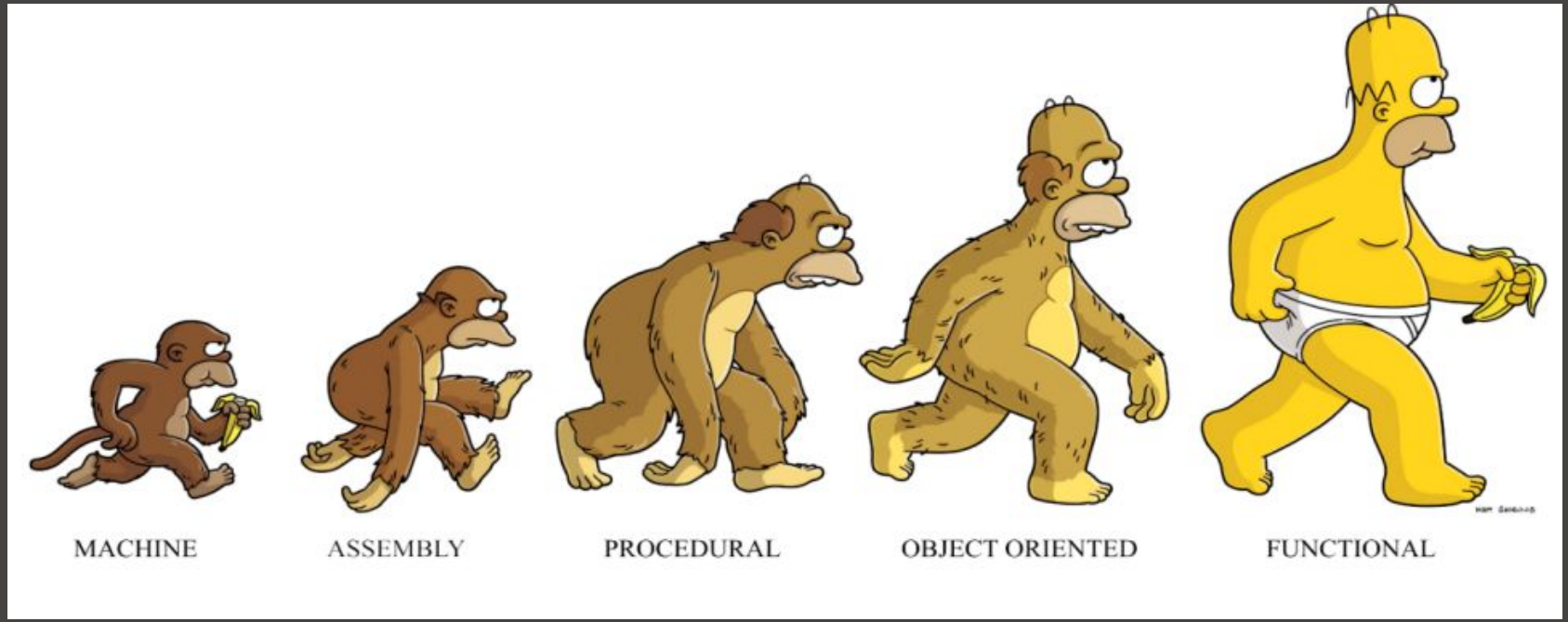
การเขียนโปรแกรมเชิงวัตถุ

Object Oriented Programming

เอกสารแจกฟรี ห้ามจำหน่าย!!!!

แนวความคิดภาษาเชิงวัตถุ

คล้ายๆจำลองชีวิตความเป็นอยู่ของมนุษย์ที่จะประกอบด้วย คน สัตว์ สิ่งของ และใช้สิ่งต่างๆ เพื่อแก้ปัญหา คือแต่ละสิ่งมีหน้าที่ในการแก้ปัญหา



https://www.youtube.com/channel/UCQ1r_4x-P-fETLIU4pqf98w https://www.facebook.com/KongRuksiamTutorial/

