5. OOP & data struct

(1) Overloading/Overriding ของฟังก์ชั่น

(2) การกำหนดค่าเริ่มต้นให้ตัวแปรและพารามิเตอร์ของฟังก์ชั่น

---

(3) Abstraction class (คลาสโครงร่างต้นแบบ)

BY SOMSIN THONGKRAIRAT

ATTASIT LASAKUL (EDITION)

(1) การสร้าง function/methods ใหม่ในคลาสลูก

หากชื่อ function/methods เหมือนกันเรามีแบบการสร้างดังนี้

- Overloading function
- Overriding function (*Polymorphism*)

Overloading function/method

Function/Method ← ชื่อเหมือน แต่ พารามิเตอร์หรือการส่งค่ากลับไม่เหมือน

Overriding function/method

Function/Method ← เหมือนทั้งชื่อและพารามิเตอร์ (มักใช้กับ Class ลูก)

Overloading เหมือนกรณีของ constructor ของ C++

```cpp
class processor
{
public:
    string name;
    string brand;
    float max_speed; // CPU clock speed in Hz unit
    int number_of_thread;

    processor()
    { // default constructor
    }

    processor(string n, string b, float s, int t)
    { // 4 parameter constructor
        name = n;
        brand = b;
        max_speed = s;
        number_of_thread = t;
    }
};
```

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

ในภาษา C เดิม จะไม่สามารถมีฟังก์ชั่นที่ชื่อเหมือนกันได้

(ดังตัวอย่างโปรแกรมที่ 5_1.C)

C compile ไม่ผ่าน

```c
int multiply(int a,int b){
    int c = a * b;
    return c;
}
int multiply(int a,int b){
    int c = a * b;
    return c;
}
int multiply(int a,float b){
    int c = a * b;
    return c;
}
```

C compile ผ่าน

```c
int multiply_int_int(int a,int b){
    int c = a * b;
    return c;
}

float multiply_int_float(int a,float b){
    float c = a * b;
    return c;
}

float multiply_float_int(float a,int b){
    float c = a * b;
    return c;
}
```

Function/methods ที่อยู่ในคลาสเดียวกัน

```
int multiply(int a,int b){
    int c = a * b;
    return c;
}
int multiply(int a,int b){
    int c = a * b;
    return c;
}
int multiply(int a,float b){
    int c = a * b;
    return c;
}
```

แต่ในภาษา C++ ในคลาสเดียวกัน เราสามารถเขียนแบบ Overloading function ได้ เพราะจะแยกที่พารามิเตอร์ที่รับเข้ามาที่ไม่เหมือนกัน

แต่ในภาษา C++ ก็ไม่สามารถเขียนเหมือนทั้งชื่อฟังก์ชั่นและพารามิเตอร์ที่รับเข้ามาได้ (*)

