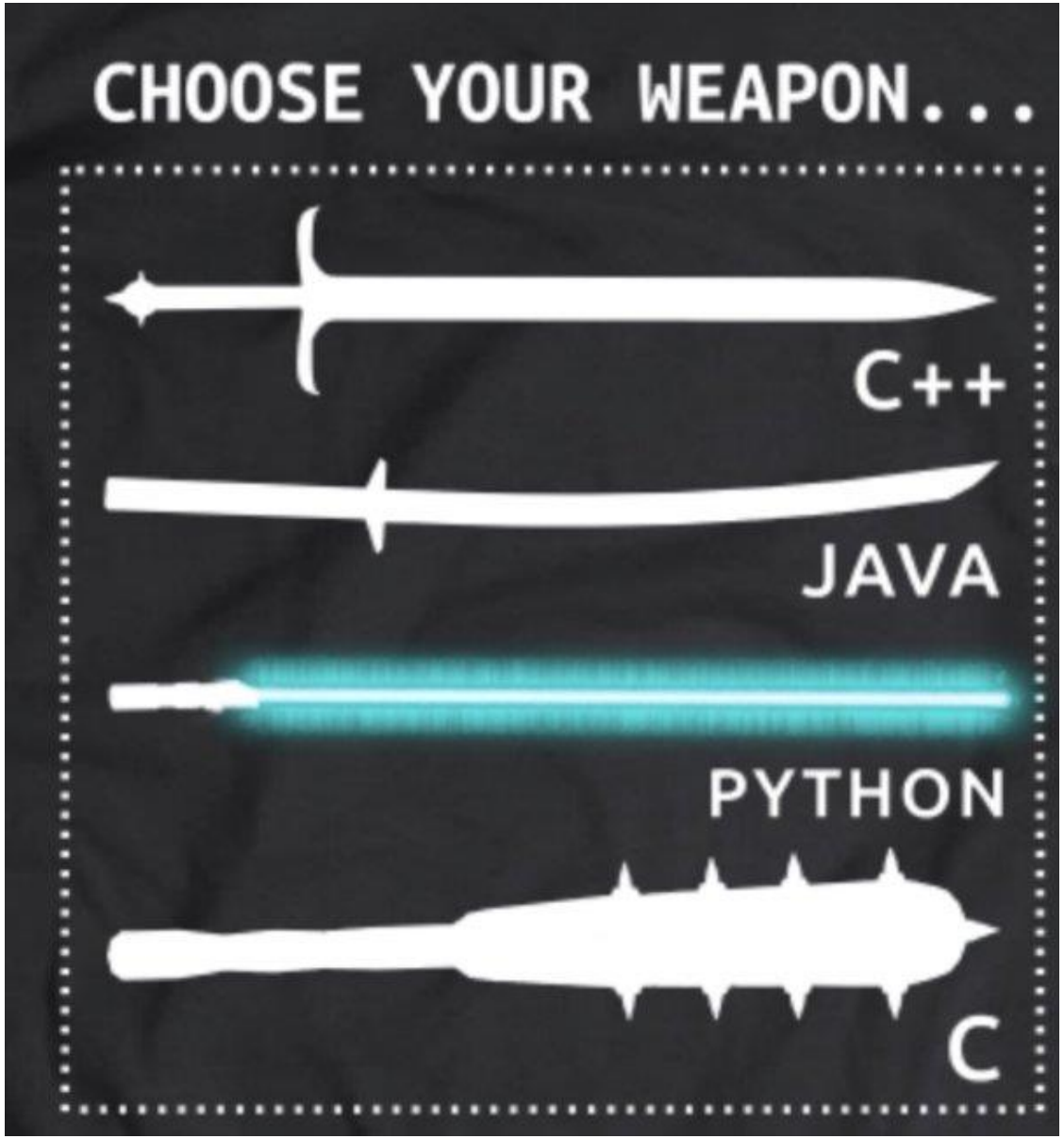
OOP & data struct

4. Inheritance con. & abstract

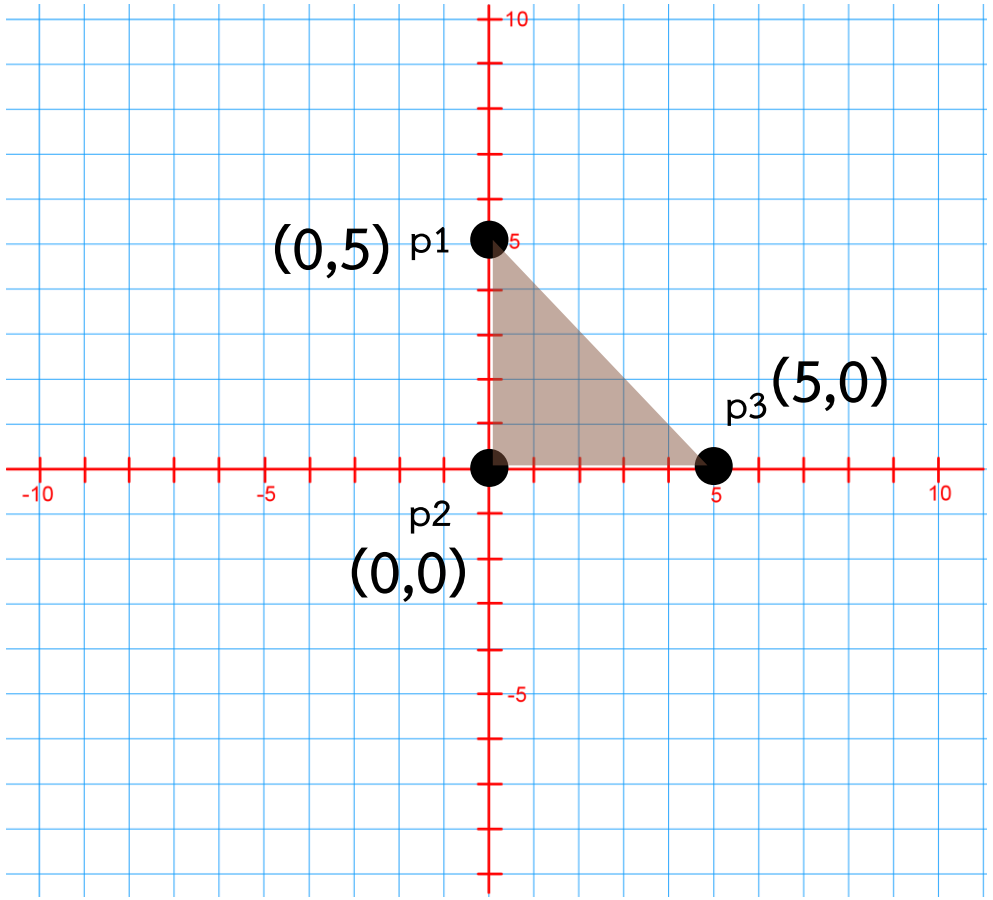
BY SOMSIN THONGKRAIRAT


CHOOSE YOUR WEAPON...
C++
JAVA
PYTHON
C

Class composition vs inheritance

Cartesian plane

- 2d point
- calculate area of triangle



Point class for collect point data

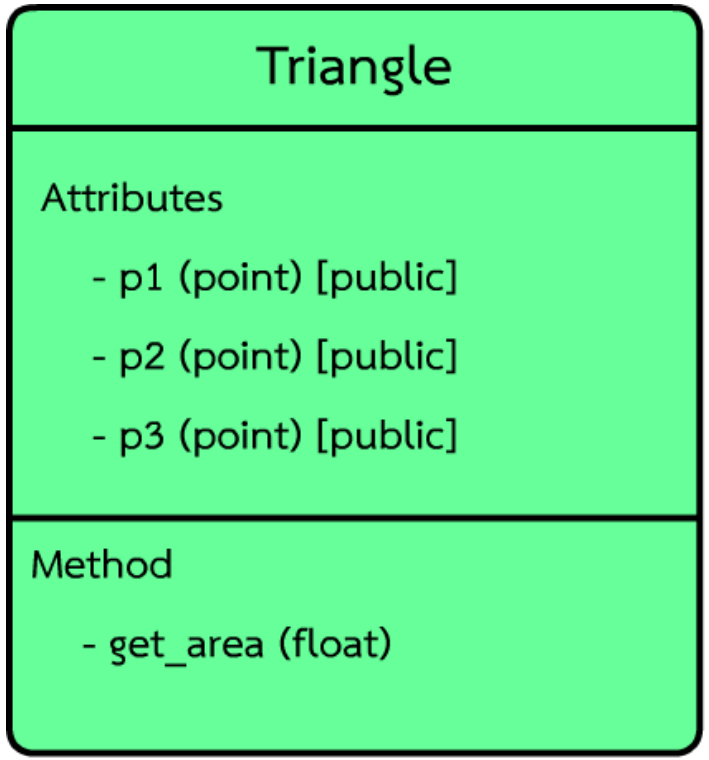
```
class point{
public :
    float x,y;

    point(){
        x = 0.00;
        y = 0.00;
    }

    point(float _x, float _y){
        x = _x;
        y = _y;
    }
};
```

Assign point data :

