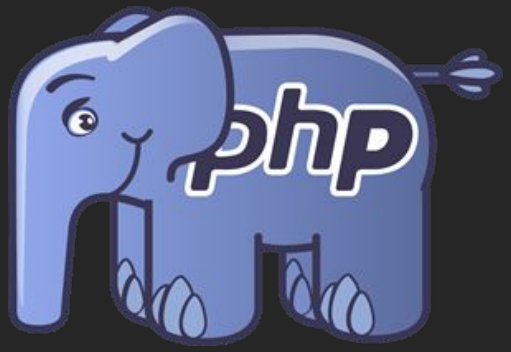


เขียนโปรแกรมเชิงวัตถุ (OOP) ด้วยภาษา PHP

https://www.youtube.com/c/KongRuksiamOfficial/

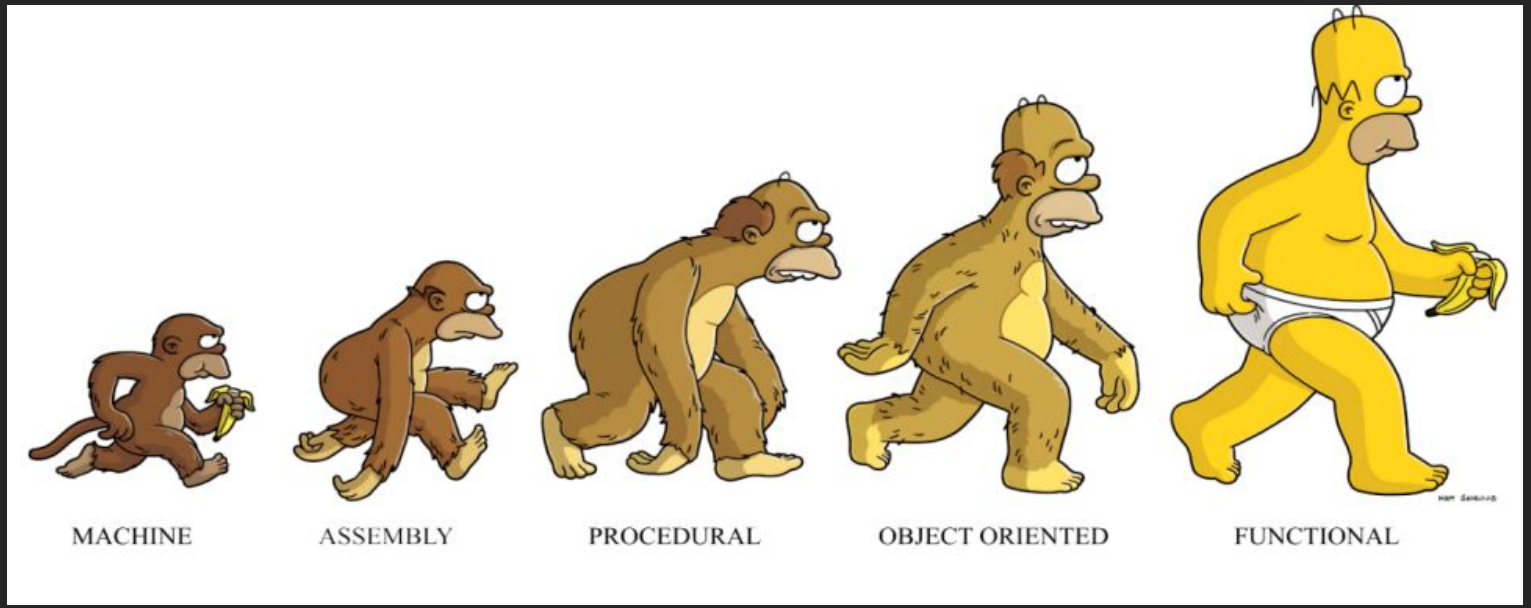
https://www.facebook.com/KongRuksiamTutorial/