เชิงกระบวนการ VS เชิงวัตถุ

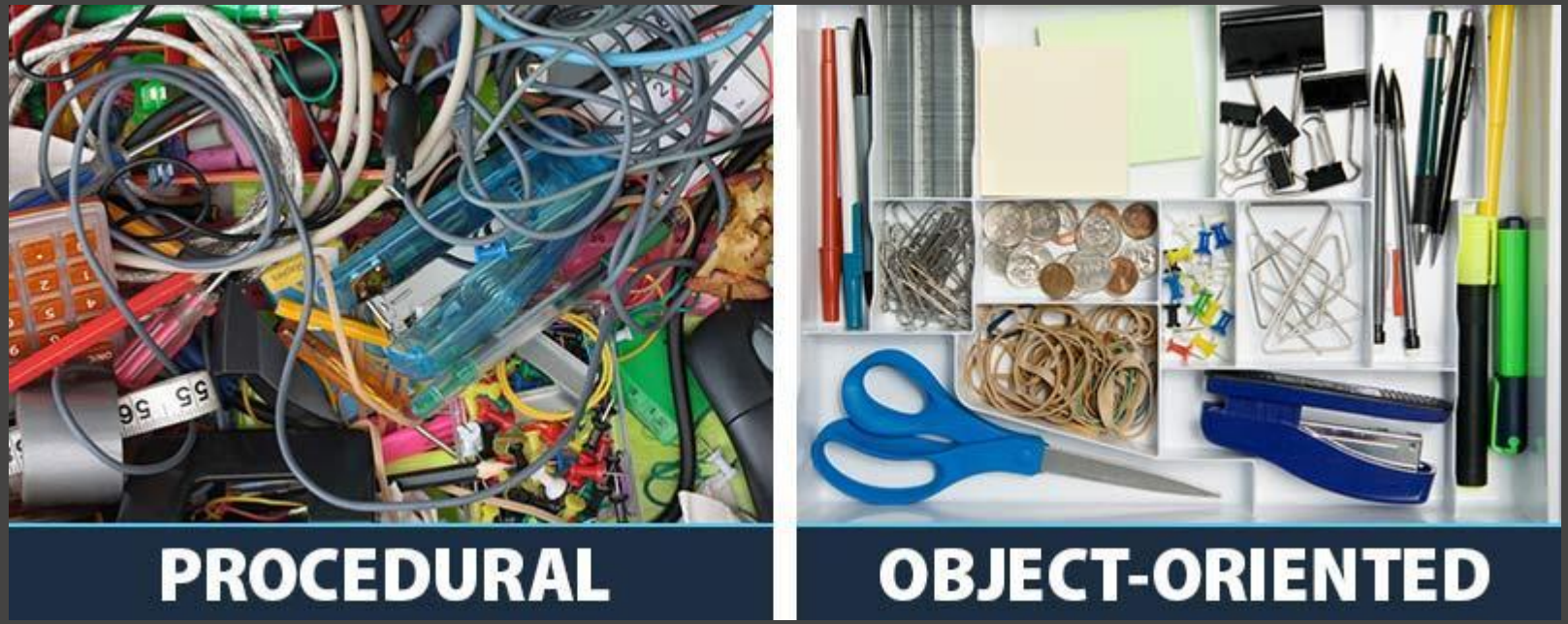
ภาษาเชิงกระบวนการ (Procedural Programming Language)

โปรแกรมจะแบ่งออกเป็นส่วนย่อยๆ เรียกว่าโมดูล (Module)
แต่ละโมดูลควรออกแบบให้มีการทำงานเพียง 1 งานเท่านั้น
การออกแบบให้แต่ละโมดูลมีความเป็นอิสระต่อกันนั้นทำได้ยาก

ภาษาเชิงวัตถุ (Object Oriented Programming Language)

การพัฒนาโปรแกรมเป็นการเลียนแบบการทำงานเชิงออปเจ็ค
การออกแบบให้วัตถุมีความเป็นอิสระต่อกันทำได้ง่ายด้วยคุณสมบัติเชิงวัตถุ
สามารถนำโปรแกรมกลับมาใช้ใหม่ (Reuse) ได้ดีกว่าภาษาเชิงกระบวนการ

เชิงกระบวนการ VS เชิงวัตถุ



https://www.youtube.com/channel/UCQ1r_4x-P-fETLIU4pqf98w https://www.facebook.com/KongRuksiamTutorial/

