



Object-Oriented Programming

การเขียนโปรแกรมเชิงวัตถุ

เนื้อหารายวิชา

■ คำอธิบายรายวิชา :

แนวคิดพื้นฐานเกี่ยวกับการออกแบบ และพัฒนาโปรแกรมเชิงวัตถุ โครงสร้างของภาษาเชิงวัตถุ เช่น คลาส ออบเจ็กต์ เมธอด อินเทอร์เฟส แพคเกจ การสืบทอดคลาส เอนแคปซูเลชัน และพอลิมอร์ฟิซึม เอพีไอของ ภาษาที่ใช้ในการเขียนโปรแกรมเชิงวัตถุ

วัตถุประสงค์ :

1. เพื่อให้ผู้ศึกษาเข้าใจและศึกษาแนวคิดการ จำลอง สรรพลึงด้วยเทคนิคเชิงวัตถุ
2. เพื่อให้ผู้ศึกษาเข้าใจความหมายของวัตถุและวิธีเขียน โปรแกรมเชิงวัตถุ
หลักการสำคัญของการเขียน โปรแกรมเชิงวัตถุ
3. เพื่อให้ผู้ศึกษาสามารถเขียน โปรแกรมภาษาจาวา โดยใช้คำสั่งพื้นฐานต่าง ๆ ได้

วิธีการวัดผลประเมินผล :

วิธีการวัดผลประเมินผล :

● คะแนนระหว่างภาค 70 % มีรายละเอียดดังนี้		
คะแนนความสนใจในการเรียน	10	%
งานที่มอบหมาย	30	%
คะแนนสอบระหว่างเรียน	30	%
● คะแนนสอบปลายภาค	30	%
รวม	100	%

เกณฑ์การประเมินผล

การประเมินผล :

อิงกลุ่ม (T-Score)

*

หัวข้อ (Topic)

1.1 แนวคิดเกี่ยวกับการโปรแกรมเชิงวัตถุ (OOP Concept)

1.2 ภาษาการเขียนโปรแกรมเชิงวัตถุ (OOP Language)

วัตถุประสงค์การเรียนรู้ (Learning Objective)

- 1. อธิบายหลักการเขียนโปรแกรมเชิงวัตถุได้
- 2. เปรียบเทียบภาษาในการเขียนโปรแกรมเชิงวัตถุกับภาษาเขียนโปรแกรมแบบโครงสร้างได้

งานส่งท้ายชั่วโมง ใบงานที่ 1

ให้ความหมายพร้อมทั้งยกตัวอย่างของคำศัพท์ต่อไปนี้

1. Polymorphism
2. Inheritance
3. Encapsulation
4. Reusable
5. OOP
6. Object
7. Class
8. Attribute และ Method
9. Overload
10. Super class
11. Sub class
12. Instance
13. Information Hiding

1.1 แนวคิดเกี่ยวกับการโปรแกรมเชิงวัตถุ (OOP Concept)

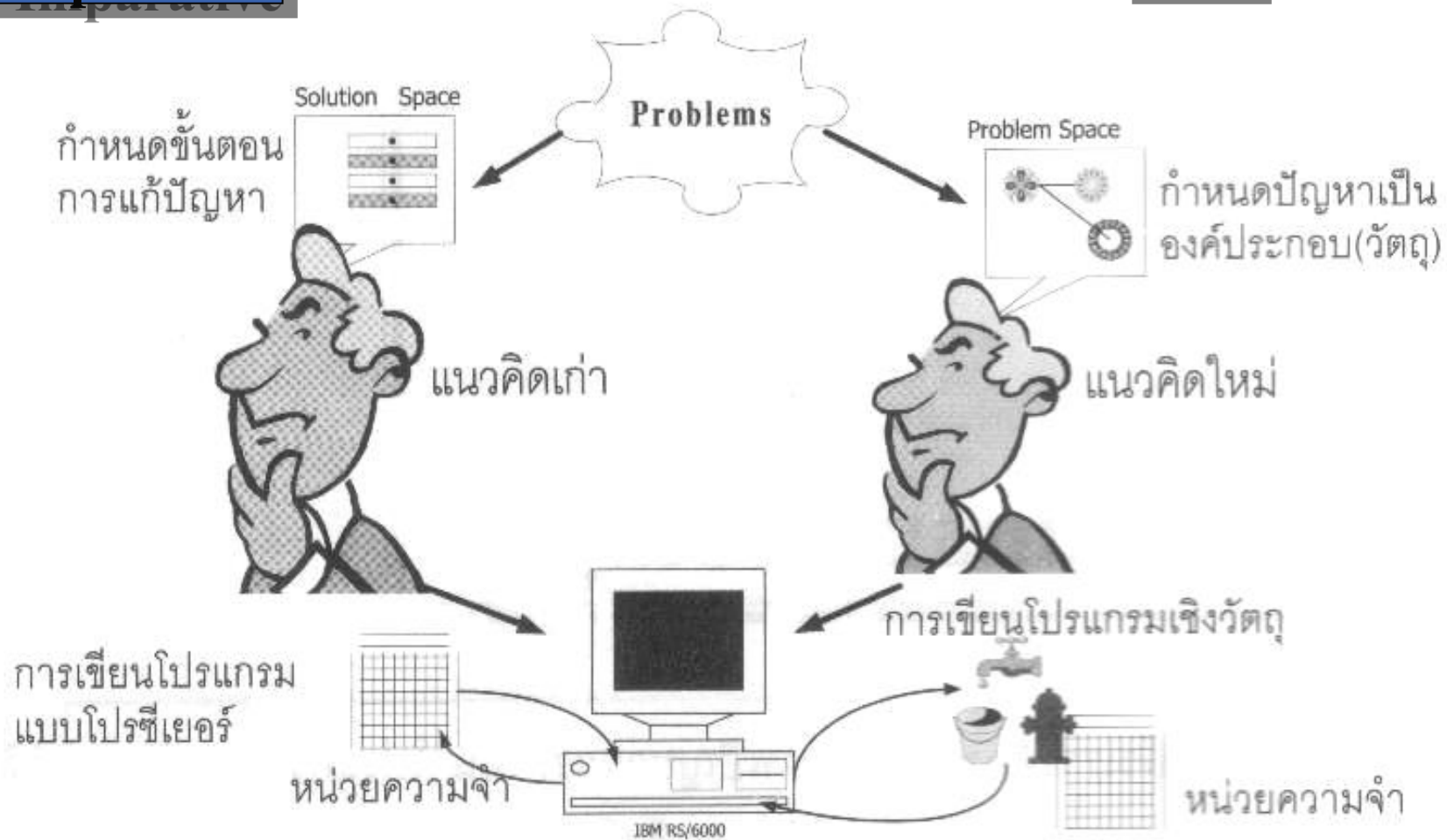
เพราะเหตุใดในปัจจุบันการเขียนโปรแกรมเชิงวัตถุ จึงเข้ามาแทนที่รูปแบบการเขียนโปรแกรมแบบเดิม ๆ การเขียนโปรแกรมในรูปแบบเดิมมีจุดบกพร่องอย่างไร และแบบใหม่สามารถแก้ไขจุดบกพร่องเหล่านั้นได้หรือไม่ เรามาดูความแตกต่างระหว่างการเขียนโปรแกรมภาษารูปแบบเดิมกับรูปแบบใหม่กัน