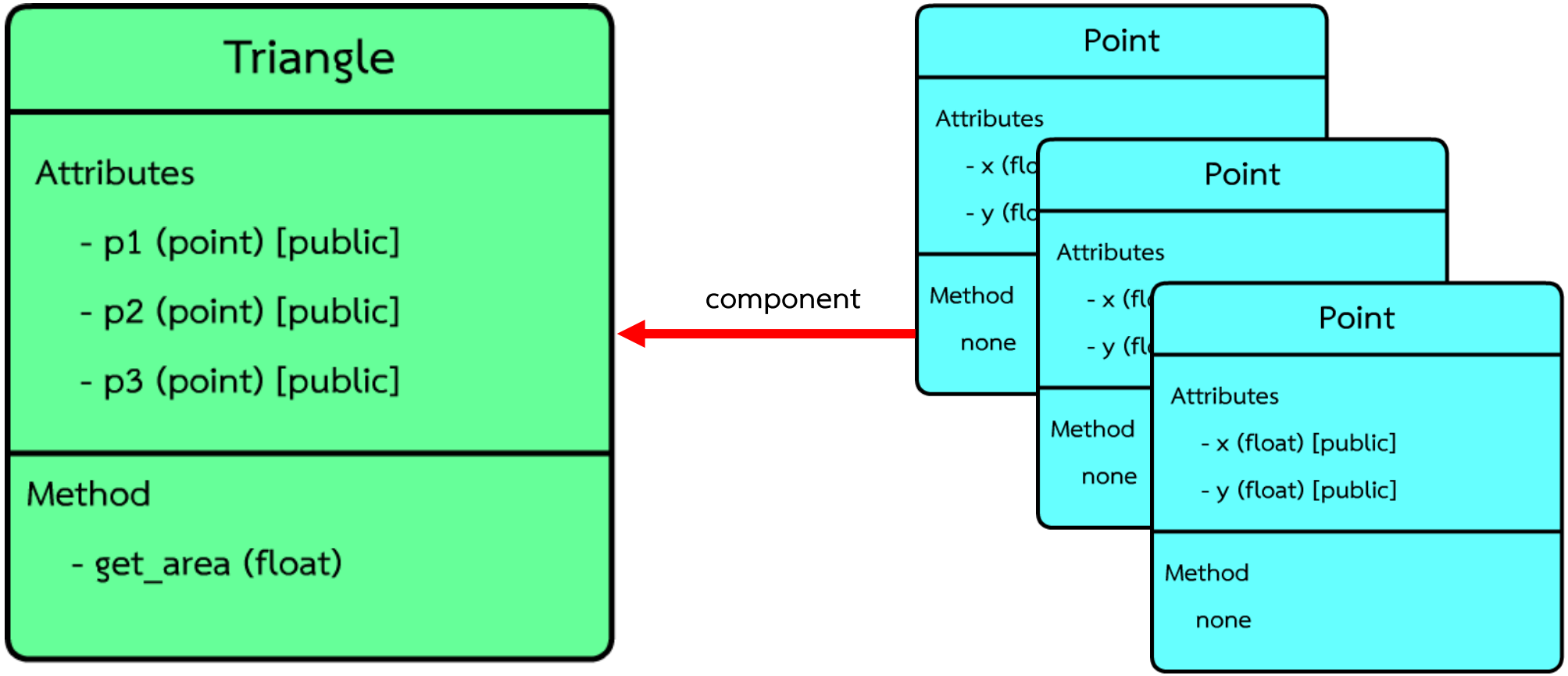
```
    point a1(0,0);
    point a2(0,5);
    point a3;
    a3.x = 5; a3.y = 0;
```

Triangle class

```
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)));
    }
};
```

