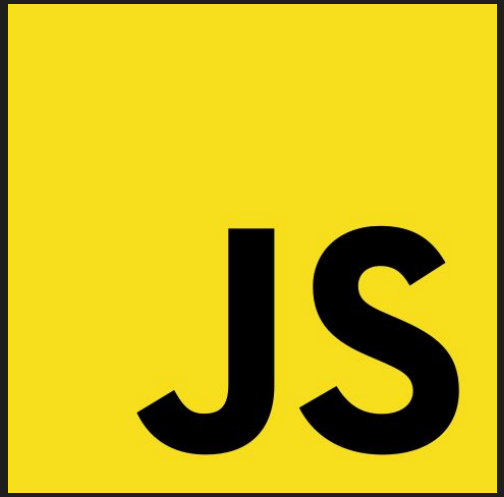


เขียนโปรแกรมเชิงวัตถุ
(Object Oriented Programming)

เขียนโปรแกรมเชิงวัตถุ

คือ การเขียนโปรแกรมอีกรูปแบบหนึ่ง โดยมองสิ่งต่างๆ เป็นวัตถุ โดยในวัตถุจะมีคุณลักษณะและพฤติกรรม โดยแนวคิดการเขียนโปรแกรม มาจากพื้นฐานความจริงในชีวิตประจำวัน

เขียนโปรแกรมเชิงวัตถุ

JS

ในภาษา JavaScript นั้นรองรับการสร้าง Object อยู่แล้ว แต่ข้อกำหนดของ Object ยังไม่เพียงพอสำหรับการประยุกต์ใช้งานในรูปแบบที่ซับซ้อน

เพราะ Object แบบดั่งเดิมของ JavaScript นั้นไม่ตรงตามหลักการเขียนโปรแกรมเชิงวัตถุที่ใช้ในภาษาคอมพิวเตอร์อื่นๆ ทำให้บางครั้งสร้างความสับสนให้กับผู้เขียนโปรแกรม

เขียนโปรแกรมเชิงวัตถุ

JS

```
const obj_name = {
    property,
    method

}
```

```
class class_name{

    Property & Method

}
```

เขียนโปรแกรมเชิงวัตถุ

JS

ตั้งแต่ ECMAScript 6 เป็นต้นมา จึงได้มีการเพิ่มทางเลือกสำหรับการสร้าง Object ให้สามารถเขียนในรูปแบบการเขียนโปรแกรมเชิงวัตถุมากยิ่งขึ้น โดยสามารถเขียนในรูปแบบ Class รวมถึงกำหนดคุณสมบัติการเขียนโปรแกรมเชิงวัตถุให้มีความใกล้เคียงกับภาษาอื่นๆ

องค์ประกอบพื้นฐาน

JS

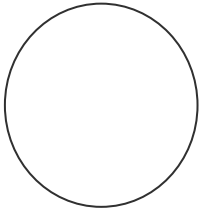
คลาส (class) คือ ต้นแบบของวัตถุการจะสร้างวัตถุขึ้นมาได้จะต้องสร้างคลาสขึ้นมาเป็นต้นแบบของวัตถุก่อนเสมอ

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

องค์ประกอบพื้นฐาน

JS

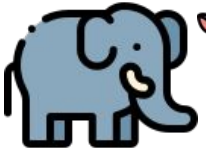
Class



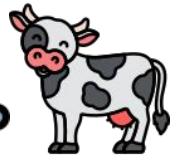
Animal

Object

สิงโต

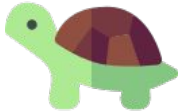


ช้าง



วัว

ไก่

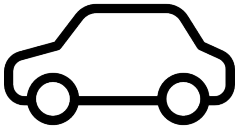


เต่า

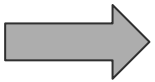
องค์ประกอบพื้นฐาน

JS

Class



รถยนต์



Object

รถเก๋ง

รถบรรทุก

รถสปอร์ต

องค์ประกอบของวัตถุ (Object)

JS

- **คุณลักษณะ (Property)** คือ สิ่งที่บ่งบอกลักษณะทั่วไปของวัตถุ หรือข้อมูลประจำตัวของวัตถุ