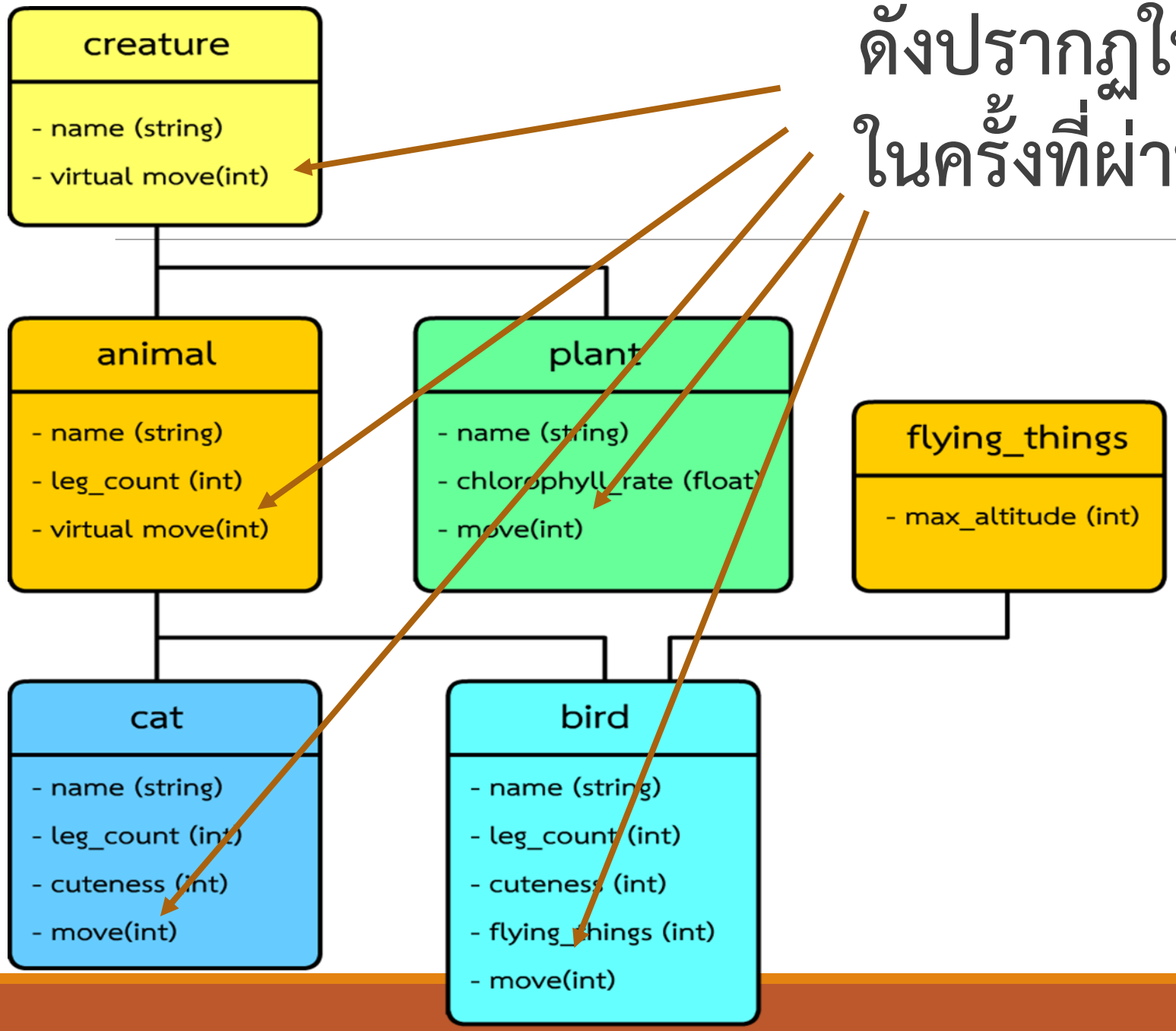
การเขียนฟังก์ชั่นแบบ Overriding methods จะสามารถทำได้บน C++ แต่ต้องอยู่คนละคลาสกัน (เช่น คลาสแม่ กับคลาสลูก)

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

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

ดังปรากฏในเรื่องการสืบทอด ในครั้งที่ผ่านมาดังรูป



ตัวอย่าง C++ Overloading function โปรแกรมที่ 5_2.CPP

```cpp
#include <iostream>
#include <string>
#include <cstdlib>
#include <ctime>
```

```cpp
class lotto
{
    string winner_number;

public:
    void random_number()
    { // real psudo random
        winner_number[0] = (rand() % 10) + '0';
        winner_number[1] = (rand() % 10) + '0';
        winner_number[2] = (rand() % 10) + '0';
        winner_number[3] = (rand() % 10) + '0';
        winner_number[4] = (rand() % 10) + '0';
        winner_number[5] = (rand() % 10) + '0';
    }

    void print()
    {
        cout << "winner price is " << winner_number << endl;
    }

    lotto()
    {
        winner_number = "000000";
    }
};
```

```cpp
int main()
{
    lotto a;
    srand(time(0));
    a.random_number();
    a.print();
}
```