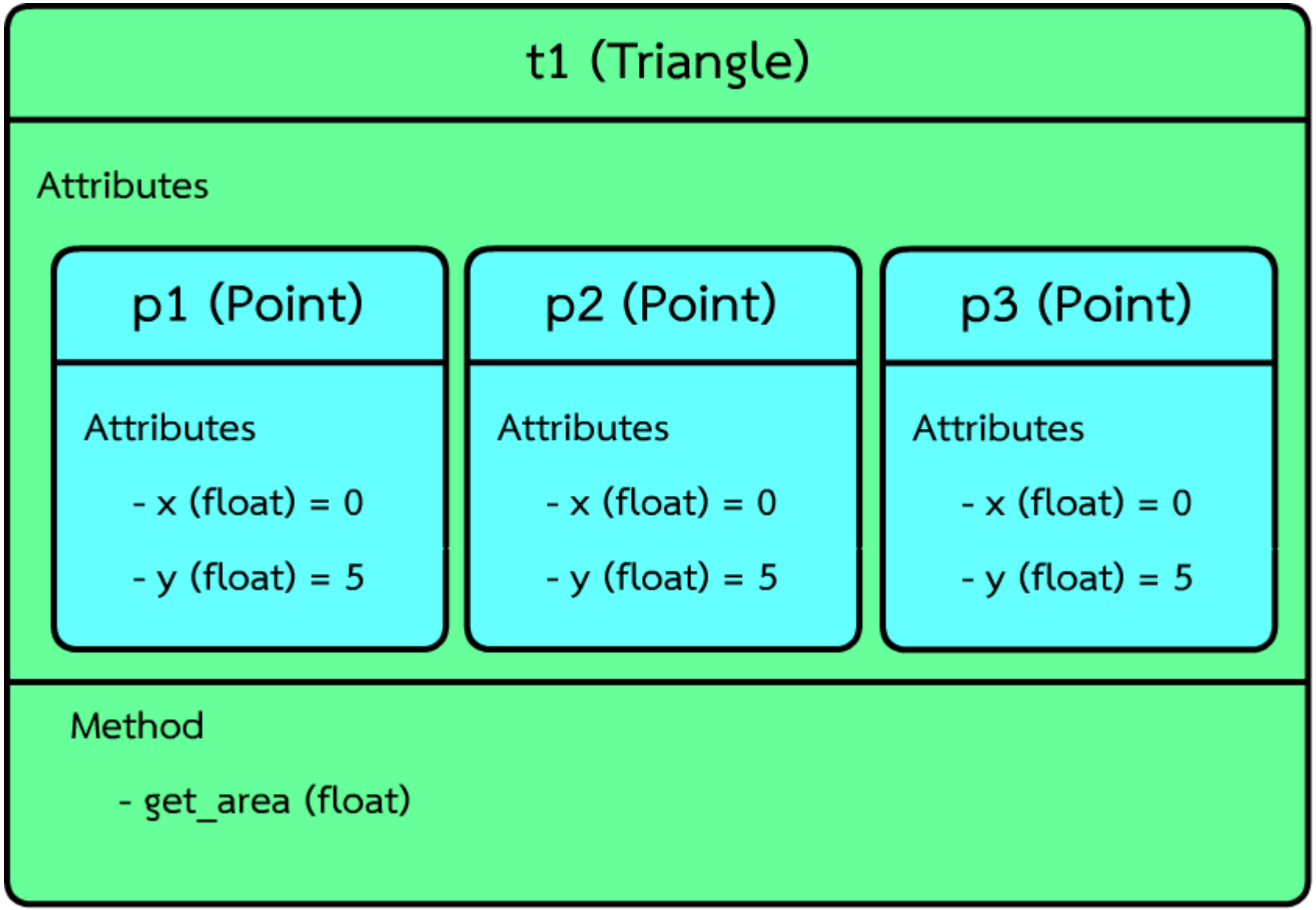


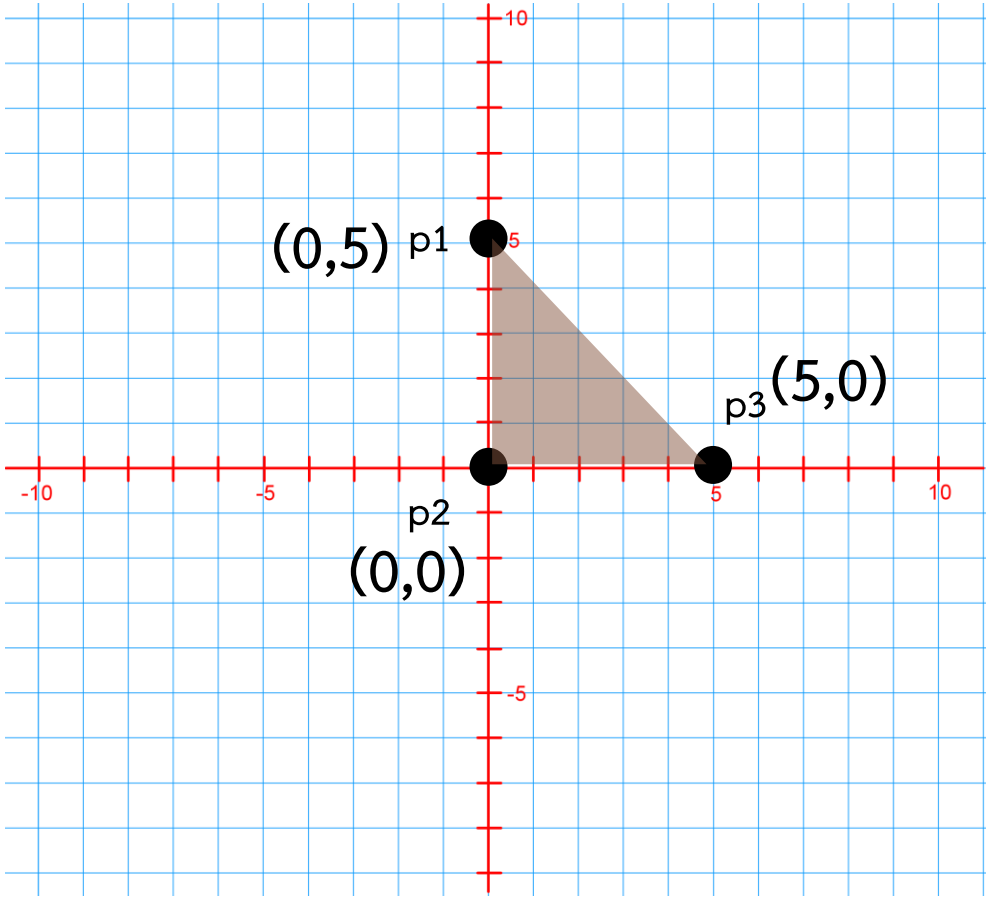
Class composition



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;
```



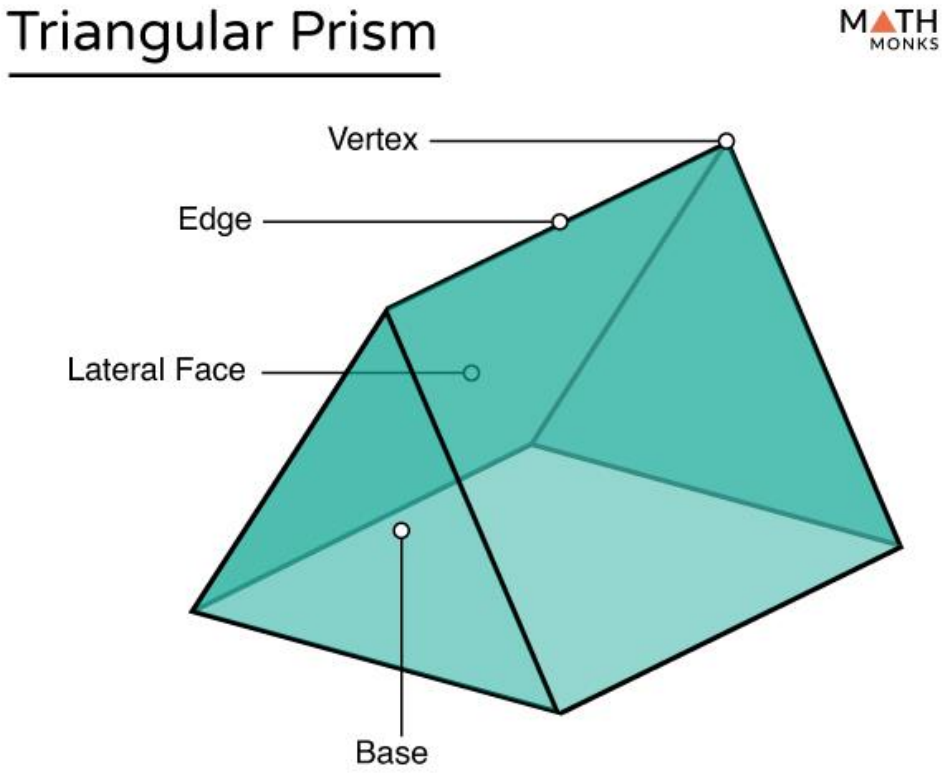



```
49  int main(){
50      
51      triangle t1;
52      t1.p1.x = 0; t1.p1.y = 5;
53      t1.p2.x = 0; t1.p2.y = 0;
54      t1.p3.x = 5; t1.p3.y = 0;
55
56      cout << t1.get_area() << " sq.unit" << endl;
57
```

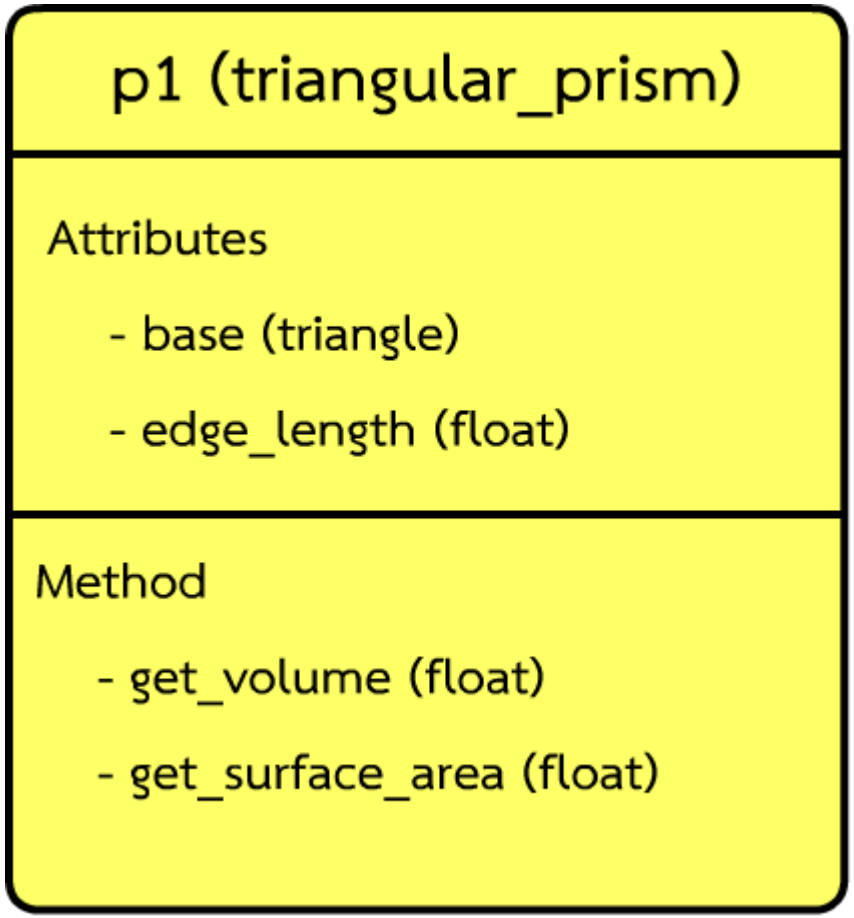
Result :

12.5 sq.unit

More complex example




```
class triangular_prism{
public :
    triangle base;
    float edge_length;

    float get_volume(){
        return base.get_area() * edge_length;
    }

    float get_surface_area(){
        return 0;
    }
};
```




```
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;

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

Inheritance not only one way to organize data!

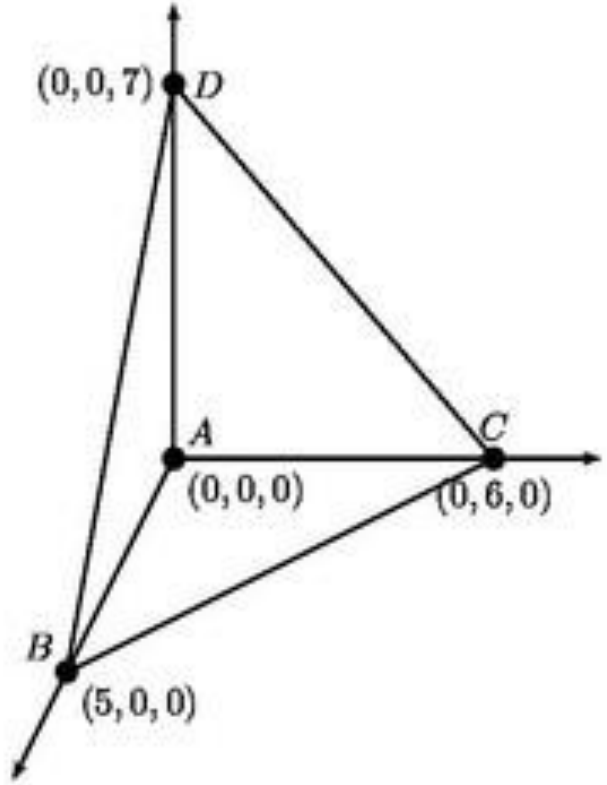
- composition or Inheritance or both
- สามารถเลือกใช้ได้ทั้ง composition หรือ Inheritance หรือทั้งคู่

Combine solution

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

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

P1 (x,y,z)

Tetrahedron (ทรงสี่หน้า)




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

}
```

Get volume ?

```
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;
    }
}
```

main

Code :