- **พฤติกรรม (Method)** คือ พฤติกรรมทั่วไปของวัตถุที่สามารถกระทำได้

Animal

คุณลักษณะ (Property)

ชื่อ : ช้าง

สี : ฟ้าอ่อน

ประเภท : สัตว์บก

น้ำหนัก : 6 ตัน

จำนวนเท้า : 4 เท้า

พฤติกรรม (Method/Behavior)

- วิ่ง
- นอน
- ส่งเสียงร้อง

Animal

คุณลักษณะ (Property)

ชื่อ : นก

สี : เหลือง

ประเภท : สัตว์ปีก

น้ำหนัก : 0.8 กิโลกรัม

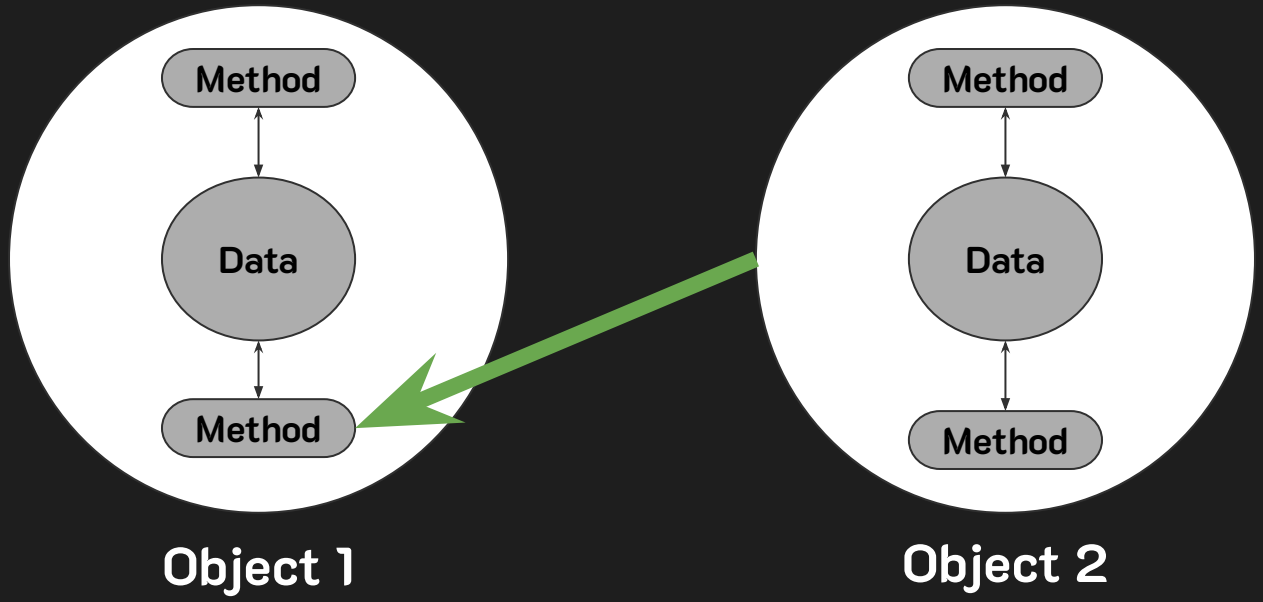
จำนวนเท้า : 2 เท้า

พฤติกรรม (Method/Behavior)