Result :

```
winner price is 007999
```

5_4_....cpp

```
void random_number(int fifth_digit, int sixth_digit)
{ // lock 2 digit random
    random_number();

    if (fifth_digit >= 0)
        winner_number[4] = fifth_digit + '0';
    if (sixth_digit >= 0)
        winner_number[5] = sixth_digit + '0';
}

void random_number(int forth_digit, long fifth_digit, long sixth_digit)
{ // lock 3 digit random
    random_number();

    if (forth_digit >= 0)
        winner_number[3] = forth_digit + '0';
    if (fifth_digit >= 0)
        winner_number[4] = fifth_digit + '0';
    if (sixth_digit >= 0)
        winner_number[5] = sixth_digit + '0';
}
```

Overloading function

Overloading function/methods

```
a.random_number();
a.print();
a.random_number();
a.print();
a.random_number(5, 5);
a.print();
a.random_number(4, 5);
a.print();
a.random_number(6, 6, 6);
a.print();
a.random_number(7, 6, 6);
a.print();
```

Result :

```
winner price is 849103
winner price is 861287
winner price is 415055
winner price is 652145
winner price is 796666
winner price is 688766
```

ส่วน Overriding จะใช้ในกรณีเขียนเพื่อใช้เองใหม่ในคลาสลูก

```
class Parent
{
public:
    void showMessage()
    {
        cout << "This is the Parent class." << endl;
    }
};
class Child : public Parent
{
public:
    void showMessage()
    {
        cout << "This is the Child class." << endl;
    }
};
```

showMessage( ) ของ Base class

showMessage( ) ของ Derive class

5_2.cpp

```
Calling method from Parent object:
This is the Parent class.
Calling method from Child object:
This is the Child class.
Calling Parent method from Child object:
This is the Parent class.
```

```cpp
int main()
{
    Parent p1;
    Child c1;
    cout << "Calling method from Parent object:" << endl;
    p1.showMessage();
    cout << "Calling method from Child object:" << endl;
    c1.showMessage();
    cout << "Calling Parent method from Child object:" << endl;
    c1.Parent::showMessage();
    return 0;
}
```

เราสามารถเรียก showmessage( ) ของ Base Class ผ่าน Derived class ก็ได้

(2) วิธีการกำหนดค่าเริ่มต้นให้ตัวแปร

```
class calculator{
    //float result = 0;
    float result {0};

public :

    calculator(){
        result = 0;
    }

    void print(){
        cout << "the result is " << result << endl;
    }

};
```

assigned when declared

C++ Initializer

constructor

Result :

result is 0

เราสามารถกำหนดค่าเริ่มต้นให้พารามิเตอร์ของฟังก์ชั่นได้ด้วยเช่นกัน

Consider these method / พิจารณา 3 method นี้

```
void sum(float a, float b){
    result = a+b;
}
void sum(float a,float b,float c){
    result = a+b+c;
}
void sum(float a,float b,float c,float d){
    result = a+b+c+d;
}
```

```
    calculator c;
    c.sum(11,12,13);
    c.print();
    c.sum(11,12,13,0);
    c.print();
```

ให้ผลลัพธ์เหมือนกัน

Result :

the result is 36

the result is 36

กำหนดค่าเริ่มต้นไว้เองเลย

```
void sum(int a, int b = 5)
{
    cout << "sum(int, int) called : " << a + b << endl;
}
```

2 way to call this method

มี 2 วิธีทีจะเรียกใช้ method นี้

```
int main()
{
    sum(2);
    sum(2,3);
```

```
sum(int, int) called : 7
sum(int, int) called : 5
```

โปรแกรมที่ 5_3.CPP เป็นโปรแกรม คำนวณการบวกเลขที่มีค่า พารามิเตอร์หลากหลายแบบ Overloading และมีการกำหนดค่าเริ่มต้นไว้ก่อนด้วย

```
void sum(float  a, float b, float c, float d = 0){
      result = a + b + c + d;
  }
```

```
calculator c;
c.sum(11,12,13);
c.print();
c.sum(11,12,13,14);
c.print();
```

```
void sum(11,12,13,0){
    result = 11+12+13+0;
}
```

```
void sum(11,12,13,14){
    result = 11+12+13+14;
}
```