```
    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;
```

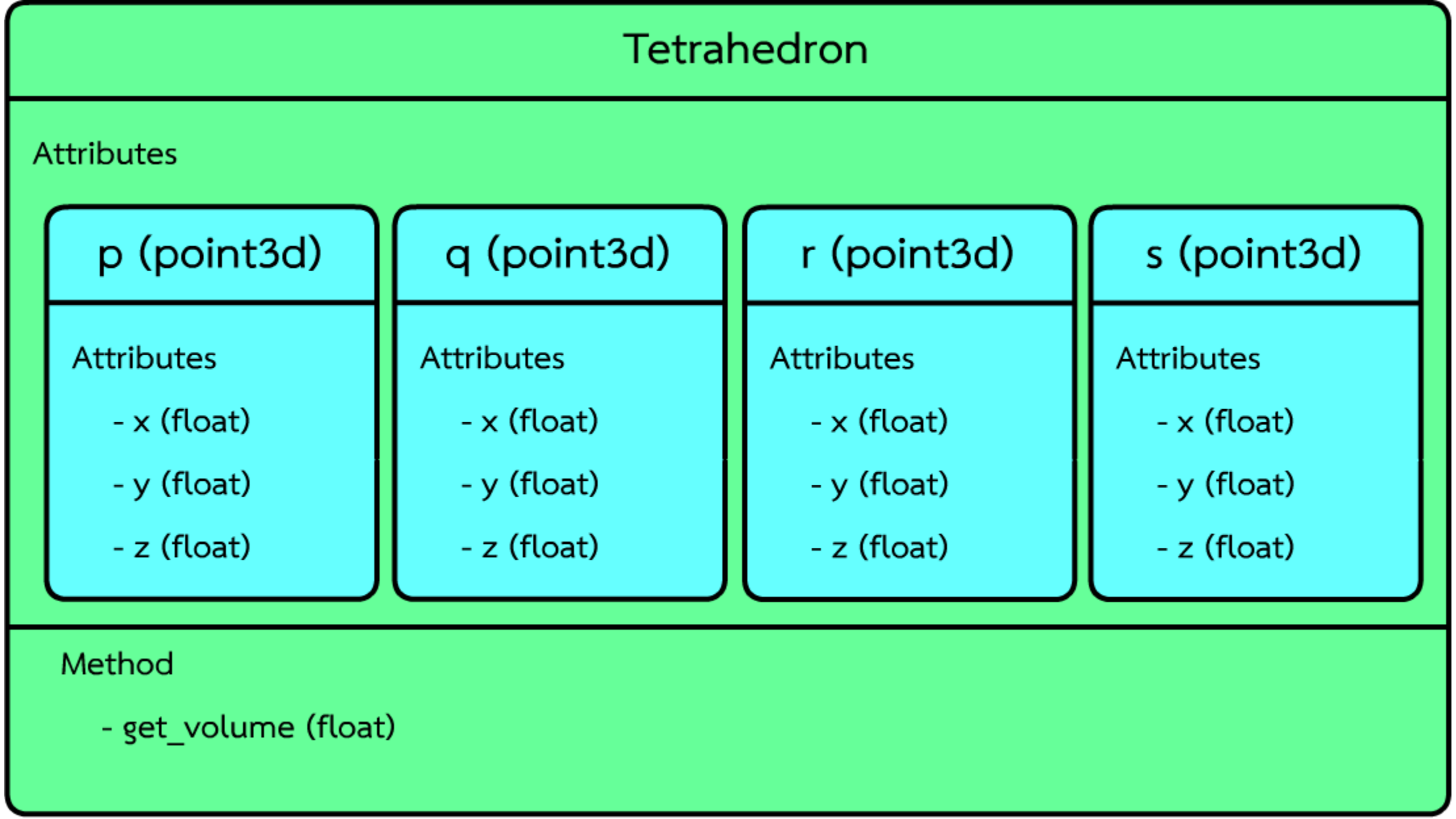
Result :

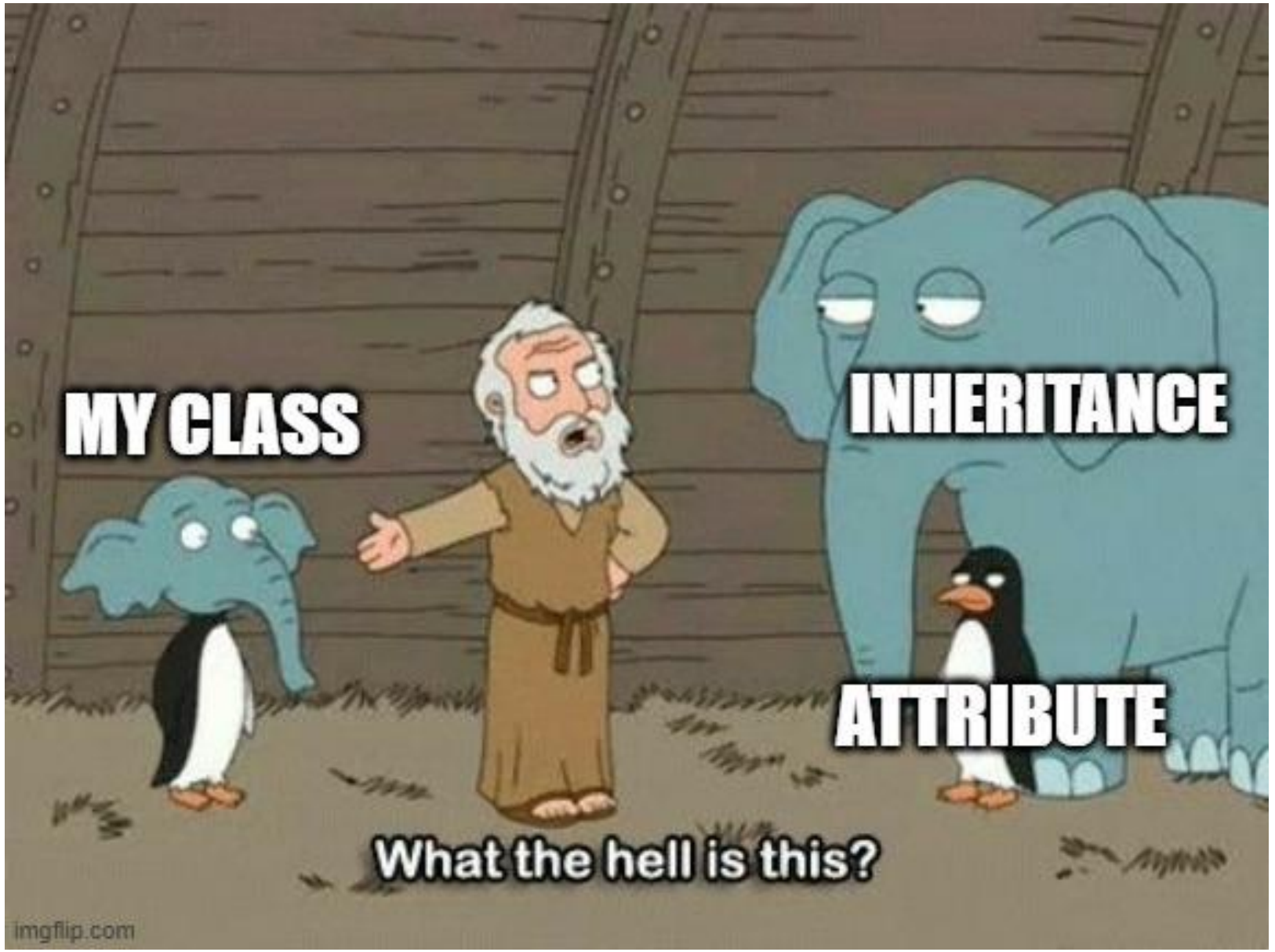
35 cubic unit

Combine method for fixability

- combine Inheritance with composition to create reasonable and non-redundance code

- ผสมระหว่าง Inheritance กับ composition ใน class เพื่อความสะดวกและความสมเหตุสมผลของ code


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)


MY CLASS
INHERITANCE
ATTRIBUTE
What the hell is this?
imgflip.com

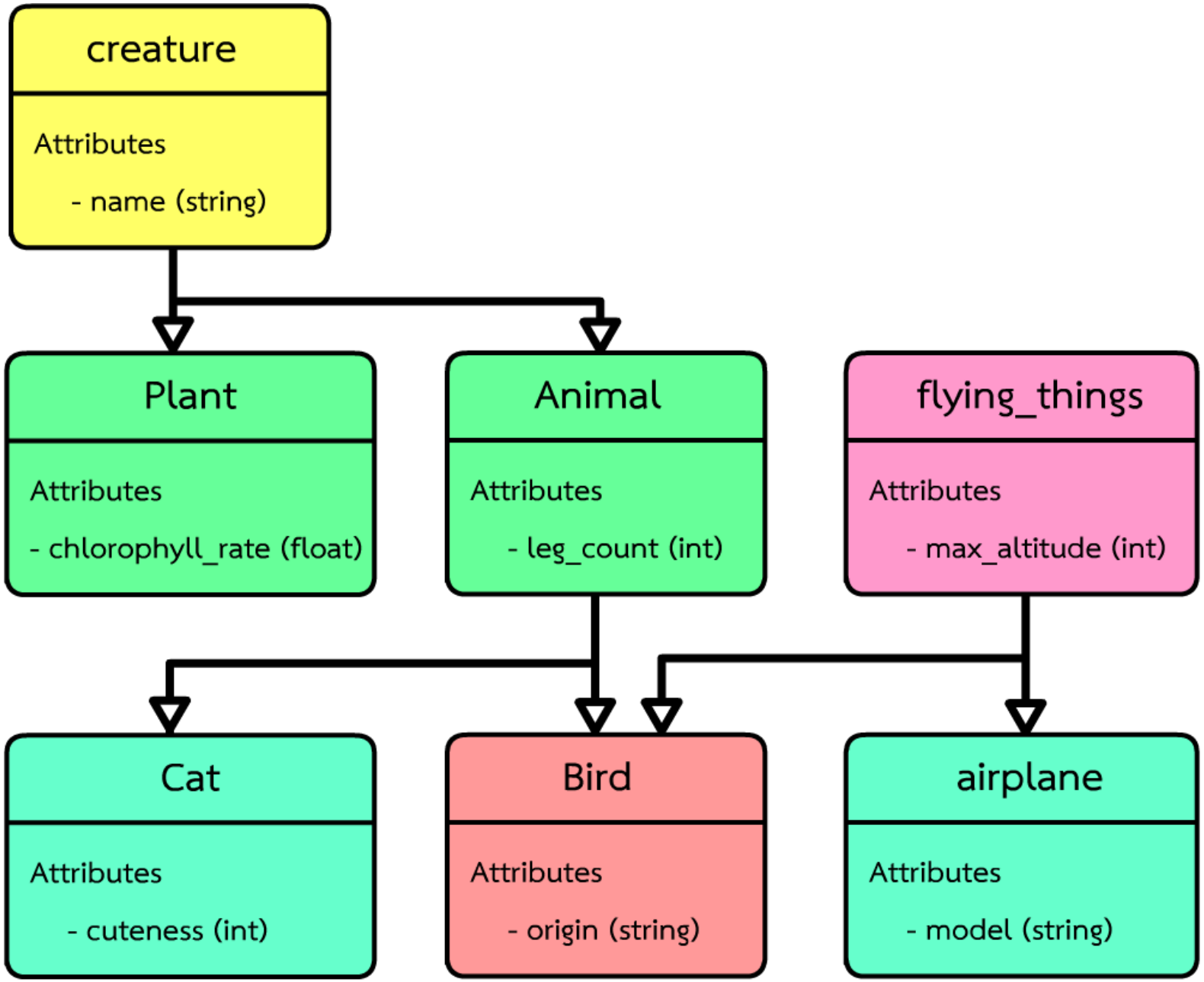
Multiple level & Multiple inheritance

- one can be inherited from serval classes

- inherited class can inherit to another class

- 1 class สามารถมรับการสืบทอดได้มากกว่า 1 class

- class ที่รับการสืบทอกมาแล้วสามารถส่งต่อการสืบทอดไปยัง class ลูกได้อีก


creature
Attributes
- name (string)
Plant
Attributes
- chlorophyll_rate (float)
Animal
Attributes
- leg_count (int)
flying_things
Attributes
- max_altitude (int)
Cat
Attributes
- cuteness (int)
Bird
Attributes
- origin (string)
airplane
Attributes
- model (string)