- บิน
- เดิน
- ส่งเสียงร้อง

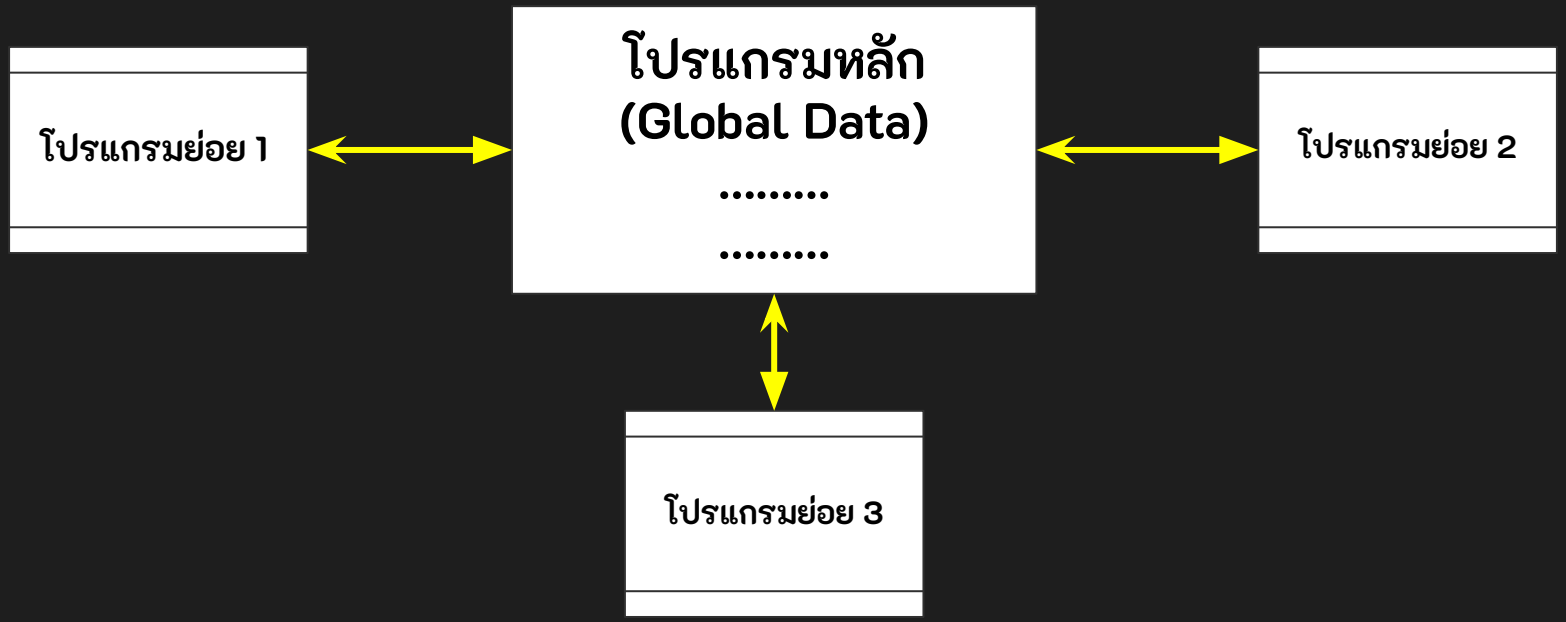
เขียนโปรแกรมเชิงวัตถุ

JS



Procedural Programming

JS



เขียนโปรแกรมเชิงวัตถุ

JS

แนวคิดการเขียนโปรแกรมเชิงวัตถุนั้นจะจัด Data ไว้ในแต่ละ Object เพื่อปกป้องข้อมูลภายใน Object และลดปัญหาการเปลี่ยนแปลงข้อมูลภายใน Object โดยไม่ได้รับอนุญาต

Object หนึ่งจะสามารถเข้าถึงข้อมูลในอีก Object หนึ่งได้ก็ต่อเมื่อมีการใช้ Method ของ Object ที่เป็นเจ้าของข้อมูลเท่านั้น จึงส่งผลให้การแก้ไขโปรแกรมในภายหลังทำได้สะดวกยิ่งขึ้น

สรุปการเขียนโปรแกรมเชิงวัตถุ

JS

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

ต้องมีพื้นฐานอะไรบ้าง

JS

- JavaScript เบื้องต้น
- JavaScript ES6

เครื่องมือพื้นฐาน

JS

- Node.js
- Visual Studio Code
- Code Runner (Extension)
- Prettier - Code formatter (Extension)

JS

การสร้าง Class & Object

การสร้าง Class

JS

```
class class_name{

    Property & Method

}
```

```
class User{

    Property & Method

}
```