OOP

=

Object Oriented Programming

วิวัฒนาการของภาษาเชิงวัตถุ

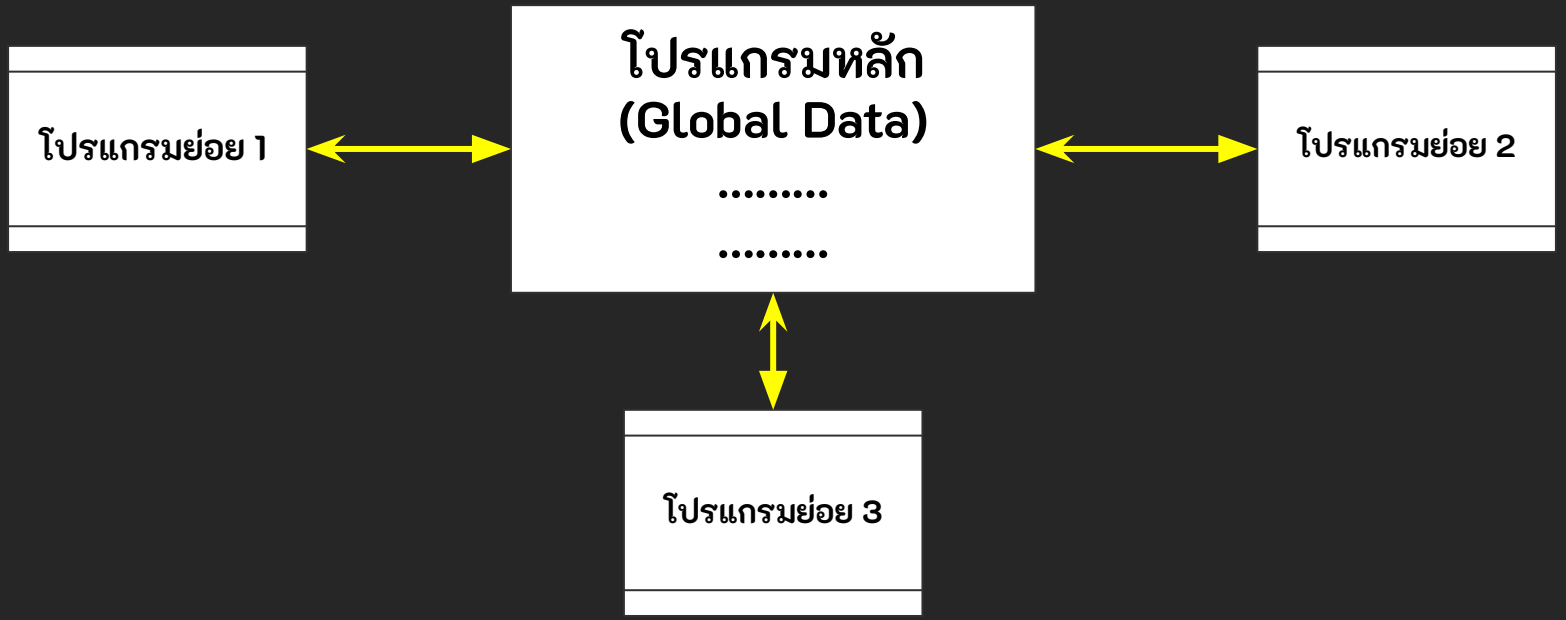


Procedural Programming

โครงสร้างโปรแกรมแบ่งออกเป็น 2 ส่วน คือ

1. โปรแกรมหลัก ที่มี Data เป็นส่วนประกอบ
2. โปรแกรมย่อย โดย Data ที่ประกาศอยู่ในโปรแกรมหลัก จะถูกเรียกใช้งานในโปรแกรมย่อยต่างๆภายในโปรแกรม ลักษณะของ Data ที่มีการประกาศใช้งานทั่วโปรแกรม ทั้งหมดจะเรียกว่า “Global Data”

Procedural Programming



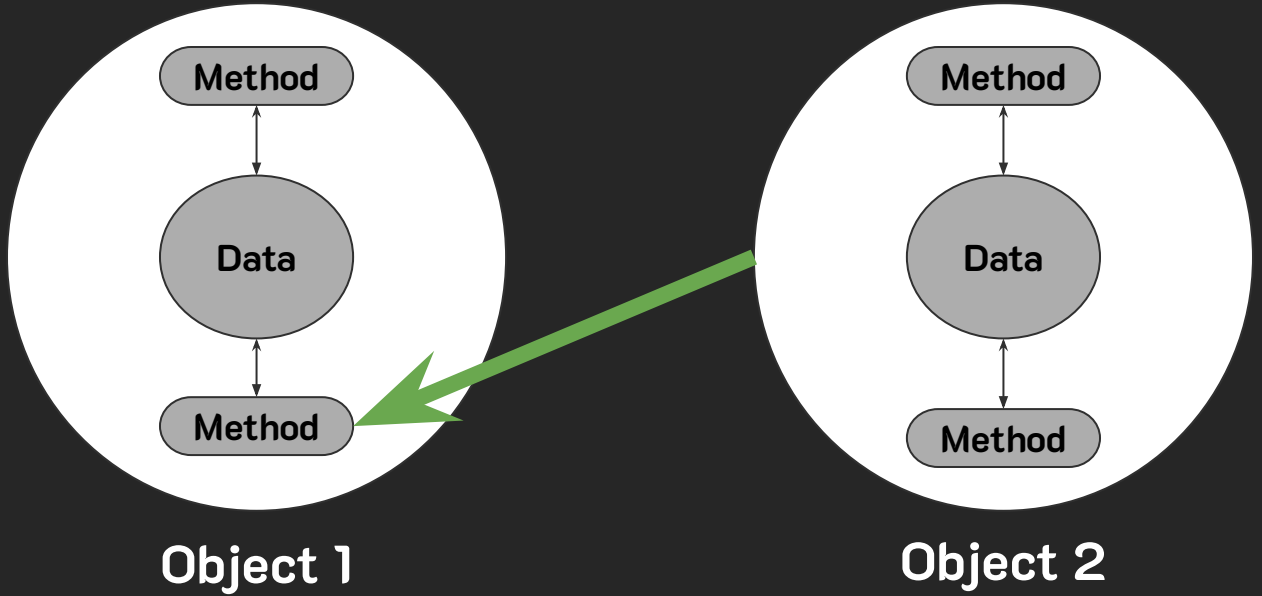
Procedural Programming

มีข้อจำกัด คือ โปรแกรมย่อยต่างเรียกใช้งานข้อมูลจากโปรแกรมหลักเดียวกัน อาจทำให้เกิดปัญหาจากการเปลี่ยนแปลงค่าของข้อมูล และส่งผลเสียต่อการควบคุมการเปลี่ยนแปลงข้อมูลของโปรแกรมซึ่งยากต่อการแก้ไขโปรแกรมในภายหลัง

Procedural Programming

สำหรับแนวทางการเขียนโปรแกรมเชิงวัตถุนั้นจะต่างออกไป Data หรือข้อมูลที่ถูกประกาศขึ้นมานั้นจะถูกใช้งานเฉพาะภายในแต่ละ Object กล่าวคือ 1 Object จะมี Data และการทำงานของโปรแกรม (Method) รวมอยู่ด้วยกัน