```
class creature{
    string name;
};

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

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

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{
    int cuteness;
};
```

Multiple inheritance

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

- Bird Class contain both of animal (leg_count) and flying_things property

- Class bird มีคุณสมบัติทั้ง animal (leg_count) และ flying_things (max_altitude)

Multiple level inheritance

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

- Cat class contain its grand parent class property (name)

- class cat มีคูณสมบัติของ class ปู่ (grand parent) นั้นก็คือ name

Code :

```
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;
```

Result :

No error

Quiz

Holding and casting

interesting questions from inheritance

- is seagull an animal?
- seagull (นกนางนวล) เป็นสัตว์หรือไม่?

- if yes, can we assign it to animal object
- ถ้าใช่ เราสามารถกำหนดค่านี้ให้กับตัวแปรประเภท animal ได้หรือไม่

Let’s try assign to object

- normally C is type restrict language, mean you can only assign variable with the same type of data such as string->string int->int float->float

- ภาษา C เป็นภาษาที่ type restrict หมายความว่าต้องกำหนดค่าของตัวแปรให้ตรงตามชนิดเท่านั้น เช่น string->string int->int float->float

Inheritance assignment

- parent class can be assigned as child class no matter how deep level of inherited such as creature object can be assigned to be a bird object , flying_things can be an airplane

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

Example :

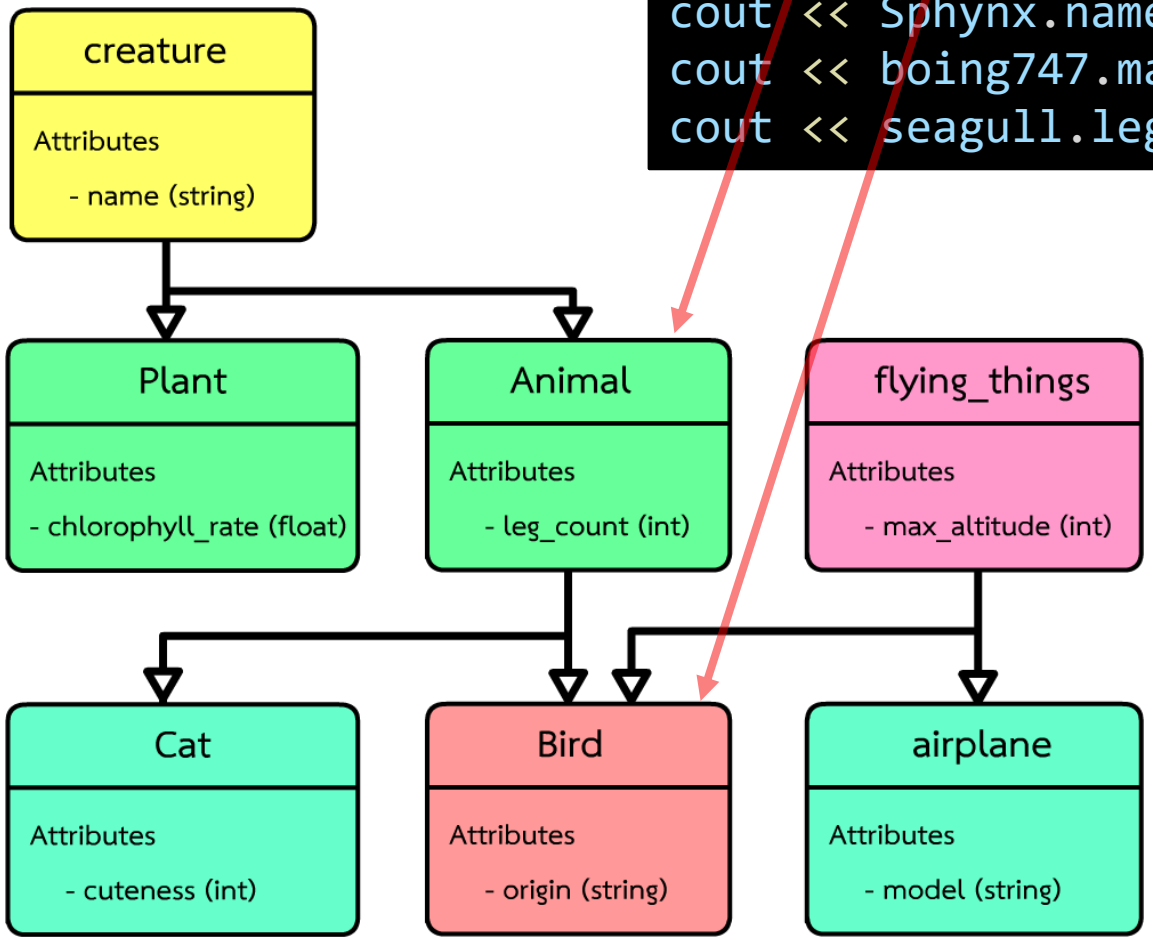
```
43    bird seagull;
44    seagull.name = "seagull";
45    seagull.leg_count = 2;
46    seagull.max_altitude = 100; // 100 meter
47    seagull.origin = "Australia";
48
49    cat Sphynx;
50    Sphynx.name = "Sphynx";
51    Sphynx.leg_count = 4;
52    Sphynx.cuteness = -10;
53
54    airplane boing747;
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
```

Attribute and method access?

- object can access attribute and method only define on its class but may contain extra data from child class

- object สามารถเข้าถึงเพียงแค่ attribute และ method ที่ประกาศไว้ใน class ของมันเท่านั้น แต่อาจมีข้อมูลเพิ่มเติมจาก class ที่อยู่ต่ำกว่าได้

```
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;
```



- Parent can assign to be a child object and access component
- Class ที่สูงกว่า สามารถกำหนดให้เป็น class ที่ต่ำกว่าได้ และยังสามารถเข้าถึงส่วนประกอบได้

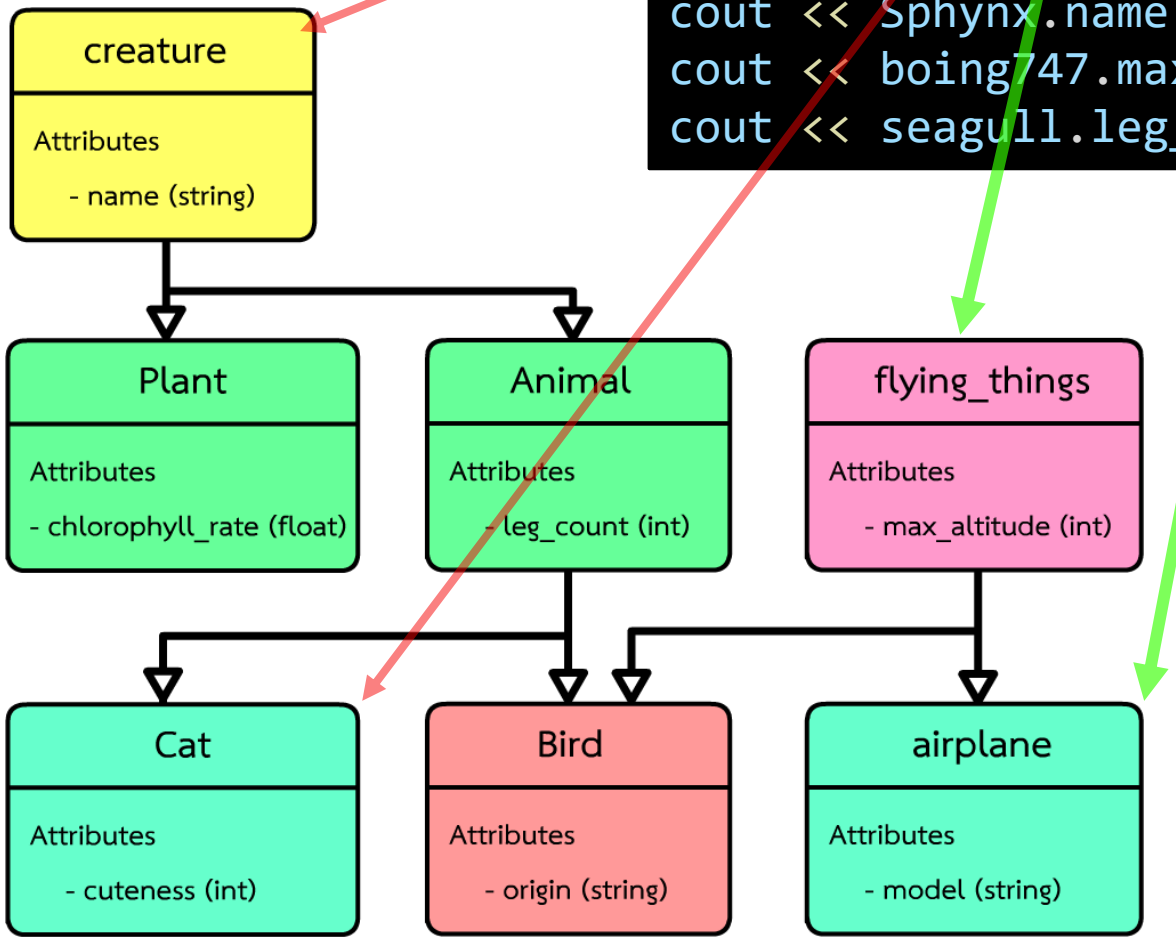
Result :

Sphynx : Sphynx

13610 : 13610

2 : 2

