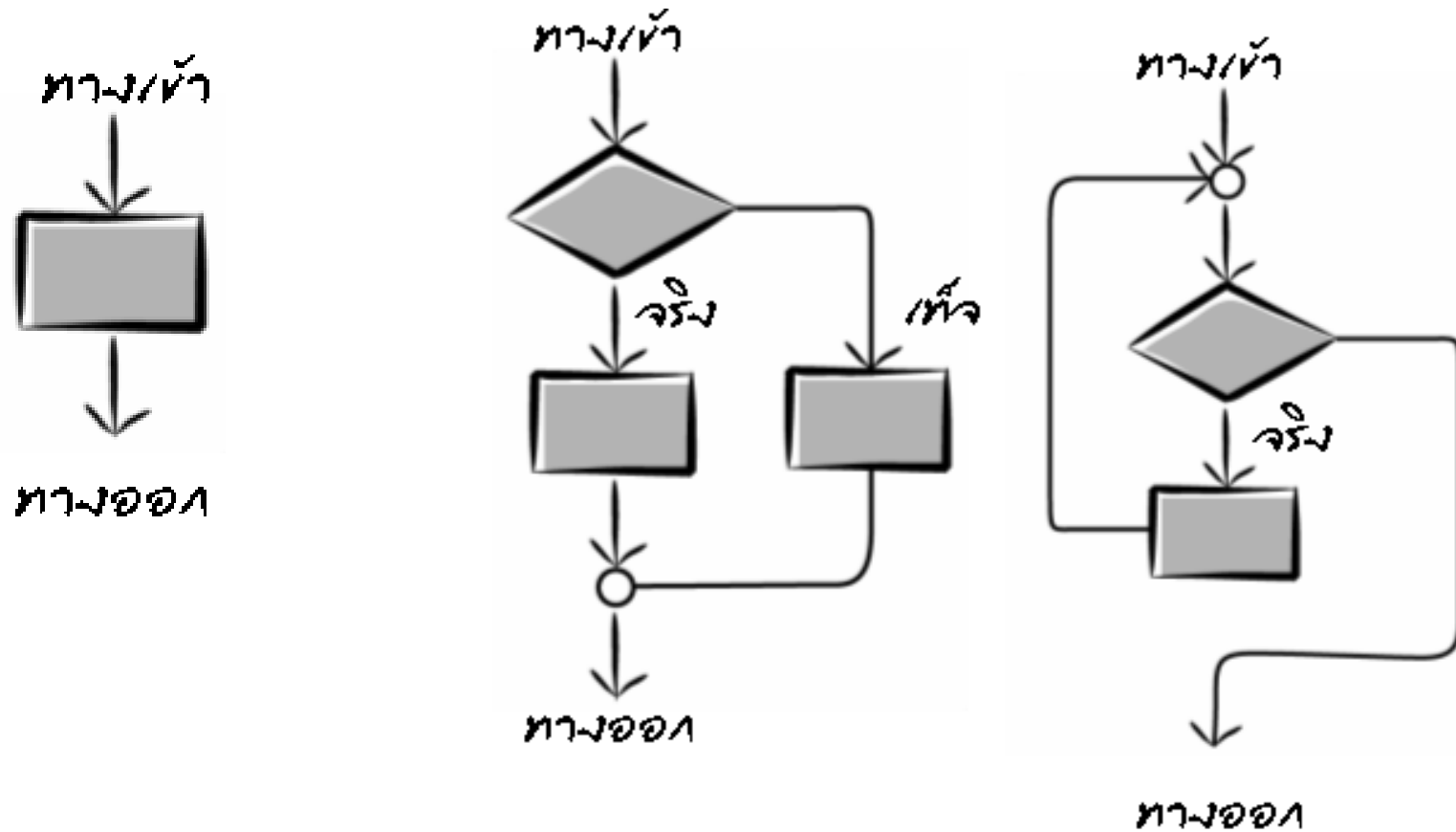


แนะนำการ โปรแกรมเชิงวัตถุด้วย ภาษา C++

การโปรแกรมเชิงโครงสร้าง

- โครงสร้างการทำงานของโปรแกรมควรมีอย่างจำกัด
- แต่ในขณะเดียวกันก็ต้องเพียงพอที่จะใช้แก้ปัญหาต่าง ๆ ได้
- โครงสร้างที่จำเป็นต่อการเขียนโปรแกรมมีอยู่ 3 แบบ
 - การทำงานเรียงลำดับ
 - การทำงานตามเงื่อนไข
 - การทำซ้ำ

โครงสร้างที่จำเป็นต่อการเขียนโปรแกรม



ชั้นรูทีนและฟังก์ชัน

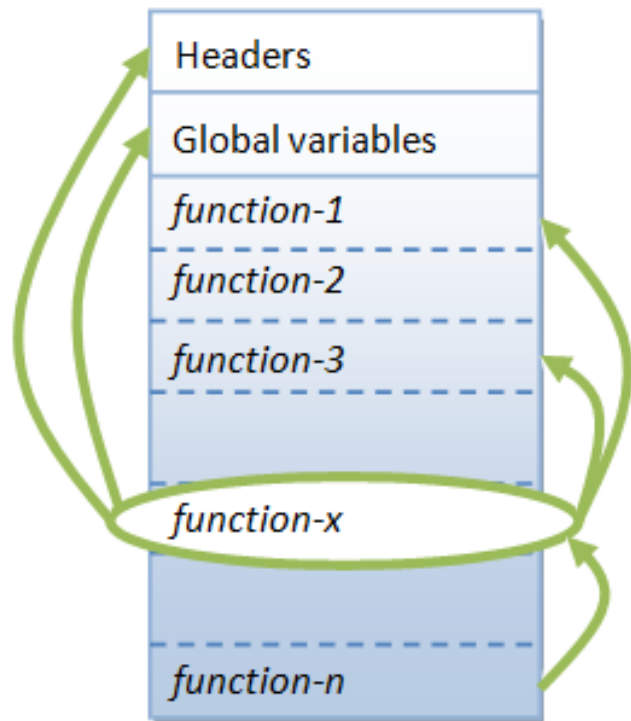
- จัดระเบียบคำสั่งในโปรแกรม
- แบ่งโปรแกรมออกเป็นส่วนย่อย ๆ
- ทำความเข้าใจได้ง่าย



ความแตกต่างของการ โปรแกรมเชิง โครงสร้างและ
การ โปรแกรมเชิงวัตถุ

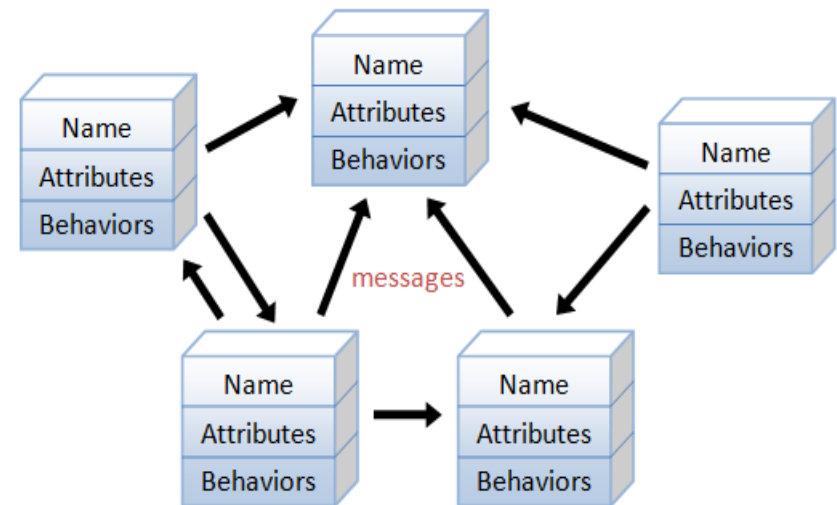
การเปรียบเทียบการโปรแกรมเชิงโครงสร้างและ การโปรแกรมเชิงวัตถุ

Traditional Procedural- Oriented languages



A function (in C) is not well-encapsulated

Object-Oriented Programming Languages



An object-oriented program consists of many well-encapsulated objects and interacting with each other by sending messages

https://www3.ntu.edu.sg/home/ehchua/programming/cpp/cp3_OOP.html

การโปรแกรมเชิงโครงสร้างและการโปรแกรมเชิงวัตถุ

- การโปรแกรมเชิงโครงสร้าง

- โปรแกรมจะแบ่งออกเป็นส่วนย่อยๆที่เรียกว่าโมดูล (module)
- แต่ละโมดูลจะต้องเป็นอิสระต่อกัน
- การออกแบบให้แต่ละโมดูลมีความเป็นอิสระต่อกันนั้นทำได้ยาก
- ต้นทุนในการพัฒนาโปรแกรมสูง

- การโปรแกรมเชิงวัตถุ

- การพัฒนาโปรแกรมเป็นการเลียนแบบการทำงานเชิงออบเจ็ค
- สามารถนำโปรแกรมกลับมาใช้ใหม่ (reuse) ได้ดีกว่าภาษาเชิงโครงสร้าง