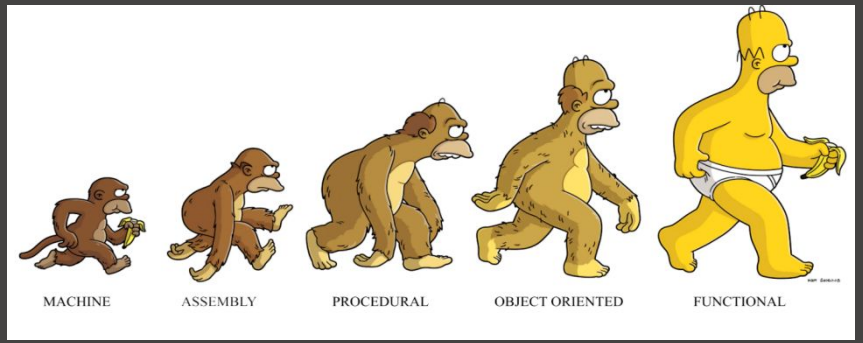
เชิงกระบวนการ VS เชิงวัตถุ

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

แนวความคิดภาษาเชิงวัตถุ

การพัฒนาโปรแกรมที่ใช้วัตถุเป็นเครื่องมือโดยที่วัตถุนั้นมีลักษณะพื้นฐานดังนี้

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



แนวความคิดภาษาเชิงวัตถุ

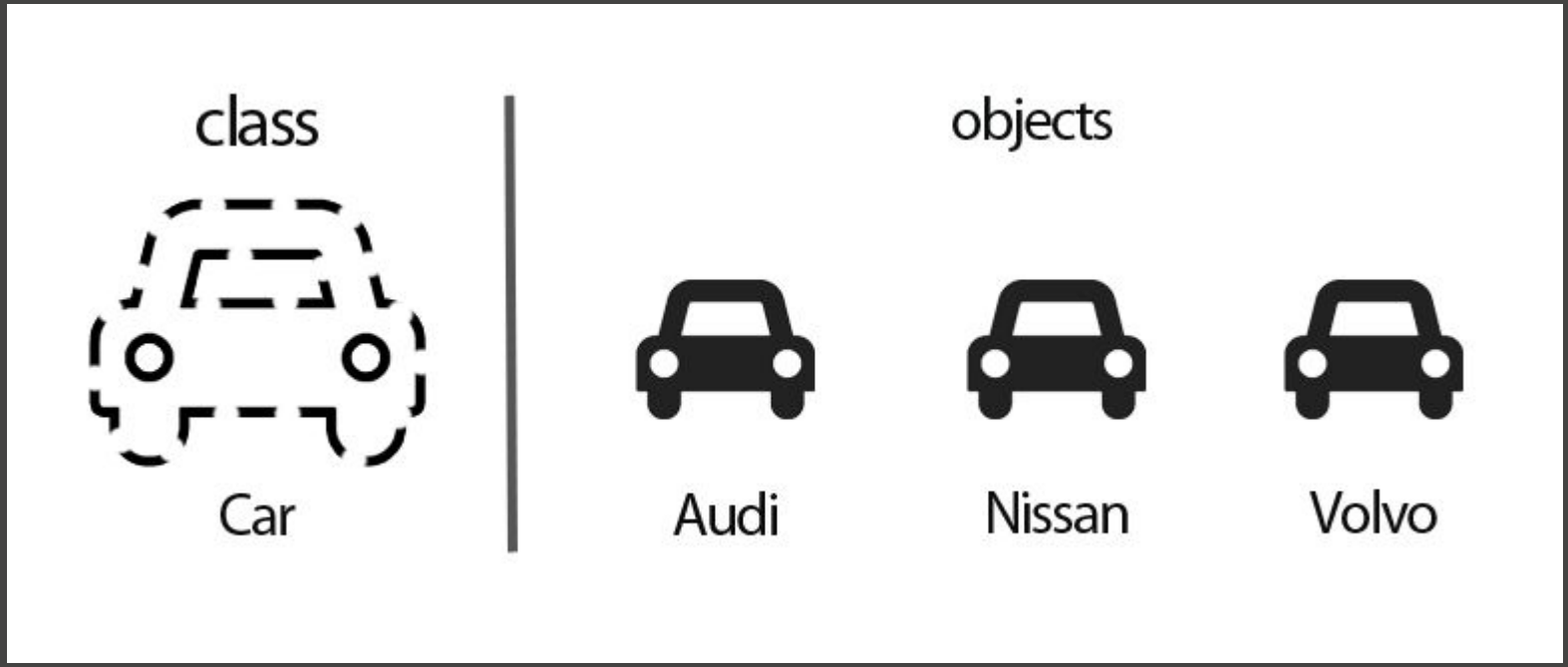
คลาส (class) คือต้นแบบของวัตถุ การจะสร้างวัตถุขึ้นมาอย่างหนึ่งจะต้องสร้างคลาสขึ้นมาเป็นโครงสร้างต้นแบบสำหรับวัตถุก่อนเสมอ

วัตถุหรือออบเจ็ค (object) คือสิ่งที่ประกอบไปด้วยคุณสมบัติ 2 ประการ คือ คุณลักษณะ และพฤติกรรม

คุณลักษณะ (attribute หรือ data member) คือ สิ่งที่บ่งบอกลักษณะทั่วไปของวัตถุ

พฤติกรรม (behavior หรือ method) คือ พฤติกรรมทั่วไปขอวัตถุที่สามารถกระทำได้

แนวความคิดภาษาเชิงวัตถุ



Pokemon
Name: Pikachu
Type: Electric
Health: 70
attack()
dodge()
evolve()

Fields

Methods

https://www.youtube.com/channel/UCQ1r_4x-P-fETLIU4pqf98w https://www.facebook.com/KongRuksiamTutorial/






https://www.youtube.com/channel/UCQ1r_4x-P-fETLIU4pqf98w https://www.facebook.com/KongRuksiamTutorial/

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

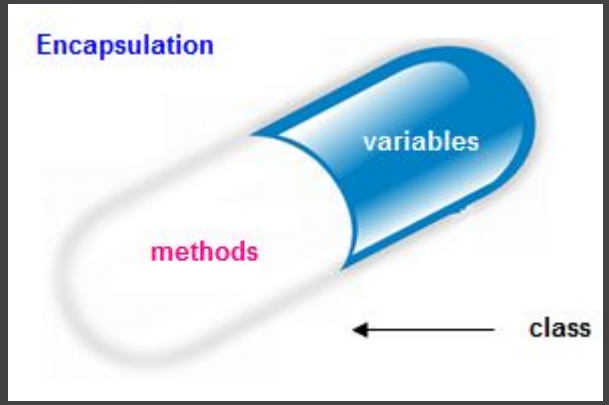
Public เป็นการประกาศระดับการเข้าถึงที่เข้มงวดน้อยที่สุด หรือกล่าวได้ว่าใคร ๆ ก็สามารถเข้าถึงและเรียกใช้งานได้