```
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;
```

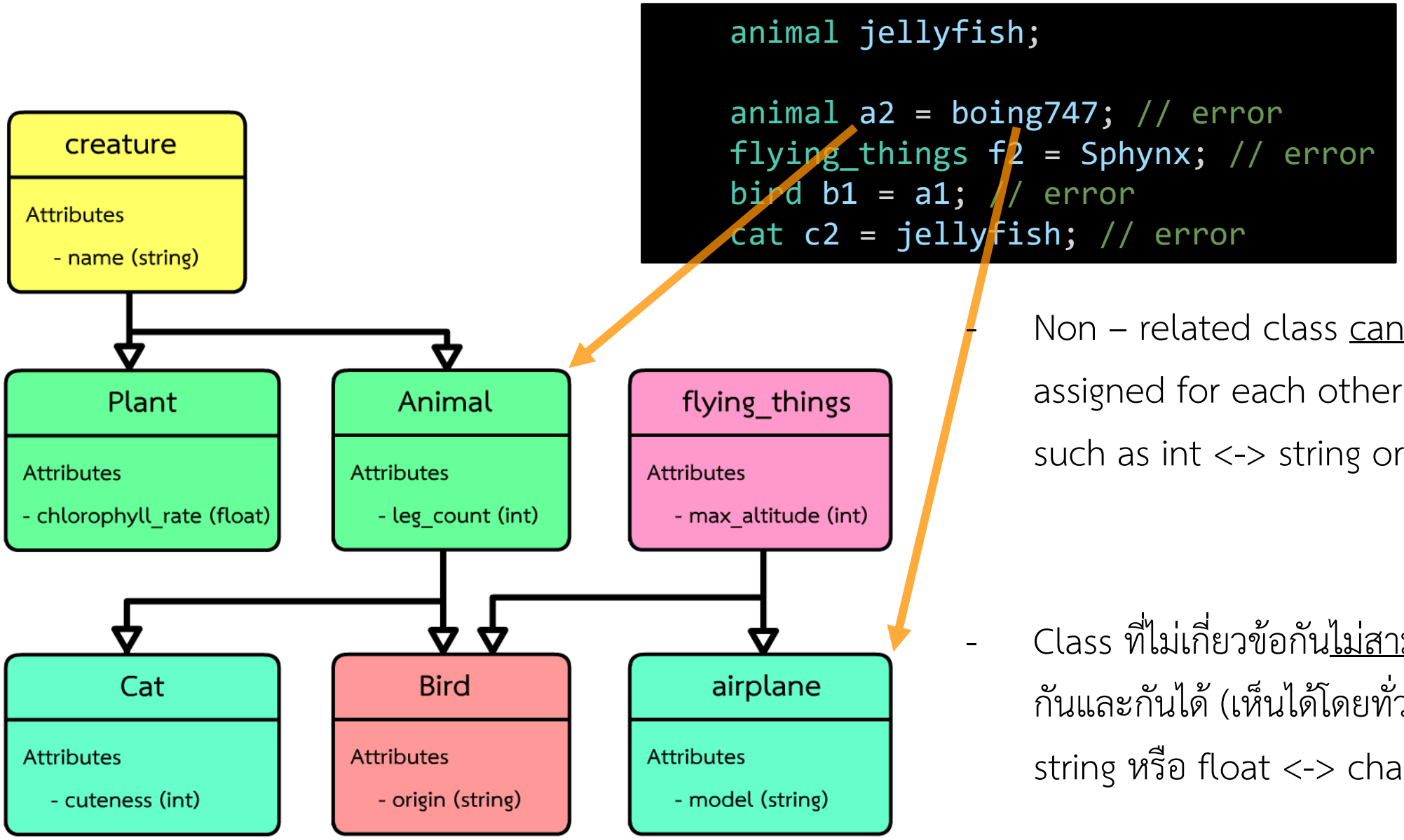


Result :

Sphynx : Sphynx

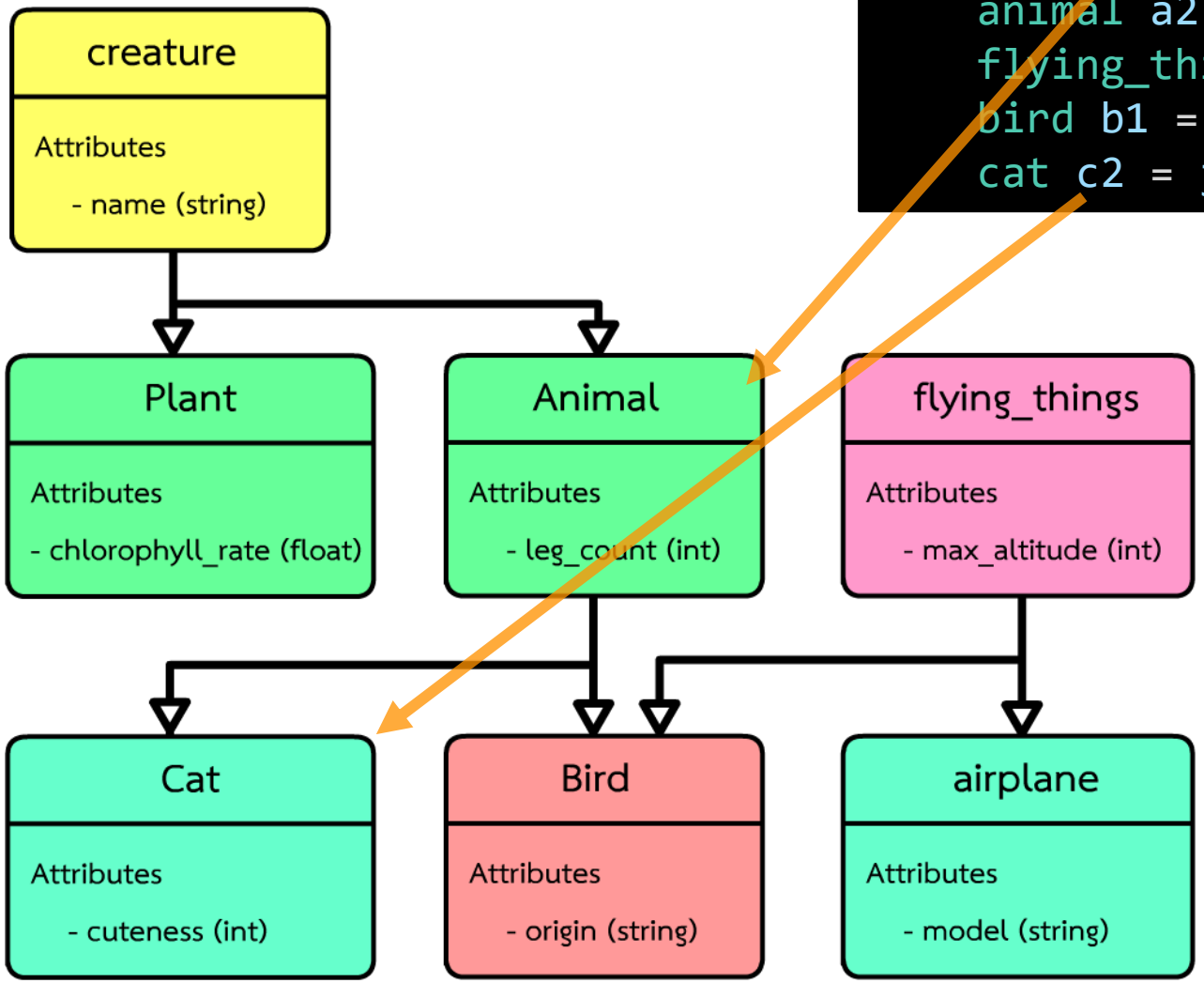
13610 : 13610

2 : 2



- Non – related class cannot be assigned for each other (obviously such as int <-> string or float <-> char)

- Class ที่ไม่เกี่ยวข้องกันไม่สามารถกำหนดค่าให้กันและกันได้ (เห็นได้โดยทั่วไป เช่น int <-> string หรือ float <-> char)



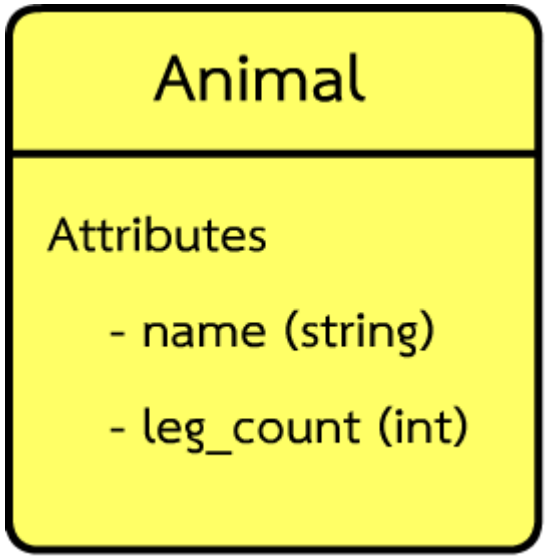