จุดอ่อนของภาษาโปรแกรมเชิงโครงสร้าง

- โปรแกรมประกอบด้วยคำสั่งและข้อมูล การจัดระเบียบคำสั่งเพียงอย่างเดียวนั้นจึงไม่สมบูรณ์
- แยกส่วนที่เป็นข้อมูลออกจากคำสั่ง ด้วยเหตุนี้ฟังก์ชันที่จัดการกับข้อมูลจึงอาจจะถูกวางไว้อย่างกระจัดกระจายสร้างความลำบากในการติดตามสืบหาว่าข้อมูลนี้ถูกเปลี่ยนแปลงโดยฟังก์ชันใด



การโปรแกรมเชิงวัตถุ

การโปรแกรมเชิงวัตถุ

- วิเคราะห์ปัญหาโดยมองปัญหาว่าประกอบไปด้วยออบเจ็กต์ต่าง ๆ
- จำลองคุณลักษณะและพฤติกรรมของออบเจ็กต์
- ออบเจ็กต์จะส่งข้อมูลกันโดยผ่านข่าวสาร (Message)
- แตกต่างจากการโปรแกรมเชิงโครงสร้างที่วิเคราะห์ปัญหาโดยพิจารณาจากลำดับการทำงานและแบ่งการทำงานของโปรแกรมตามฟังก์ชันต่าง ๆ

ระบบทะเบียนนิสิต

- วิธีการโปรแกรมเชิงโครงสร้าง
 - ลงทะเบียนรายวิชา
 - ชำระเงิน
 - เพิ่มวิชา
- วิธีการโปรแกรมเชิงวัตถุ
 - นิสิต
 - ใบลงทะเบียน
 - รายชื่อรายวิชา

- วัตถุ (Object) ชนิด นิสิต
 - คุณลักษณะ
 - พฤติกรรม

ข้อดีของการพัฒนาการโปรแกรมเชิงวัตถุ

- แนวคิดการวิเคราะห์ปัญหาใกล้เคียงกับธรรมชาติของมนุษย์
- ระบบจริง (real life) แบ่งตามวัตถุ (Object) ไม่ได้ขึ้นอยู่กับฟังก์ชันการทำงาน
- ขบวนการพัฒนาโปรแกรมทำได้รวดเร็วขึ้น
- ง่ายต่อการพัฒนาและแก้ไข
- นำโปรแกรมกลับมาใช้ใหม่ได้ง่าย



คุณลักษณะเด่นของโปรแกรมเชิงออบเจ็ค

- การห่อหุ้ม (Encapsulation)
 - รวมข้อมูลกับคำสั่งที่เกี่ยวข้องกับข้อมูลนั้นเข้าด้วยกัน
- ช้อนวิธีการทำงาน (Inheritance)
 - ช้อนวิธีการทำงานของวัตถุ
 - ทำให้เกิดความยืดหยุ่น ผู้สร้างวัตถุสามารถเปลี่ยนแปลงวิธีการทำงานของวัตถุได้โดยไม่กระทบต่อผู้ใช้วัตถุ
- การมองที่เป็นธรรมชาติ (Polymorphism)
 - เห็นภาพแล้วแยกเป็นวัตถุโดยอัตโนมัติ
 - ในขณะเดียวกันก็นึกถึงประโยชน์ของวัตถุได้ทันที

ความรู้เรื่อง คลาส วัตถุ แอททริบิวต์
และเมธอด

หัวข้อ

- หลักการเชิงวัตถุ
- คลาส (Class)
- วัตถุ (Object)
- แอททริบิวต์ (Attribute)
- เมธอด (Method)
- ข้อความ (Message)

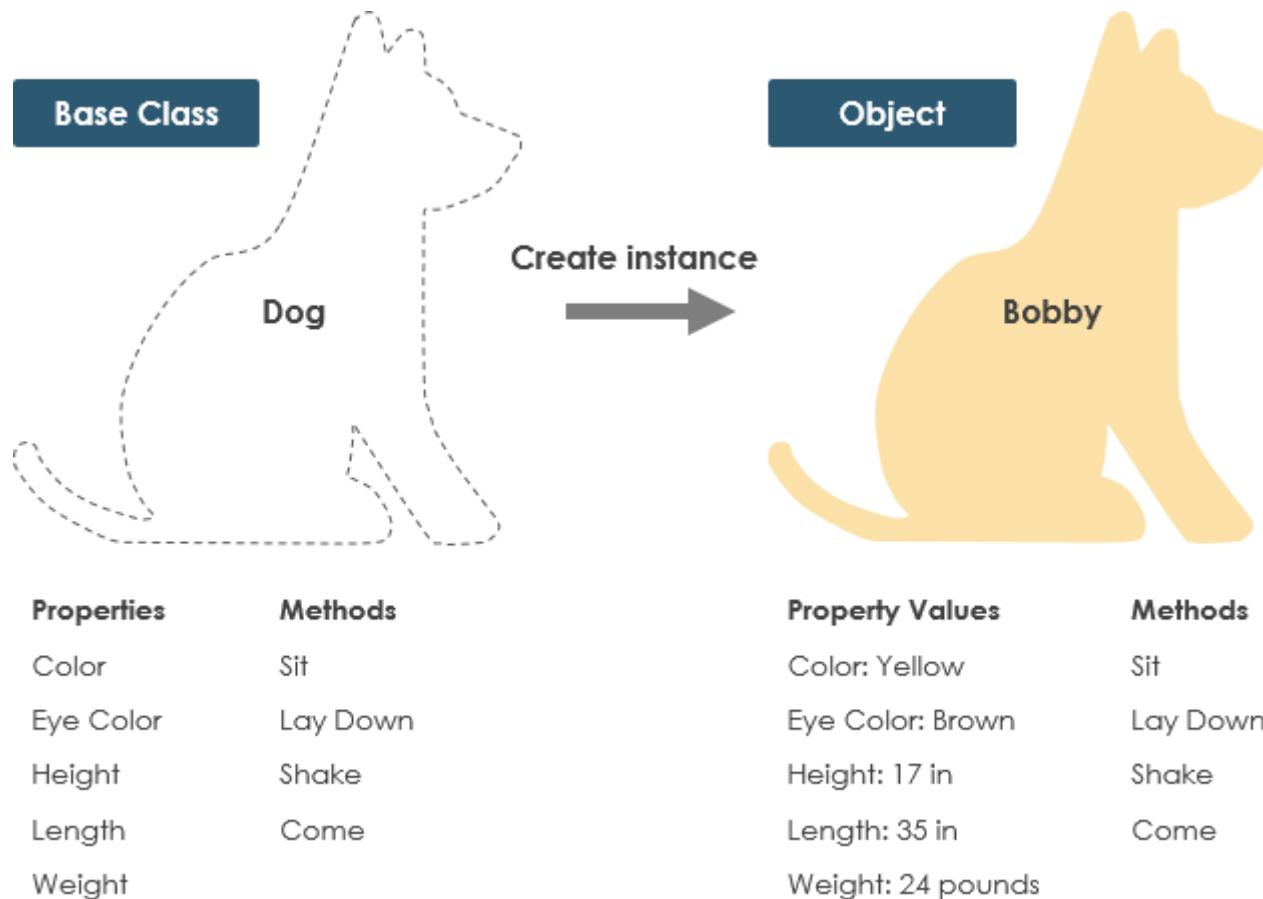
หลักการเชิงวัตถุ

คลาส (Class)

คลาส (Class)

- สิ่งที่ใช้อธิบายลักษณะและความสามารถของวัตถุ
- คลาสอาจจะเปรียบได้กับพิมพ์เขียวหรือแบบแปลนของวัตถุ
- วัตถุ (object) คือ สิ่งที่สร้างขึ้นมาจากแบบแปลนนั้น
- คลาสหนึ่งคลาสสามารถสร้างวัตถุได้หลายวัตถุ อาทิ
 - คลาสชื่อ bird อาจสร้างวัตถุชื่อ นกเพนกวิน, นกฟิราบ หรือ นกกระเจือกเทศ ซึ่งเป็นวัตถุชนิด bird
 - คลาสชื่อ Student อาจสร้างวัตถุชื่อ Stu1, Stu2 หรือ Stu3 ซึ่งเป็นวัตถุชนิด Student

ตัวอย่างคลาสและวัตถุ



<https://www.visual-paradigm.com/guide/uml-unified-modeling-language/uml-class-diagram-tutorial/>

Classname
(Identifier)

Data Member
(Static attributes)

Member Functions
(Dynamic Operations)

Student	Circle
name grade	radius color
getName() printGrade()	getRadius() getArea()

SoccerPlayer	Car
name number xLocation yLocation	plateNumber xLocation yLocation speed
run() jump() kickBall()	move() park() accelerate()

Examples of classes

Classname	<u>paul:Student</u>	<u>peter:Student</u>
Data Members	name="Paul Lee" grade=3.5	name="Peter Tan" grade=3.9
Member Functions	getName() printGrade()	getName() printGrade()

