# OOP & data struct

## 5. Polymorphism & abstraction

BY SOMSIN THONGKRAIRAT

## Polymorphism

- override function

- default parameter

## Back to constructor

```
anime(){
    cout << "this is default constructor" << endl;

    playing_episode = 1;
    playing_sec = 0;
}

anime(string _name,string _author,int _ep,int length){ // 4 parameter function
    cout << "this is constructor for " << _name << endl;

    full_name = _name;
    author = author;
    total_episode = _ep;
    length_per_episode = length;

    playing_episode = 1;
    playing_sec = 0;
}
```

## How compiler select constructor

```
    anime a1("The Melancholy of Haruhi Suzumiya","Nagaru Tanigawa",24,1200);

    anime a2;
    a2.full_name = "Spy x Family Part 1";
    a2.author = "Tatsuya Endo";
    a2.total_episode = 12;
    a2.length_per_episode = 1440;
```

Yes! a1 has 4 parameter (string,string,int,int)

a2 has 0 parameter (default constructor)

this constructor is morphing!

## What about normal function (C)

```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;
}
```

- compile ผ่าน

## How about this (in C)

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

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

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

```
conflicting types for 'multiply’
- compile error
- same name function (C)
- function ชื่อซ้ำกัน (C)
```

Interesting Question is / คำถามคือ

- are these all the same function?
- ทั้งหมดนี้คือฟังก์ชันเดียวกันหรือไม่?

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

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

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

are these all the same function?

ทั้งหมดนี้คือฟังก์ชันเดียวกันหรือไม่?

In the terms of purpose : Yes

ในแง่ของจุดประสงค์การใช้งาน : ใช่

In the terms of mechanism : Not at all!

ในแง่ของหลักการทำงาน : ไม่มีทางแน่นอน!

## Overloading function

```
int multiply(int a,int b){
    int c = a * b;
    std::cout << "multiply(int,int) : ";
    return c;
}

float multiply(int a,float b){
    float c = a * b;
    std::cout << "multiply(int,float) : ";
    return c;
}

float multiply(float a,int b){
    float c = a * b;
    std::cout << "multiply(float,int) : ";
    return c;
}
```

- same name function but difference parameter can be defined as difference function

- ฟังก์ชันที่มีชื่อหมือนกันแต่รับ parameter ไม่เหมือนกันสามารถมองเป็นคนละฟังก์ชันได้

## Selection function by parameter

```
int multiply(int a,int b){
    int c = a * b;
    std::cout << "multiply(int,int) : ";
    return c;
}

float multiply(int a,float b){
    float c = a * b;
    std::cout << "multiply(int,float) : ";
    return c;
}

float multiply(float a,int b){
    float c = a * b;
    std::cout << "multiply(float,int) : ";
    return c;
}
```

```
printf("%d\n", multiply_int_int(2,1) );
printf("%f\n", multiply_int_float(2,2.0) );
printf("%f\n", multiply_float_int(2.0,3.0) );
std::cout << multiply(2,4) << std::endl;
std::cout << multiply(2,5.0F) << std::endl;
std::cout << multiply(2.0F,6) << std::endl;
```

Result :

```
2
4.000000
6.000000
multiply(int,int) : 8
multiply(int,float) : 10
multiply(float,int) : 12
```

