OOAD

สยามทูเดฟ

www.siam2dev.com

วิชา การวิเคราะห์และออกแบบระบบเชิงวัตถุ (4122506)

Lec02_3_Generalization Abstraction
(Generalization Abstraction)

By
ผู้ช่วยศาสตราจารย์ ดร. นัฐพงศ์ ส่งเนียม
Asst. Dr. Nattapong Songneam
เว็บไซต์ :: http://www.siam2dev.com
อีเมล์ :: siam2dev@gmail.com

UNIFIED MODELING LANGUAGE

สยามทูเดฟ

Last Update : 18/05/2567

สอบระหว่างภาค 2/2567

- บทที่ใช้สอบ
  - บทที่ 1, 2, 2.1, 2.2, 2.3, 2.4, 3 , 4
- สอบ วันที่.... .... ม.ค. 2568
- ห้องสอบ ...

อาจารย์ ดร. นัฐพงศ์ ส่งเนียม

- Website : http://www.siam2dev.com

- E-mail1 : dr.nattapong_s@hotmail.com
- E-mail2 : siam2dev@hotmail.com
- E-mail3 : xnattapong@hotmail.com
- Facebook : siam2dev@hotmail.com

Abstractions

1. Classification abs.
2. Aggregation abs.
3. *Generalization/Specialization abs.*
4. *Association abs.*

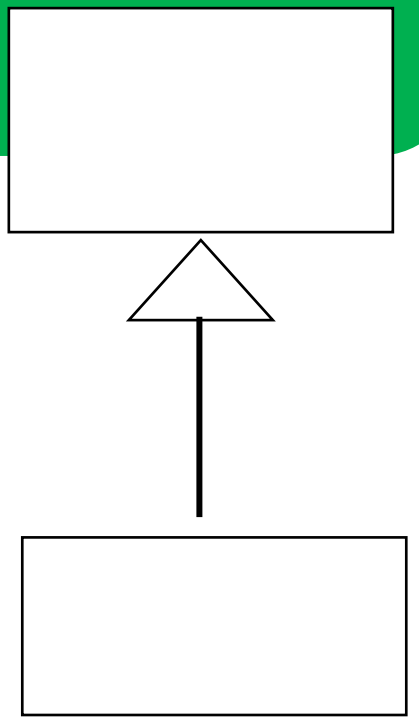
OOP

***** กลไกเหล่านี้เกิดจากการทำ Generalization**

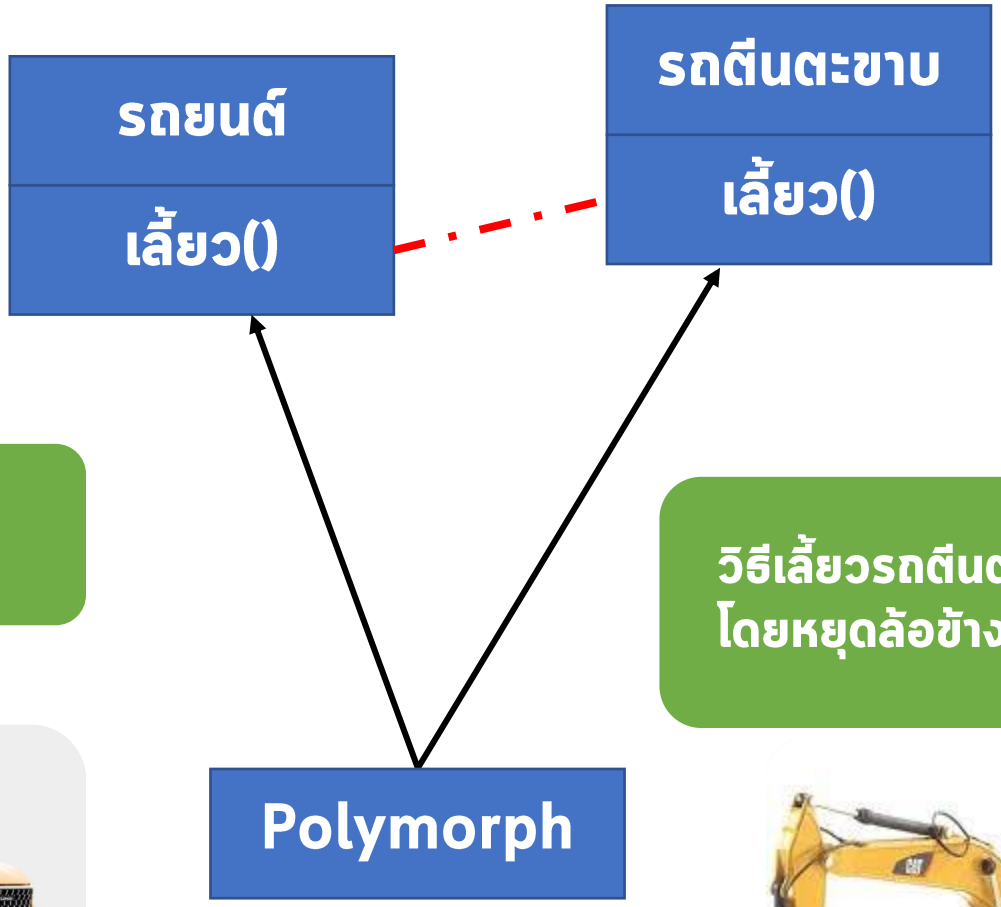
- **Reusable ความสามารถในการนำกลับมาใช้ใหม่**
- **Polymorphism การพ้องรูป หรือ หนึ่งรูปหลายพฤติกรรม**
  - **Override**
  - **Overloading**

คน
กิน()

แมว
กิน()



Polymorph การพ้องรูป หรือ หนึ่งรูปหลายพฤติกรรม



วิธีเลี้ยวรถยนต์ใช้พวงมาลัย

วิธีเลี้ยวรถตีนตะขาบเลี้ยวโดยหยุดล้อข้างใดข้างหนึ่ง

Lec02_3_Generalization Abstraction

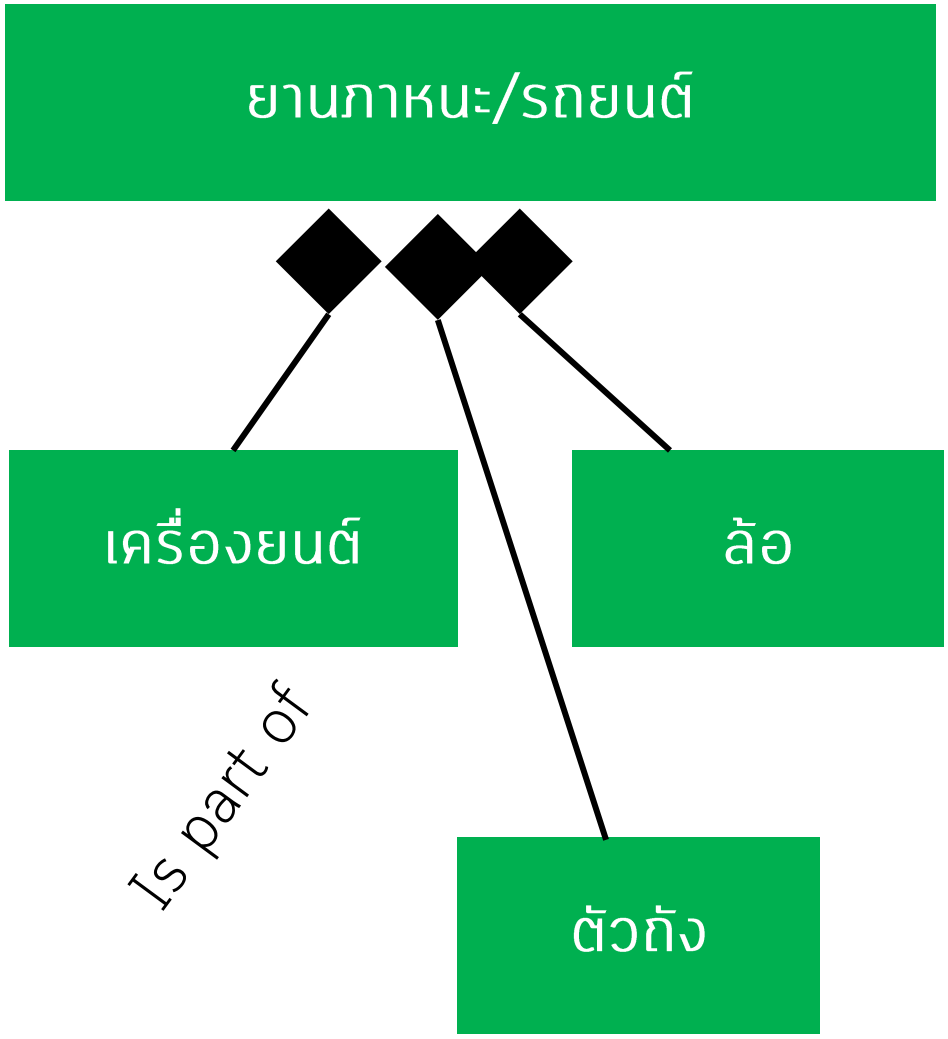
- เพื่อให้ผู้อ่านเข้าใจหลักการของ Generalization Abstraction และการ Inheritance
- เพื่อให้ผู้อ่านเข้าใจวัตถุประสงค์ กลไก และวิธีการทำ inheritance
- เพื่อให้ผู้อ่านสามารถแยกความแตกต่างและสร้างความสัมพันธ์ระหว่าง Super class และ Sub class ได้
- เข้าใจหลักการของ Polymorphism ได้