Two instances of the **Student** class

https://www3.ntu.edu.sg/home/e_hchua/programming/cpp/cp3_OP.html

การสร้างวัตถุจากคลาส Student

class Student

Studennt
แอททริบิวต์
เมธอด

สร้าง 3 objects มาจาก class Student

Stu1:Studennt
แอททริบิวต์
เมธอด

Stu2:Studennt
แอททริบิวต์
เมธอด

Stu3:Studennt
แอททริบิวต์
เมธอด

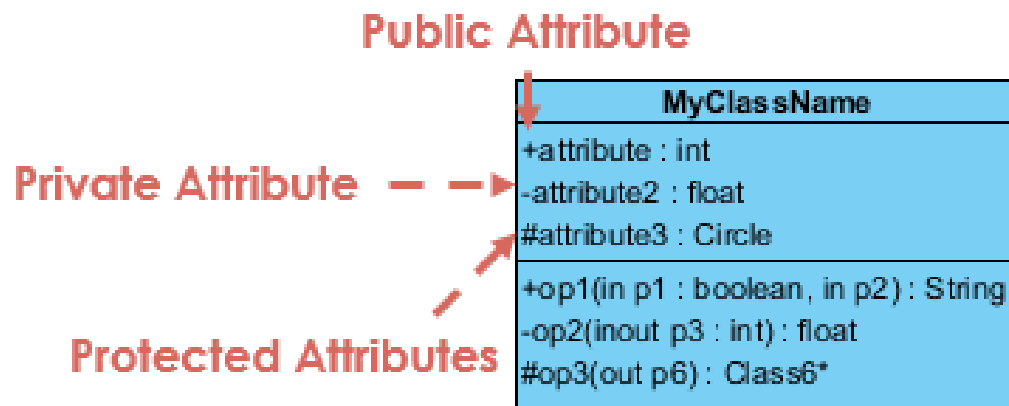
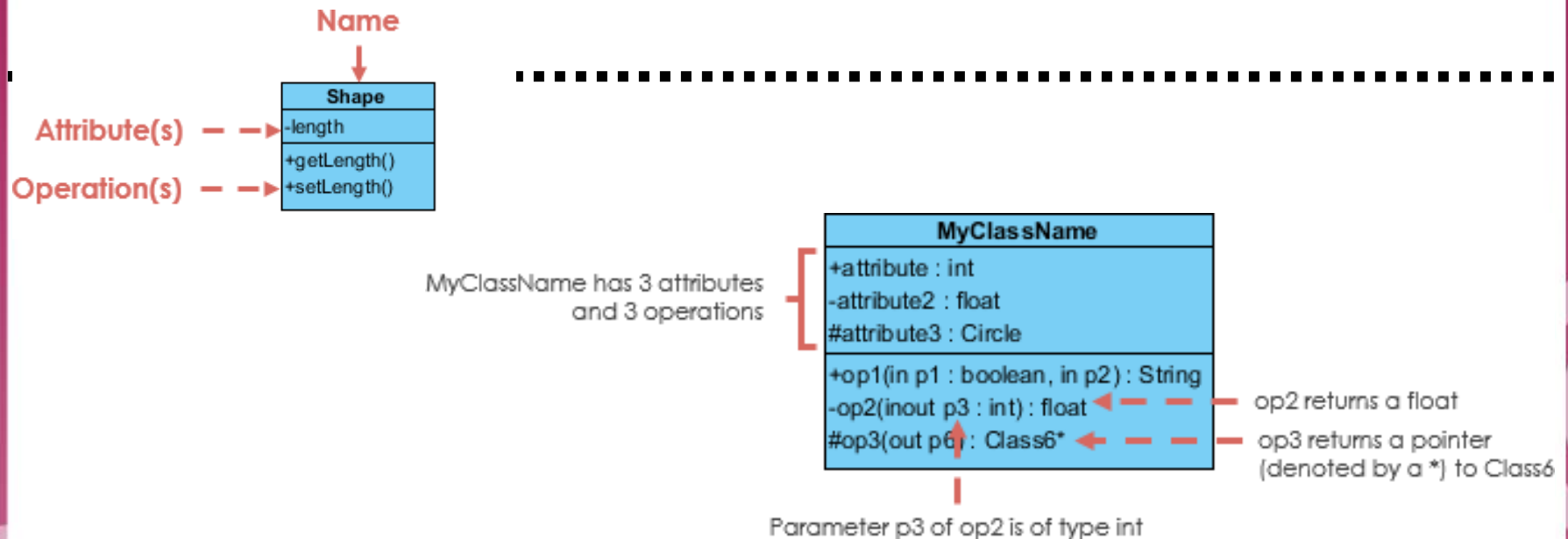
<https://www.uml-diagrams.org/class-reference.html>

โครงสร้างของคลาสประกอบด้วย

1. ชื่อคลาส (Class name)
2. คุณลักษณะ (Attributes) หรือ ข้อมูลสมาชิก (Data member) เป็นสิ่งที่ใช้อธิบายคุณลักษณะของคลาสหรือเรียกอย่างหนึ่งว่าตัวแปร
 - เช่น student, gpa, major
3. เมธอด (Methods) หรือ พฤติกรรม (Behaviors) เป็นสิ่งที่แสดงถึงฟังก์ชันการทำงานของคลาสหรือเรียกอย่างหนึ่งว่าฟังก์ชัน
 - เช่น calculateGpa, setRadius, addValue



แผนภาพ Unified Modeling Language (UML)



<https://www.visual-paradigm.com/guide/uml-unified-modeling-language/uml-class-diagram-tutorial/>

การกำหนดสาระสำคัญ (abstraction)

- เลือกจำลองลักษณะที่สำคัญ
 - จำลองเฉพาะความสามารถของวัตถุที่สนใจหรือเกี่ยวข้องกับปัญหาที่ต้องการจะแก้ไข
 - เช่น ถ้าต้องการคำนวณหาน้ำหนักเฉลี่ยของแมว ก็อาจจะให้แมวจำลองมี "น้ำหนัก" เป็นลักษณะที่สำคัญ
- ไม่จำลองลักษณะทั้งหมดของวัตถุ
 - การจำลองแมวจะไม่ต้องจำลองหมวดทุกเส้นของแมว
 - การจำลองรถยนต์จะไม่ต้องจำลองว่ารถยนต์ สีดำ ปี 2018

การกำหนดสาระสำคัญ (abstraction)

Student
Student_ID
Student_Name
Student_Gpa
calculateGpa()

Circle
Color
Radius
getRadius()

FootballPlayer
Name
Number
Position
turnLeft()
turnRight()
run()

Rectangle
Color
Width
Height
getArea()

การกำหนดสาระสำคัญ (abstraction)

- คำถาม

1. สร้างโปรแกรมต้องการคำนวณหาพื้นที่ของสามเหลี่ยม
— ต้องกำหนดสาระสำคัญ (abstraction) คือ ?
2. สร้างโปรแกรมคำนวณเกรดของวิชา 01418113
— ต้องกำหนดสาระสำคัญ (abstraction) คือ ?

การกำหนดสาระสำคัญ (abstraction)

- โปรแกรมระบบจัดการบัญชีเงินฝากของธนาคาร
- ตัวอย่างของวัตถุ
 - ลูกค้า, เครื่อง ATM
- ลูกค้า
 - มีแอตทริบิวต์ (attribute)
 - เลขที่บัญชี ชื่อเจ้าของบัญชี วันที่เปิดบัญชี และยอดเงินคงเหลือ
 - มีเมธอด (method)
 - ฝาก ถอน และโอนเงิน
- เครื่อง ATM
 - มีแอตทริบิวต์ (attribute) ?
 - มีเมธอด (method) ?

แอททริบิวต์ (attribute)

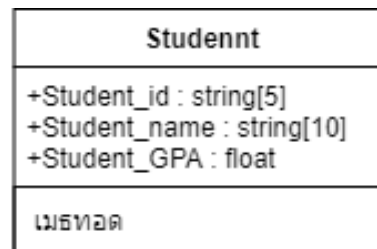
แอททริบิวต์ของวัตถุ

- ข้อมูลที่เก็บอยู่ในวัตถุ
- แบ่งเป็นตัวแปร (variable) และค่าคงที่ (constant)
 - **ตัวแปร** คือ คุณลักษณะที่สามารถเปลี่ยนค่าได้
 - **ค่าคงที่** คือ คุณลักษณะที่ไม่สามารถเปลี่ยนค่าได้

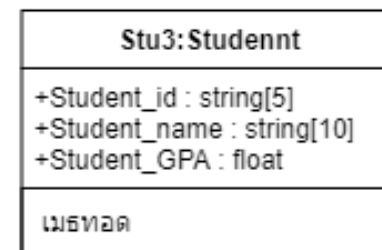
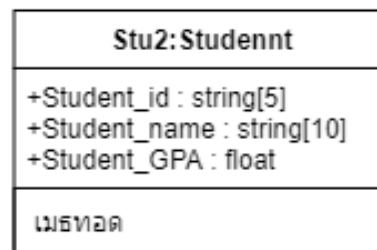
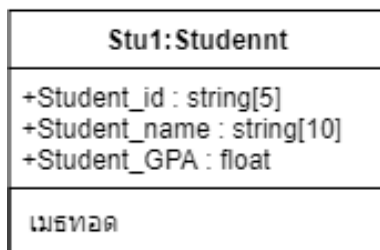
แอททริบิวต์ (attribute)