ตัวอย่างการสร้าง Object

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

ตัวอย่างการสร้าง Object

```
const user1 = new User();
```

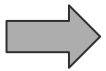
การสร้าง Property

การสร้าง Property

JS

Class

User



ชื่อ : เจน
รหัสผ่าน : xxxx

Object

ชื่อ : โจโจ้
รหัสผ่าน : xxxx

ชื่อ : ก้อง
รหัสผ่าน : xxxx

การเรียกใช้งาน

JS

เรียกใช้งานภายใน Class

- this.propertyName

เรียกใช้งานภายนอก Class

- obj_name.propertyName

กฎการตั้งชื่อ

JS

1. ชื่อ Class ควรกำหนดให้ตัวอักษรตัวแรกเป็นตัวพิมพ์ใหญ่ ที่เหลือเป็นพิมพ์เล็ก เช่น MyClass , User เป็นต้น
2. ชื่อ Object กำหนดเป็นตัวพิมพ์เล็กทั้งหมด
3. Property กำหนดเป็นตัวพิมพ์เล็ก เช่น name , age เป็นต้น

Constructor

Constructor

JS

เป็นฟังก์ชั่นพิเศษที่จะถูกเรียกใช้งาน เมื่อสร้างวัตถุขึ้นมาและจะทำงานอัตโนมัติในตอนเริ่มต้นเพียงครั้งเดียว

โครงสร้าง Constructor

```
constructor([parameter]){
}
```

ประเภทของ Constructor

JS

- Default Constructor คือ Constructor เริ่มต้นที่มีอยู่ในทุกคลาส

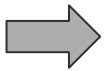
- Parameterized Constructor คือ Constructor ที่สามารถส่งพารามิเตอร์เข้าไปทำงานได้

กำหนดค่าเริ่มต้น

JS

Class

User



ชื่อ : เจน
รหัสผ่าน : xxxx

Object

ชื่อ : โจโจ้
รหัสผ่าน : xxxx

ชื่อ : ก้อง
รหัสผ่าน : xxxx

คีย์เวิร์ด this

การใช้คีย์เวิร์ด this จะเป็นตัวชี้หรือตัวที่บ่งบอกว่าตอนนี้เราทำงานกับวัตถุใด ให้บอกตัวตนของวัตถุนั้นๆ เช่น การกำหนดคุณสมบัติต่างๆ ในวัตถุ เป็นต้น

การสร้างเมธอด (Method)

การสร้างเมธอด

JS

การสร้างเมธอดในคลาสจะคล้ายหลักการสร้างฟังก์ชั่น ใน JavaScript แต่มีลักษณะสำคัญ คือ

- ระบุแค่ชื่อเมธอด แต่ไม่มีคำว่า function นำหน้า
- สามารถกำหนดพารามิเตอร์ได้ เหมือนกับฟังก์ชั่น
- อ้างอิง Property หรือ Method เดียวกันภายใน Class โดยใช้คีย์เวิร์ด this

การเรียกใช้งาน

JS

เรียกใช้งานภายใน Class

- this.methodName()

เรียกใช้งานภายนอก Class

- obj_name.methodName()

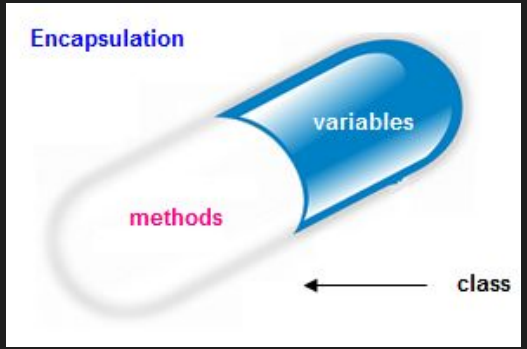
JS

การห่อหุ้ม (Encapsulation)