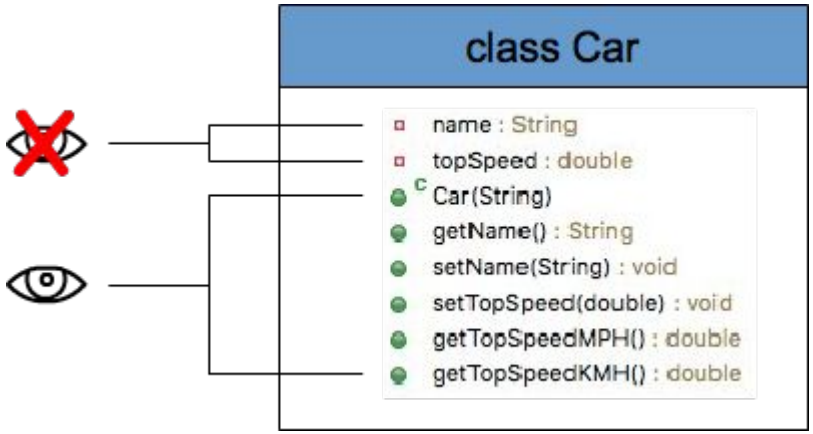
Protected เป็นการประกาศระดับการเข้าถึงที่เกี่ยวข้องกับเรื่องการสืบทอด (Inheritance) คลาสที่อยู่ในแพ็กเกจเดียวกันกับคลาสที่ถูกกำหนด modifier เป็น protected จะสามารถเรียกใช้งานสมาชิกของคลาสที่ถูกกำหนดเป็น protected ได้

Private เป็นการประกาศระดับการเข้าถึงที่เข้มงวดที่สุด กล่าวคือ จะมีแต่คลาสของตัวมันเองเท่านั้นที่มีสิทธิใช้งานได้

การห่อหุ้ม (Encapsulation)

- เป็นกระบวนการซ่อนรายละเอียดการทำงานและข้อมูลไว้ภายในไม่ให้ภายนอกสามารถมองเห็นได้
- ทำให้ภายนอกไม่สามารถทำการเปลี่ยนแปลงแก้ไขข้อมูลภายในได้ ซึ่งเป็นผลทำให้เกิดความเสียหายแก่ข้อมูล
- ข้อดีของการห่อหุ้มคือสามารถสร้างความปลอดภัยให้แก่ข้อมูลได้ เนื่องจากข้อมูลจะถูกเข้าถึงจากผู้มีสิทธิ์เท่านั้น

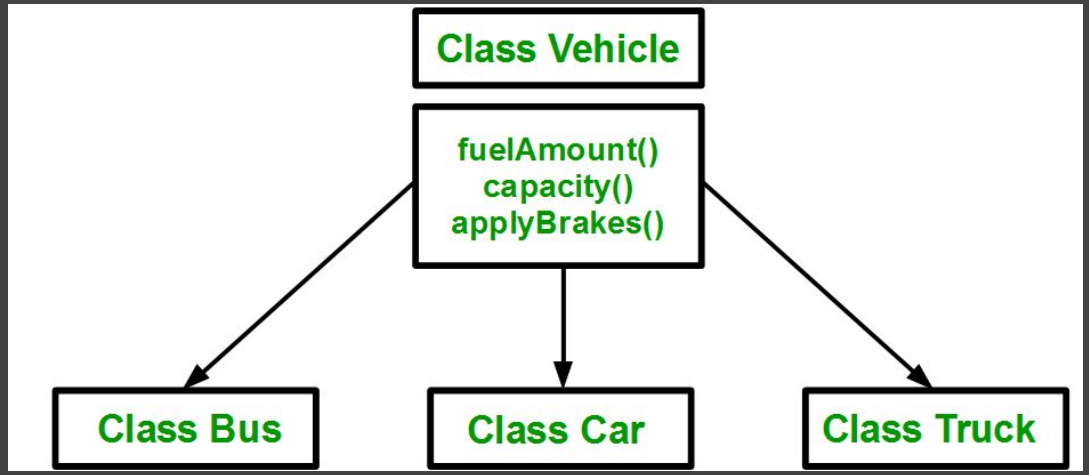




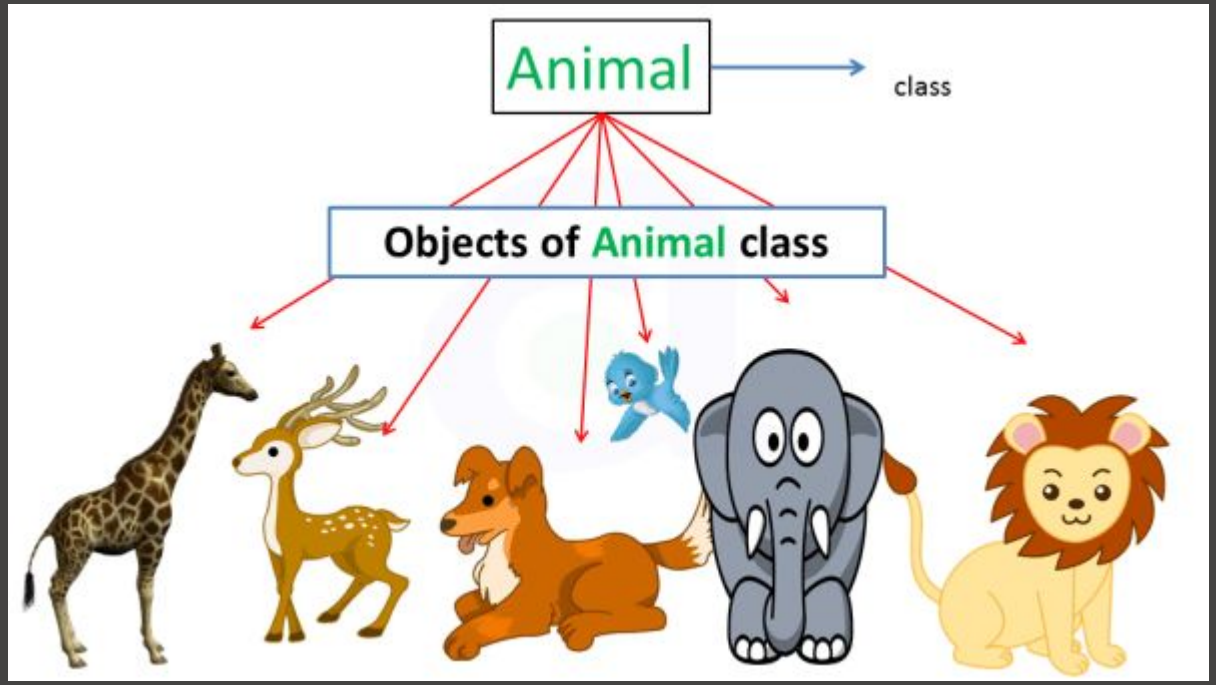
https://www.youtube.com/channel/UCQ1r_4x-P-fETLIU4pqf98w https://www.facebook.com/KongRuksiamTutorial/