- แอททริบิวต์ หมายถึง ข้อมูลที่เก็บอยู่ในวัตถุ แบ่งเป็น
 - ตัวแปร (variable) และค่าคงที่ (constant)

class Student



สร้าง 3 objects มาจาก class Student



<https://www.uml-diagrams.org/class-reference.html>

แอททริบิวต์ (attribute)

ตัวแปร (variable)

Studennt
+Student_id +Student_name +Student_GPA
เมธอด

Stu1: Studennt
+Student_id = "1111" +Student_name = "Jirawan" +Student_GPA = 3.00
เมธอด

Stu2: Studennt
+Student_id = "2222" +Student_name = "Steve" +Student_GPA = 3.99
เมธอด

Stu3: Studennt
+Student_id = "3333" +Student_name = "Bill" +Student_GPA = 3.98
เมธอด

<https://www.uml-diagrams.org/class-reference.html>

แอททริบิวต์ (attribute)

- เป็นแอททริบิวต์ที่ใช้ร่วมกันของทุกวัตถุ (ค่าคงที่)
- ทุกวัตถุจะใช้แอททริบิวต์ ร่วมกันทำให้ประหยัดพื้นที่ในหน่วยความจำ
 - เช่น แอททริบิวต์ที่กำหนดให้เป็นค่าคงที่ชื่อ MIN_GPA

แอททริบิวต์ของคลาส (ค่าคงที่)

ค่าคงที่ (constant)

แอททริบิวต์ที่ใช้ร่วมกัน
= ค่าคงที่ (**formal**)

Studentnt
+Student_id +Student_name +Student_GPA -Min_GPA = 2.00
เมธอด

Stu1:Studentnt
+Student_id = "1111" +Student_name = "Jirawan" +Student_GPA = 3.00 <u>-Min_GPA = 2.00</u>
เมธอด

Stu2:Studentnt
+Student_id = "2222" +Student_name = "Steve" +Student_GPA = 3.99 <u>-Min_GPA = 2.00</u>
เมธอด

Stu3:Studentnt
+Student_id = "3333" +Student_name = "Bill" +Student_GPA = 3.98 <u>-Min_GPA = 2.00</u>
เมธอด

ข้อความ (Message) และเมธอด (Method)

ข้อความและเมธอด

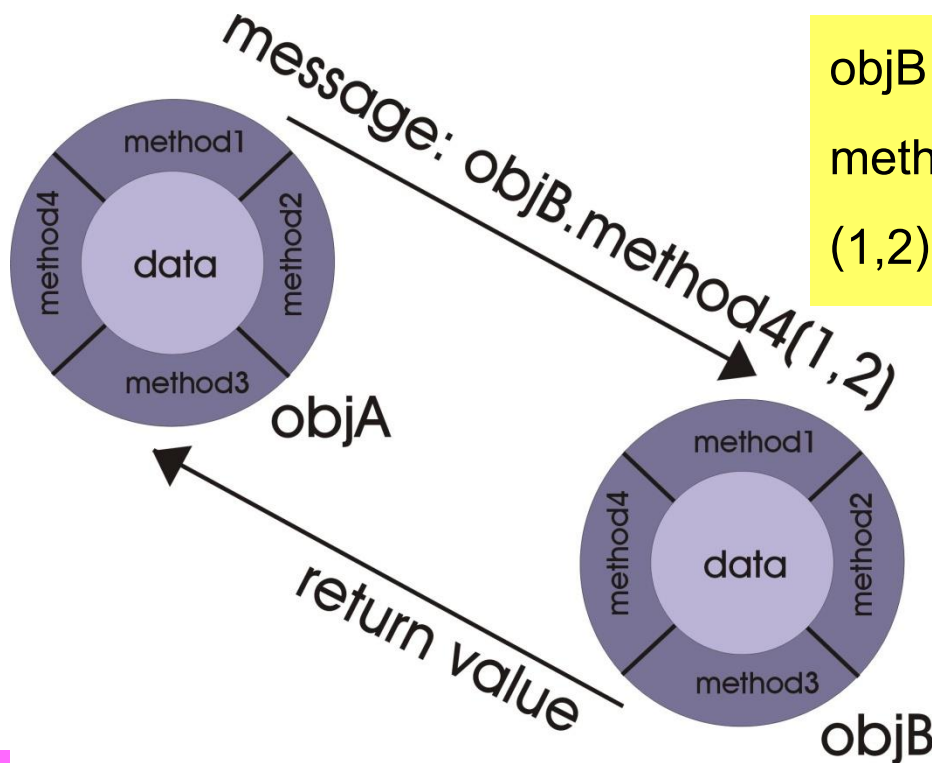
- สั่งให้วัตถุทำงานโดยการส่งข้อความ (message) หรือคำสั่งไปหาวัตถุ
- เมื่อวัตถุได้รับข้อความแล้ว วัตถุนั้นก็จะทำงานตามวิธีการหรือคำสั่งต่างๆ ที่เขียนไว้ในเมธอด (method)
- ชื่อของเมธอดควรเป็นคำกริยา

เมธอด

- วิธีการหรือการกระทำที่กำหนดอยู่ในคลาสหรือวัตถุเพื่อใช้จัดการกับแอตทริบิวต์ของวัตถุ
- เปรียบเทียบได้กับ function, procedure หรือ subroutine ของโปรแกรมโครงสร้าง
 - เมธอด `weight()` ใช้เป็นเมธอดหาน้ำหนักของนก
 - เมธอด `setColor()` ใช้เป็นเมธอดกำหนดค่าสีของนก
 - เมธอด `register()` ใช้เป็นเมธอดลงทะเบียนของนิสิต
 - เมธอด `deposit()` ใช้เป็นเมธอดฝากเงินของลูกค้าธนาคาร

การสื่อสารระหว่างวัตถุ

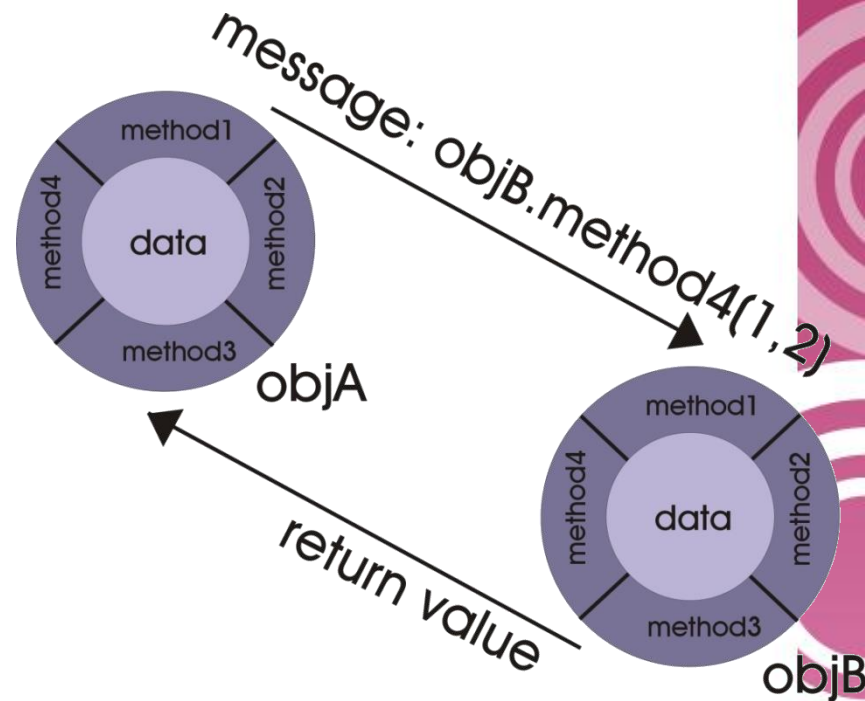
- การสื่อสารระหว่างกันของวัตถุทำได้โดยการผ่านข่าวสาร (message)



objB = ชื่อวัตถุ
method4 = ชื่อเมธอด
(1,2) = อาร์กิวเมนต์(argument)




การสื่อสารระหว่างวัตถุ

- ข่าวสารจะส่งผ่านจากวัตถุ objA ที่เป็นผู้ส่ง (sender) เพื่อเรียกการทำงานของเมธอดที่ชื่อ **method4** และ อาร์กิวเมนต์ (1,2) จากวัตถุ objB ที่เป็นผู้รับ (receiver)
- **objB** อาจส่งค่า (return value) บางค่ากลับมายัง **objA**





ตัวอย่างคลาส เมธอด และแอททริบิวต์

ตัวอย่างคลาสเมธอด

คลาส		เมธอด
นก		กิน(), บิน(), ร้องเพลง()
โทรทัศน์		เปิด(), ปิด(), เปลี่ยนช่อง() 
สี่เหลี่ยม		คำนวณพื้นที่() 