Result :

the result is 36

the result is 50

ดังนั้นอาจสร้าง function/method รอรับพารามิเตอร์ตามความเป็นไปได้หลายๆแบบใน function เดียวก็ได้

```
void sum(float a, float b = 1,float c = 0, d = 0){
    result = a+b+c+d; float
}
```

เวลาเรียก Function ก็สามารถส่งพารามิเตอร์ได้หลายแบบดังนี้

```
void sum(float, float, float, float);
void sum( float, float, float);
void sum( float, float);
void sum(float);
```

(3) Abstract class

- เป็น class ที่ออกแบบมาเพื่อการสืบทอด (inherit) และชี้ไปยัง object อื่น เท่านั้นไม่สามารถสร้าง object จาก class นี้ได้

- สร้างได้โดยการสร้าง pure virtual function อย่างน้อยหนึ่งฟังก์ชันใน class

Pure virtual function คืออะไร??

- คือฟังก์ชันที่มีเพียงแค่กำหนดให้เท่ากับ “0” (ไม่ต้องเขียนส่วนเนื้อหา)

Pure virtual function syntax

- กำหนดส่วนของ declaration ให้เท่ากับศูนย์

```
class creature{
public :
    string name;
    int x = 0;

    virtual void move(int _x) = 0;
};
```

```
class creature{
public :
    string name;
    int x = 0;

    virtual void move(int _x) = 0;
};
```

ตอนนี้ class creature เป็น abstract แล้วไม่สามารถนำไปสร้าง Object ได้

main() {

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

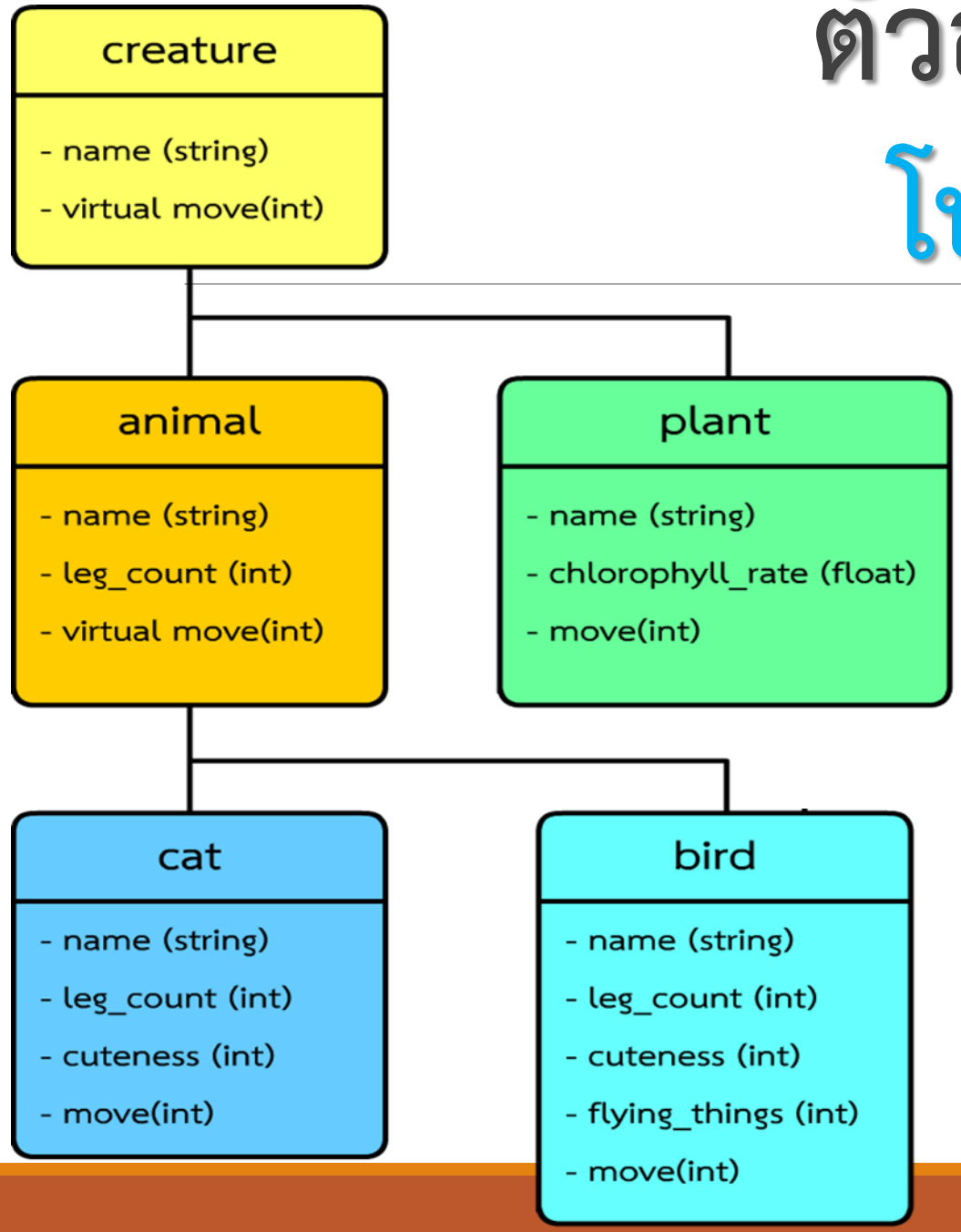
```
Error :
cannot declare variable
 'alien' to be of abstract type
'creature'
```