การสืบทอดคุณสมบัติ (Inheritance)

หลักการของ inheritance คือ ทำการสร้างสิ่งใหม่ขึ้นด้วยการสืบทอด หรือรับเอา (inherit) คุณสมบัติบางอย่างมาจากสิ่งเดิมที่มีอยู่แล้ว โดยการสร้างเพิ่มเติมจากสิ่งที่มีอยู่แล้วได้เลย ข้อดีของการ inheritance คือ จากการที่สามารถนำสิ่งที่เคยสร้างขึ้นแล้วนำกลับมาใช้ใหม่ (re-use) ได้ ทำให้ช่วยประหยัดเวลาการทำงานลง เนื่องจากไม่ต้องเสียเวลาพัฒนาใหม่หมด

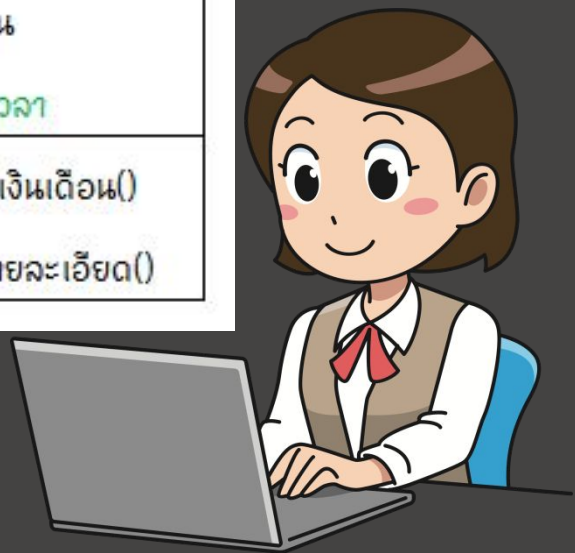


คลาสแม่ (Superclass) คลาสลูก (Subclass)



Employee

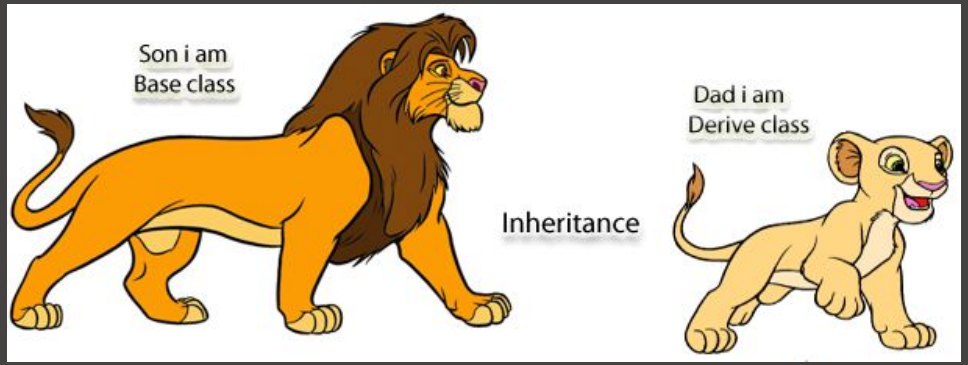
ผู้จัดการ	พนักงานขาย	พนักงานฝ่ายผลิต
- รหัสพนักงาน - ชื่อ - เงินเดือน - ที่จอดรถ	- รหัสพนักงาน - ชื่อ - เงินเดือน - ค่าคอมมิชั่น	- รหัสพนักงาน - ชื่อ - เงินเดือน - ค่าล่วงเวลา
+ คำนวณเงินเดือน() + แสดงรายละเอียด()	+ คำนวณเงินเดือน() + แสดงรายละเอียด()	+ คำนวณเงินเดือน() + แสดงรายละเอียด()