Generalization Abstraction

Generalization Abstraction คือกระบวนการในการนำ Class ที่มีลักษณะเหมือนหรือคล้ายกันหรือมีคุณสมบัติอย่างใดอย่างหนึ่งร่วมกัน (General) มาจัดหมวดหมู่ไว้เป็น Class เดียวกัน ซึ่งกระบวนการย้อนกลับของGeneralization Abstraction เรียกว่า **Specialization** คือการตอบคำถามว่าใน Class หนึ่ง ๆ นั้นสามารถจำแนกเป็น Class อะไรได้บ้าง

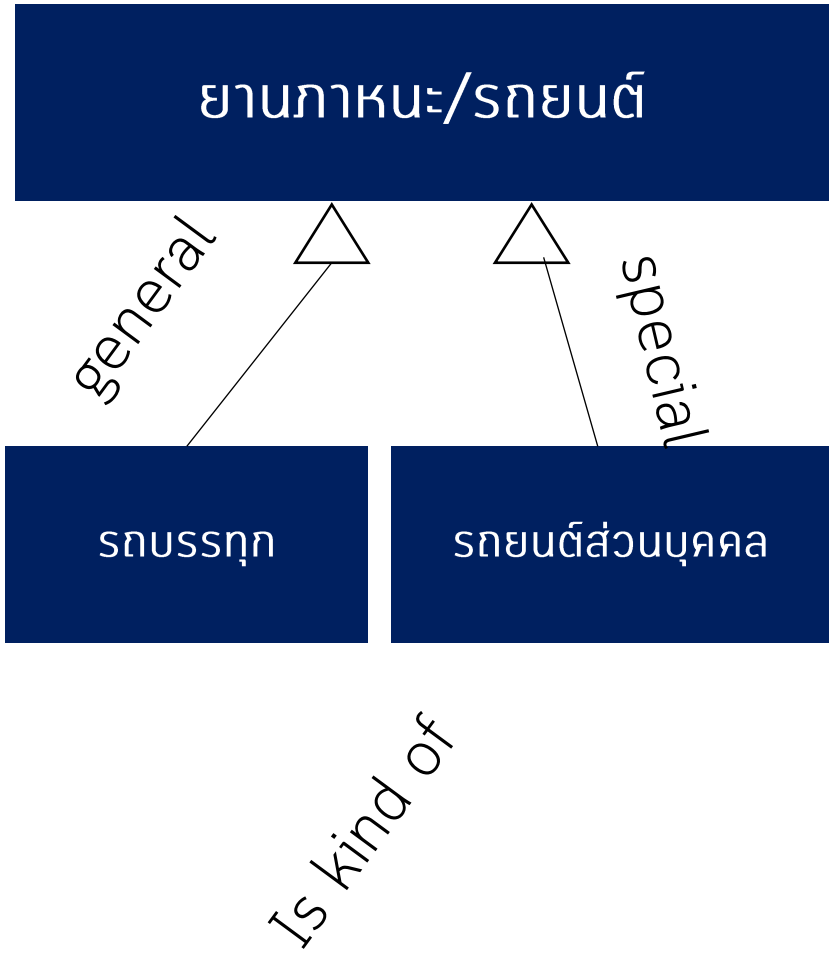
หากสนใจ slide นี้เพื่อการเรียนการสอนโปรดติดต่อ siam2dev@hotmail.com