ตัวอย่างคลาสแอททริบิวต์

คลาส		แอททริบิวต์
นก		สี, เพศ, น้ำหนัก
โทรทัศน์		ยี่ห้อ, รุ่น, หมายเลข, ขนาดจอภาพ
สี่เหลี่ยม		ความกว้าง, ความสูง 



การแทนค่าใช้ UML

ชื่อคลาส

แอสโทริบิวท์

เมธอด

Rectangle

ความกว้าง

ความสูง

คำนวณพื้นที่()

นก

สี

เพศ

น้ำหนัก

กิน()

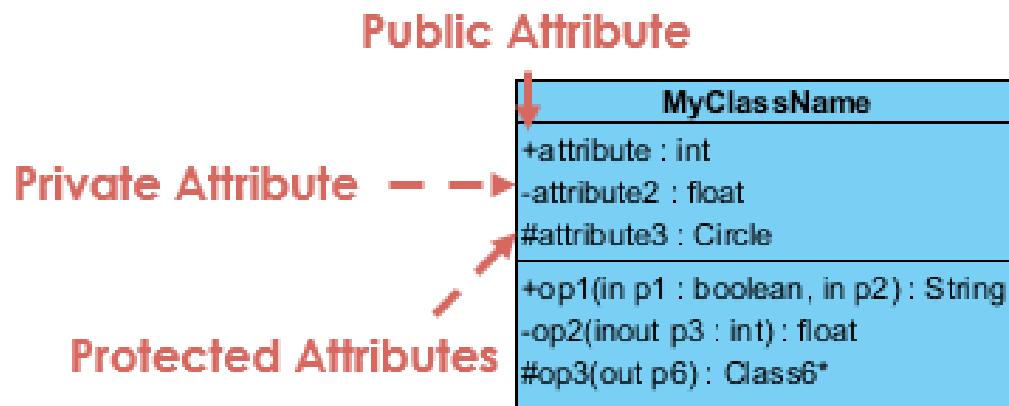
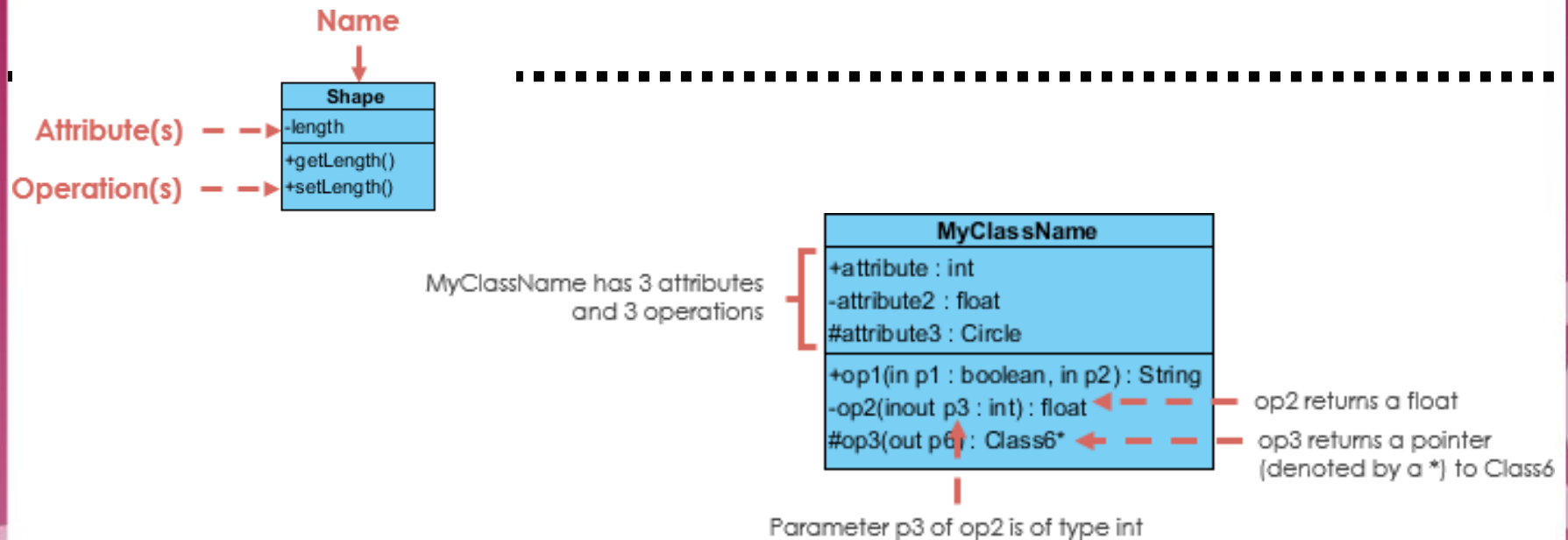
บิน()

ร้องเพลง()

แผนภาพ Unified Modeling Language (UML)

- การอธิบายโครงสร้างของคลาสและวัตถุ (Object)
- การอธิบายความสัมพันธ์คุณลักษณะ (Attributes) และเมธอด ของแต่ละวัตถุ
- สามารถใช้แผนภาพ Class diagram ของ Unified Modeling Language (UML) เพื่ออธิบายความสัมพันธ์ระหว่างโปรแกรมเชิงวัตถุดังกล่าวได้ดี Class diagram

แผนภาพ Unified Modeling Language (UML)



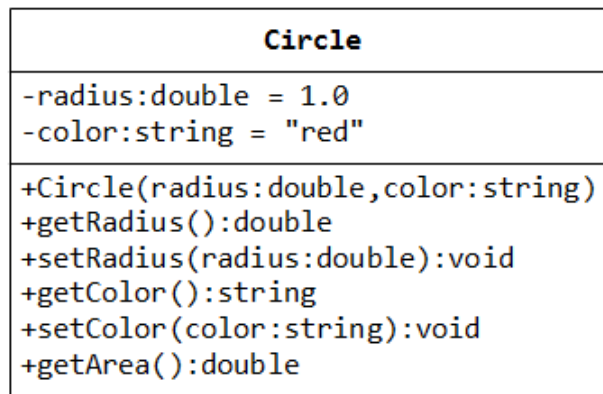
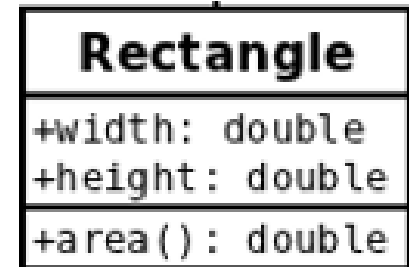
<https://www.visual-paradigm.com/guide/uml-unified-modeling-language/uml-class-diagram-tutorial/>

องค์ประกอบของ Class diagram

1. คลาสและวัตถุจะใช้สัญลักษณ์ที่แทนด้วยสี่เหลี่ยม แบ่งออกเป็น 3 ส่วน ซึ่งแต่ละส่วนนั้น (จากบนลงล่าง) ประกอบด้วย ชื่อคลาส, แอททริบิวต์ และ เมธอด

2. แอททริบิวต์ ต้องกำหนดสิทธิ์การเข้าถึง

— ได้แก่ เครื่องหมาย + public , เครื่องหมาย # protected และ เครื่องหมาย - private

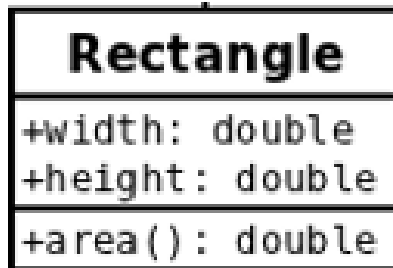


https://icarus.cs.weber.edu/~dab/cs1410/textbook/9.Classes_And_Objects/uml.html

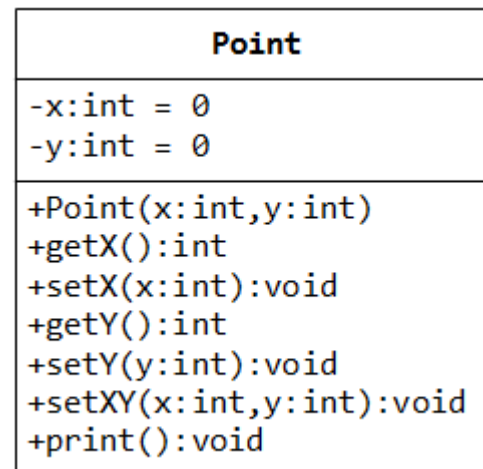
<http://cse230.artifice.cc/lecture/classes-and-object-orientation-2.html>

องค์ประกอบของ Class diagram

3. เมธอด ต้องมีการระบุพารามิเตอร์ (parameter) ที่จะใช้ส่งผ่านกันระหว่างเมธอดไว้หลังชื่อเมธอดเสมอ โดยชื่อพารามิเตอร์จะอยู่ภายในเครื่องหมาย ()
- ถ้าเมธอดไม่มีพารามิเตอร์ที่ก็ต้องใส่เครื่องหมาย () ที่เป็นค่าว่างไว้เสมอ