ความแตกต่างระหว่าง

การเขียนโปรแกรมเชิงวัตถุ

และ

การเขียนโปรแกรมแบบเดิม



แนวคิดเก่าและใหม่ของการเขียนคำสั่งคอมพิวเตอร์

การเขียนโปรแกรมแบบ Procedural

ภายในโครงสร้างของโปรแกรมจะแบ่งออกเป็น 2 ส่วนได้แก่

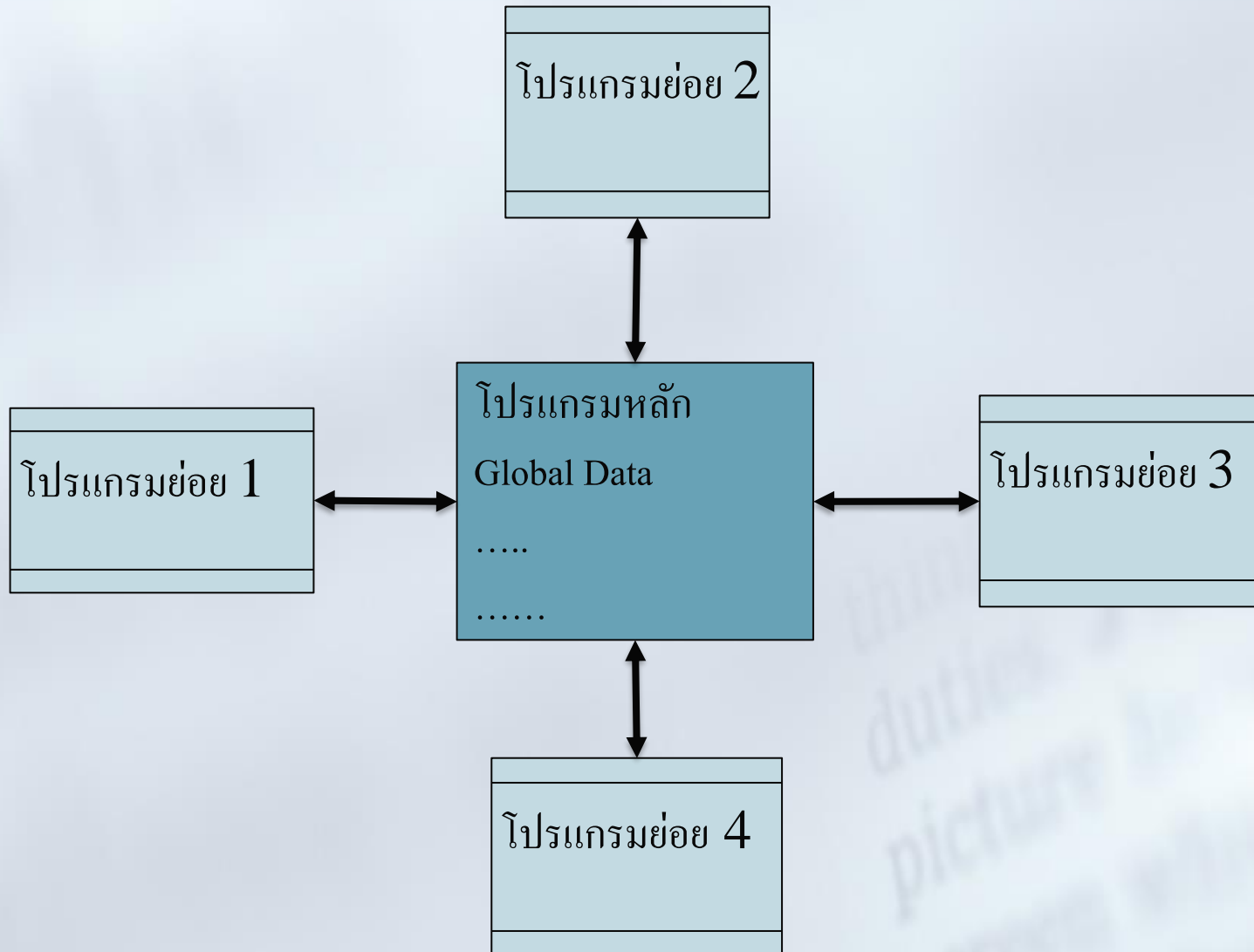
- ส่วนโปรแกรมหลัก (Main Program)

ที่มี Data เป็นส่วนประกอบ

- ส่วนโปรแกรมน้อย (Procedure หรือ Function)

โดย Data ที่ประกาศใช้อยู่ในภายในโปรแกรมหลักนั้น จะถูกเรียก
โดยโปรแกรมน้อยต่าง ๆ ที่อยู่ภายในโปรแกรม

แสดงโครงสร้างการเขียนโปรแกรมแบบ Procedural



การเขียนโปรแกรมแบบ Procedural

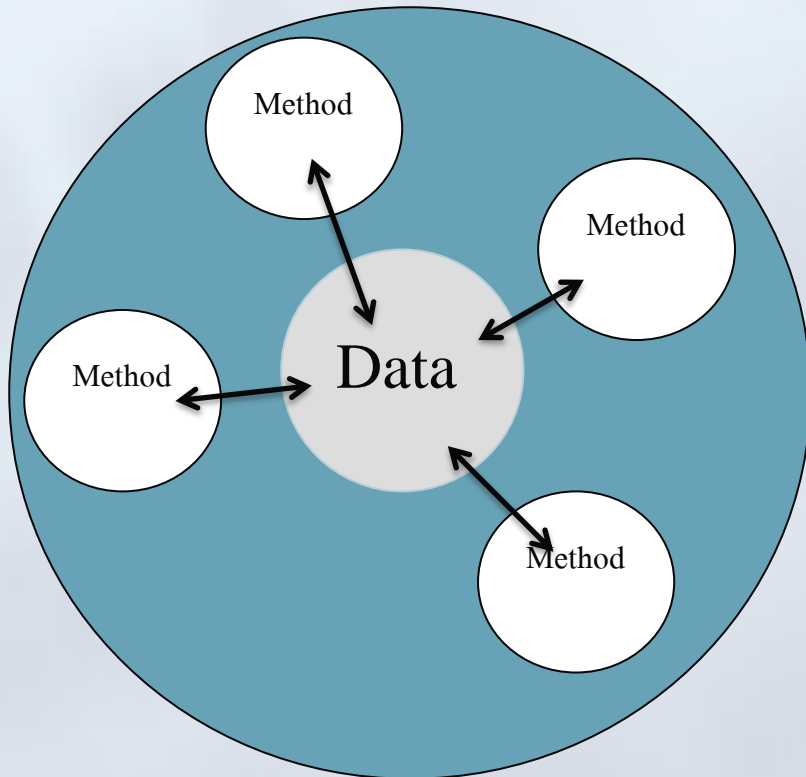
- ❑ Data ที่มีการประกาศใช้ทั่วทั้งโปรแกรมนั้นเรียกว่า การประกาศใช้แบบ “Global”
 - ❑ จะเห็นว่าโปรแกรมย่อยต่าง ก็เรียกใช้ข้อมูลจากโปรแกรมหลักเดียวกัน
- ดังนั้น อาจจะทำให้เกิดปัญหาจากการเปลี่ยนแปลงค่าของข้อมูลตัวเดียวกันได้ ส่งผลเสียต่อการควบคุมการเปลี่ยนแปลง Data ของโปรแกรมซึ่งยากต่อการแก้ไขโปรแกรมภายหลัง