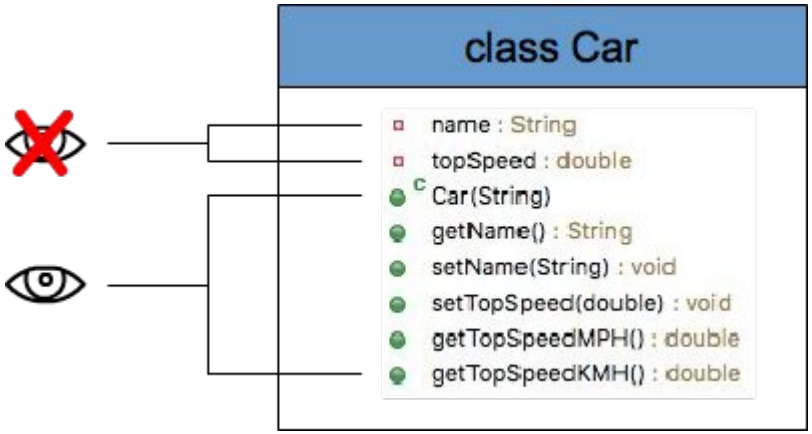
การห่อหุ้ม (Encapsulation)

- เป็นกระบวนการซ่อนรายละเอียดการทำงานและข้อมูลไว้ภายใน ไม่ให้ภายนอกสามารถมองเห็นและเปลี่ยนแปลงแก้ไขข้อมูลภายในได้ ซึ่งเป็นผลทำให้เกิดความเสียหายแก่ข้อมูล

- สามารถสร้างความปลอดภัยให้แก่ข้อมูลได้ เนื่องจากข้อมูลจะถูกเข้าถึงจากผู้มีสิทธิ์เท่านั้น



การห่อหุ้ม (Encapsulation)



Access Modifier

JS

Access Modifier

คือ ระดับในการเข้าถึง Class, Property, Method และอื่น ๆ ในภาษาเชิงวัตถุ มีประโยชน์อย่างมากในเรื่องของการกำหนดสิทธิในการเข้าใช้งาน การซ่อนข้อมูล และอื่น ๆ

Access Modifier

JS

- **Public** เป็นการประกาศระดับในรูปแบบสาธารณะหรือกล่าวได้ว่าใครๆ ก็สามารถเข้าถึงและเรียกใช้งานได้
- **Protected** เป็นการประกาศระดับการเข้าถึงที่เกี่ยวข้องกับเรื่องการสืบทอด (Inheritance) ทำให้คลาสนั้นๆ สามารถเรียกใช้งานสมาชิกของคลาสที่ถูกกำหนดเป็น Protected ได้ **(เครื่องหมาย _)**
- **Private** เป็นการประกาศระดับการเข้าถึงที่เข้มงวดที่สุด กล่าวคือ จะมีแต่คลาสของตัวมันเองเท่านั้นที่มีสิทธิ์ใช้งานได้ **(เครื่องหมาย #)**

Access Modifier

โครงสร้างคำสั่ง

```
class className {

    modifier property

}
```

Getter , Setter Method

Accessor (get , set)

Accessor

JS

คำสั่งที่ช่วยให้สามารถจัดการ Property ได้ง่ายมากยิ่งขึ้น โดยสามารถกำหนดได้ว่าต้องการอยากทำงานกับ Property ใด โดยมีองค์ประกอบ 2 ส่วน คือ

- Get คือ ตัวช่วยสำหรับเรียกดูข้อมูลใน Property
- Set คือ ตัวช่วยสำหรับกำหนดหรือเขียนข้อมูลใน Property

Static

Static Property

JS

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

โครงสร้างคำสั่ง

```
static property = value
```

การเรียกใช้งาน

```
className.property
```

Static Method

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

Static Method

JS

โครงสร้างคำสั่ง

```
static method_name (parameter){
    //statement
}
```

การเรียกใช้งาน

```
className.method()
```

JS

การสืบทอดคุณสมบัติ (Inheritance)

การสืบทอดคุณสมบัติ (Inheritance)

JS

คือ ทำการสร้างสิ่งใหม่ขึ้นด้วยการสืบทอด หรือรับเอา (inherit) คุณสมบัติบางอย่างมาจากสิ่งเดิมที่มีอยู่แล้ว

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

คลาสแม่ (Superclass) คลาสลูก (Subclass)

JS

SuperClass

User

SubClass

Teacher

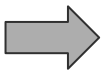
SubClass

Student

คุณสมบัติต่างๆจากแม่จะถูกถ่ายทอดไปยังลูก

SuperClass

User



SubClass

Teacher

SubClass

Student

*** ยกเว้น Private Property & Private Method

การสืบทอดคุณสมบัติ (Inheritance)