Aggregate

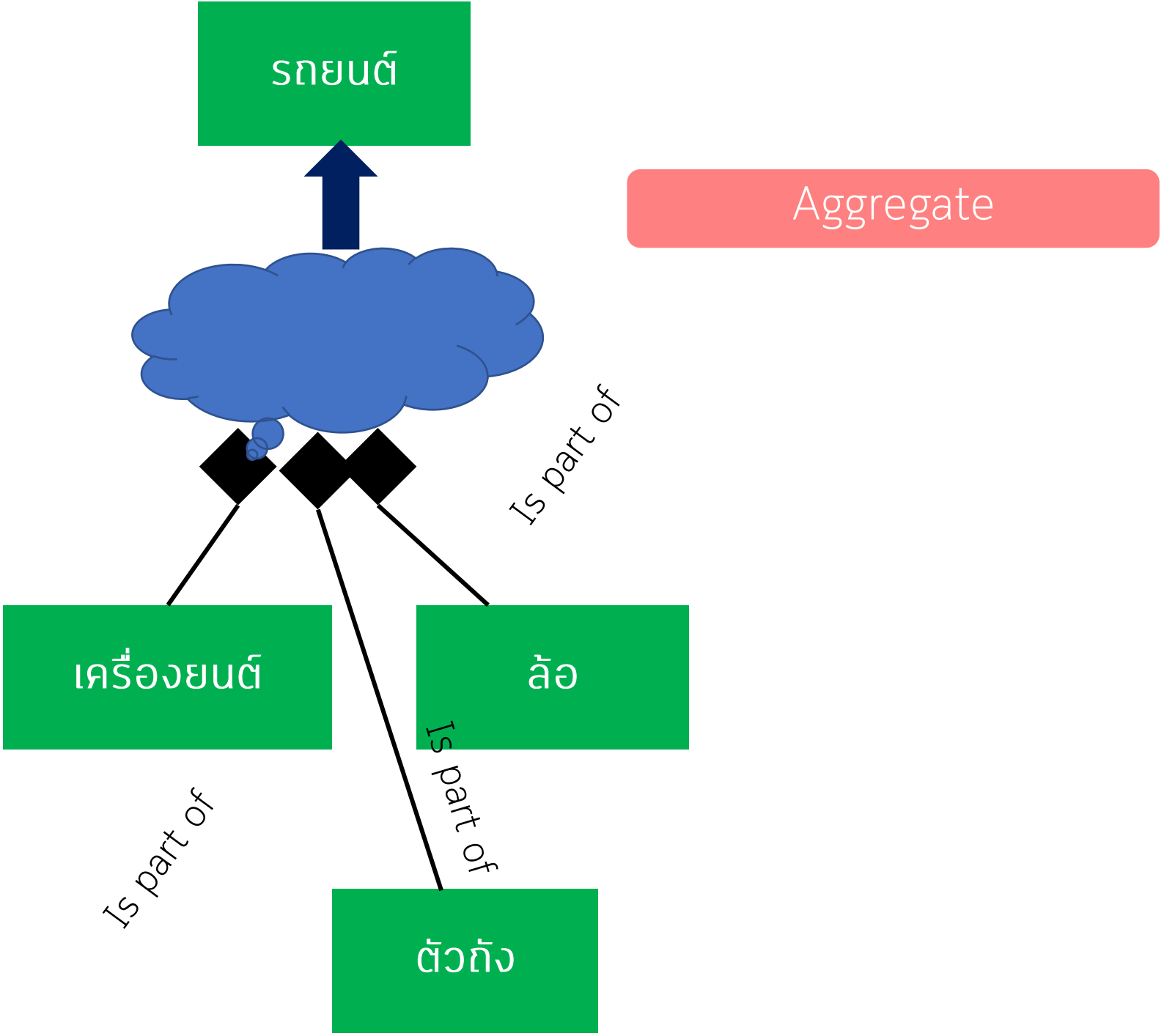


Aggregate ดูว่าคลาสมีส่วนประกอบอะไรบ้าง

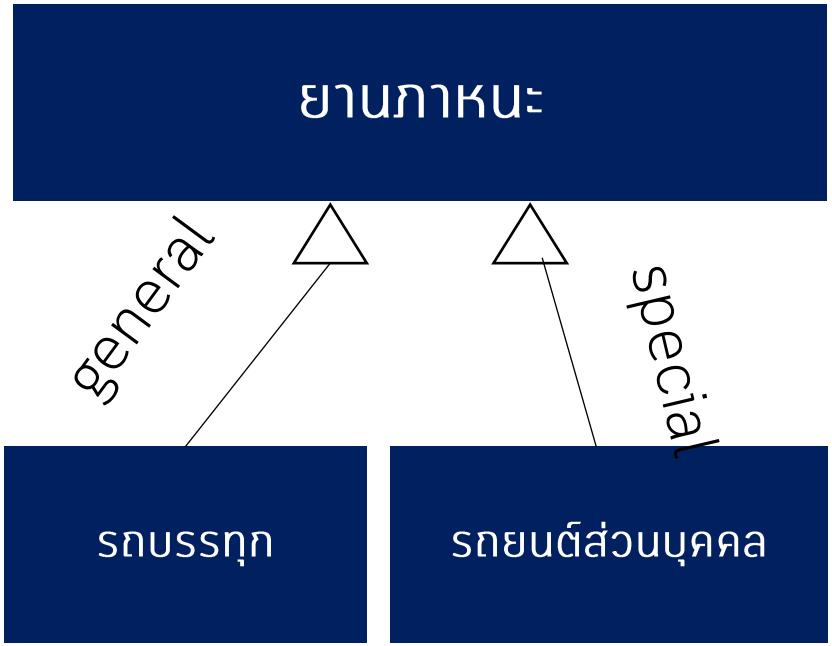
Generalization



Generalization ดูว่าคลาสจำแนกเป็นคลาสอะไรบ้าง


รถยนต์
Aggregate
Is part of
เครื่องยนต์
ล้อ
Is part of
Is part of
ตัวถัง

Generalization

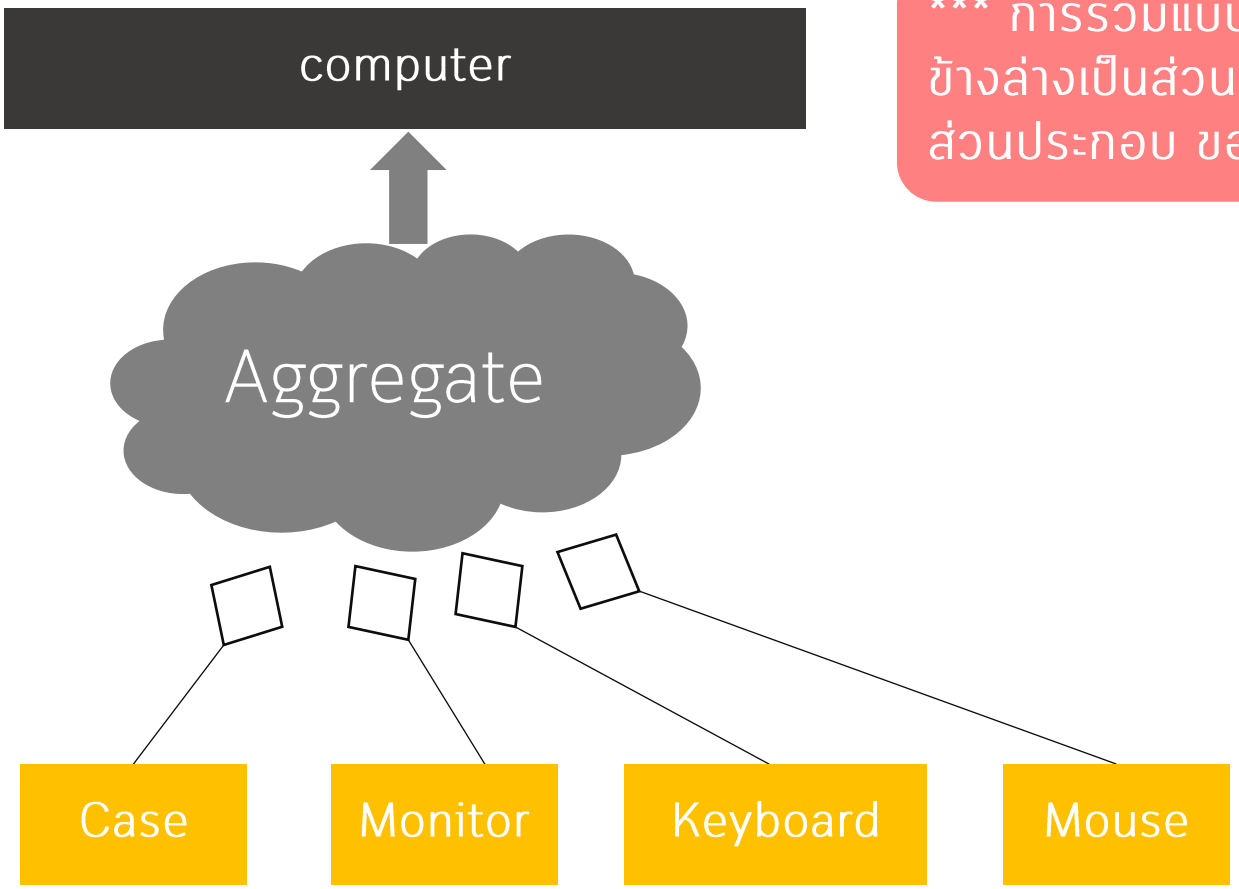


สองคลาส มีบางอย่างคล้ายกัน ทำให้สามารถมารวมกันได้ทำให้เกิดคลาสใหม่ขึ้นมาได้

- **ล้อ**
- **เครื่องยนต์**
- **ตัวถัง**
- **พวงมาลัย**

Generalization คือ นำคลาสต่าง ๆ ที่มีลักษณะบางคล้ายกันมาจัดรวมเป็นคลาสเดียวกัน
Aggregation คือ นำคลาสย่อย มารวมกัน ไม่จำเป็นที่ต้องมีอะไรที่เหมือนกัน

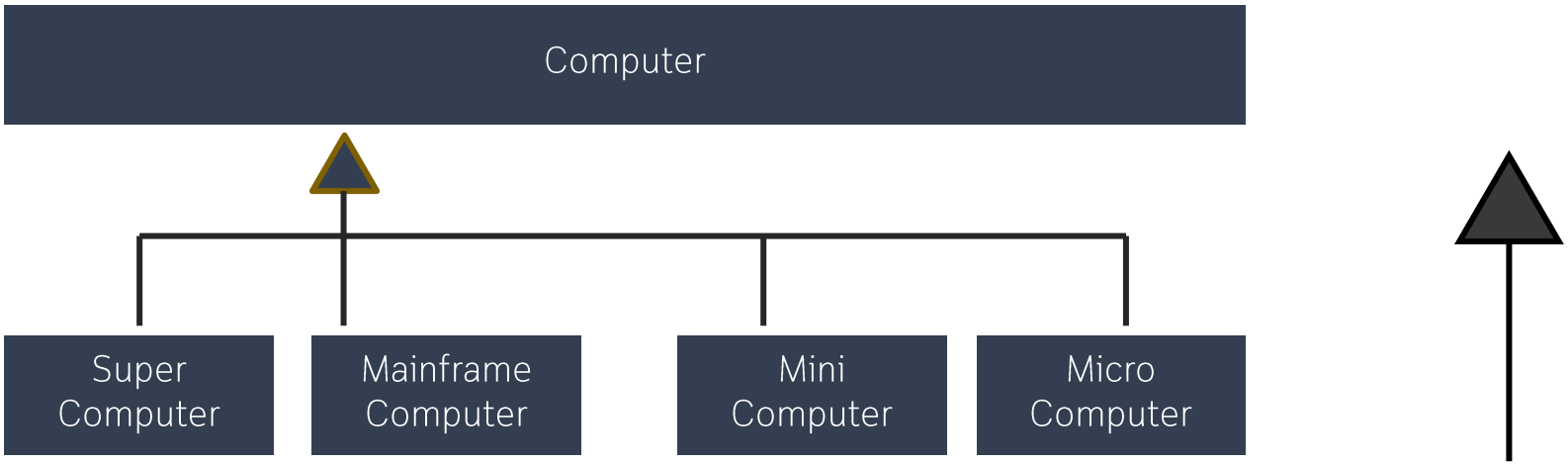
*** การรวมแบบ aggregate คือ ข้างล่างเป็นส่วนหนึ่ง หรือ เป็นส่วนประกอบ ของคลาสด้านบน