Abstract class มีไว้เพื่ออะไร?

- เพื่อเป็นแนวทางเดียวกันในการพัฒนา class ลูกที่อยู่ต่ำกว่า
- เพื่อกำหนดแนวทางหรือกฎของการพัฒนาเป็นลำดับชั้น
- เพื่อสร้าง class กลางที่สามารถกำหนดให้เป็น object ใดก็ได้ในระบบ

ตัวอย่าง Abstract class

โปรแกรมที่ 5_4.CPP



การใช้งาน virtual function จะทำให้คุณสมบัติ polymorphism เป็นการพร้องรูปที่สมบูรณ์ คือสามารถรัน function/methods ของคลาสลูกได้ถูกต้องทั้งโดยผ่าน Object และ Pointer

conclude

- Overloading, Overriding function
- Default parameter & default initialization
- Pure virtual function & abstract class

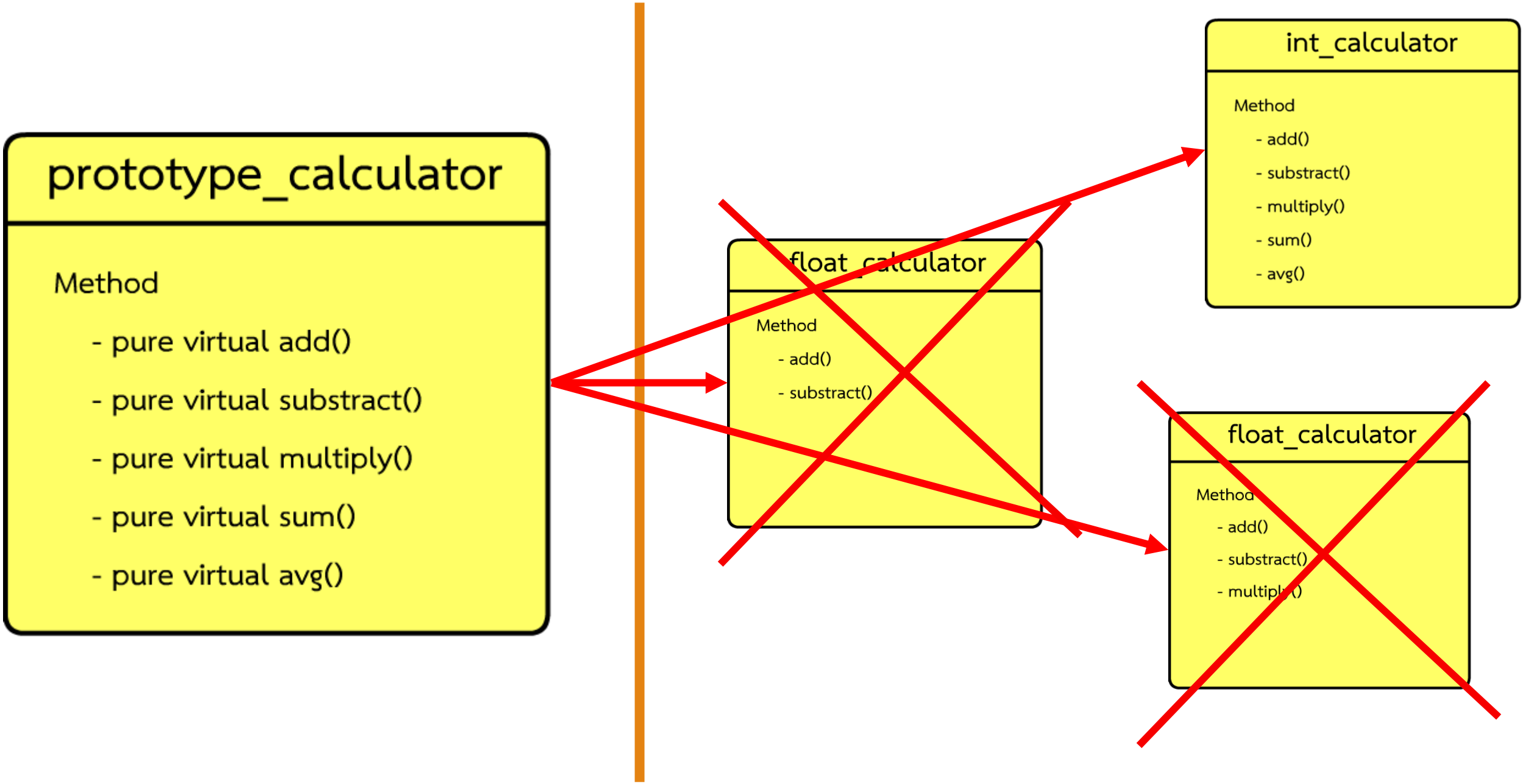
LAB

แก้ไขโปรแกรมเพื่อให้ได้คำตอบตามต้องการ

1) แก้ไขโปรแกรม 5_5_lab.cpp ที่ให้ไว้ เพื่อให้ได้คำตอบดังนี้

(ห้ามแก้ไข class prototype_calculator)