Object Oriented Programming

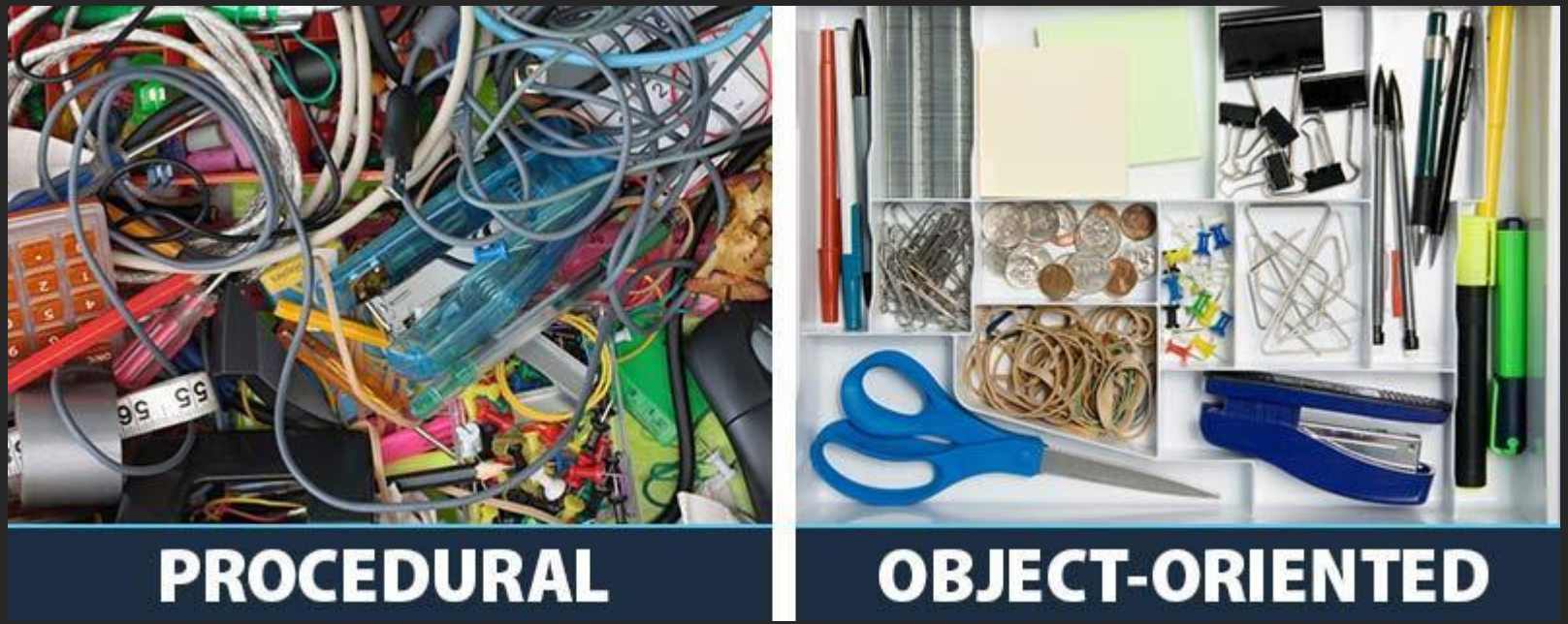


Object Oriented Programming

แนวคิดการเขียนโปรแกรมเชิงวัตถุนั้นจะจัด Data ไว้ในแต่ละ Object เพื่อปกป้องข้อมูลภายใน Object และลดปัญหาการเปลี่ยนแปลงข้อมูลภายใน Object โดยไม่ได้รับอนุญาต

Object หนึ่งจะสามารถเข้าถึงข้อมูลในอีก Object หนึ่งได้ก็ต่อเมื่อมีการใช้ Method ของ Object ที่เป็นเจ้าของข้อมูลเท่านั้น จึงส่งผลให้การแก้ไขโปรแกรมในภายหลังทำได้สะดวกยิ่งขึ้น

Procedural VS Object Oriented



https://www.youtube.com/c/KongRuksiamOfficial/

https://www.facebook.com/KongRuksiamTutorial/

Procedural VS Object Oriented

ภาษาเชิงกระบวนการ (Procedural Programming Language)

- โปรแกรมจะแบ่งออกเป็นส่วนย่อยๆ เรียกว่าโมดูล (Module)
- แต่ละโมดูลควรออกแบบให้มีการทำงานเพียง 1 งานเท่านั้น
- การออกแบบให้แต่ละโมดูลมีความเป็นอิสระต่อกันนั้นทำได้ยาก

ภาษาเชิงวัตถุ (Object Oriented Programming Language)

- การพัฒนาโปรแกรมเป็นการเลียนแบบการทำงานเชิงวัตถุ
- ออกแบบให้วัตถุมีความเป็นอิสระต่อกันทำได้ง่ายด้วยคุณสมบัติเชิงวัตถุ
- สามารถนำโปรแกรมกลับมาใช้ใหม่ (Reuse) ได้ดีกว่าภาษาเชิงกระบวนการ

Procedural VS Object Oriented

ภาษาเชิงกระบวนการ	ภาษาเชิงวัตถุ
กำหนดขั้นตอนการแก้ปัญหา	กำหนดปัญหาเป็นองค์ประกอบ (วัตถุ)
โปรแกรมและข้อมูลอยู่คนละส่วนกัน	เอาส่วนโปรแกรมและข้อมูลไว้ด้วยกัน
ออกแบบจากล่างขึ้นบน	ออกแบบเป็นวัตถุ
แก้ไขง่ายเพราะแต่ละส่วนไม่มีความสัมพันธ์กัน	การแก้ไขไม่กระทบส่วนอื่นๆของโปรแกรม เพราะวัตถุจะมีความสมบูรณ์ในตัวเอง

การเขียนโปรแกรมเชิงวัตถุ
(Object Oriented Programming)


https://www.youtube.com/c/KongRuksiamOfficial/


https://www.facebook.com/KongRuksiamTutorial/

เขียนโปรแกรมเชิงวัตถุ

คือ การเขียนโปรแกรมอีกรูปแบบหนึ่ง โดยมองสิ่งต่างๆ เป็นวัตถุ โดยในวัตถุจะมีคุณสมบัติและพฤติกรรมซึ่งมีมุมมองจากพื้นฐานความจริงในชีวิตประจำวัน

องค์ประกอบพื้นฐาน

- คลาส (Class) & วัตถุ (Object)
- เมธอด (Method)
- การห่อหุ้ม (Encapsulation)
- การสืบทอดคุณสมบัติ (Inheritance)
- การพ้องรูป (Polymorphism)