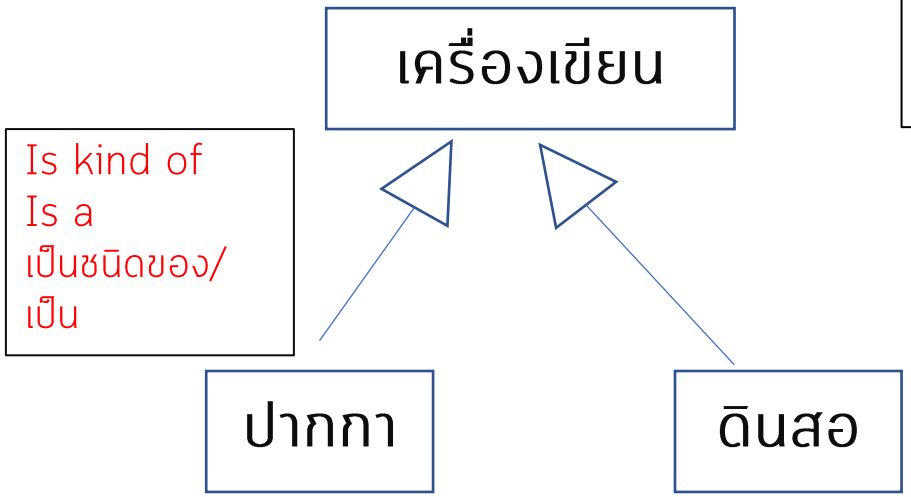


หมายถึง เมาส์ จอ เคส คีย์บอร์ด เป็นชิ้นส่วนหนึ่งของคอมพิวเตอร์

Generalization คือ นำคลาสต่าง ๆ ที่มีบางคล้ายกันมาจัดรวมเป็นคลาสเดียวกัน
Aggregation คือ นำคลาส มารวมกัน ไม่จำเป็นที่ต้องมีอะไรที่เหมือนกัน

*** การแบ่งประเภทของคอมพิวเตอร์ ว่า มี 4 ประเภท 1. super 2. mainframe 3. mini 4. micro

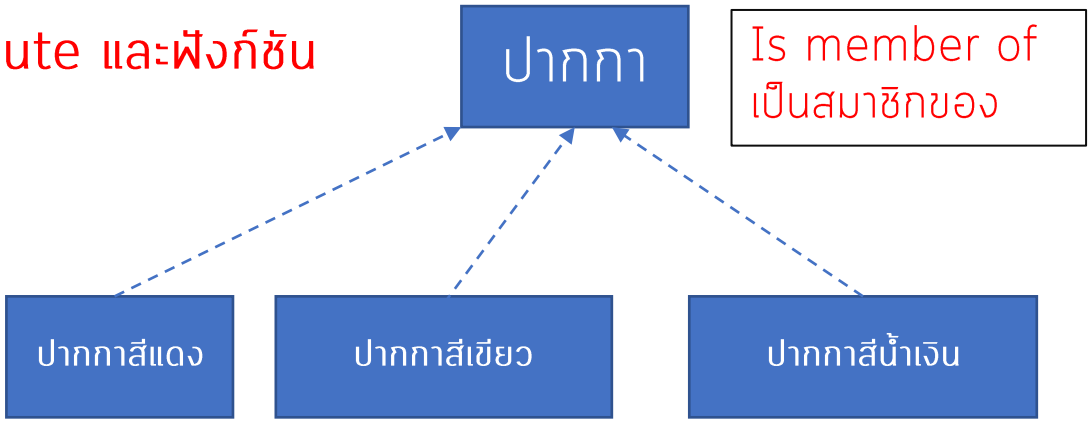




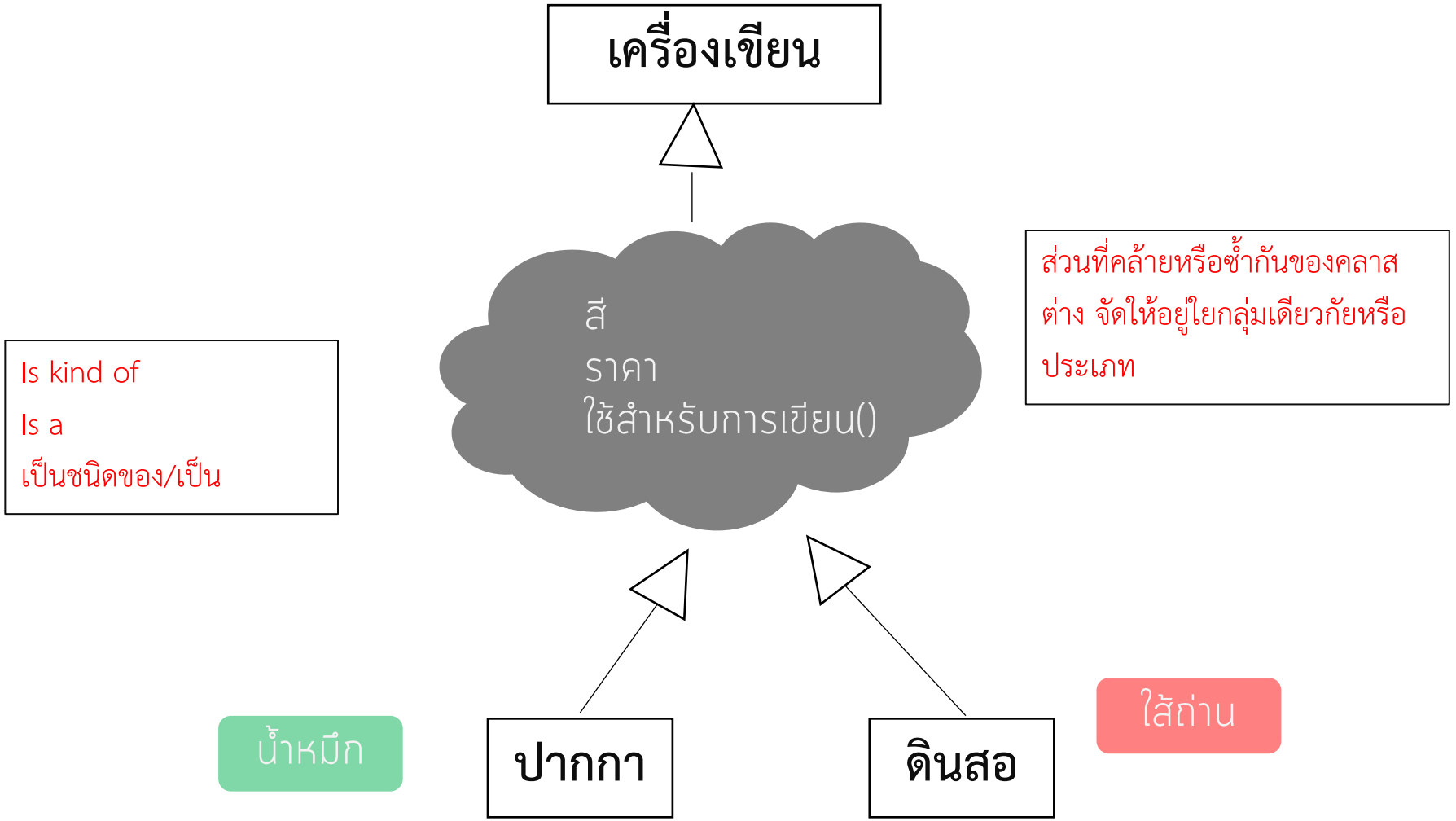
*** ปากกา กับ ดินสอ มีอะไรที่เหมือนกันบ้าง ถ้าพอจะมีอะไรที่เหมือนกัน ก็จัดให้อยู่กลุ่มเดียวกันได้ โดยไม่สนใจ สิ่งที่แตกต่างกัน

*** ปากกา กับ ดินสอ มี ความสามารถในการเขียน และ มีสี มีราคา มีความยาว มีขนาด จึงจัดให้อยู่ในกลุ่มเดียวหรือประเภทกันได้ เรียก กลุ่มใหม่นี้ ว่า คลาสเครื่องเขียน