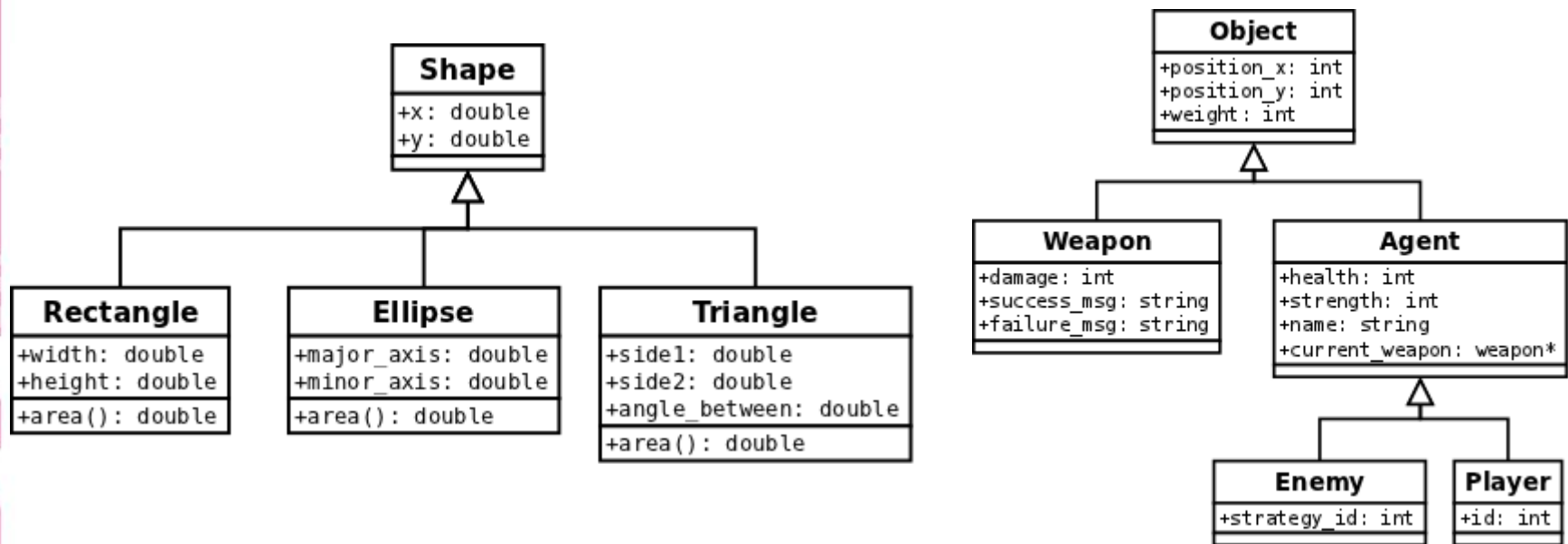
<http://cse230.artifice.cc/lecture/classes-and-object-orientation-2.html>



https://www3.ntu.edu.sg/home/ehchua/programming/cpp/cp6_Inheritance.html

องค์ประกอบของ Class diagram

4. เส้นแสดงความสัมพันธ์ แสดงให้เห็นว่าคลาสและวัตถุมีการใช้แอตทริบิวต์ และเมธอด เกี่ยวข้องกันและการเชื่อมโยงกัน ซึ่งส่วนใหญ่จะใช้ความสัมพันธ์แบบ Inheritance relationship เพื่อแสดงถึงการถ่ายทอดคุณสมบัติระหว่างคลาสดับวัตถุ หรือ วัตถุกับวัตถุด้วยกัน



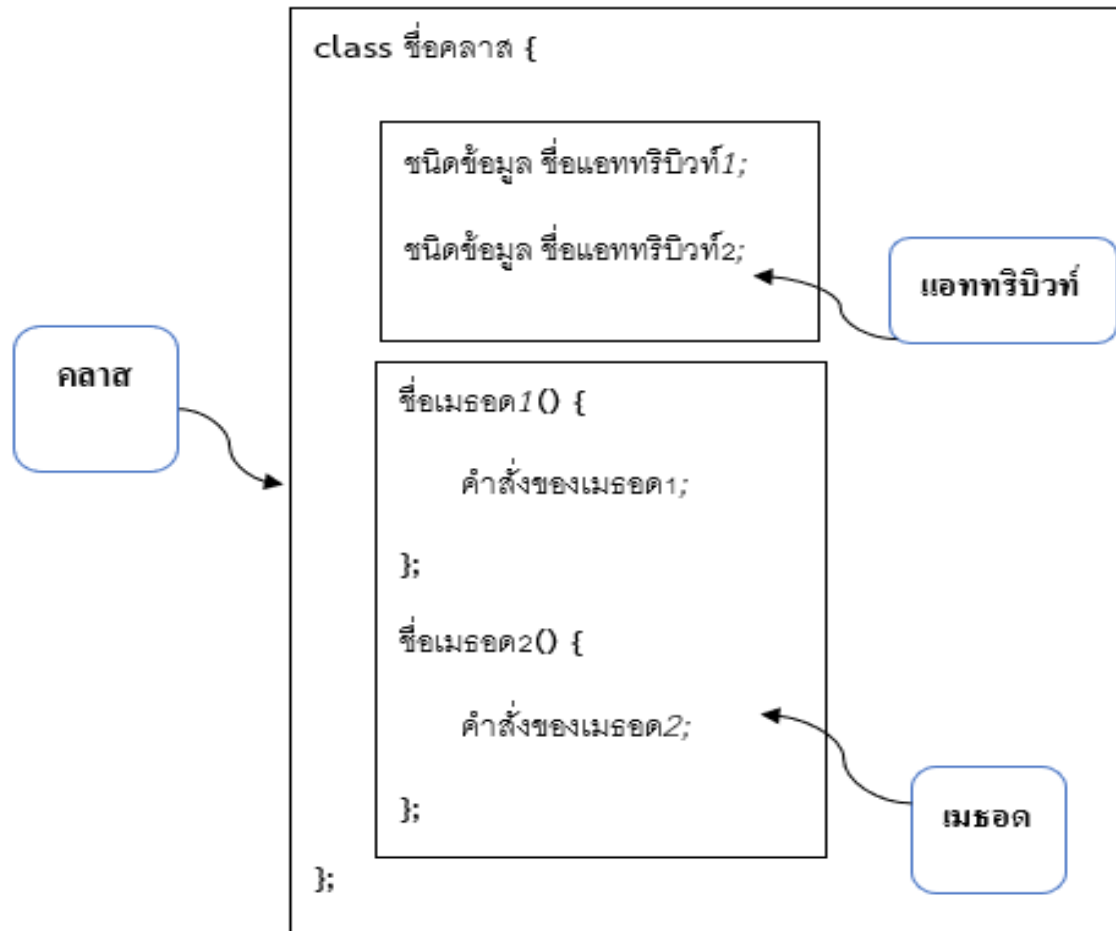
<http://cse230.artifice.cc/lecture/classes-and-object-orientation-2.html>

<https://bhattaca.github.io/cse212/lecture/class-inheritance.html>

การเขียนโปรแกรมเชิงวัตถุโดยใช้ภาษา C++

- การประกาศคลาส
- การกำหนดสิทธิ์ในการใช้งาน (Access Modifier)
- การประกาศแอททริบิวต์ (Attribute)
- การประกาศเมธอด (Method)
- การประกาศและสร้างวัตถุ
- การเรียกใช้สมาชิกของวัตถุ

โครงสร้างและส่วนประกอบของคลาส



การประกาศคลาส

- รูปแบบการประกาศคลาส โดยไม่มีแอมพลิฟายเออร์และเมธอด

```
class ชื่อคลาส{  
    // แอมพลิฟายเออร์  
    //เมธอด  
};
```

ตัวอย่าง

```
class Circle{  
  
};
```

การประกาศคลาส

- รูปแบบการประกาศคลาสพร้อม
กับแอตทริบิวต์และเมธอด

ตัวอย่าง

```
class ชื่อคลาส{  
    ชนิดข้อมูล ชื่อแอตทริบิวต์1;  
    ชื่อเมธอด1() {  
        คำสั่งของเมธอด1;  
    };  
};
```

```
class Circle{  
    private: float radius;  
    public:  
        void setRadius (int val) { radius = val;};  
        float area (void) { return (3.14* radius *  
            radius); };  
        float girth (void) { return (3.14*2* radius); };  
};
```

```
1. #include <iostream>
2. using namespace std;
```

```
3. class Circle {
```

```
4. private:
```

```
5.     int r;
```

```
6. public:
```

```
7.     void setRadius (int val) { r = val;};
```

```
8.     float area (void) { return 3.14*r*r; };
```

```
9.     float girth (void) { return 3.14*2*r; };
```

```
10.} cir1;
```

```
11.main ()
```

```
12.{
```

```
13.     int val;
```

```
14.     cout << "Enter circle radius: ";
```