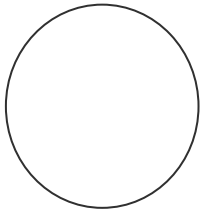
องค์ประกอบพื้นฐาน

คลาส (class) คือ ต้นแบบของวัตถุการจะสร้างวัตถุขึ้นมาได้จะต้องสร้างคลาสขึ้นมาเป็นโครงสร้างต้นแบบสำหรับวัตถุก่อนเสมอ

วัตถุหรือออบเจ็ค (object) คือ สิ่งที่ถูกสร้างจากคลาส ประกอบด้วยคุณสมบัติ 2 ประการ คือ คุณลักษณะ และ พฤติกรรม

องค์ประกอบพื้นฐาน

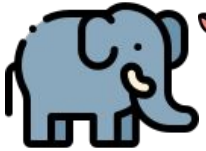
Class



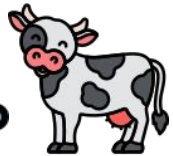
Animal

Object

สิงโต

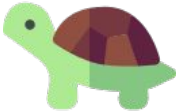


ช้าง



วัว

ไก่

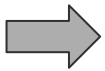


เต่า

องค์ประกอบพื้นฐาน

Class

Employee



Accounting

Object

Programmer

Sale

องค์ประกอบพื้นฐาน

- **คุณลักษณะ (Attribute หรือ Data member)** สิ่งที่บ่งบอกลักษณะทั่วไปของวัตถุ

- **พฤติกรรม (Behavior หรือ Method)** คือ พฤติกรรมทั่วไปของวัตถุที่สามารถกระทำได้

Animal

คุณสมบัติ (Attribute)

ชื่อ : ช้าง

สี : ฟ้าอ่อน

ประเภท : สัตว์บก

น้ำหนัก : 6 ตัน

จำนวนเท้า : 4 เท้า

พฤติกรรม (Method/Behavior)

- วิ่ง
- นอน
- ส่งเสียงร้อง

Animal

คุณสมบัติ (Attribute)

ชื่อ : นก

สี : เหลือง

ประเภท : สัตว์ปีก

น้ำหนัก : 0.8 กิโลกรัม

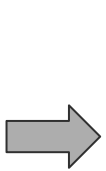
จำนวนเท้า : 2 เท้า

พฤติกรรม (Method/Behavior)

- **บิน**
- **เดิน**
- **ส่งเสียงร้อง**

องค์ประกอบพื้นฐาน

Class



Object

Employee Accounting Programmer Sale

Employee (พนักงาน)

Accounting	Programmer	Sale
Attribute - ชื่อ - เงินเดือน	**Attribute** - ชื่อ - เงินเดือน - ประสบการณ์ทำงาน	**Attribute** - ชื่อ - เงินเดือน - เขตพื้นที่รับผิดชอบ
Method - คำนวณเงินเดือน - แสดงรายละเอียด	**Method** - คำนวณเงินเดือน - ค่าล่วงเวลา - แสดงรายละเอียด	**Method** - คำนวณเงินเดือน - ค่าคอมมิสชั่น - แสดงรายละเอียด

สรุป

- Class – ต้นแบบของวัตถุ
- Object – สิ่งที่ถูกสร้างขึ้นมาจาก Class ประกอบด้วย
  - คุณสมบัติ (Attribute)
  - พฤติกรรม (Method)
- คุณสมบัติของการเขียนโปรแกรมเชิงวัตถุ
  - การห่อหุ้ม(Encapsulation)
  - การสืบทอด (Inheritance)
  - การพ้องรูป (Polymorphism)

การสร้าง Class & Object

การสร้าง Class

```
class class_name{

    Attribute & Method

}
```

```
class Employee{

    Attribute & Method

}
```

การสร้าง Object

```
$obj_name = new class_name();
```

การสร้าง Object

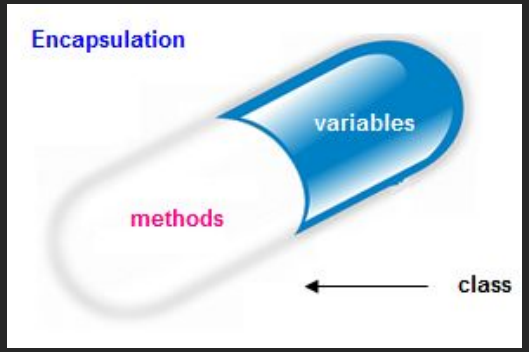
```
$empl = new Employee();
```

การห่อหุ้ม (Encapsulation)

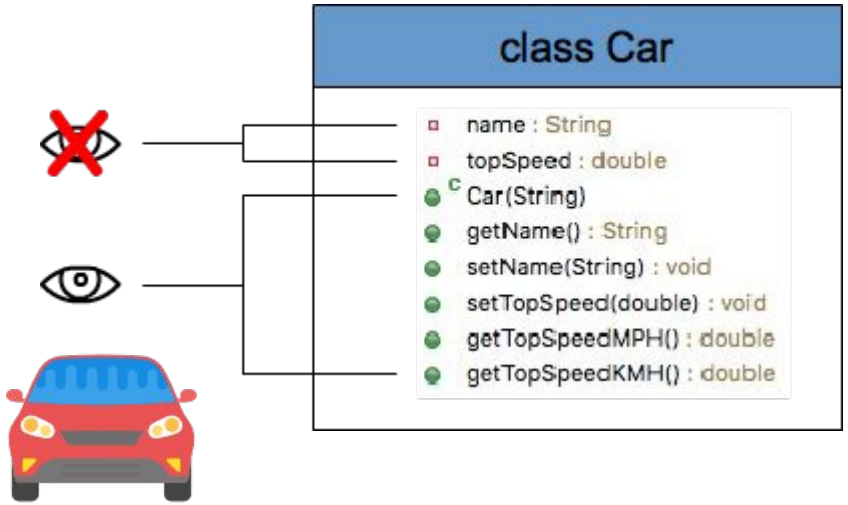
การห่อหุ้ม (Encapsulation)