## No except for method in class

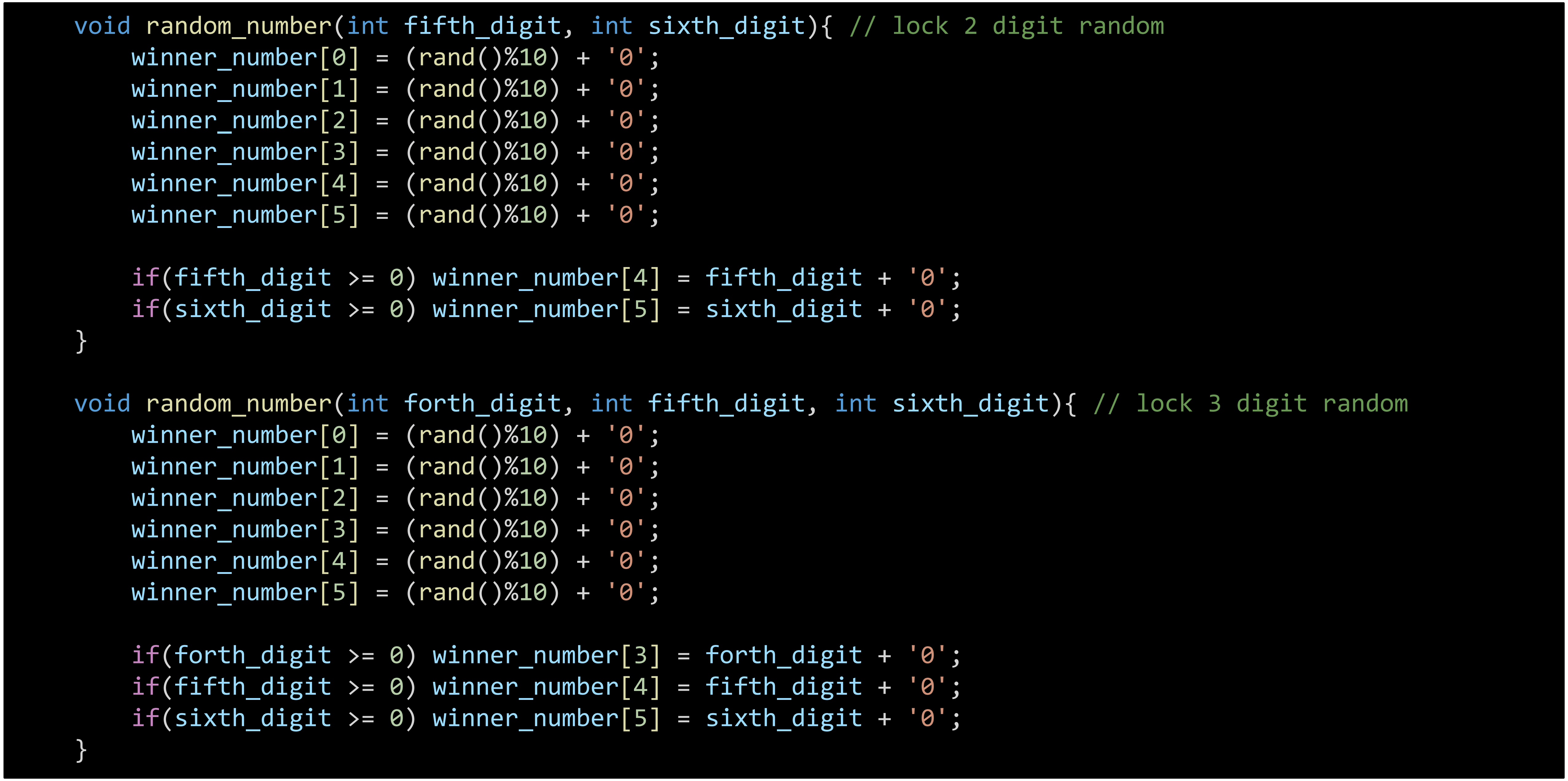
```
class lotto{
    string winner_number;

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";
    }
};
```

Code :

```
lotto a;
a.random_number();
a.print();
```

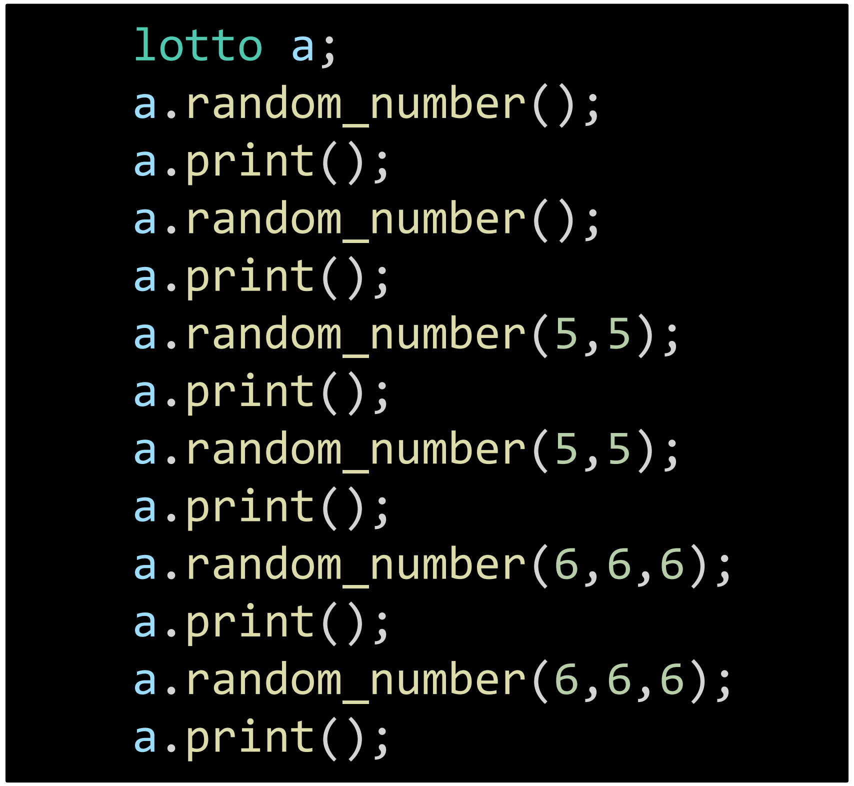
Result :