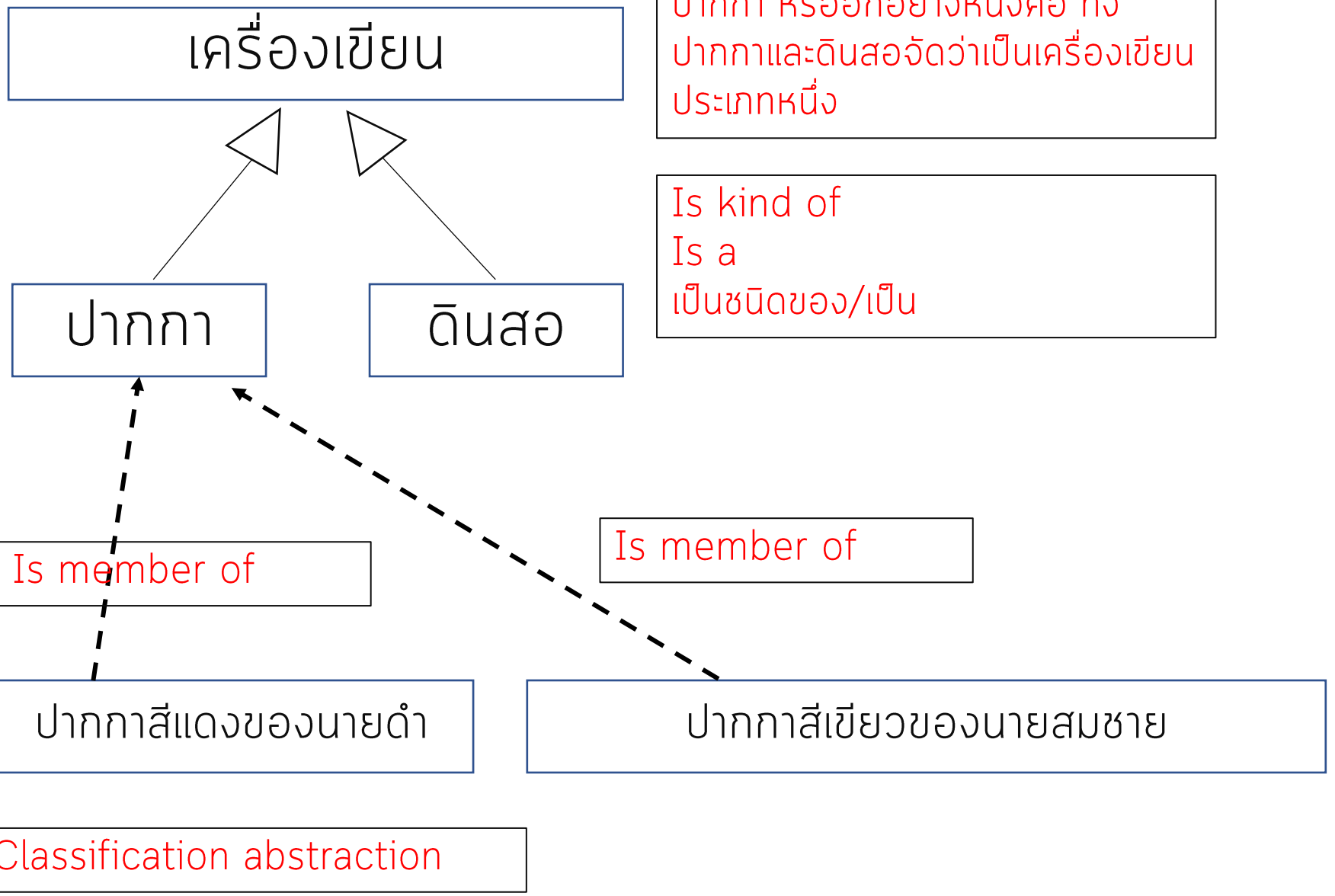
*** คำว่าเหมือนกันให้ดู ทั้ง attribute และฟังก์ชัน



*** คลาสสิฟิเคชัน แอบแทรคชัน
classification abstraction

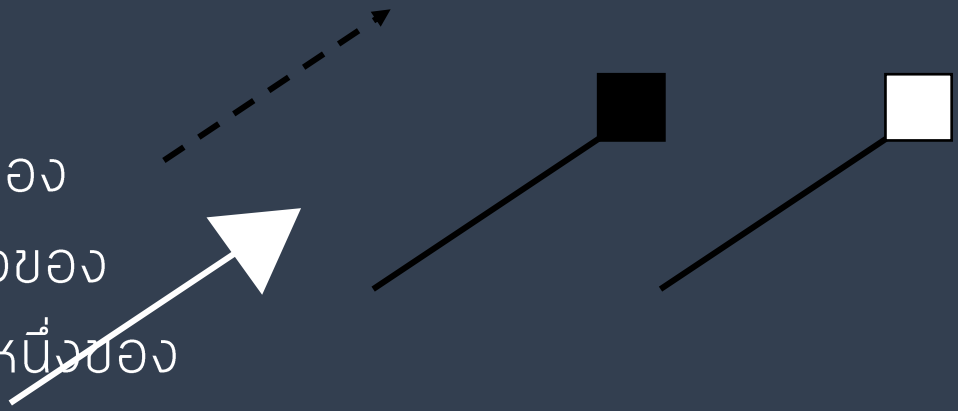


Generalization คือการพิจารณาความสามารถและลักษณะของคลาส ที่ร่วมกัน และจัดให้อยู่ในประเภทเดียวกัน


เครื่องเขียน
เครื่องเขียนแบ่งออกเป็นดินสอและปากกา หรืออีกอย่างหนึ่งคือ ทั้งปากกาและดินสอจัดว่าเป็นเครื่องเขียนประเภทหนึ่ง
ปากกา
ดินสอ
Is kind of
Is a
เป็นชนิดของ/เป็น
Is member of
Is member of
ปากกาสีแดงของนายดำ
ปากกาสีเขียวของนายสมชาย
Classification abstraction

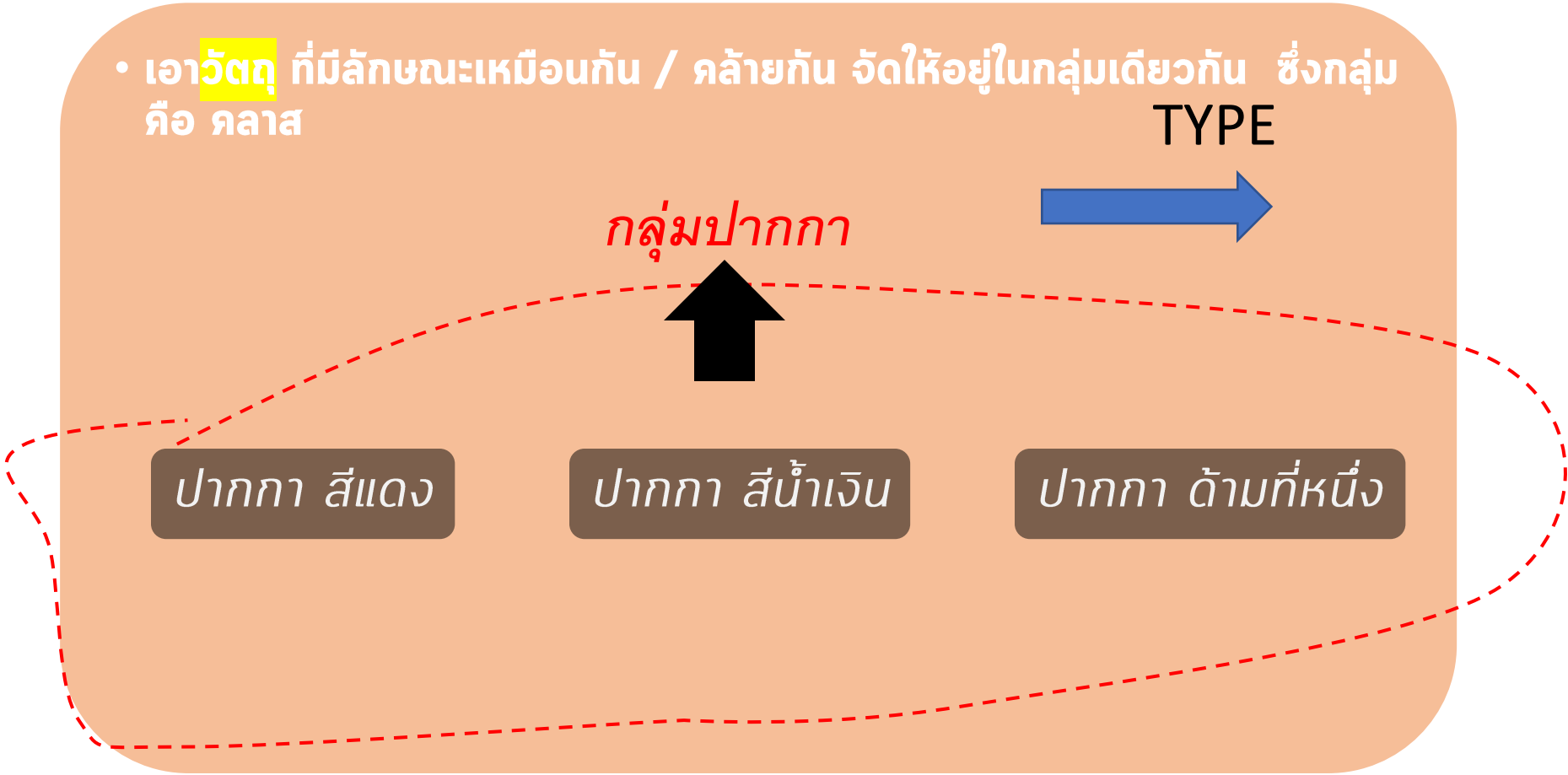
ประเภทของ relationship

- Is member of เป็นสมาชิกของ
- Is part of เป็นส่วนหนึ่งของ
- Is kind of เป็นประเภทหนึ่งของ



Classification Abs.