คีย์เวิร์ด THIS & SUPER

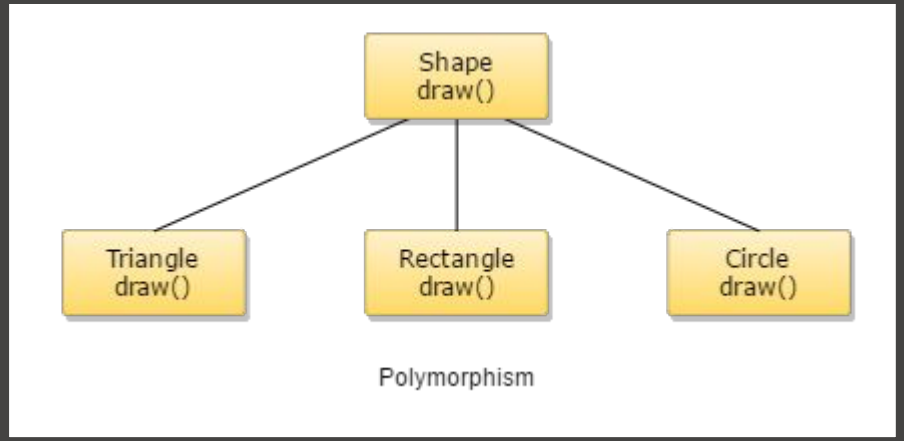
คีย์เวิร์ด **this** การใช้คีย์เวิร์ด **this** คือ เมื่อต้องการเรียกใช้งานคอนสตรัคเตอร์อื่นๆ ที่อยู่ภายในคลาสเดียวกัน

คีย์เวิร์ด **super** เมื่อต้องการเรียกคอนสตรัคเตอร์ของคลาสแม่ให้ทำงาน คีย์เวิร์ด **super** ในการเรียกใช้งานคอนสตรัคเตอร์ของคลาสแม่**จะต้องทำการเรียกที่บรรทัดแรกสุดของคอนสตรัคเตอร์นั้นๆ เท่านั้น**



การพ้องรูป (POLYMORPHISM)

Polymorphism เกิดจาก **poly** (หลากหลาย) + **morphology** (รูปแบบ)



ในทางโปรแกรมคือการที่เมธอดชื่อเดียวกัน สามารถรับอาร์กิวเมนต์ที่แตกต่างกันได้หลายรูปแบบ โดยเมธอดนี้จะถูกเรียกว่า overload method (เมธอดถูกโอเวอร์โหลด)

Overloading & Overriding Method

Overloading method คือ เมธอดที่มีชื่อเหมือนกัน และอยู่ภายในคลาสเดียวกัน สิ่งที่แยกความแตกต่างของเมธอดที่เป็น overload method คือ **พารามิเตอร์** (เป็นผลมาจากคุณสมบัติ OO คือ **polymorphism**)

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

Class & Object

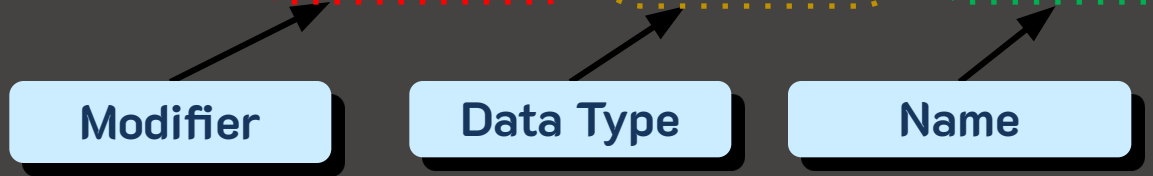
คลาส (class) คือ ต้นแบบของวัตถุ หรือ เป็นแม่แบบสำหรับวัตถุ (Template, Prototype)

วัตถุ (Object) คือ สิ่งที่ถูกสร้างขึ้นจากคลาส

https://www.youtube.com/channel/UCQ1r_4x-P-fETLIU4pqf98w https://www.facebook.com/KongRuksiamTutorial/