winner price is 882455



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

        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, int fifth_digit, int sixth_digit){ // lock 3 digit 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';

        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 method



```
    lotto a;
    a.random_number();
    a.print();
    a.random_number();
    a.print();
    a.random_number(5,5);
    a.print();
    a.random_number(5,5);
    a.print();
    a.random_number(6,6,6);
    a.print();
    a.random_number(6,6,6);
    a.print();
```

Result :

```
winner price is 174094

winner price is 882455

winner price is 171155

winner price is 761455

winner price is 221666

winner price is 761666
```

## Overload can call another overload inside

```
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, int fifth_digit, int 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';
}
```

```
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, int fifth_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';
}
```

```
void lotto::random_number(int, int)' cannot be overloaded with 'void
lotto::random_number(int, int)'
```

Cannot overload same type parameter method

ไม่สามารถ overload ฟังก์ชันที่มี parameter เหมือนกันได้

## Default parameter & Default initialization

```
class calculator{
    float result;

public :

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

};
```

- How to initial value of result
- เราสามารถกำหนดค่าเริ่มต้นของ result ได้ด้วยวิธีได้บ้าง

## Default initialization

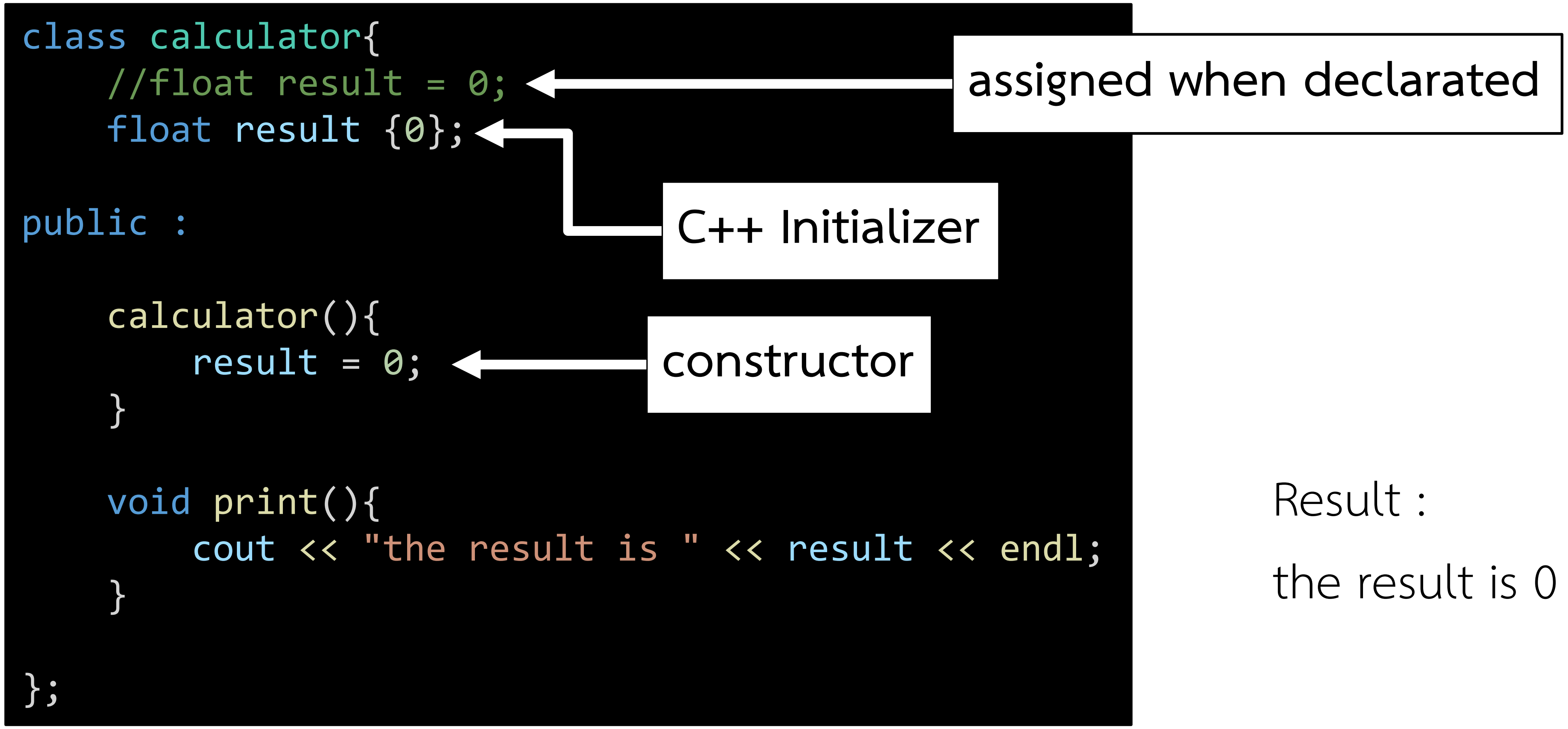
- constructor

- assigned when declarated

- C++ Initializer (advance)

Same result / ผลลัพธ์เหมือนกัน

## Default initialization



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

public :

    calculator(){
        result = 0;
    }

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

};
```

Result :

the result is 0

## Default parameter

```
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;
}
```

Consider these method / พิจารณา 3 method นี้

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

Exact same result or not?

ให้ผลลัพธ์เหมือนกันหรือไม่?

Yes Result :

```
the result is 36
the result is 36
```

## Default parameter

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

2 way to call this method

มี 2 วิธีที่จะเรียกใช้ method นี้

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

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

Same as original

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

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

Assign d to default (zero)

```