การเขียนโปรแกรม OOP

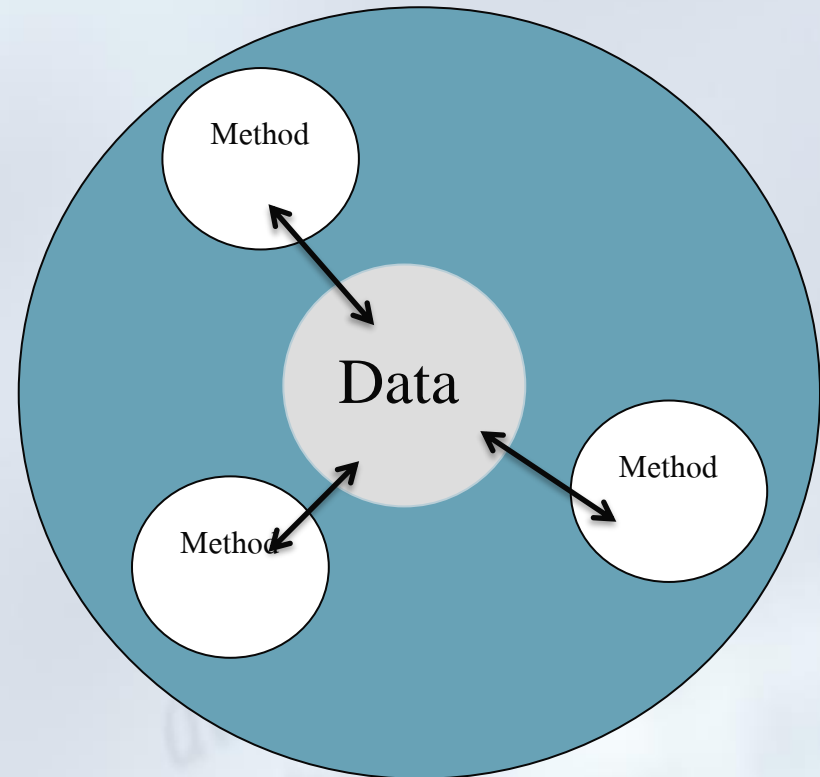
แต่สำหรับแนวทางการเขียนโปรแกรมแบบ OOP แล้ว Data จะถูกประกาศใช้เฉพาะภายในแต่ละ Object เท่านั้น(ไม่ประกาศเป็น Global)

การเขียนโปรแกรม OOP

Object 1



Object 2

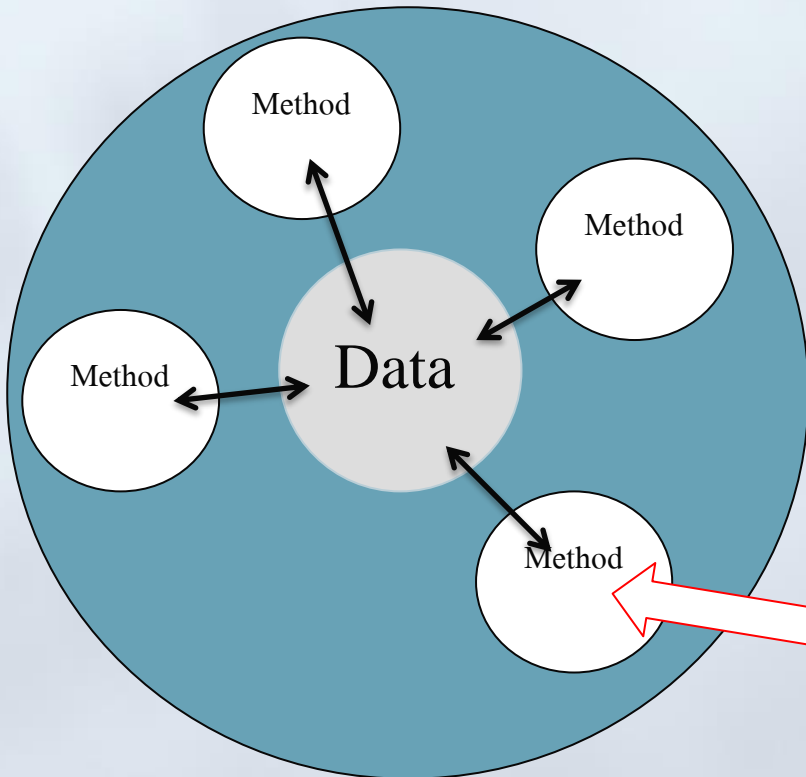


การเขียนโปรแกรม OOP

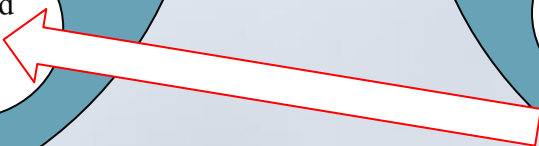
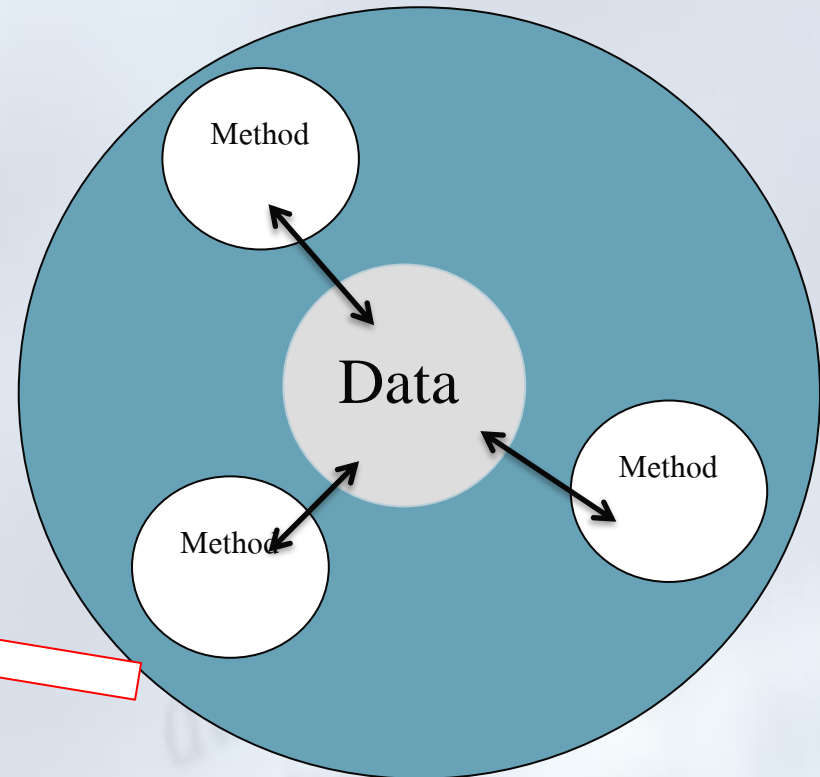
- การที่ OOP จัด Data ไว้ในแต่ละ Object นั้น นับว่าเป็นการปกป้องข้อมูลภายใน Object ซึ่งลดปัญหาการเปลี่ยนแปลงข้อมูลภายใน Object โดยไม่ได้รับอนุญาต (*Data และ Method จะถูกครอบหรือห่อหุ้มเอาไว้*)
- Object หนึ่งจะสามารถเข้าถึง Data ของ Object อื่นได้ก็ต่อเมื่อเรียกใช้ Method ของ Object ที่เป็นเจ้าของ Data เท่านั้น จึงส่งผลให้การแก้ไขโปรแกรมในภายหลังสะดวกยิ่งขึ้น อีกทั้งยังสามารถนำไปใช้ในโปรแกรมใหม่ได้ด้วย (Reusability)