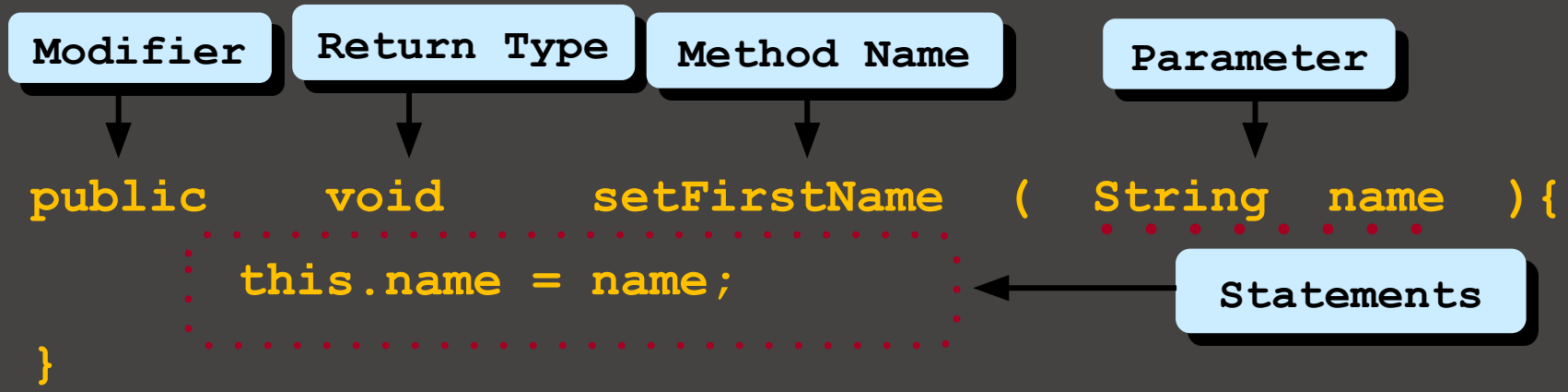
ตัวอย่างการสร้าง Class และ Properties

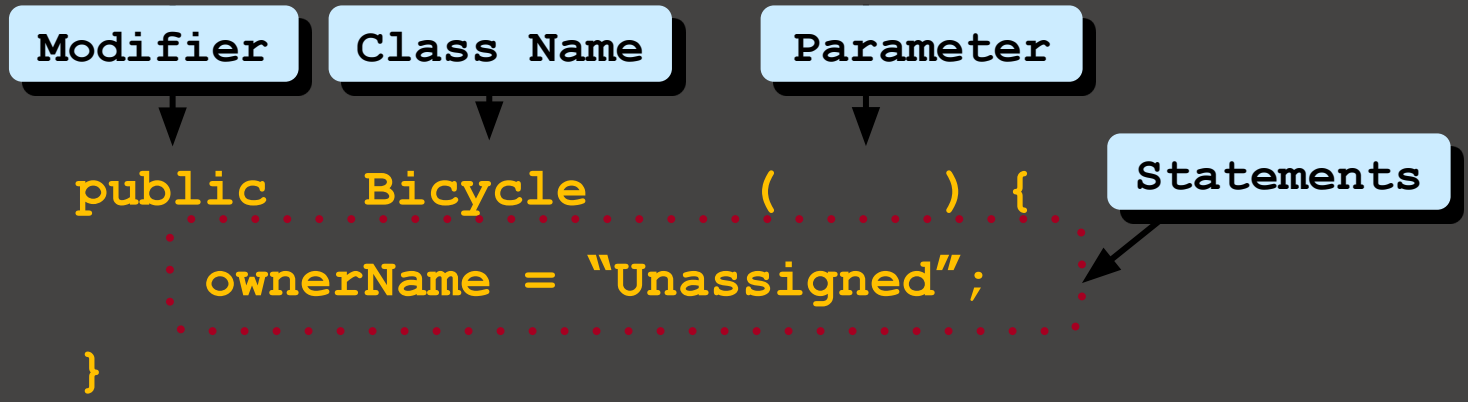
```
public class Employee{
    private String id;
    private String name;
    private double salary;
}
```



Method เป็นกลไกที่กำหนดพฤติกรรมให้กับคลาส

Method เป็นกลไกที่กำหนดพฤติกรรมให้กับคลาส



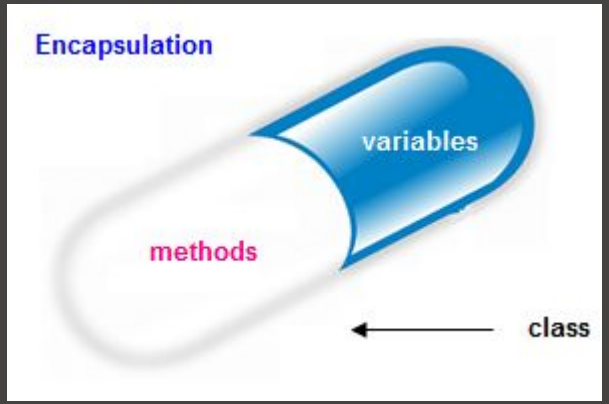


Constructor เป็น Method พิเศษที่จะถูกสั่งให้ทำงานโดยอัตโนมัติ เมื่อมีการสร้างวัตถุขึ้น เพื่อกำหนดค่าเริ่มต้นให้กับตัวแปรคลาสของวัตถุ

- **มีชื่อเหมือนกับชื่อคลาส**
- Method มีการคืนค่ากลับ แต่ Constructor **ไม่มีการคืนค่ากลับ**
- ประกาศการเข้าถึงเป็นแบบ **public**

Encapsulation

การห่อหุ้ม คือ กระบวนการซ่อนรายละเอียดการทำงานและข้อมูลไว้ภายในไม่ให้ภายนอกสามารถมองเห็นได้ เพื่อไม่ให้ภายนอกสามารถทำการเปลี่ยนแปลง แก้ไข หรือสร้างความเสียหายให้กับสิ่งต่างๆ ที่อยู่ภายในได้



การนำ Access Modifier มาใช้งาน

Access Modifier	ใช้ได้ทั้งหมด	package เดียวกัน	ต่าง package กัน	ต่าง package กัน แต่เป็นคลาสแม่ คลาสลูกกัน	คลาสเดียวกัน
public	✓	✓	✓	✓	✓
protected	☐	✓	☐	✓	✓
package	☐	✓	☐	☐	✓
private	☐	☐	☐	☐	✓

Non-access modifier: static

ใช้กำหนดหน้า Method หรือตัวแปรเพื่อให้เป็นแบบ static

การกำหนดให้ method เป็น static เรียกว่า static method จะทำให้เราสามารถเรียกใช้งาน method นั้นโดยไม่ต้องสร้างออปเจ็ค

การกำหนดให้ข้อมูลเป็น static เรียกว่า static attribute จะทำให้เราสามารถเรียกใช้งานโดยไม่ต้องสร้างออปเจ็คขึ้นมาก่อน

Inheritance

การนิยามคลาสใหม่จากรูปแบบของคลาสที่มีอยู่แล้ว โดยคลาสใหม่จะนำ attribute และ method ของคลาสเดิมมาใช้ได้