    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
```

## Default parameter

- Generate method or function for all possible default parameter

- สร้าง function หรือ method ตาความเป็นไปได้ทั้งหมดของ default parameter

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

Generate these method :

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

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

```
void sum(float,float,float,float);
void sum(float,float,float,0);
void sum(float,float,0,0);
void sum(float,1,0,0);
```

## Warning redundant with origin method

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

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

Error :

```
call of overloaded 'sum(int, int, int)' is ambiguous
```

*ข้อควรระวัง สามารถซ้ำกับ method เดิมได้

## Another example

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

    void max(float a,float b,float c = 0,float d = 0){
        if(a > b && a > c && a > d) result = a;
        else if(b > a && b > c && b > d) result = b;
        else if(c > a && c > b && c > d) result = c;
        else result = d;
    }
```

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

Result :

```
the result is 23
the result is 36
the result is 50
the result is 14
the result is 13
the result is 12
```

## Quiz

## Abstract class (Abstraction)

- a class that designed to inherit and point to another object only, cannot create any object

- create by create at less one pure virtual function in class

- เป็น class ที่ออกแบบมาเพื่อการสืบทอด (inherit) และชี้ไปยัง object อื่นเท่านั้นไม่สามารถสร้าง object จาก class นี้ได้

- สร้างได้โดยการสร้าง pure virtual function อย่างน้อยหนึ่งฟังก์ชันใน class

## Pure virtual function

- a function that have only declaration and no need to write definition

- ฟังก์ชันที่มีเพียงแค่ declaration ไม่ต้องสร้างในส่วนของ definition

## Pure virtual function syntax

- assign a declaration of function to zero

- 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;
};
```

- Creature is now abstract class
- ตอนนี้ class creature กลายเป็น abstract แล้ว

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

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

## Why we want abstract class

- we want to provide common class to be a guidelines creating child class
- provide a hierarchy paradigm or rule to develop same project
- provide common class to assigned to be any object in the project
- etc.
- เพื่อเป็นแนวทางเดียวกันในการพัฒนา class ลูกที่อยู่ต่ำกว่า
- เพื่อกำหนดแนวทางหรือกฎของการพัฒนาเป็นลำดับชั้น
- เพื่อสร้าง class กลางที่สามารถกำหนดให้เป็น object ได้ก็ได้ในระบบ
- อื่นๆ

## Another example