การเขียนโปรแกรม OOP

Object 1



Object 2



ภาษา OOP

การเขียนโปรแกรม OOP มีหลายภาษาเช่น C++ ,C# ,Java เป็นต้น

การเขียนโปรแกรมเชิงวัตถุคืออะไร

- โปรแกรมที่จะรองรับงานของเราได้นั้นมักจะซับซ้อนเกินกว่าการเขียนโปรแกรมแบบ Structure Programming จะทำได้ผลดีพอแนวคิดของการเขียนโปรแกรมเชิงวัตถุจึงถูกคิดขึ้นมารองรับงานที่ซับซ้อน
- เทียบการเขียนโปรแกรมแบบ OOP กับ **รถยนต์** อย่างแรกโปรแกรมแบบ OOP แยกเป็นชิ้นส่วนที่มีมาตรฐานเหมือนรถยนต์ ซึ่งเราเรียกชิ้นส่วนนั้นว่า วัตถุ หรือ Object
- พอมองเป็น Object ก็ง่าย เราจะถอดเข้าถอดเปลี่ยนด้วยชิ้นส่วนใดก็ได้ ขอให้มันมีมาตรฐานเดียวกัน (เหมือนที่เรามองว่า เราถอดเปลี่ยนชิ้นส่วนรถยนต์ได้สะดวกนั่นเอง) ฉะนั้นมันก็ง่ายที่เราจะยอมให้ใครหลายๆ คนมาช่วยสร้างรถยนต์ หรือปรับแต่งรถยนต์

ความหมายของ Object

- ความหมายของ Object แบ่งได้เป็น 2 ประเภทคือ สิ่งที่เป็นรูปธรรมและนามธรรม ที่มีอยู่จริงบนพื้นโลก (real-world)
- สิ่งที่มีลักษณะเป็นรูปธรรม (จับต้องได้) เช่น จักรยาน รถ สุนัข องค์กร ใบรายการสินค้า เป็นต้น
- สิ่งที่มีลักษณะเป็นนามธรรม (จับต้องไม่ได้) เช่น ความเป็นเจ้าของ เทียบบิน การวิ่ง แสง เป็นต้น

องค์ประกอบพื้นฐานของ OOP

1. **Object** : ออบเจ็กต์
2. **Class** : คลาส

ทำความรู้จักกับ Class และ Object

ก่อนที่จะสร้าง**ออบเจกต์**ขึ้นมาได้เราต้องสร้าง**คลาส**ขึ้นมาก่อน คลาสก็เปรียบเสมือน**แม่แบบ** หรือ**พิมพ์เขียว** ในการสร้างออบเจกต์ต่างๆ ขึ้นมา Object คือสิ่งที่ประกอบด้วยคุณสมบัติ 2 ประการคือ

- ❑ คุณลักษณะ (**Attribute** หรือ Data) คือส่วนที่บ่งบอกลักษณะทั่วไปของวัตถุ
- ❑ พฤติกรรม (**Behavior** หรือ **Method**) คือสิ่งที่วัตถุสามารถกระทำออกมาได้

ทำความรู้จักกับ Class และ Object

- ❑ คุณลักษณะ (**Attribute** หรือ Data) คือส่วนที่บ่งบอกลักษณะทั่วไปของวัตถุ
- ❑ พฤติกรรม (Behavior หรือ **Method**) คือสิ่งที่วัตถุสามารถกระทำออกมาได้

ตัวอย่าง



Class พนักงานบริษัท	
Attribute	รหัสพนักงาน, เงินเดือน, เวลาเข้า-ออกงาน
Method	รูดบัตรพนักงาน, รับเงินเดือน