- เอาวัตถุ ที่มีลักษณะเหมือนกัน / คล้ายกัน จัดให้อยู่ในกลุ่มเดียวกัน ซึ่งกลุ่มคือ คลาส

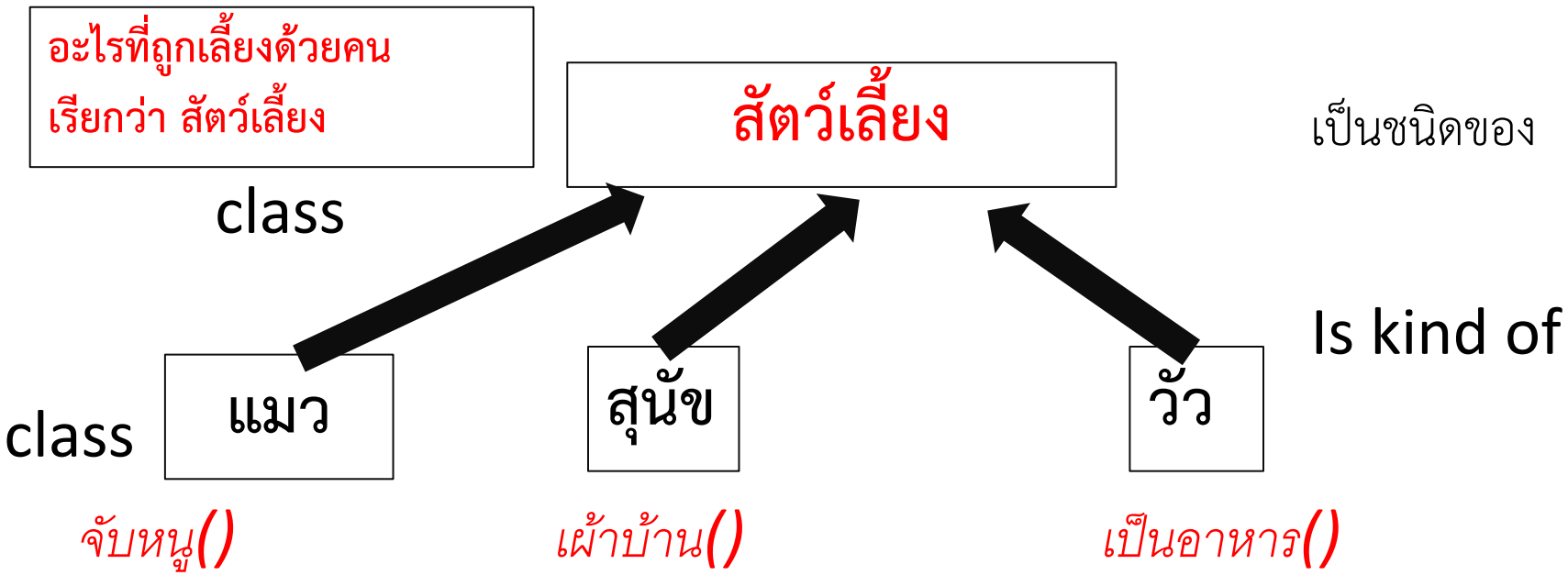


Generalization

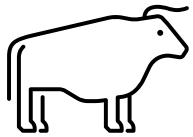
คลาสหนึ่งมีบางอย่างที่ไปคล้ายคลึงกับอีกคลาสหนึ่ง

- เอาคลาส ที่มีลักษณะเหมือนกัน / คล้ายกัน จัดให้อยู่ในกลุ่มเดียวกัน ซึ่งกลุ่มคือ คลาส
- ให้ตัดส่วนที่แตกต่างออก (ignore)
- พิจารณาเฉพาะส่วนที่เหมือนกัน

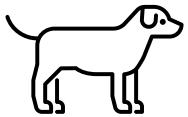
Is member of
Is part of
Is kind of



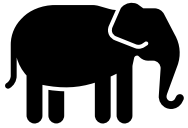
Aggregation abstraction



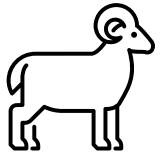
วัว + เขา + ขา + หาง = กินหญ้า

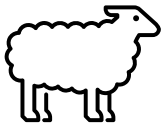


สนัข + เขี้ยว + หู + หาง = เห่า



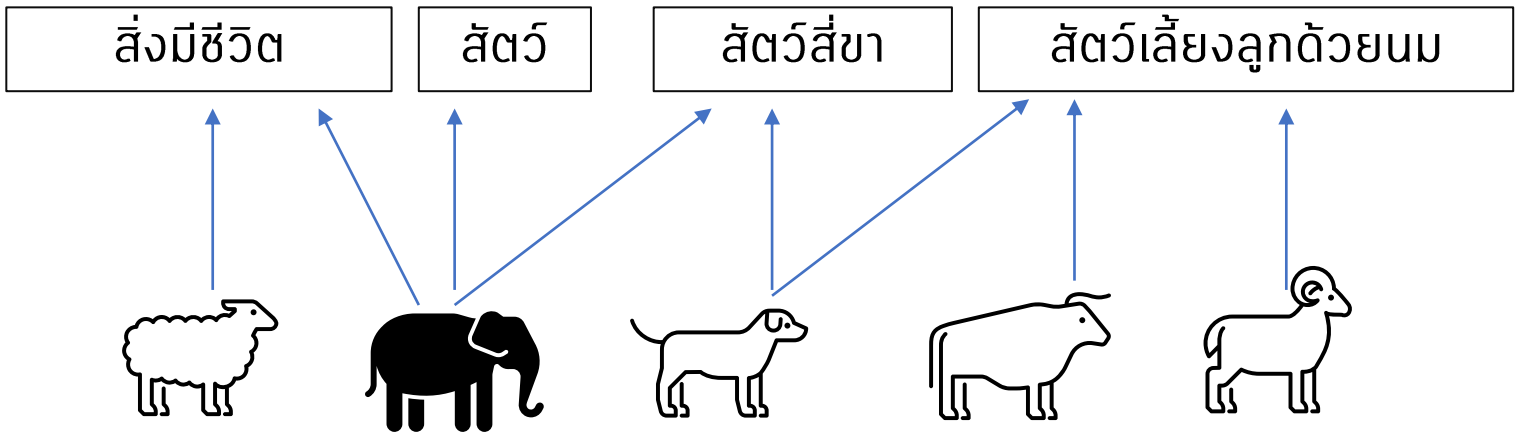
ช้าง + งา + หู + หาง = กินอ้อย / ชน





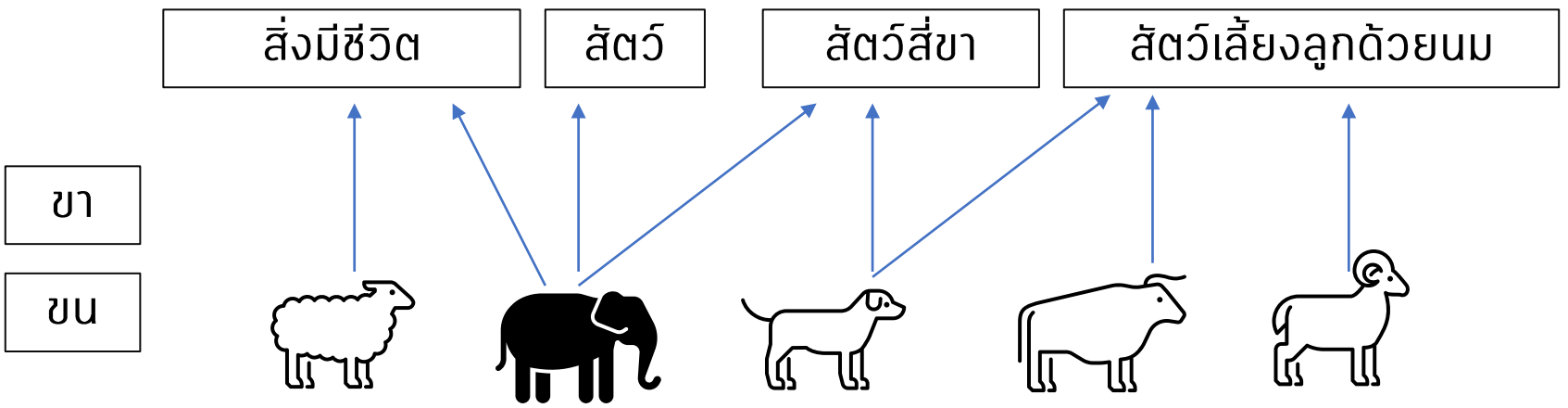
Generalization abstraction

วัว	เขา	+	ขา	+	หาง	กินหญ้า
สนัข	เขี้ยว	+	หู	+	หาง	เห่า
ช้าง	งา	+	หู	+	หาง	กินอ้อย / ชน



Generalization abstraction

วัว	เขา	+	ขา	+	หาง	กินหญ้า
สนุข	เขี้ยว	+	หู	+	หาง	เห่า
ช้าง	งา	+	หู	+	หาง	กินอ้อย / ชน