1. เป็นกระบวนการซ่อนรายละเอียดการทำงานและข้อมูลไว้ภายใน ไม่ให้ภายนอกสามารถมองเห็นและสามารถทำการเปลี่ยนแปลงแก้ไขข้อมูลภายในได้ ซึ่งเป็นผลทำให้เกิดความเสียหายแก่ข้อมูล

2. สามารถสร้างความปลอดภัยให้แก่ข้อมูลได้ เนื่องจากข้อมูลจะถูกเข้าถึงจากผู้มีสิทธิ์เท่านั้น



การห่อหุ้ม (Encapsulation)



Access Modifier

Access Modifier

คือ ระดับในการเข้าถึง Class, Attribute, Method และอื่น ๆ ในภาษาเชิงวัตถุ โดยมีประโยชน์อย่างมากในเรื่องของการกำหนดระดับการเข้าถึง สิทธิในการเข้าใช้งาน การซ้อนข้อมูล และอื่น ๆ

Access Modifier

- **Public** เป็นการประกาศระดับการเข้าถึงที่เข้มงวดน้อยที่สุดหรือกล่าวได้ว่าใครๆ ก็สามารถเข้าถึงและเรียกใช้งานได้
- **Protected** เป็นการประกาศระดับการเข้าถึงที่เกี่ยวข้องกับเรื่องการสืบทอด (Inheritance) ทำให้คลาสนั้นๆ สามารถเรียกใช้งานสมาชิกของคลาสที่ถูกกำหนดเป็น Protected ได้
- **Private** เป็นการประกาศระดับการเข้าถึงที่เข้มงวดที่สุด กล่าวคือ จะมีแต่คลาสของตัวมันเองเท่านั้นที่มีสิทธิ์ใช้งานได้

การสร้าง Attribute

this keyword

$this

การใช้คีย์เวิร์ด $this จะเป็นตัวชี้หรือตัวที่บ่งบอกว่าตอนนี้เราทำงานกับวัตถุใด ให้บอกตัวตนของวัตถุนั้นๆ เช่น การกำหนดคุณสมบัติต่างๆ ในวัตถุ เป็นต้น

การสร้าง Attribute

```
class Employee {
    private $id;
    private $name;
    private $salary;
}
```

Modifier

Attribute

เรียกใช้งานภายใน Class

- $this->attr_name

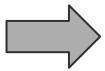
เรียกใช้งานภายนอก Class

- $obj_name->attr_name

กำหนด Attribute ให้วัตถุ

Class

Employee



ชื่อ : เจน
แผนก : บัญชี

Object

ชื่อ : ก้อง
แผนก : ไอที

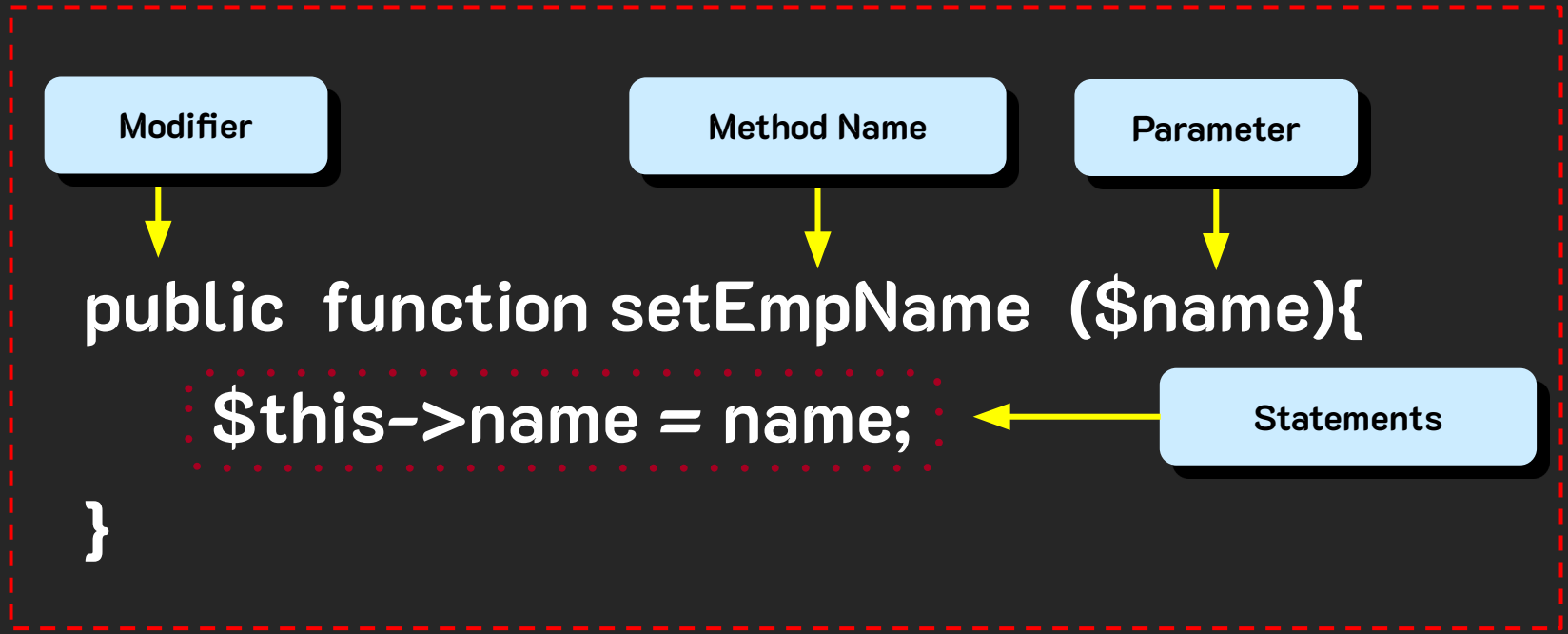
ชื่อ : โจ้
แผนก : ฝ่ายขาย

การสร้าง Method

การสร้าง Method

```
modifier  function method_name (parameter ){
      // statement
}
```