```
animal jellyfish;

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

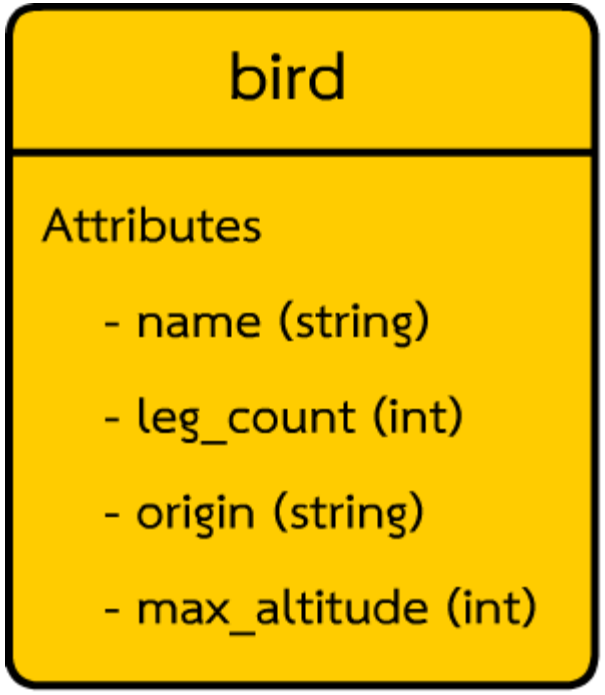
- Child class cannot assign to be a parent or higher class
- Class ที่ตำกว่าไม่สามารถกำหนดให้เป็น class ที่อยู่สูงกว่าได้

Quiz

So, how to access child data via parent

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

animal a1 = seagull; // no error
cout << a1.origin << endl; // error
```





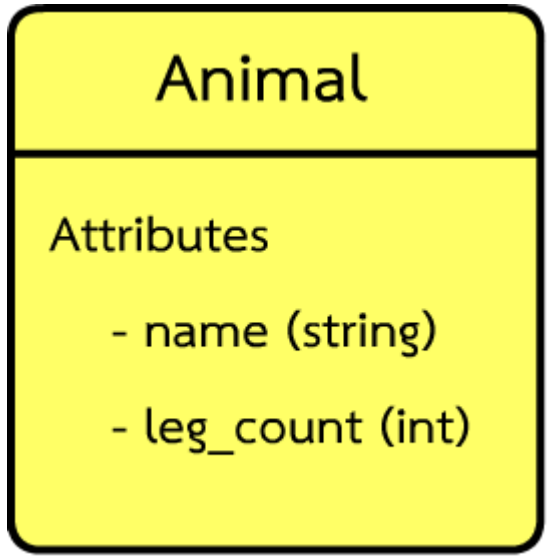
- How to access origin and max_altitude via a1 (animal class)
- จะเข้าถึง origin กับ max_altitude ผ่านทาง a1(animal class) ได้อย่างไร

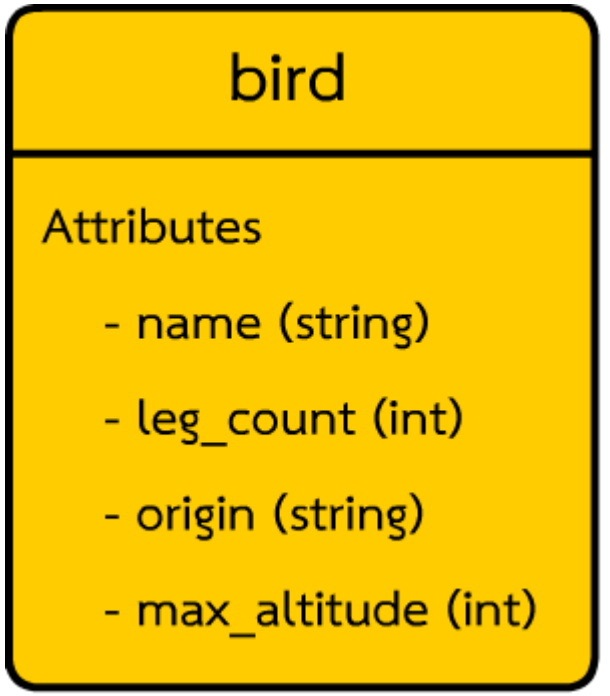
Pointer! (complicate)

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

- Cast parent pointer to child pointer

- Cast object ที่ class สูงกว่าให้เป็น class ที่ต้องการใช้ component





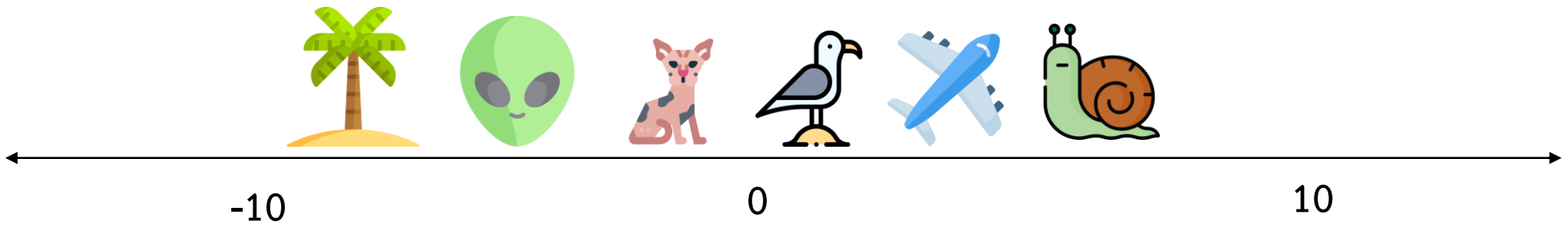
Virtual function (basic Abstraction & Polymorphism)

virtual function

- A function that has declared in base class and can be re-defined (overridden) by child class or more priority section

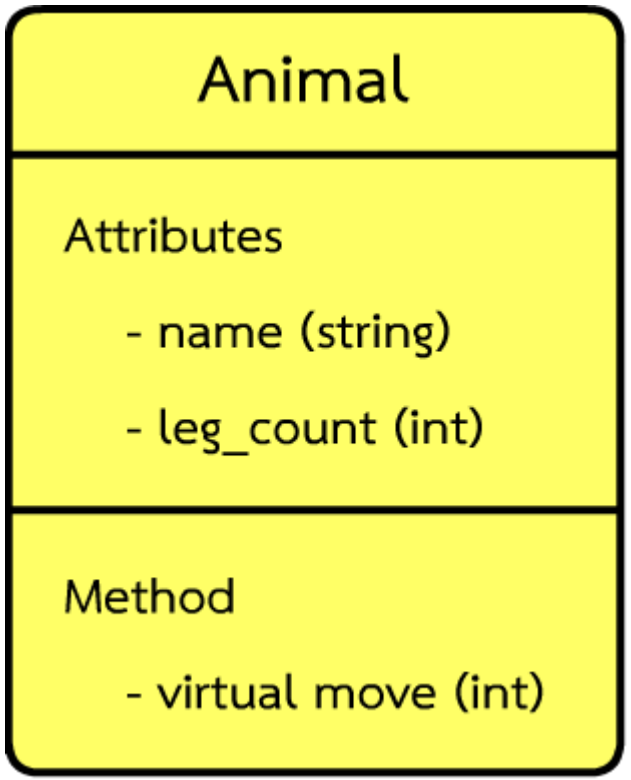
- คือ function ที่ถูกประกาศไว้ใน base class (parent class) และสามารถถูกเขียนทับ(override) ได้ใน class ลูก หรือ class ที่มีลำดับความสำคัญสูงกว่า

Example (1D position system)

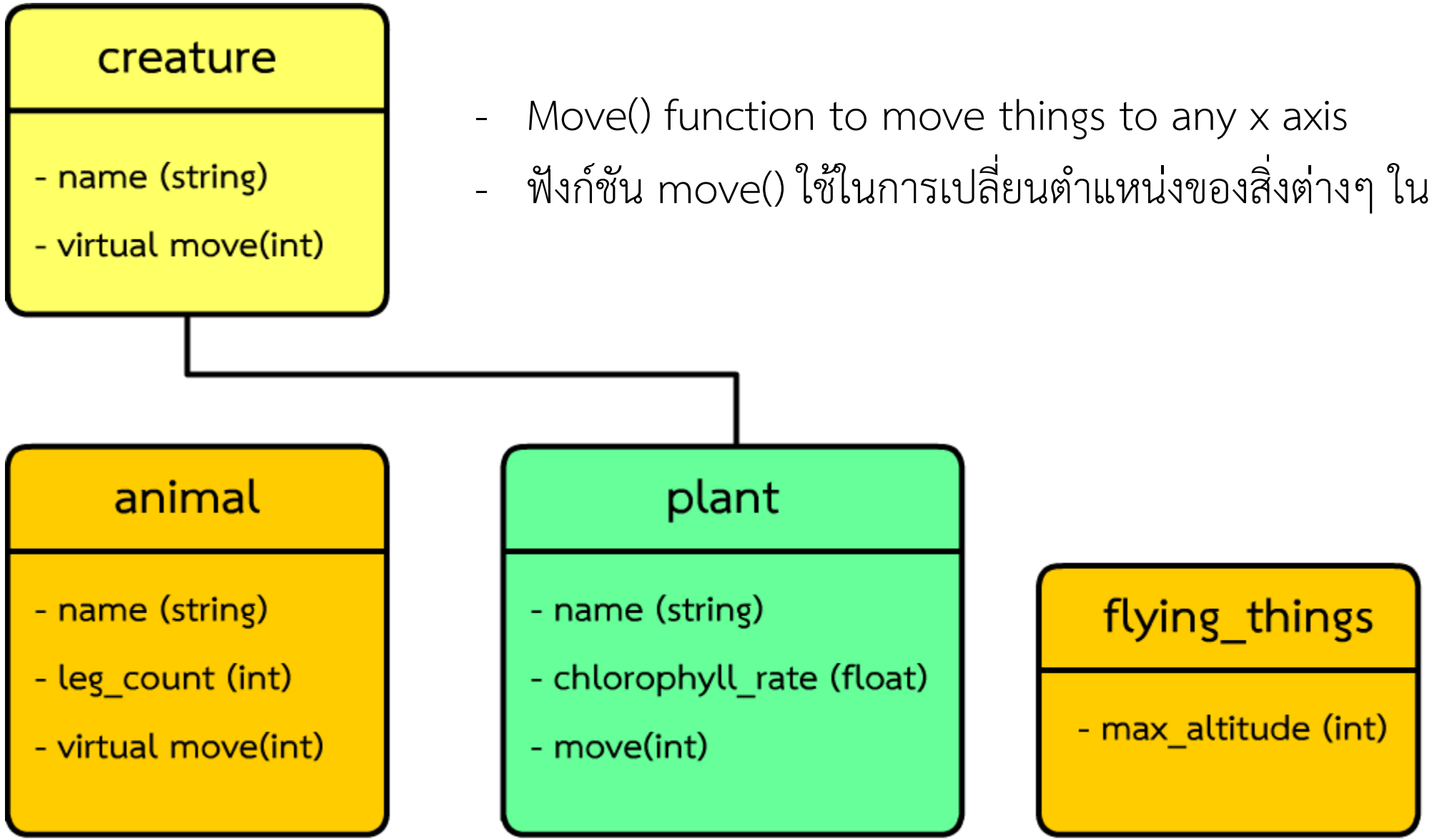


- Everything has position on 1d position system (X axis)
- ทุกอย่างมีตำแหน่งบนระบบ 1 มิติ (แกน X)

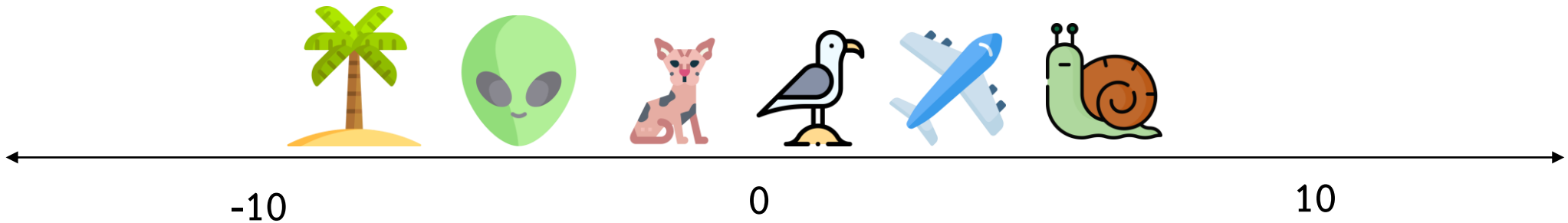
```
class creature{
public :
    string name;
    int x;