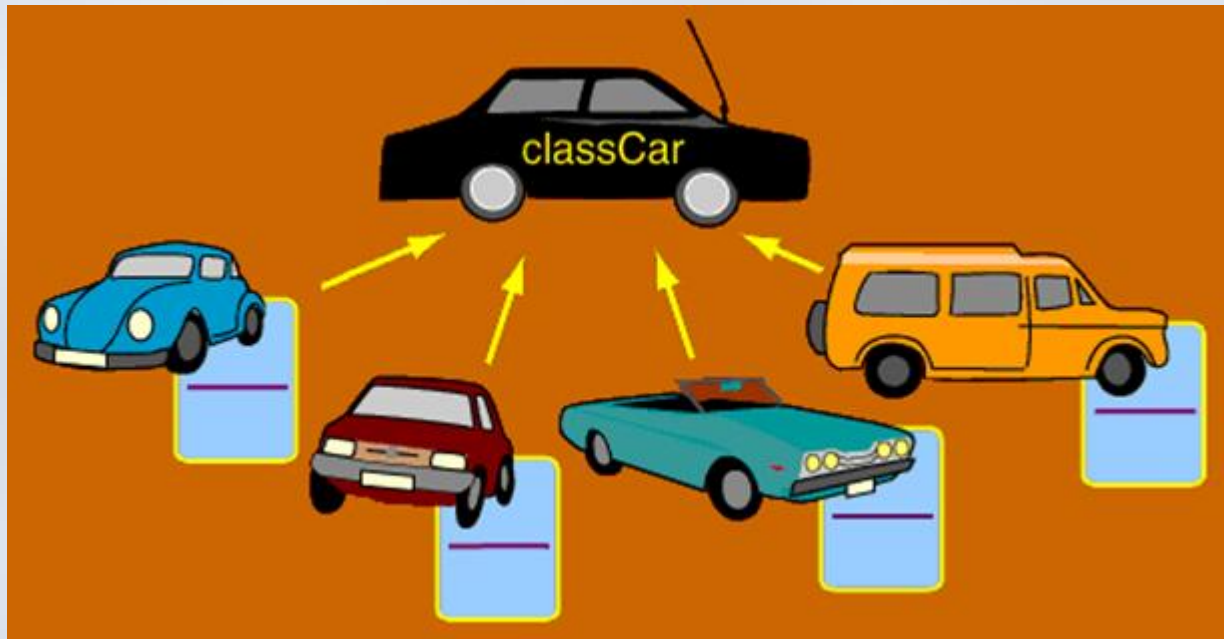
ตัวอย่าง class รถยนต์



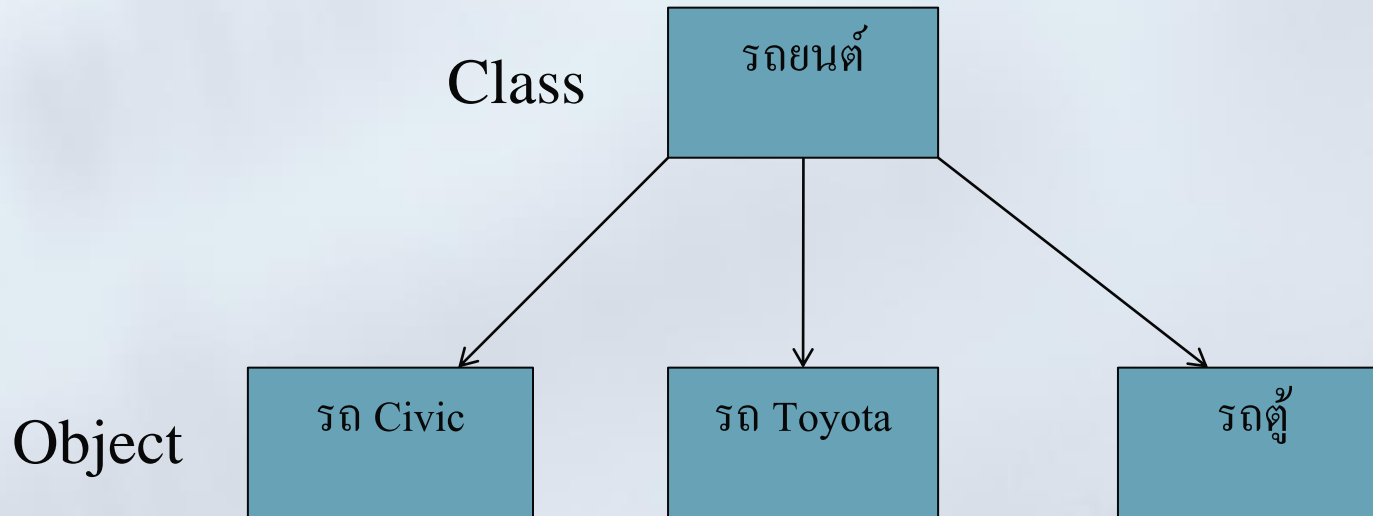
Class รถยนต์	
Attribute	ยี่ห้อ, รุ่น, สี, ความเร็ว
Method	สตาร์ท, ดับเครื่อง, เปลี่ยนเกียร์, เบรค

จาก Class นำไปสร้างเป็น → Object


หากเขียน โปรแกรมประมวลเกี่ยวกับรถ จะต้องสร้าง Object ของรถขึ้นมา โดยรถ 1 คันหมายถึง Object 1 Object



. Class : คณิตศาสตร์ “รถ”



Object mycar, BillsCar



myCar

reg_no	ABC 123
colour	gray
weight	930
speed	0

=====
=====



BillsCar

reg_no	DEF 456
colour	black
weight	1020
speed	40

=====
=====

ตัวอย่าง Object Car

Attribute

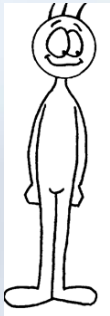
ยี่ห้อ = BMW
Model : M6
Color : แดง



Method

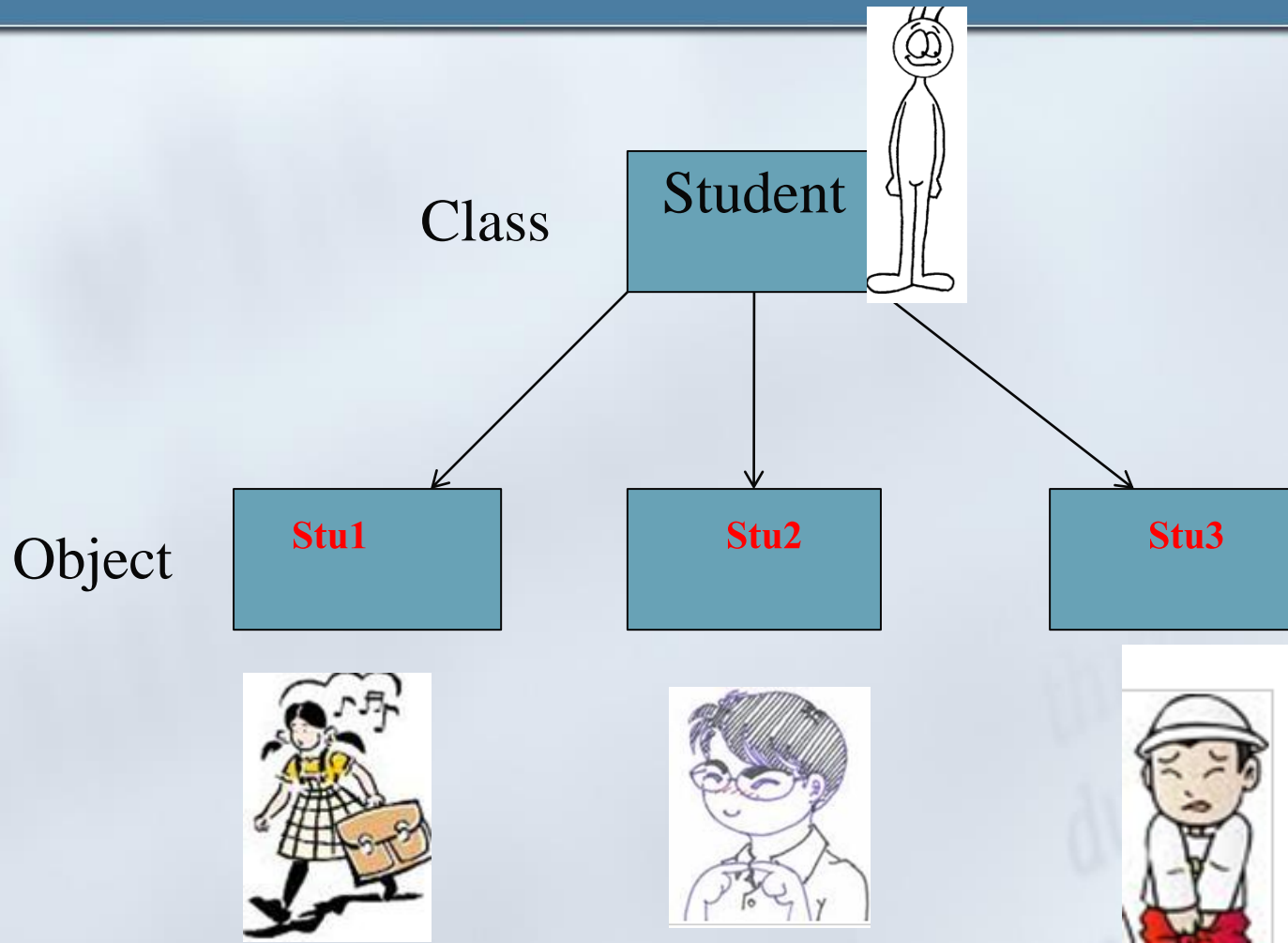
สตาร์ทเครื่อง
ดับเครื่อง
เปลี่ยนเกียร์
เบรค

ตัวอย่าง 2 : Class “นักเรียน”



Class Student	
Attribute	ชื่อ รหัส อายุ
Method	ลงทะเบียนเรียน

สร้าง Object นักเรียน จาก Class Student



สร้าง Object คน จาก Class person

Class

Student	
ID	<input type="text"/>
Name	<input type="text"/>
Age	<input type="text"/>

Object

Student	
ID	<input type="text" value="1234"/>
Name	<input type="text" value="สมหญิง"/>
Age	<input type="text" value="15"/>

Stu1

Student	
ID	<input type="text" value="1235"/>
Name	<input type="text" value="สมหมาย"/>
Age	<input type="text" value="16"/>

Stu2

Student	
ID	<input type="text" value="1236"/>
Name	<input type="text" value="สมชาย"/>
Age	<input type="text" value="9"/>

Stu3

Note :