การสร้าง Method



การสร้าง Method

- เรียกใช้งานภายใน Class
  - $this->method_name()
- เรียกใช้งานภายนอก Class
  - $obj_name->method_name()

Constructor

Constructors

เป็นฟังก์ชั่นที่จะถูกใช้งานเมื่อตอนเริ่มต้นสร้างวัตถุและทำงานอัตโนมัติเพียงครั้งเดียวในตอนเริ่มต้น

โครงสร้าง Constructor

```
function __construct([parameter]){
}
```

Destructors

Destructors

เป็นฟังก์ชั่นพิเศษที่ตรงข้ามกับ Constructor จะถูกใช้งานเมื่อสิ้นสุดการทำงานของ class

โครงสร้าง Destructor

```
function __destruct(){
}
```

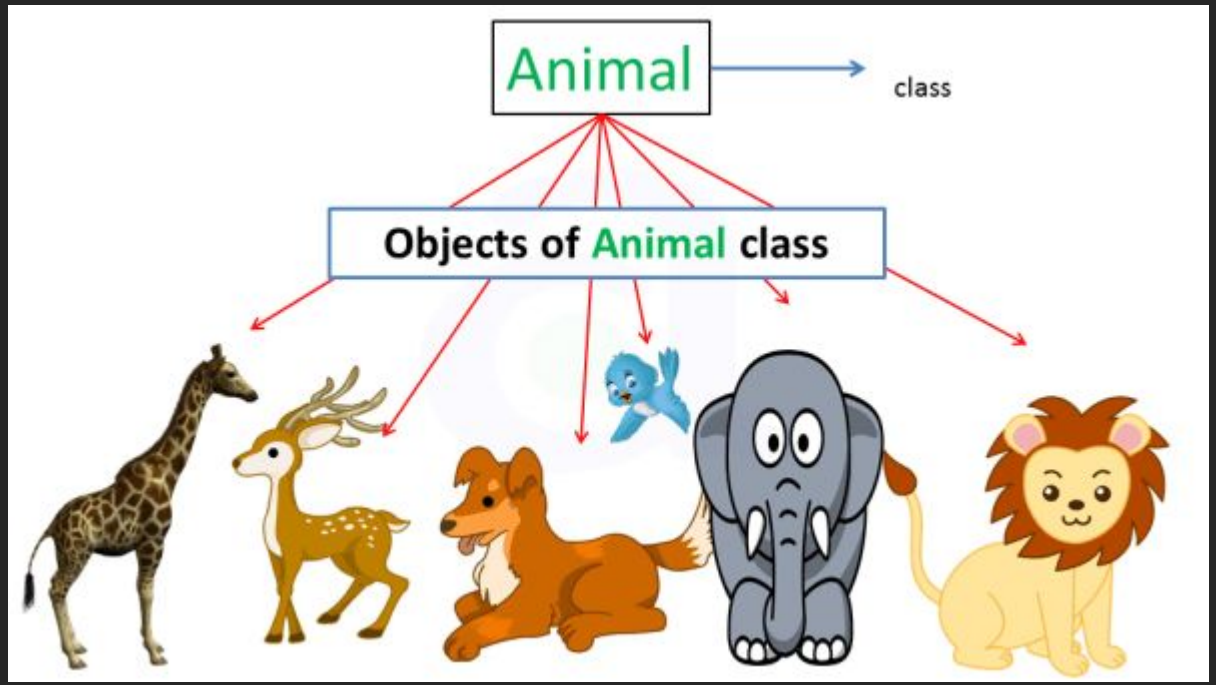
การสืบทอดคุณสมบัติ (Inheritance)

การสืบทอดคุณสมบัติ (Inheritance)

คือ ทำการสร้างสิ่งใหม่ขึ้นด้วยการสืบทอด หรือรับเอา (inherit) คุณสมบัติบางอย่างมาจากสิ่งเดิมที่มีอยู่แล้ว

ข้อดีของการ inheritance คือ สามารถนำสิ่งที่เคยสร้างขึ้นแล้วนำกลับมาใช้ใหม่ (re-use) ได้ ทำให้ช่วยประหยัดเวลาการทำงานลง เนื่องจากไม่ต้องเสียเวลาพัฒนาใหม่หมด

คลาสแม่ (Superclass) คลาสลูก (Subclass)



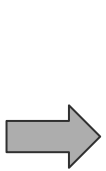
Employee (พนักงาน)

Accounting	Programmer	Sale
Attribute - ชื่อ - เงินเดือน	**Attribute** - ชื่อ - เงินเดือน - ประสบการณ์ทำงาน	**Attribute** - ชื่อ - เงินเดือน - เขตพื้นที่รับผิดชอบ
Method - คำนวณเงินเดือน - แสดงรายละเอียด	**Method** - คำนวณเงินเดือน - ค่าล่วงเวลา - แสดงรายละเอียด	**Method** - คำนวณเงินเดือน - ค่าคอมมิสชั่น - แสดงรายละเอียด

คุณสมบัติต่างๆจากแม่จะถูกถ่ายทอดไปยังลูก

Class

ยกเว้น Private Attribute & Private Method



Employee

Accounting

Programmer

Sale

การสืบทอดคุณสมบัติ (Extends)

การสืบทอดคุณสมบัติ (Inheritance)