Generalization abstraction

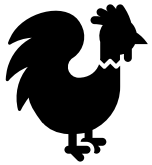
ขา

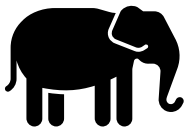
ขน

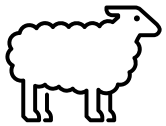
กินได้()

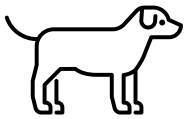
เดินได้()

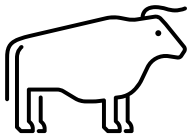
นอน()

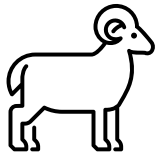












Generalization abstraction

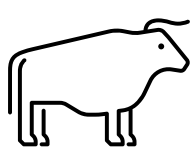
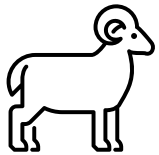
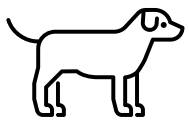
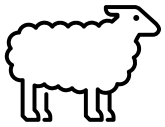
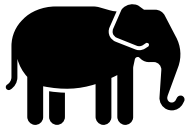
จระเข้

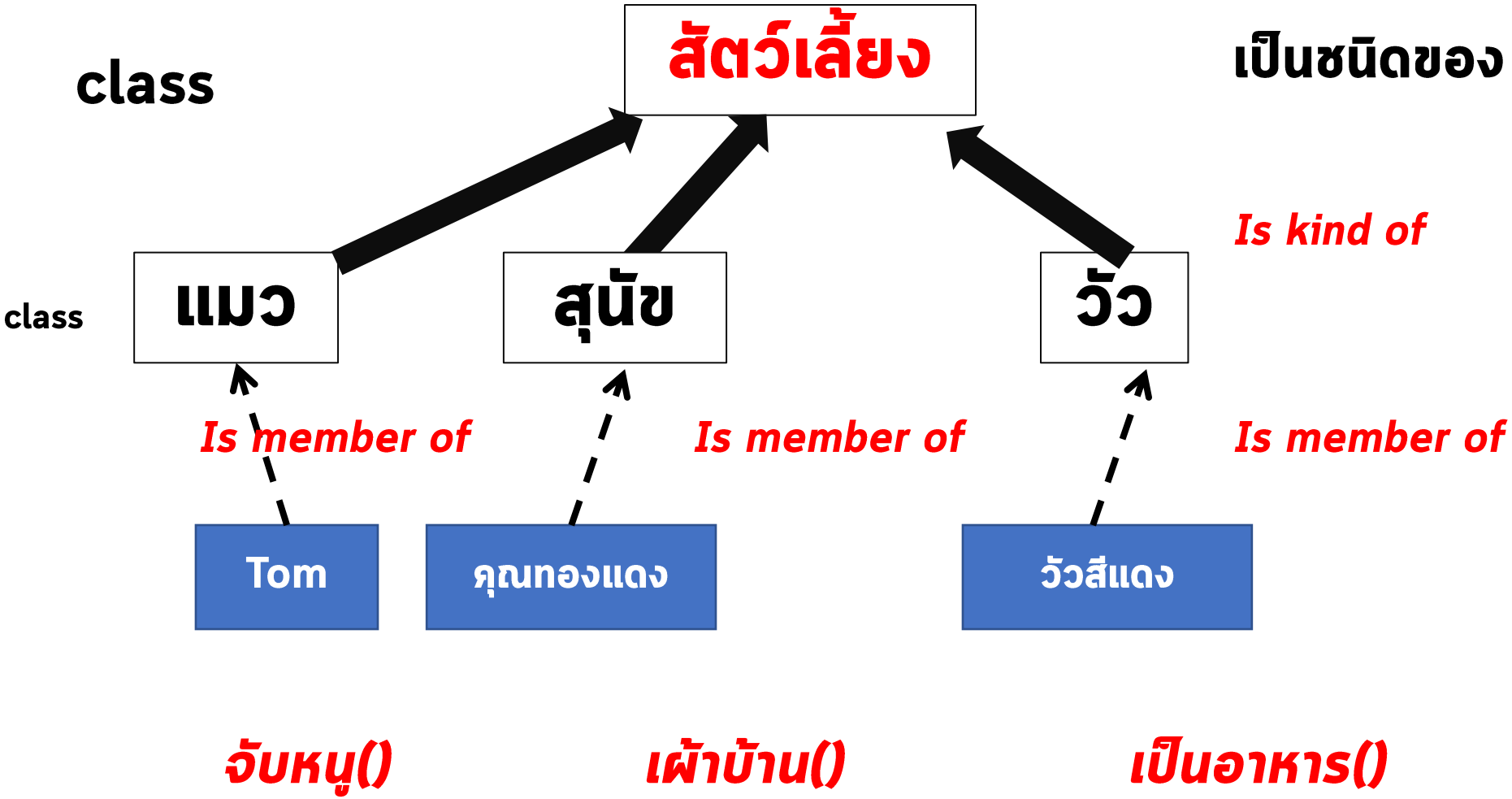