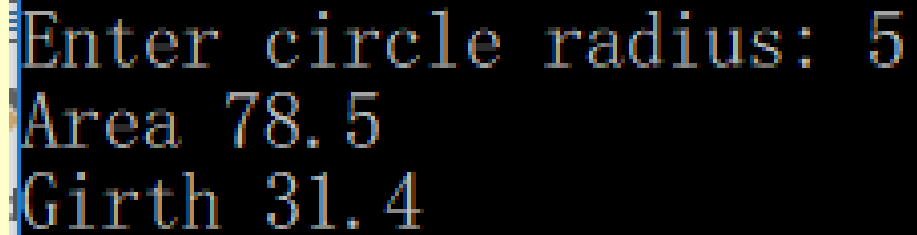
```
15.     cin >> val;
```

```
16.     cir1.setRadius(val);
```

```
17.     cout << "Area " << cir1.area() << endl;
```

```
18.     cout << "Girth " << cir1.girth() << endl;
```

```
19.}
```



```
Enter circle radius: 5
Area 78.5
Girth 31.4
```

<http://marcuscode.com/lang/cpp/classes1>

การแทนค่าคลาสโดยใช้ UML

```
class Circle {  
private:  
    float radius;  
public:  
    void setRadius (int val) { radius = val;};  
    float area (void) { return (3.14* radius * radius); };  
    float girth (void) { return (3.14*2* radius); };  
}
```

ชื่อคลาส

แอตทริบิวต์

เมธอด

Circle

Circle

- radius : float

+setRadius (int val) : void

+area (void) : float

+ girth (void): float

<http://marcuscode.com/lang/cpp/classes1>

การแทนค่าคลาสโดยใช้ UML

C++

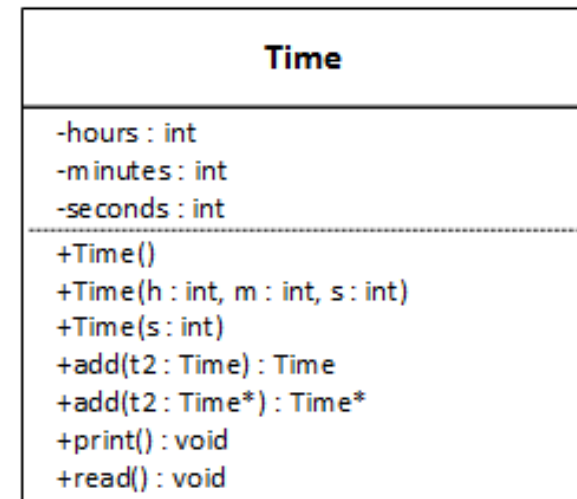
```
class Time
{
    private:
        int    hours;
        int    minutes;
        int    seconds;

    public:
        Time();
        Time(int h, int m, int s);
        Time(int s);

        Time    add(Time t2);
        Time*   add(Time* t2);

        void    print();
        void    read();
};
```

UML



https://icarus.cs.weber.edu/~dab/cs1410/textbook/9.Classes_And_Objects/uml.html

การแทนค่าคลาสโดยใช้ UML

Circle.h - Header

```
1  /* The Circle class Header (Circle.h) */
2  #include <string>    // using string
3  using namespace std;
4
5  // Circle class declaration
6  class Circle {
7  private:    // Accessible by members of this class only
8      // private data members (variables)
9      double radius;
10     string color;
11
12 public:     // Accessible by ALL
13     // Declare prototype of member functions
14     // Constructor with default values
15     Circle(double radius = 1.0, string color = "red");
16
17     // Public getters & setters for private data members
18     double getRadius() const;
19     void setRadius(double radius);
20     string getColor() const;
21     void setColor(string color);
22
23     // Public member Function
24     double getArea() const;
25 };
```

Circle
-radius:double = 1.0 -color:string = "red"
+Circle(radius:double,color:string) +getRadius():double +setRadius(radius:double):void +getColor():string +setColor(color:string):void +getArea():double

https://www3.ntu.edu.sg/home/ehchua/programming/cpp/cp3_OOP.html

ข้อเสนอแนะสำหรับการตั้งชื่อคลาส

- ชื่อคลาสควรเป็นคำนาม
- นิยมให้ตัวอักษรตัวแรกเป็นตัวใหญ่
 - ถ้าชื่อคลาสประกอบด้วยคำมากกว่า 1 คำ นิยมให้ตัวแรกของคำเป็นอักษรตัวใหญ่
 - Student, MobilePhone, CalulateArea

Access Modifier

Access Modifier ในภาษา C++

- Access Modifier มี 3 ประเภท คือ
 - ❖ `access modifier` ได้แก่ `private`, `public`, `protected` และ (ลำดับความเข้มงวดที่สุดไล่ลำดับถึงอิสระที่สุด)
- Access Modifier จะมีหรือไม่มีก็ได้ ถ้าไม่มีจะถือว่าเป็น `private`

Access Modifier ในภาษา C++

Modifier	คำอธิบาย
public (สาธารณะ)	คลาสอื่นๆ สามารถเข้าใช้งานแอตทริบิวต์ และเมธอด ที่ถูกกำหนด public ได้อย่างอิสระ
protected (ถูกปกป้อง)	ฟังก์ชันที่ประกาศภายใน Class และ Sub-class เท่านั้นที่สามารถอ้างถึงหรือเรียกใช้ข้อมูลหรือฟังก์ชันในส่วน protected ได้
private (ส่วนบุคคล)	ปิดกั้นไม่ให้คลาสอื่นๆ สามารถเข้าใช้งาน แอตทริบิวต์ และเมธอด ได้ยกเว้นคลาสของตัวเอง

สรุปการนำ Access Modifier แต่ละแบบไปใช้งาน

Modifier	สัญลักษณ์ UML	ใช้กับ คลาส	ใช้กับ แอตทริบิวต์	ใช้กับ เมธอด
public	+	✗	✓	✓
protected	#	✗	✓	✓
private	-	✗	✓	✓

ตัวอย่าง

<https://www.geeksforgeeks.org/access-modifiers-in-c/>

Access Modifier

- การกำหนดสิทธิ์จะต้องกำหนดสิทธิ์ที่ต้องการว่าเป็น private, public หรือ protected ตามด้วย เครื่องหมาย “:” แอททริบิวต์และเมธอดที่อยู่หลังเครื่องหมาย : จะถูกกำหนดสิทธิ์ทันที ยกเว้นมีการกำหนดสิทธิ์ใหม่อีกครั้ง

```
class Student{  
    public:  
        char    student_ID[10];  
        char    student_Name[10];  
        void    showGPA( );  
    private:  
        float    score;  
        float    calGPA( );  
};
```

แอททริบิวต์ (Attribute)

การประกาศแอตทริบิวต์

- รูปแบบ

[access modifier]: ชนิดข้อมูล ชื่อแอตทริบิวต์;

ตัวอย่าง

```
public:  
    double   gpa;  
    int      money;  
private:  
    float radius;
```

```
class Circle {  
    private: float radius;  
};
```

การประกาศแอตทริบิวต์

- รูปแบบ

[access modifier] : ชนิดข้อมูล ชื่อแอตทริบิวต์;

ตัวอย่าง

```
class Student {  
    public:      char    id[10];  
    private:    char    StudentName[20];  
    protected: double   gpa;  
}
```


การแทนค่าคลาสและแอตทริบิวต์โดยใช้ UML

```
class Student {  
    public:      char  id[10];  
    private:    char  StudentName[20];  
    protected: double gpa;  
};
```

Student

+ id : string หรือ id : string[10]
- StudentName : string
gpa : double

```
class Rectangle {  
    double width;  
    double height;  
    double getArea() {  
        return width * height;  
    }  
};
```

Rectangle

- width : double
- height : double
- getArea() : double

ข้อเสนอแนะสำหรับการตั้งชื่อแอททริบิวต์

- ชื่อแอททริบิวต์ควรเป็นคำนาม
- นิยมให้ใช้ตัวอักษรเป็นตัวเล็กทั้งหมด
 - ถ้าชื่อแอททริบิวต์ประกอบด้วยคำมากกว่า 1 คำ นิยมให้ตัวแรกของคำเป็นอักษรตัวใหญ่ (ยกเว้นตัวแรก)
 - name, studentName, birdColor

เมธอด (Method)

การประกาศเมธอด

• รูปแบบ

[access modifier] :

```
return_type ชื่อเมธอด ([อาร์กิวเมนต์]){  
    //คำสั่งที่ใช้ในเมธอด  
    [return]; //ส่วนใหญ่ใช้ในกรณีมีการส่งค่ากลับผู้เรียก  
};
```

return_type = ชนิดข้อมูลของค่าที่ส่งกลับ หลังจากเสร็จสิ้นการทำงาน
กรณีที่ ไม่มีการส่งค่าใดๆ กลับควรระบุชนิดข้อมูลเป็น void

ตัวอย่าง

public :

```
float calRectangle(float width, float height) {  
    if( width <= 0 || height <= 0)  
        cout << " width or height is wrong " < endl;  
    return (width * height);  
};
```