คลาสแม่

```
class SuperClass{


// Attribute & Method
}
```

คลาสลูก

```
class SubClass extends SuperClass{


// Attribute & Method
}
```

ตัวอย่าง

คลาสแม่

```
class Employee{


// Attribute & Method
}
```

คลาสลูก

```
class Programmer extends Employee{


// Attribute & Method
}
```

Abstract Class & Method

Abstract คืออะไร

- abstract เป็นคีย์เวิร์ดในภาษา PHP สามารถกำหนดคีย์เวิร์ด abstract นี้ให้กับ คลาส หรือเมธอด ก็ได้ ใช้สำหรับกำหนดโครงสร้างโดยไม่ระบุรายละเอียดการทำงานด้านใน
- กฎของ abstract หากคลาสใดสืบทอดมาจาก abstract class คลาสนั้นจะต้องทำการระบุเมธอดทุกเมธอดที่เป็น abstract method ใน abstract class ไว้เสมอ (ไม่กำหนดรายละเอียดก็ได้แต่จะต้องมีการเขียน abstract method ทุกเมธอดลงไปในคลาสนั้นด้วย)

Abstract Class & Method

- **abstract class** หากคลาสใดเป็น abstract class ต้องสร้างเมธอดที่เป็น abstract method เพื่อกำหนดโครงสร้าง
- **abstract method** คือ เมธอดว่างเปล่าที่ยังไม่ได้มีการกำหนดรายละเอียดการทำงานลงไป จะถูกกำหนดรายละเอียดลงไปภายหลัง โดยคลาสลูกที่ได้รับการสืบทอดจากคลาสของ abstract method เหล่านั้น

Employee (พนักงาน)

Accounting	Programmer	Sale
Attribute - ชื่อ - เงินเดือน	**Attribute** - ชื่อ - เงินเดือน - ประสบการณ์ทำงาน	**Attribute** - ชื่อ - เงินเดือน - เขตพื้นที่รับผิดชอบ
Method - คำนวณเงินเดือน - แสดงรายละเอียด	**Method** - คำนวณเงินเดือน - ค่าล่วงเวลา - แสดงรายละเอียด	**Method** - คำนวณเงินเดือน - ค่าคอมมิสชั่น - แสดงรายละเอียด

Final Keyword

Final Keyword

เนื่องจากการเขียนโปรแกรมเชิงวัตถุสามารถสืบทอด (Inherit) Class ได้ ถ้าในกรณีที่ไม่ต้องการให้ Class ถูกสืบทอดไปใช้งานหรือไม่ต้องการให้ SubClass เขียน Method ใหม่ทับ Method ของ Super Class ได้ หรือไม่ต้องการให้แก้ไขค่า Attribute สามารถป้องกันได้โดย

- ใช้ “final” ร่วมกับ Class และ Method

สรุป Final Keyword

- เป็นคีย์เวิร์ดที่สามารถกำหนดให้กับ class , method , attribute ได้
- กำหนด final ให้คลาสจะทำให้คลาสนั้นไม่สามารถมี subclass ได้
- กำหนด final ให้เมธอดจะทำให้เมธอดนั้นไม่สามารถ override method นั้นได้
- กำหนด final ให้ data หรือ attribute จะทำให้เป็นค่าคงที่ (constant)

ตัวอย่างการใช้งาน

```
final class class_name{



}
```

```
class class_name{

    final function name(){

    }

}
```

Static Keyword

Static Attribute

คือ Attribute ที่สามารถเรียกใช้งานได้โดยตรง ไม่ต้องเรียกผ่าน Object การสร้าง Static Attribute จะเหมือนกับการสร้าง Attribute โดยทั่วไปเพียงแค่เติม static นำหน้า Attribute

```
static $attribute_name = value
```

การเรียกใช้งาน Static Attribute

- เรียกใช้งานภายใน Class
  - self::$attribute_name
- เรียกใช้งานภายนอก Class
  - class_name::$attribute_name

Static Method

คือ Method ที่สามารถเรียกใช้งานได้โดยตรง ไม่ต้องเรียกผ่าน Object การสร้าง Static Method จะเหมือนกับการสร้าง Method โดยทั่วไปเพียงแค่เติม static นำหน้า function

```
static function method_name([parameter]){
        //statement
}
```

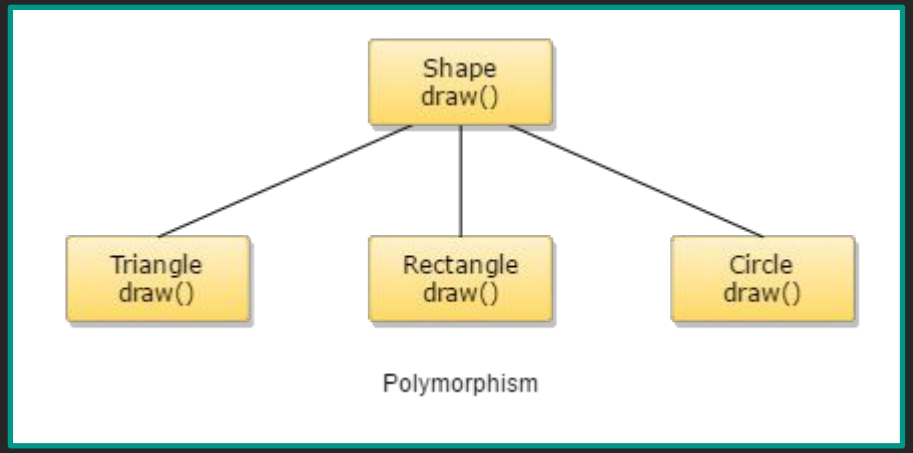
การเรียกใช้งาน Static Method

- เรียกใช้งานภายใน Class
  - self::method_name()
- เรียกใช้งานภายนอก Class
  - class_name::method_name()

การพ้องรูป (Polymorphism)

การพ้องรูป (Polymorphism)

Polymorphism เกิดจาก poly (หลากหลาย) + morphology (รูปแบบ)