- ก่อนที่จะสร้าง**ออบเจกต์**ขึ้นมาได้เราต้องสร้าง**คลาส**ขึ้นมาก่อน คลาสก็เปรียบเสมือน**แม่แบบ** หรือ**พิมพ์เขียว** ในการสร้างออบเจกต์ต่างๆ ขึ้นมา
- ออบเจกต์ที่สร้างมาจากคลาสเดียวกันอาจมีรายการของคุณสมบัติที่เหมือนกัน, มีความสามารถที่เหมือนกัน แต่จะแตกต่างกันด้วย ค่าของคุณสมบัติต่างๆ
- ออบเจกต์ที่ถูกสร้างจากคลาสจะเรียกว่า “**อินสแตนซ์**” instance

Class "รถยนต์"



Attribute

ยี่ห้อ

สี

รุ่น

จำนวนล้อ

ฯลฯ

Method

สตาร์ทเครื่อง()

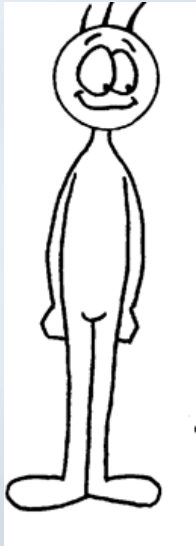
เบรก()

เปลี่ยนเกียร์()

ดับเครื่อง()

ฯลฯ

Class "people"



Attribute

หมายเลขบัตรประชาชน

ชื่อ-สกุล

วันเกิด

เพศ

ที่อยู่

ฯลฯ

Method

บอกเลขที่บัตรประชาชน()

บอกชื่อ()

บอกวันเกิด()

บอกที่อยู่()

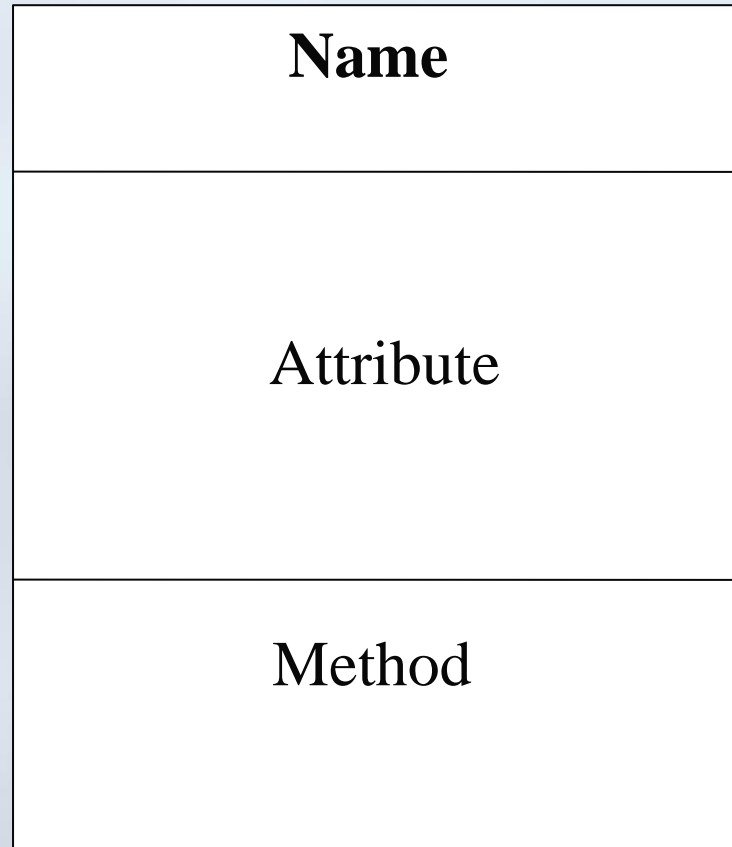
กิน()

นอน()

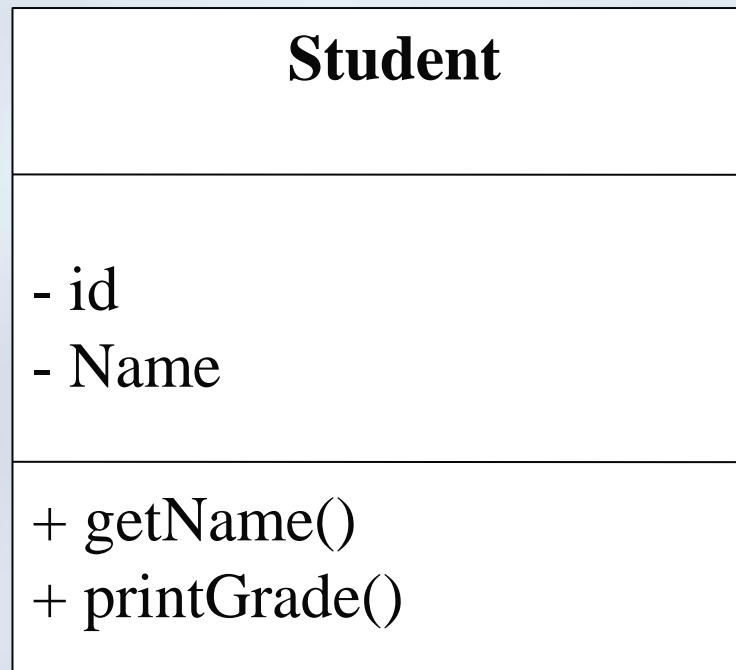
วิ่ง()

ฯลฯ

การเขียน Class diagram



ตัวอย่าง Class diagram ของ class “Student”



แสดงข้อมูล 2 instance จาก Class “Student”

Identifier

paul:Student

peter:Student

Variables

id=1888
name="Paul Lee"

id=1999
name="Peter Tan"

Methods

getName()
printGrade()

getName()
printGrade()

2 instances of the class Student

หลักการพื้นฐานของ object oriented

■ **Encapsulation** และ Information Hiding

การห่อหุ้ม และ การซ่อนรายละเอียด

■ **Inheritance** การสืบทอด และ (Reusable)