```
class vehicle{
protected :
    int speed;
    int wheel_count;

    vehicle(int wheel){
        wheel_count = wheel;
    }

public :
    void print_wheel(){
        cout << "this vehicle has [" <<  wheel_count << "] Wheel(s)" << endl;
    }

    void virtual print() = 0;
};
```

## Another example

- all class that inherited from vehicle class must implement print() function to be usable class

- ทุก class ที่สืบทอดไปจาก class vehicle ต้อง implement ฟังก์ชัน print() เพื่อให้กลายเป็น class ที่สามารถสร้าง object ได้

```
    motorbike vaspa_sprint;
    airplane a380(22);
    airplane a777(14);
    submarine S26T;

    vehicle* my_vehicles[4];
    my_vehicles[0] = &vaspa_sprint;
    my_vehicles[1] = &a380;
    my_vehicles[2] = &a777;
    my_vehicles[3] = &S26T;

    for(int i=0;i<4;i++){
        my_vehicles[i]->print();
    }
```

Result :

```
motobike is using speed [98]Kph
airplane is at [0] ft above sea level using speed [0]Kph
airplane is at [0] ft above sea level using speed [0]Kph
submarine is at [0] depth level using speed [0]Kph
```

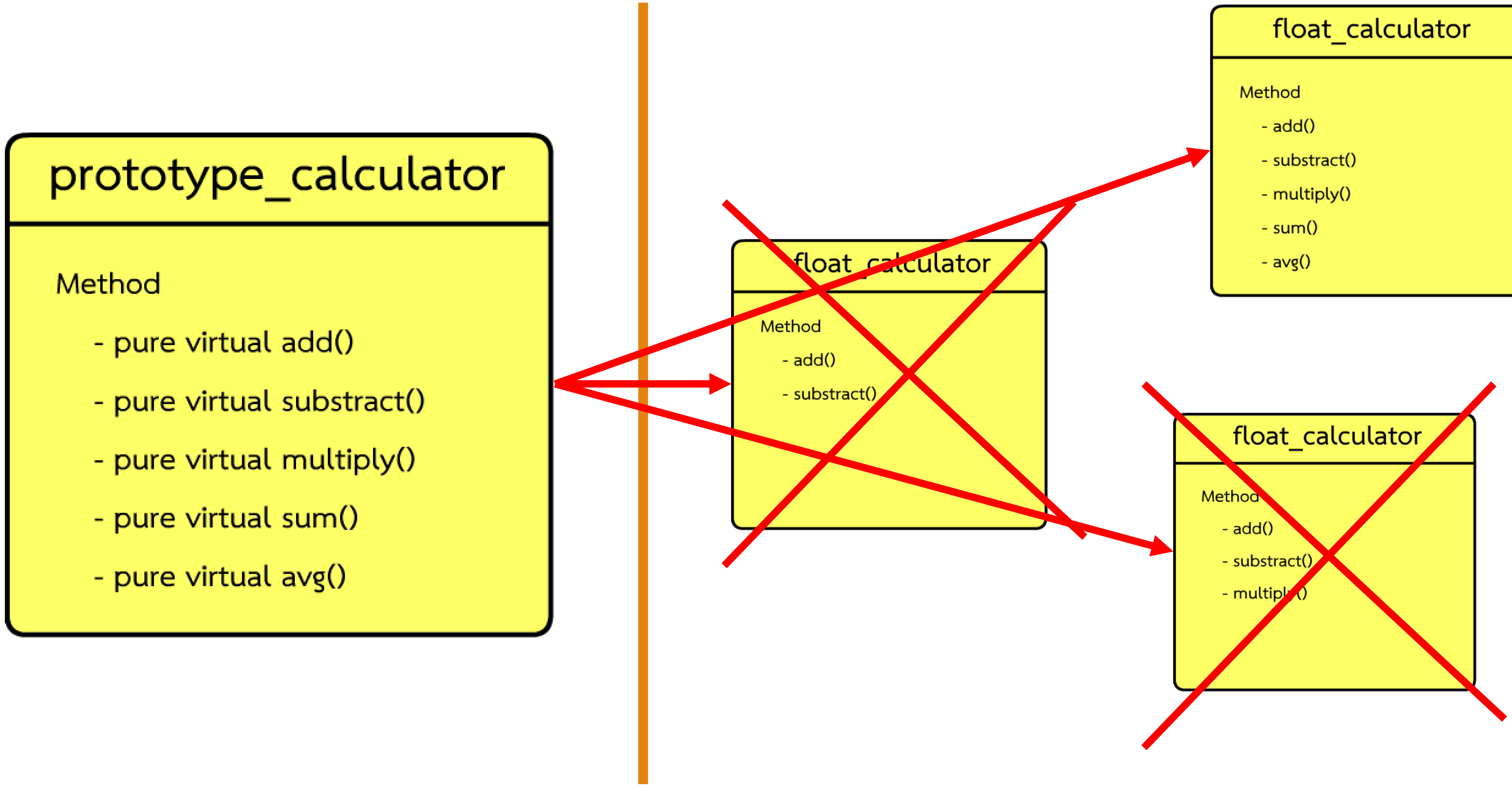
## conclude

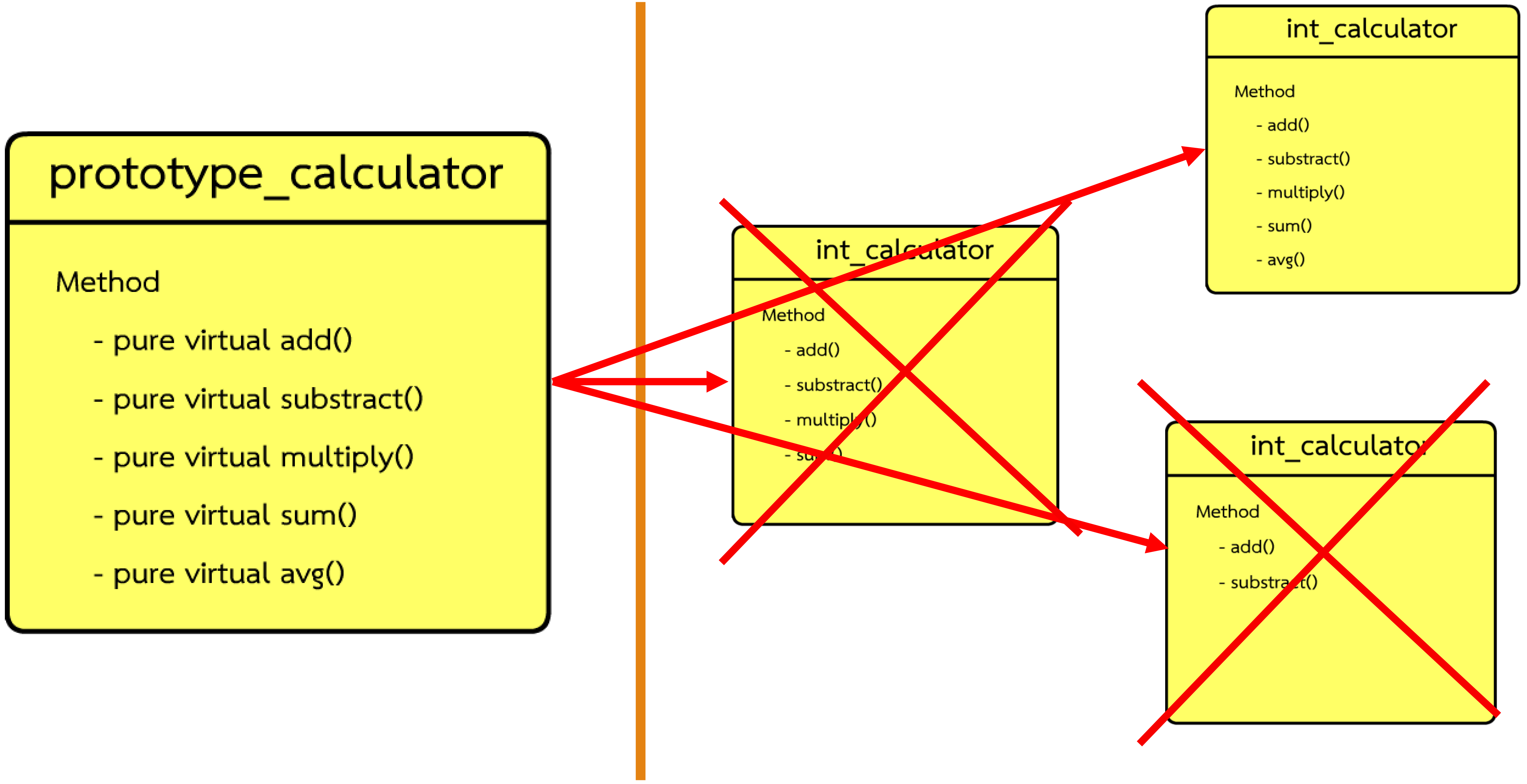
- overloading constructor & function

- default parameter & default initialization

- pure virtual function & abstract class

## LAB


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


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