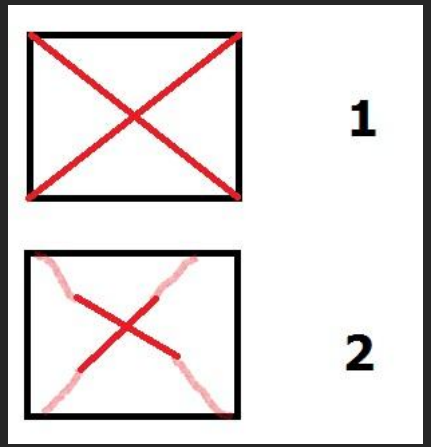


ในทางโปรแกรมคือการที่เมธอดชื่อเดียวกัน สามารถรับอาร์กิวเมนต์ที่แตกต่างกันได้หลายรูปแบบ โดยเมธอดนี้จะถูกเรียกว่า overload method (เมธอดถูกโอเวอร์โหลด)

การพ้องรูป (Polymorphism)

“ ข้อความเดียวกันแต่กระบวนการทำงานภายในแตกต่างกันนั้น เรียกว่า **การพ้องรูป** หรือ **polymorphism** ”

กา



การพ้องรูป (Polymorphism)

คุณสมบัติการพ้องรูป คือ สามารถตอบสนองต่อ Method เดียวกันด้วยวิธีการที่ต่างกันและสามารถกำหนด object ได้หลายรูปแบบมีข้อดี คือ ทำให้โปรแกรมสามารถปรับเปลี่ยนหรือเพิ่มเติมได้ง่ายขึ้น

ดำ

https://www.youtube.com/c/KongRuksiamOfficial/

https://www.facebook.com/KongRuksiamTutorial/

Overloading & Overriding

Overloading & Overriding Method

- **Overloading Method** คือ **เมธอดที่มีชื่อเหมือนกันและอยู่ภายในคลาสเดียวกัน** สิ่งที่แยกความแตกต่างของเมธอดที่เป็น overload method คือ **พารามิเตอร์** (เป็นผลมาจากคุณสมบัติ OO คือ **polymorphism**)

- **Overriding Method** คือ **เมธอดของคลาสลูก (subclass) ที่มีชื่อเหมือนกับเมธอดของคลาสแม่ (superclass)** (เป็นผลมาจากคุณสมบัติ OO คือ **inheritance**)

Employee (พนักงาน)

Accounting	Programmer	Sale
Attribute - ชื่อ - เงินเดือน	**Attribute** - ชื่อ - เงินเดือน - ประสบการณ์ทำงาน	**Attribute** - ชื่อ - เงินเดือน - เขตพื้นที่รับผิดชอบ
Method - คำนวณเงินเดือน - แสดงรายละเอียด	**Method** - คำนวณเงินเดือน - ค่าล่วงเวลา - แสดงรายละเอียด	**Method** - คำนวณเงินเดือน - ค่าคอมมิสชั่น - แสดงรายละเอียด

อินเทอร์เฟซ (Interface)

อินเทอร์เฟซ (Interface)

อินเทอร์เฟซ (interface) มีหลักการคล้ายกับ abstract class คือ สร้างอินเทอร์เฟซขึ้นมาก็เพื่อกำหนดโครงสร้างของเมธอดที่จำเป็นใช้งานขึ้นมา แต่ยังไม่ได้กำหนดรายละเอียดการทำงานใดๆ ลงไปให้กับเมธอดนั้น (abstract method) เมธอดในอินเทอร์เฟซจึงเป็นเมธอดที่ว่างเปล่า ซึ่งในภายหลังจึงมีการกำหนดรายละเอียดของเมธอดเหล่านั้นลงไป โดยถูกกำหนดโดยคลาสที่เรียกใช้อินเทอร์เฟซนั้นๆ

Interface กับ Abstract class ต่างกันอย่างไร

- เมธอดใน abstract class ไม่เป็น abstract method ก็ได้ แต่เมธอดทุกเมธอดใน interface เป็น abstract method
- คลาสที่จะเรียกใช้งานเมธอดในอินเทอร์เฟซไม่จำเป็นต้องมีความสัมพันธ์ใดๆ กับอินเทอร์เฟซทั้งสิ้น

Interface กับ Abstract class ต่างกันอย่างไร

- คลาสที่จะเรียกใช้งาน abstract method ใน abstract class จะต้องสืบทอดคุณสมบัติไปจาก abstract class นั้น แล้วจึงทำการสร้างเมธอดของตัวเองขึ้นมาให้มีชื่อเดียวกับ abstract method ใน abstract class โดยกำหนดรายละเอียดการทำงานให้กับ abstract method เหล่านั้นตามต้องการ

การสร้าง Interface

```
interface interface_name{

    function method_nameN();// สร้าง method มีกี่จำนวนก็ได้ สามารถ
กำหนด public หรือ abstract นำหน้าก็ได้ (ไม่ระบุก็ได้)

}
```

การใช้งาน Interface

```
class name [extends superclass] implements interface_N{
    function methodN(){
        //รายละเอียดการทำงาน
    }
}
```

Type Hinting

Type Hinting

โดยทั่วไปพารามิเตอร์ใน Method จะมีรูปแบบเป็น ข้อความ , ตัวเลข หรือ อาร์เรย์ เท่านั้น ถ้าต้องการอยากส่งค่า Object เข้าไปทำงานต้องทำการแปลงค่าเป็นชนิดข้อมูลก่อน ทำให้เสียเวลาในการประมวลผล จึงได้มีการพัฒนา Type Hinting ขึ้นมาเพื่อให้สามารถส่งพารามิเตอร์เป็นรูปแบบ Object ได้

Type Hinting

```
modifier  function method_name (class_name object_name ){
    // statement
}
```

คอร์สเรียน PHP บน Udemy

วิทยากร

Kong Ruksiam

Programmer , Developer

ผู้เรียนทั้งหมด	รีวิว
1,912	450

เกี่ยวกับฉัน

โปรแกรมเมอร์และนักพัฒนาเกม รวมถึงสอนเกี่ยวกับการเขียนโปรแกรมในช่องยูทูป KongRuksiam , KongRuksiam Official และเป็นเจ้าของแฟนเพจ KongRuksiam มีผู้ติดตามมากกว่า 50,000 คน

Facebook

Youtube

https://www.udemy.com/course/php-pdo-mysql-crud/