- สามารถนำโปรแกรมเดิมมาพัฒนาเพิ่มเติมใหม่ได้ง่ายขึ้น (re-use) ประหยัดเวลา
- ทำให้โปรแกรมแต่ละโปรแกรมมีขนาดเล็ก ซึ่งทำให้ง่ายต่อการเข้าใจ และปรับปรุงแก้ไข

- ส่งผลให้เกิด **Overriding method** คือ เมธอดของคลาสลูก (subclass) ที่มีชื่อเหมือนกับเมธอดของคลาสแม่ (superclass)

Employee -> Programmer

modifier: final

final เป็นคีย์เวิร์ดหนึ่งในภาษาจาวา สามารถที่จะกำหนดคีย์เวิร์ด final นี้ให้กับ class , method , attribute

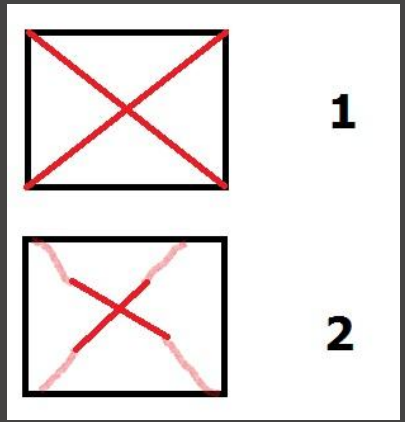
final ให้คลาส จะทำให้คลาสนั้นไม่สามารถมี subclass ได้

final ให้เมธอด จะทำให้เมธอดนั้นไม่สามารถ override method นั้นได้
final ให้ data หรือ attribute จะทำให้เป็นค่าคงที่ (constant)

Polymorphism

“ ข้อความเดียวกันแต่กระบวนการทำงานภายในแตกต่างกัน นั้น เรียกว่า **การพ้องรูป** หรือ **polymorphism** ”

กา



คุณสมบัติการพ้องรูป คือ คุณสมบัติที่สามารถตอบสนองต่อ Method เดียวกันด้วยวิธีการที่ต่างกันและสามารถกำหนด object ได้หลายรูปแบบ

ข้อดี คือ ทำให้โปรแกรมสามารถปรับเปลี่ยนหรือเพิ่มเติมได้ง่ายขึ้น

การโอเวอร์โหลดเมธอด (Method overloading) เป็นการนิยามคลาสที่ประกอบด้วยเมธอดหลายเมธอดที่มีชื่อเหมือนกัน**จะต้องมีหน้าที่เหมือนกัน แต่ต่างกันในส่วนของรายการพารามิเตอร์**

การโอเวอร์ไรดิ้งเมธอด (Method Overriding) เป็นการกำหนดการทำงานให้กับ**เมธอดที่สืบทอด** มาจาก**การถ่ายทอดคุณสมบัติ (Inheritance)**

Abstract Class

abstract เป็นคีย์เวิร์ดในภาษาจาวา สามารถกำหนดคีย์เวิร์ด abstract นี้ให้กับ คลาส หรือเมธอด ก็ได้

abstract method คือ เมธอดว่างเปล่าที่ยังไม่ได้มีการกำหนดรายละเอียดการทำงานลงไป จะถูกกำหนดรายละเอียดลงไปภายหลัง โดยคลาสลูกที่ได้รับการสืบทอดจากคลาสของ abstract method เหล่านั้น

Abstract Method

abstract class หากคลาสใดก็แล้วแต่ที่ประกอบไปด้วยเมธอดที่เป็น abstract method เพียงเมธอดเดียว จะต้องประกาศคลาสนั้นเป็น abstract ด้วย

กฎของ abstract หากคลาสใดสืบทอดมาจาก abstract class คลาสนั้นจะต้องทำการระบุเมธอดทุกเมธอดที่เป็น abstract method ใน abstract class ไว้เสมอ (ไม่กำหนดรายละเอียดก็ได้แต่จะต้องมีการเขียน abstract method ทุกเมธอดลงไปในคลาสนั้นด้วย)

อินเทอร์เฟซ (Interface)

interface มีหลักการคล้ายกับ abstract class คือ สร้างอินเทอร์เฟซขึ้นมาก็เพื่อกำหนดโครงสร้างของเมธอดที่จำเป็นใช้งานขึ้นมาแต่ยังไม่ได้กำหนดรายละเอียดการทำงานใดๆ ลงไปให้กับเมธอดนั้น (abstract method) เมธอดในอินเทอร์เฟซจึงเป็นเมธอดที่ว่างเปล่า ซึ่งในภายหลังจึงมีการกำหนดรายละเอียดของเมธอดเหล่านั้นลงไป โดยถูกกำหนดโดยคลาสที่เรียกใช้อินเทอร์เฟซนั้นๆ

Interface กับ Abstract class ต่างกันอย่างไร

เมธอดใน abstract class ไม่เป็น abstract method ก็ได้ แต่เมธอดทุกเมธอดใน interface เป็น abstract method

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

แต่คลาสที่จะเรียกใช้งานเมธอดในอินเทอร์เฟซไม่จำเป็นต้องมีความสัมพันธ์ใดๆ กับอินเทอร์เฟซทั้งสิ้น

เรียนเขียนโปรแกรมเชิงวัตถุด้วยภาษา Java
แบบวิดีโอสอน จำนวน 15 ตอน ฟรี!!! ได้ที่

http://bit.ly/2WvUTJU