การประกาศเมธอด

- รูปแบบ

[access modifier] :

```
return_type ชื่อเมธอด ([อาร์กิวเมนต์]){  
    //คำสั่งที่ใช้ในเมธอด  
    [return]; //ส่วนใหญ่ใช้ในกรณีมีการส่งค่ากลับผู้เรียก  
};
```

ตัวอย่าง

```
public : void calRectangle() { <- arugent จะมีหรือไม่ก็ได้  
    width = 7;  
    height = 3;  
    return; //จะมีหรือไม่ก็ได้  
}  
void = ไม่มีการส่งค่ากลับผู้เรียก
```

การแทนค่าคลาส,แอททริบิวต์ และเมธอดโดยใช้ UML

```
1.class Employee {  
2.    public:  
3.        char name[50];  
4.        char id[10];  
5.        float salary;  
6.        void show_employee(void) {  
7.            cout <<"Name: " <<name<<endl;  
8.            cout <<"Id: " <<id<<endl;  
9.            cout <<"Salary: " <<salary<<endl;  
10.        };  
11.};  
12.Employee worker, boss;
```

Rectangle

```
+ name : string[50]  
+ id : string[10]  
+ float salary  
  
+ show_employee () : void
```

ข้อแนะนำสำหรับการตั้งชื่อเมธอด

- เหมือนกับการตั้งชื่อแอททริบิวต์
 - นิยมให้ใช้ตัวอักษรเป็นตัวเล็กทั้งหมด
 - ถ้าชื่อแอททริบิวต์ประกอบด้วยคำมากกว่า 1 คำ นิยมให้ตัวแรกของคำเป็นอักษรตัวใหญ่ (ยกเว้นตัวแรก)
- สิ่งที่แตกต่างกัน คือ ชื่อเมธอดควรเป็นคำกริยา
 - fly, setTime, addSubjectComputer

การประกาศและสร้างวัตถุ (Object)

การประกาศและสร้างวัตถุ

- การประกาศและสร้างวัตถุ (Object) ทำได้ 2 วิธี
- 1. สร้างวัตถุพร้อมการประกาศคลาส
- 2. สร้างวัตถุหลังการประกาศคลาส

การประกาศและสร้างวัตถุ

- 1. สร้างวัตถุพร้อมการประกาศคลาส

- รูปแบบ

```
class ชื่อคลาส{  
    }ชื่อวัตถุ;
```

ตัวอย่าง

//สามารถประกาศวัตถุได้มากกว่าที่ ชนิด

```
class Circle{  
  
} cir1;
```

```
class Circle{  
  
} cir1,cir2, cir3;
```

1. สร้างวัตถุพร้อมการประกาศคลาส

```
1. class Circle {  
2.     private:  
3.         float radius;  
4.     public:  
5.         void setRadius (float val) { radius = val;};  
6.         float area (void) { return (3.14* radius * radius); };  
7.         float girth (void) { return (3.14*2*radius); }  
8.     ;} cir1;
```

การประกาศและสร้างวัตถุ

- 2. สร้างวัตถุหลังการประกาศคลาส

- รูปแบบ

```
class ชื่อคลาส{  
};  
ชื่อคลาส ชื่อวัตถุ;
```

ตัวอย่าง

```
class Circle{  
  
};  
Circle cir1;
```

```
class Circle{  
  
};  
Circle cir1,cir2, cir3;
```

การประกาศและสร้างวัตถุ

- 2. สร้างวัตถุหลังการประกาศคลาส

```
1.class Circle {  
2.private:  
3.    int r;  
4.public:  
5.    void setRadius (int val) { r = val;};  
6.    float area (void) { return 3.14*r*r; };  
7.    float girth (void) { return 3.14*2*r; };  
8.};  
9.Circle cir1;
```

การเรียกใช้สมาชิกของวัตถุ

การเรียกใช้สมาชิกของวัตถุ

- วัตถุสามารถเรียกใช้งานแอตทริบิวต์และเมทอดของคลาส
ทำได้โดยใช้การใช้เครื่องหมาย dot operator
 1. ใช้เครื่องหมาย . (dot operator) ในการให้ค่า (assign)
แอตทริบิวต์ของคลาส
 2. ใช้เครื่องหมาย . (dot operator) ในการเรียกเมทอดของ
คลาส

```

1. #include <iostream>
2. #include <string.h>
3. using namespace std;
4. class Employee {
5.     public:
6.         char name[50];
7.         char id[10];
8.         float salary;
9.         void showEmployee(void) {
10.             cout <<"Name: "<<name<<endl;
11.             cout <<"Id: "<<id<<endl;
12.             cout <<"Salary: "<<salary<<endl;
13.         };
14. };
15. main( ) {
16.     Employee worker, boss;
17.     strcpy(worker.name, "Steve Job");
18.     strcpy(worker.id,"12345");
19.     worker.salary=20000;
20.
21.     strcpy(boss.name, "Mark Zuckerberg");
22.     strcpy(boss.id,"11111");
23.     boss.salary=500000;
24.
25.     worker.showEmployee();
26.     cout <<endl<<"*****"<<endl;
27.     boss.showEmployee();
28. }

```

```

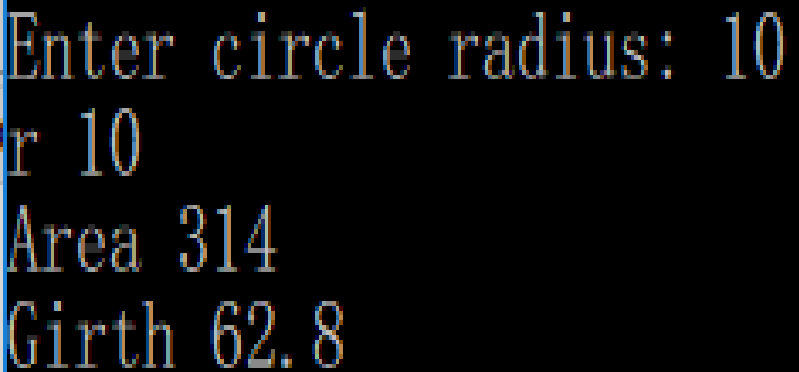
Name: Steve Job
Id: 12345
Salary: 20000

*****
Name: Mark Zuckerberg
Id: 11111
Salary: 500000

```



```
1. #include <iostream>
2. using namespace std;
3. class Circle {
4. public:
5.     int radius;
6. public:
7.     void setRadius (int val) { radius = val;};
8.     float area (void) { return 3.14*radius*radius; };
9.     float girth (void) { return 3.14*2*radius; };
10. };
11. main ()
12. {
13.     Circle cir1;
14.     int val,n_val;
15.     cout << "Enter circle radius: ";
16.     cin >> val;
17.     cir1.setRadius(val);
18.     n_val = cir1.radius;
19.     cout << "radius " << n_val << endl;
20.     cout << "Area " << cir1.area() << endl;
21.     cout << "Girth " << cir1.girth() << endl;
22. }
```



```
Enter circle radius: 10
r 10
Area 314
Girth 62.8
```

สรุปเนื้อหาของบท

- โปรแกรมเชิงวัตถุจะมีคำนิยามที่สำคัญสองคำคือ **วัตถุ** และ **คลาส**
- **วัตถุ** -> สิ่งต่าง ๆ ที่มีอยู่ในชีวิตประจำวันจะประกอบไปด้วยคุณลักษณะและเมธอด
- **คลาส** -> เปรียบเสมือนพิมพ์เขียว วัตถุที่ถูกสร้างมาจากคลาส วัตถุหลายวัตถุสามารถถูกสร้างจากคลาสหนึ่งคลาสได้

สรุปเนื้อหาของบท

- **แอททริบิวต์ของวัตถุ** -> ข้อมูลที่เก็บอยู่ในวัตถุ ซึ่งจะแบ่งออกเป็น ตัวแปรและค่าคงที่
- **เมธอด** -> วิธีการเพื่อใช้ในการจัดการกับคุณลักษณะของวัตถุหรือคุณลักษณะของคลาส
- ภาษา C++ มีรูปแบบการเขียนโปรแกรมเชิงวัตถุเพื่อประกาศคลาส คุณลักษณะ เมธอด และวัตถุ อย่างชัดเจน

คำถาม



Quiz 11

1. จงเขียนโปรแกรมที่มีเงื่อนไขดังต่อไปนี้

- มี Class จำนวน 1 class
- มี private attribute อย่างน้อยจำนวน 1 attribute
- มี public attribute อย่างน้อยจำนวน 1 attribute
- มี method อย่างน้อยจำนวน 3 method
- สร้าง object อย่างน้อยจำนวน 2 object
- เขียน UML ไดอะแกรมของคลาสเพื่ออธิบาย Attribute และ Method

Quiz 11

2. โปรแกรมธนาคารมีการให้บริการฝาก-ถอน เงินให้ลูกค้า
3. โปรแกรมเปลี่ยนหน่วยเงินสกุลไทยไปเป็นเงินสกุลอื่น
4. โปรแกรมเก็บคะแนนของนิสิตในรายวิชา 01418113
คะแนนเต็ม 100 คะแนน