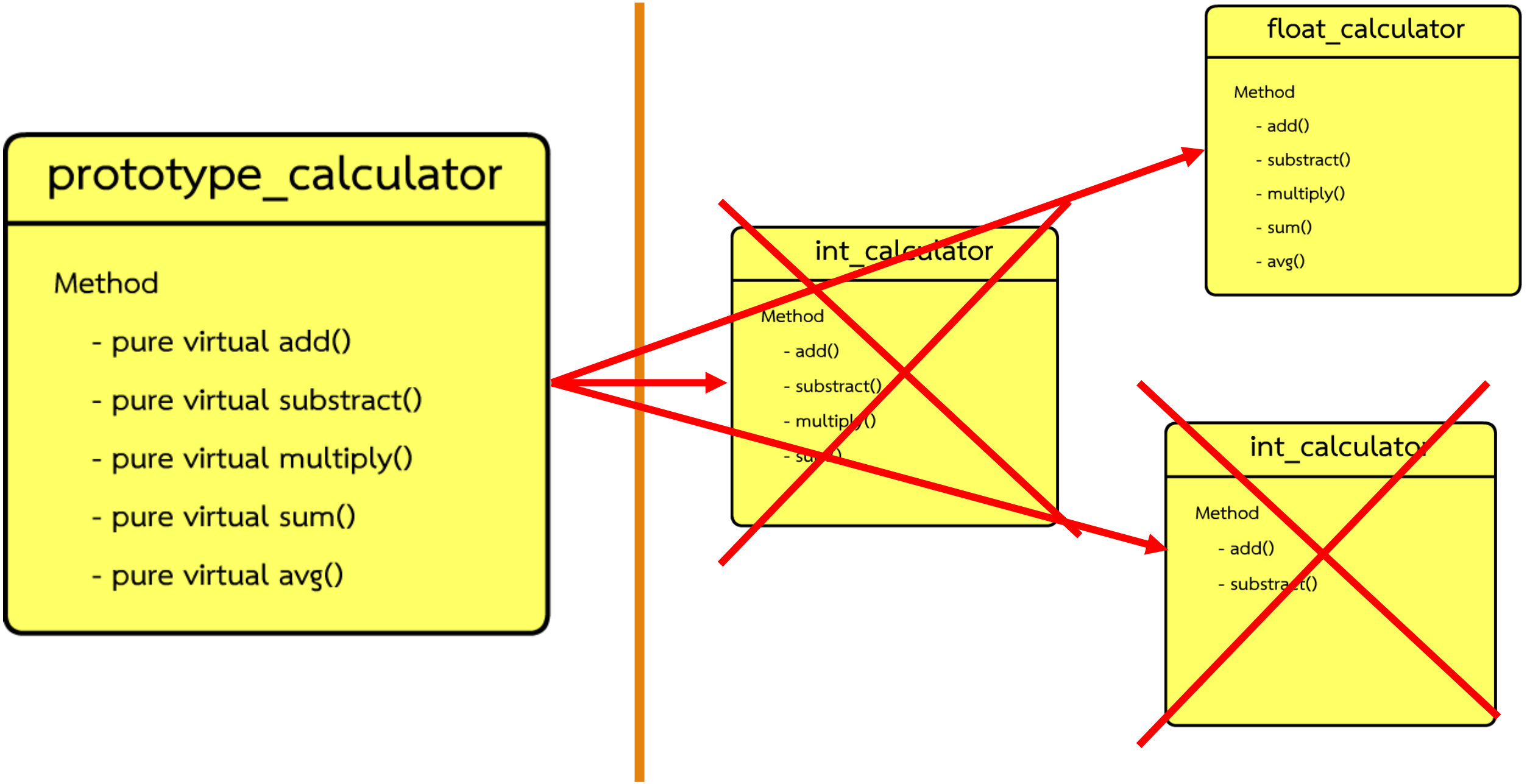
```
--add--:--substract--:--multiply--
   15   :          -3   :       54
-----------------sum-----------------
55,45,36,28,21,15,10,6,3,1
```


prototype_calculator
Method
- pure virtual add()
- pure virtual substract()
- pure virtual multiply()
- pure virtual sum()
- pure virtual avg()
float_calculator
Method
- add()
- substract()
int_calculator
Method
- add()
- substract()
- multiply()
- sum()
- avg()
float_calculator
Method
- add()
- substract()

1) แก้ไขโปรแกรม 5_6_lab.cpp ที่ให้ไว้ เพื่อให้ได้คำตอบดังนี้

(ห้ามแก้ไข class prototype_calculator)

```
 -[add]- : -[substract]- : -[multiply]-
  15.00  :          -3.00  :        54.00
--------------------sum-----------------
55.00,45.00,36.00,28.00,21.00,15.00,10.00,6.00,3.00,1.00
----------------------avg---------------
 3.00, 2.50, 2.00, 1.50, 1.00
```


prototype_calculator
Method
- pure virtual add()
- pure virtual substract()
- pure virtual multiply()
- pure virtual sum()
- pure virtual avg()
int_calculator
Method
- add()
- substract()
float_calculator
Method
- add()
- substract()
- multiply()
- sum()
- avg()
Method
- add()

สรุปการพิมพ์ตัวเลขใช้ cout<<fixed<<setprescistion(2)<<endl;

กำหนดแสดงทศนิยมสองตำแหน่ง เขียนไว้ครั้งเดียวเท่านั้น

cout<<setw(7)<<12.1245<<endl;

กำหนดจองพื้นที่อักษรจำนวน 5 อักษร ต้องเขียนไว้หน้าข้อมูลแต่ละข้อมูล