การนำวัตถุมาริใช้ใหม่

■ **Polymorphism** การพ้องรูป

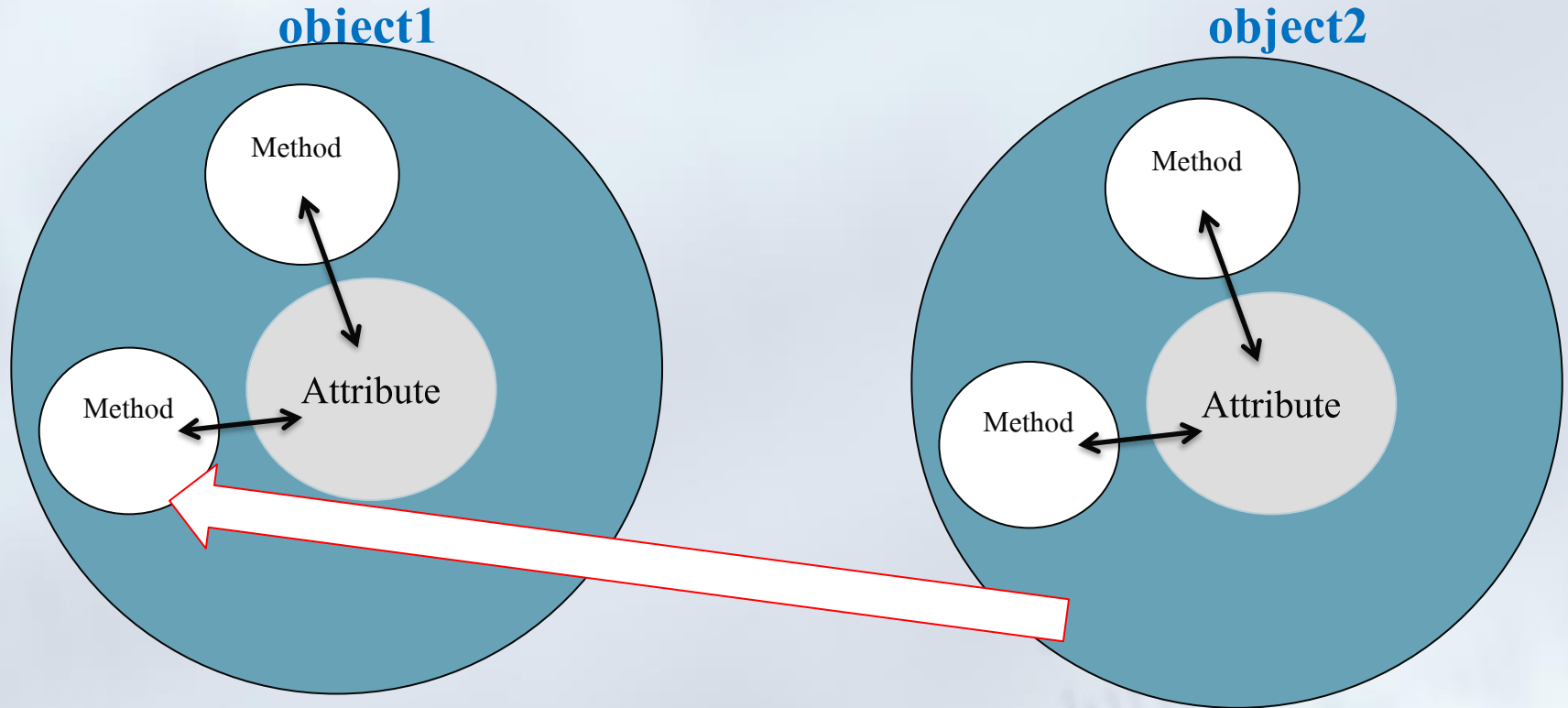
การซ่อนรายละเอียด(Information Hiding)

- มีผู้สังเกตเห็นว่าในชีวิตประจำวันของเรานั้น ไม่จำเป็นต้องทราบหรือรู้ทุกเรื่องก็สามารถใช้งานสิ่งของนั้นๆ ได้ เช่น

โทรทัศน์ที่มีอยู่ตามบ้าน เราไม่ต้องทราบการทำงานของอุปกรณ์ภายในของมัน แต่เราก็สามารถเปิดเครื่องและดูรายการโปรดของเราได้ เพียงรู้วิธีการกดปุ่มต่างๆ



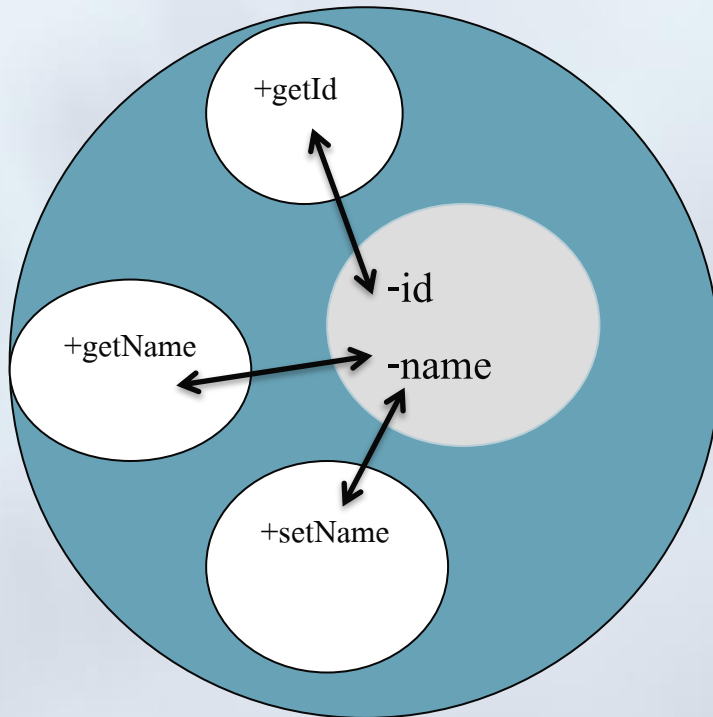
Encapsulation การห่อหุ้ม



การเข้าถึง Attribute จะต้องผ่าน Method

Encapsulation & Information Hiding

Student



Student
- id - Name
+getId() +getName() +setName()

Encapsulation

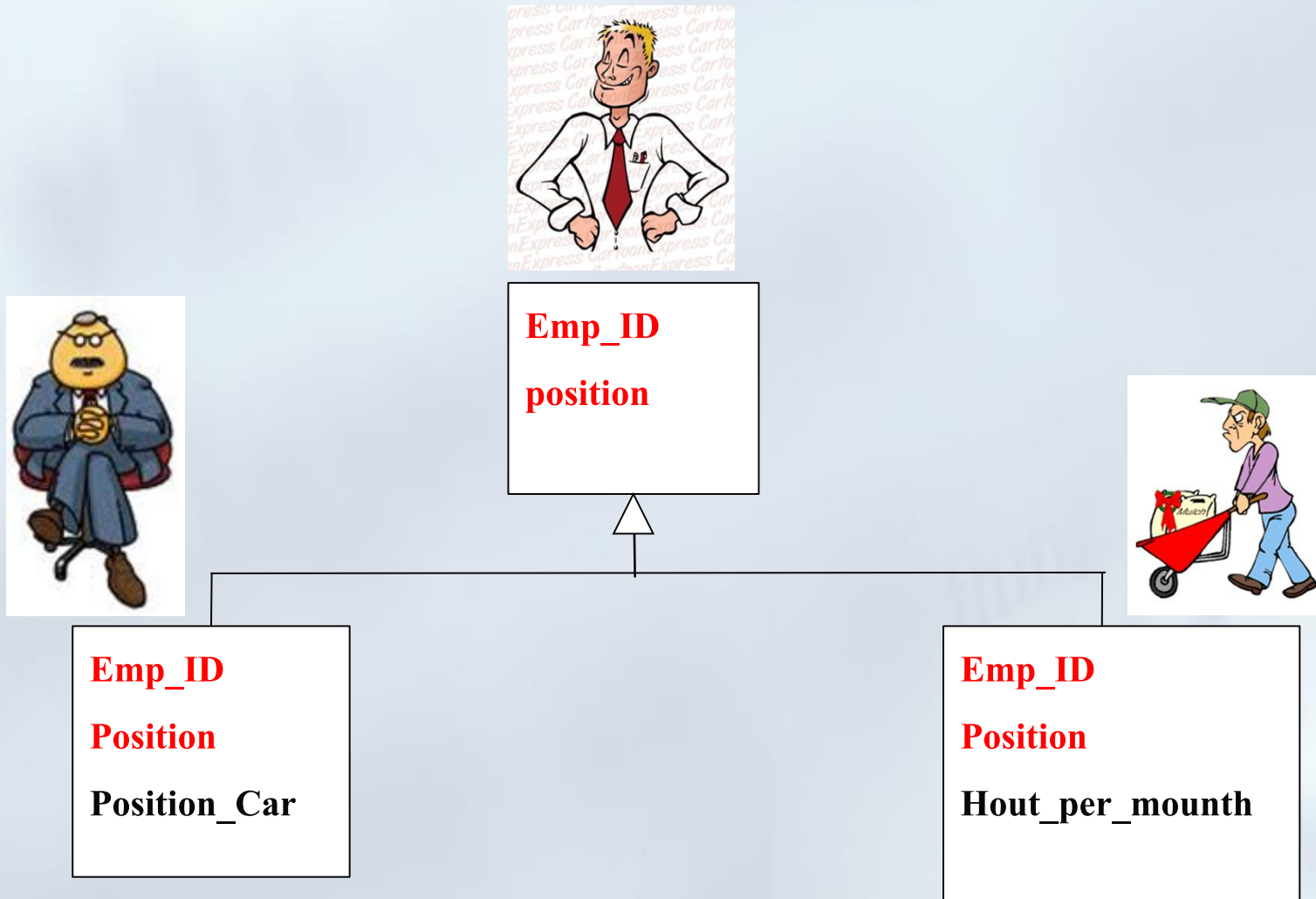
- Encapsulation คือรากฐานอย่างหนึ่งของแนวความคิดในเชิง Object-Oriented ซึ่งข้อดีของ Encapsulation คือการป้องกัน attribute (data) ของ object จากความเสียหาย เพราะถ้าส่วนของโปรแกรมทั้งหมด อนุญาตให้มีการเข้าถึง Attribute ได้ตามที่ต้องการแล้วนั้น จะส่งผลให้ attribute นั้นง่ายต่อการถูกใช้อย่างผิดๆ ทำให้ค่า attribute เปลี่ยนแปลงไป ซึ่งก่อให้เกิดความเสียหายตามมา
- Encapsulation คือการห่อหุ้ม attributes และ methods เข้าไว้ด้วยกัน
- Encapsulation จะทำหน้าที่ป้องกันมิให้ Object อื่นที่อยู่ภายนอก เข้าถึง Object หนึ่งๆ ได้อย่างอิสระจะมีเฉพาะ methods ที่อยู่ใน Object เท่านั้นจะสามารถติดต่อกับ attribute ที่อยู่ใน Object เดียวกันได้ เรียกได้ว่าการ