JS

คลาสแม่

```
class SuperClass{


// Property & Method
}
```

คลาสลูก

```
class SubClass extends SuperClass{


// Property & Method
}
```

ตัวอย่าง

คลาสแม่

```
class User{


// Property & Method
}
```

คลาสลูก

```
class Teacher extends User{


// Property & Method
}
```

ตัวอย่าง

คลาสแม่

```
class User{


// Property & Method
}
```

คลาสลูก

```
class Student extends User{


// Property & Method
}
```

Super

Super คืออะไร

เป็นคำสั่งสำหรับเรียกใช้งานเมื่อต้องการคุณสมบัติต่างๆ ที่ทำงานอยู่ในคลาสแม่ เช่น Constructor เป็นต้น

JS

User

Teacher

Student

JS

- ชื่อ
- รหัสผ่าน

User

Teacher

Student

JS

- ชื่อ
- รหัสผ่าน

User

- ชื่อ
- รหัสผ่าน
- วิชาที่สอน

Teacher

- ชื่อ
- รหัสผ่าน
- คะแนนสอบ

Student

JS

- ชื่อ
- รหัสผ่าน

User

- ชื่อ
- รหัสผ่าน
- วิชาที่สอน

Teacher

- ชื่อ
- รหัสผ่าน
- คะแนนสอบ

Student

JS

การพ้องรูป (Polymorphism)

การพ้องรูป (Polymorphism)

JS

การพ้องรูป คือ ความสามารถในการตอบสนองต่อสิ่งเดียวกันด้วยวิธีการที่แตกต่างกัน กล่าวคือวัตถุนั้นสามารถกำหนดกระบวนการทำงานได้หลายรูปแบบโดยเพิ่มเติมกระบวนการทำงานจากสิ่งเดิมที่มีอยู่แล้ว

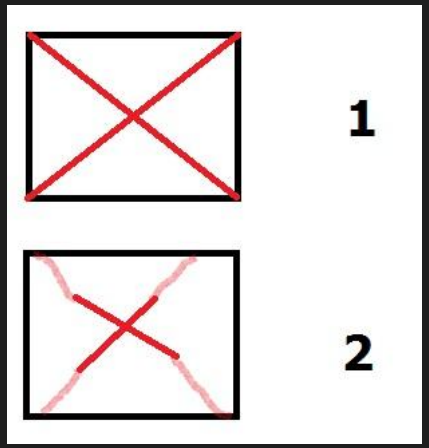
ข้อดีคือทำให้โปรแกรมสามารถปรับเปลี่ยนหรือเพิ่มเติมการทำงานได้ง่ายขึ้น

การพ้องรูป (Polymorphism)

JS

“ ข้อความเดียวกันแต่กระบวนการทำงานภายในแตกต่างกันนั้น เรียกว่า การพ้องรูป หรือ polymorphism ”

กา



ดำ

JS

JS

Overriding Method

Overriding Method

คือ เมธอดของคลาสลูก ที่มีชื่อเหมือนกับเมธอดของคลาสแม่ (เป็นผลมาจากคุณสมบัติ OO คือ inheritance) แต่มีกระบวนการทำงานด้านในแตกต่างกัน

Protected Access Modifier

Access Modifier

JS

- **Public** เป็นการประกาศระดับในรูปแบบสาธารณะหรือกล่าวได้ว่าใครๆ ก็สามารถเข้าถึงและเรียกใช้งานได้
- **Protected** เป็นการประกาศระดับการเข้าถึงที่เกี่ยวข้องกับเรื่องการสืบทอด (Inheritance) ทำให้คลาสนั้นๆ สามารถเรียกใช้งานสมาชิกของคลาสที่ถูกกำหนดเป็น Protected ได้ **(อ้างอิงด้วยเครื่องหมาย _)**
- **Private** เป็นการประกาศระดับการเข้าถึงที่เข้มงวดที่สุด กล่าวคือ จะมีแต่คลาสของตัวมันเองเท่านั้นที่มีสิทธิ์ใช้งานได้ **(#)**