    creature(){
        x = 0;
    }

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



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

    move(int _x){
        x = _x;
        cout << name << " cannot move" << endl;
    }
};

class animal : public creature{
public :
    int leg_count;

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

};
```



- Move() function to move things to any x axis
- ฟังก์ชัน move() ใช้ในการเปลี่ยนตำแหน่งของสิ่งต่างๆ ใน แกน X




```
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 boeing747;
boeing747.model = "747";
boeing747.max_altitude = 13610;
```

Our Member

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

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

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

Code :

```
seagull.move(10);
Sphynx.move(10);
//boeing747.move(10); error
alien.move(10);
snail.move(10);
palm.move(10);
```

Result :

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

Another example

```
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;
    }
};
```

Code :

```
seagull.move(10);
Sphynx.move(10);
//boeing747.move(10); error
alien.move(10);
snail.move(10);
palm.move(10);
```

Result :

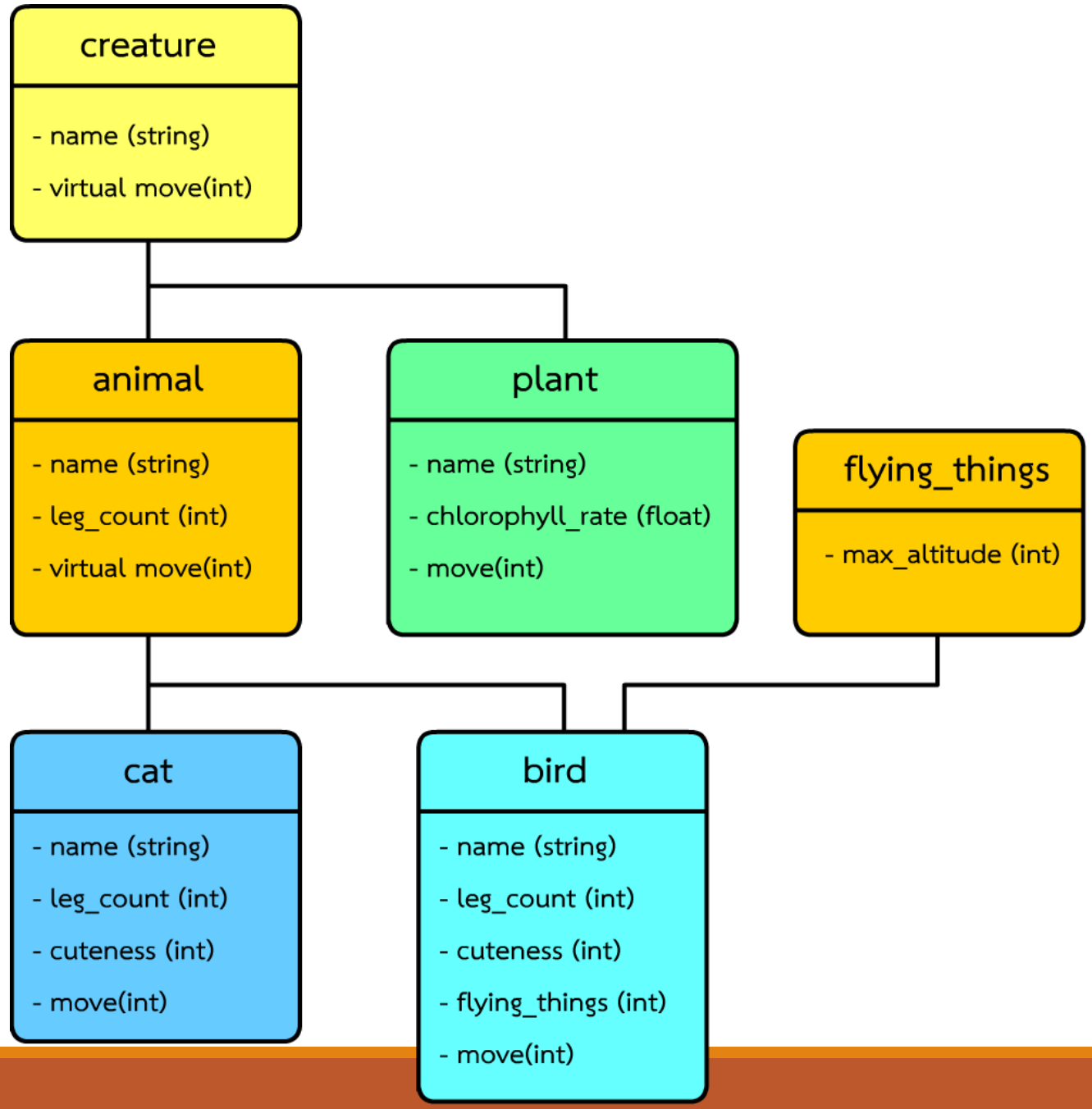
seagull fly to position : 10

Sphynx run to position : 10

alien move to position : 10

snail using 0 leg(s) move to position : 10

palm cannot move



Conclude

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

LAB

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()

input_device

Attributes

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