Encapsulation มีคุณสมบัติของ Information hiding

Information Hiding

- Information Hiding คือการจำกัดการมองเห็นข้อมูลภายใน Object เช่นการกำหนดคุณลักษณะเป็น “**public**” + เพื่อให้สามารถเชื่อมต่อกับ Object ภายนอกได้ และกำหนดเป็น “**private**” - เพื่อจำกัด attribute ให้อยู่ภายใน object เท่านั้น

Inheritance (การสืบทอดคุณสมบัติ)



Inheritance (การสืบทอดคุณสมบัติ)

- Inheritance คือ คุณสมบัติที่ Class ๓ หนึ่ง สามารถสืบลักษณะของ attribute และ method ของ อีก Class หนึ่ง ได้การทำเช่นนี้ทำให้คุณสามารถ Create Class ใหม่ขึ้น โดยนำสาระสำคัญ ที่เหมือนกันของ attribute และ behavior(method) จาก class อื่นมาใช้ได้

Inheritance (การสืบทอดคุณสมบัติ)



คลาสแม่ (Super Class)

Emp_ID
position



Emp_ID
Position
Position_Car

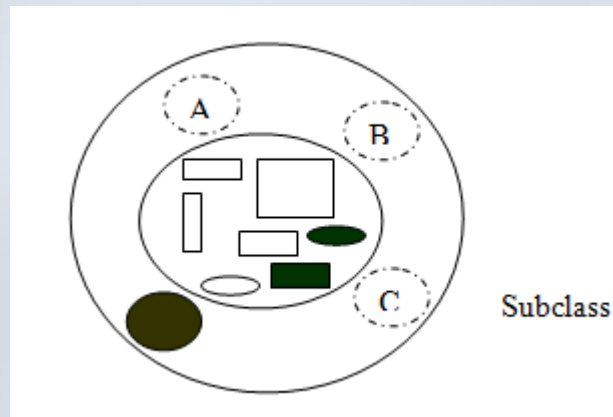
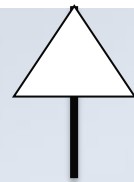
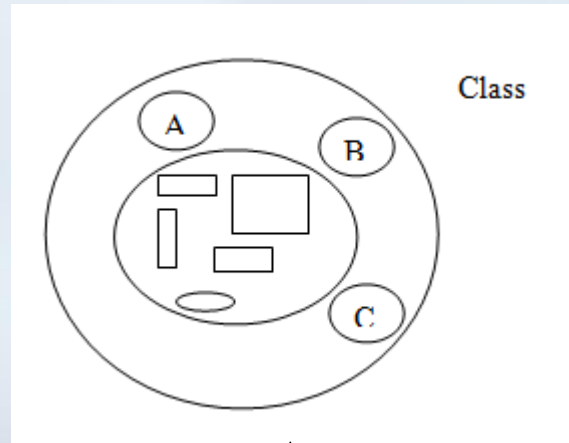
คลาสลูก (SubClass)



Emp_ID
Position
Hout_per_mounth

คลาสลูก (SubClass)

Inheritance (การสืบทอดคุณสมบัติ)



Polymorphism การพ้องรูป

คือ การทำให้สิ่งหนึ่งสามารถทำงานได้หลากหลายรูปแบบตามความต้องการที่เกิดขึ้นในขณะหนึ่ง ๆ เช่นการที่ Method ชื่อเดียวกันสามารถรับค่าที่ส่งมาแตกต่างกันได้หลายรูปแบบ

Polymorphism การพ้องรูป

ยกตัวอย่าง Method

calculate(int x)

calculate(double y)

หากมีผู้ใช้โปรแกรมส่งข้อมูลเข้ามาเป็นตัวเลขจำนวนเต็ม เราก็สามารถเรียกเมธอด calculate(int x) ให้ทำงาน

หากผู้ใช้โปรแกรมส่งข้อมูลเลขทศนิยมเข้ามาเราก็สามารถเรียกคำนวณให้ละเอียดมากขึ้น โดยเรียกเมธอด calculate(double y)

โดยไม่ต้องไปตั้งชื่อเมธอดใหม่ เพียงแค่ตั้งค่า อาร์กิวเมนต์ที่แตกต่างกัน มีศัพท์เรียกเฉพาะว่า เมธอดชื่อนั้นถูก “**Overload**”

การพ้องรูป (Polymorphism)

- รากฐานของการพ้องรูปคือคุณสมบัติการถ่ายทอด
- คุณสมบัติการถ่ายทอดยืนยันได้ว่าคลาสลูกที่เกิดจากคลาสแม่เดียวกันย่อมมีคุณสมบัติเหมือนกัน

คลาสแม่คือ **Shape**

คลาสลูกคือ **Circle, Triangle, Rectangle** มีคุณสมบัติเหมือน

คลาสแม่ทุกประการ

- เป็นที่มาของ **หนึ่งรูปหลายพฤติกรรม**

การถ่ายทอดให้เกิดลักษณะของพ็องรูป