Animal

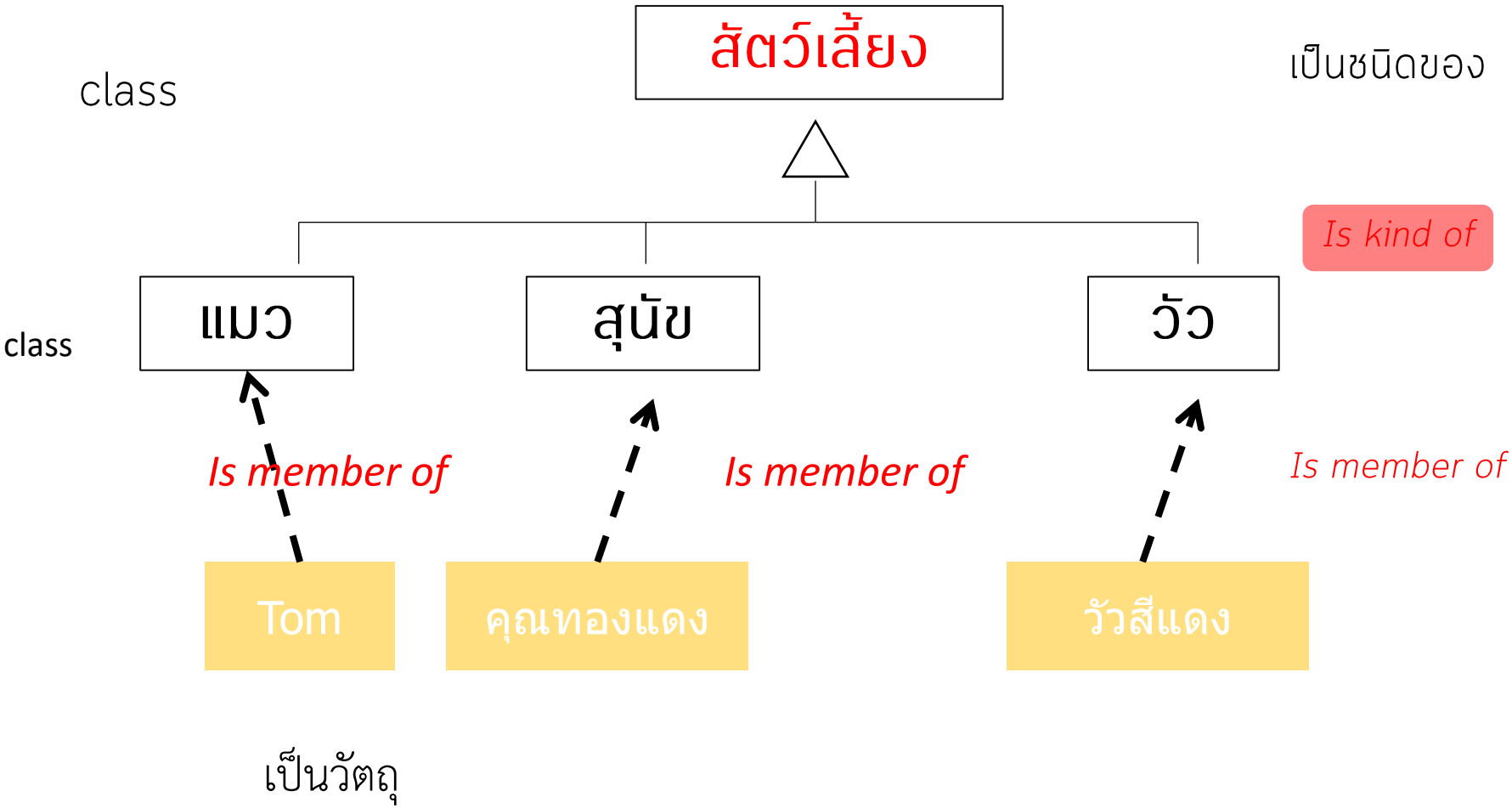
สัตว์เลี้ยง

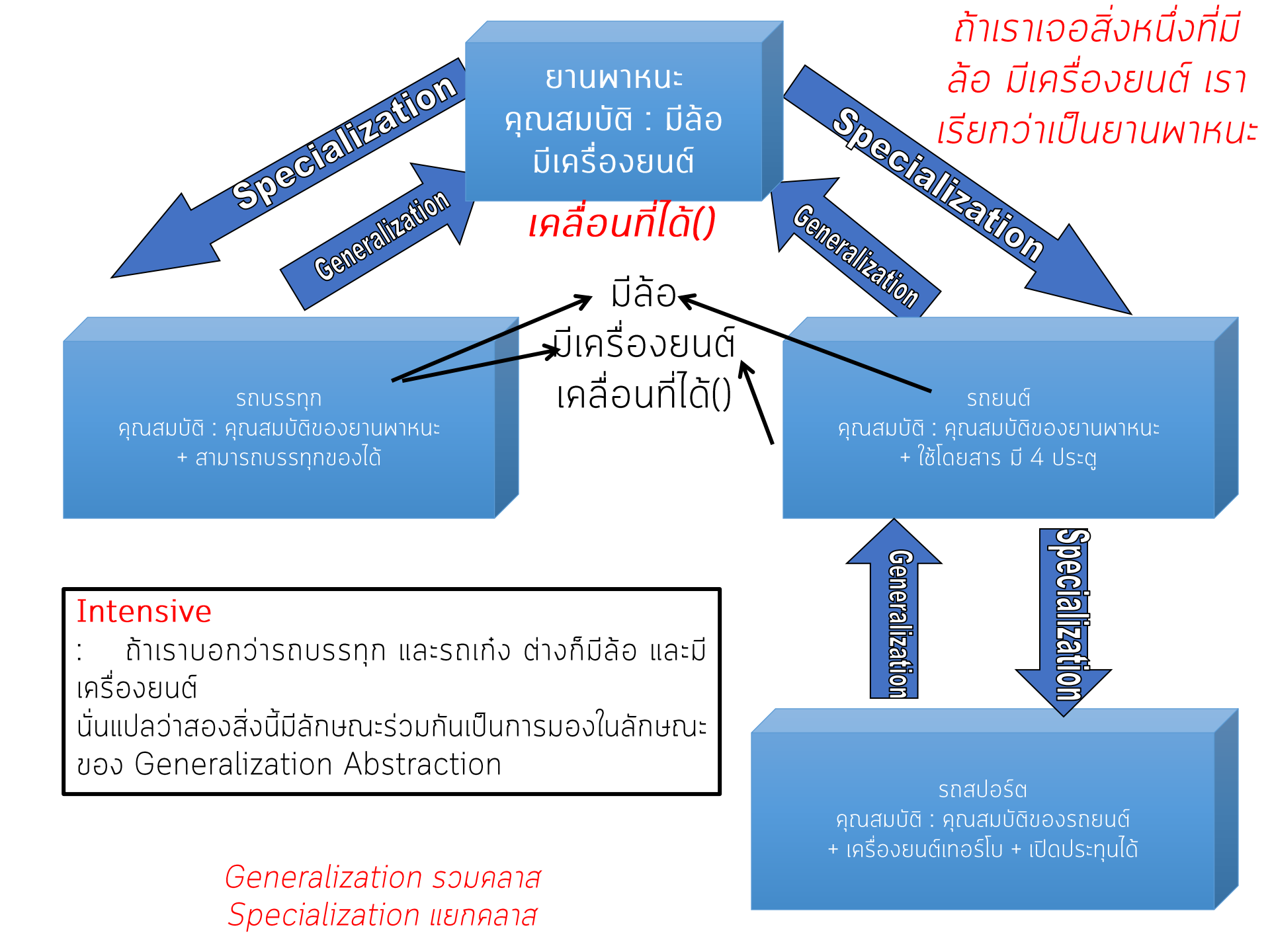
สี่ขา | หู | หาง | ... | กิน

กระทิง | แพะ | สุนัข | แกะ | ช้าง


class
สัตว์เลี้ยง
เป็นชนิดของ
Is kind of
class
แมว
สุนัข
วัว
Is member of
Is member of
Is member of
Tom
คุณทองแดง
วัวสีแดง
จับหนู()
เผ้าบ้าน()
เป็นอาหาร()


class
สัตว์เลี้ยง
เป็นชนิดของ
Is kind of
class
แมว
สุนัข
วัว
Is member of
Is member of
Is member of
Tom
คุณทองแดง
วัวสีแดง
เป็นวัตถุ



Intensive

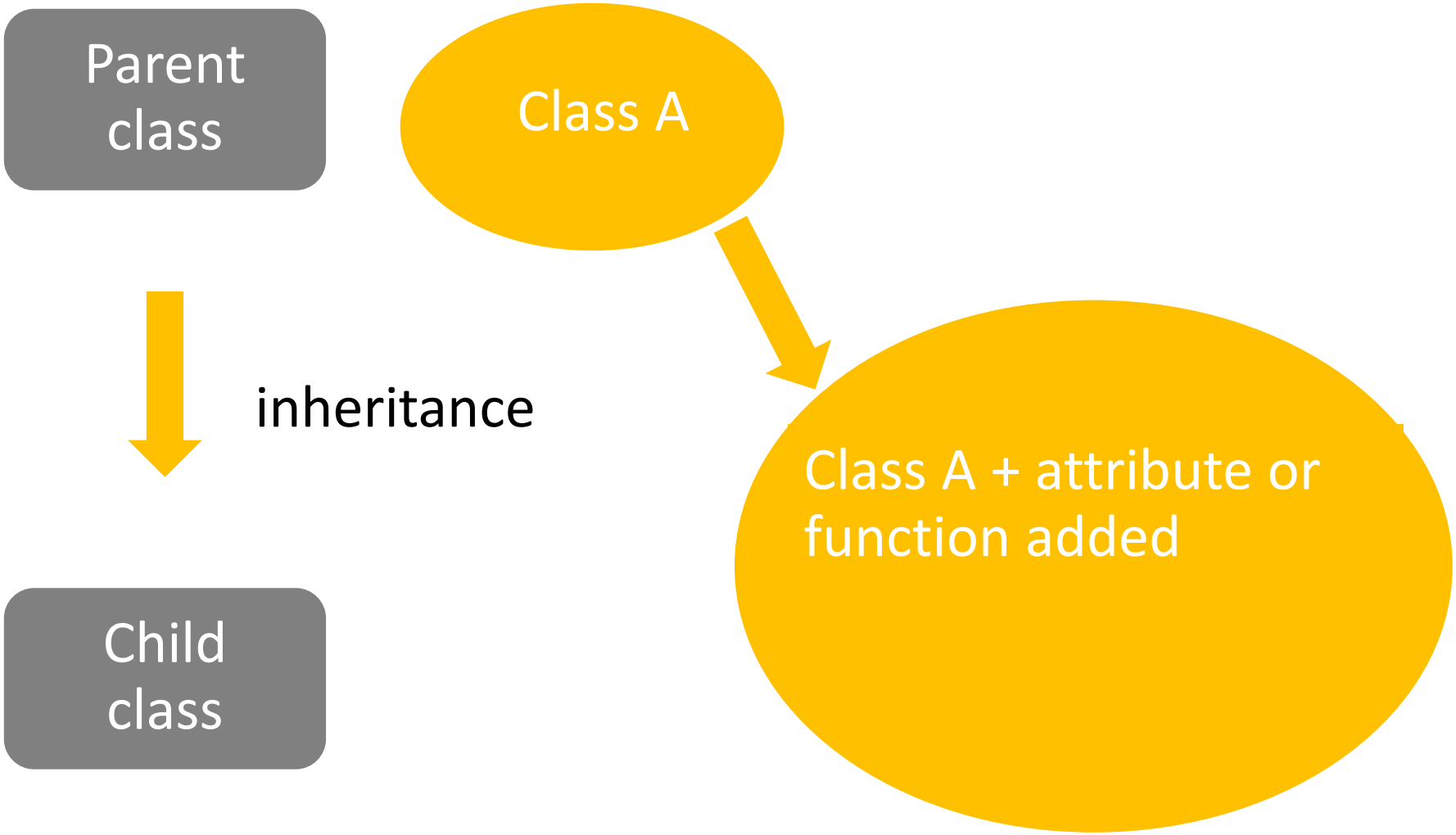
: ถ้าเราบอกว่ารถบรรทุก และรถเก๋ง ต่างก็มีล้อ และมีเครื่องยนต์

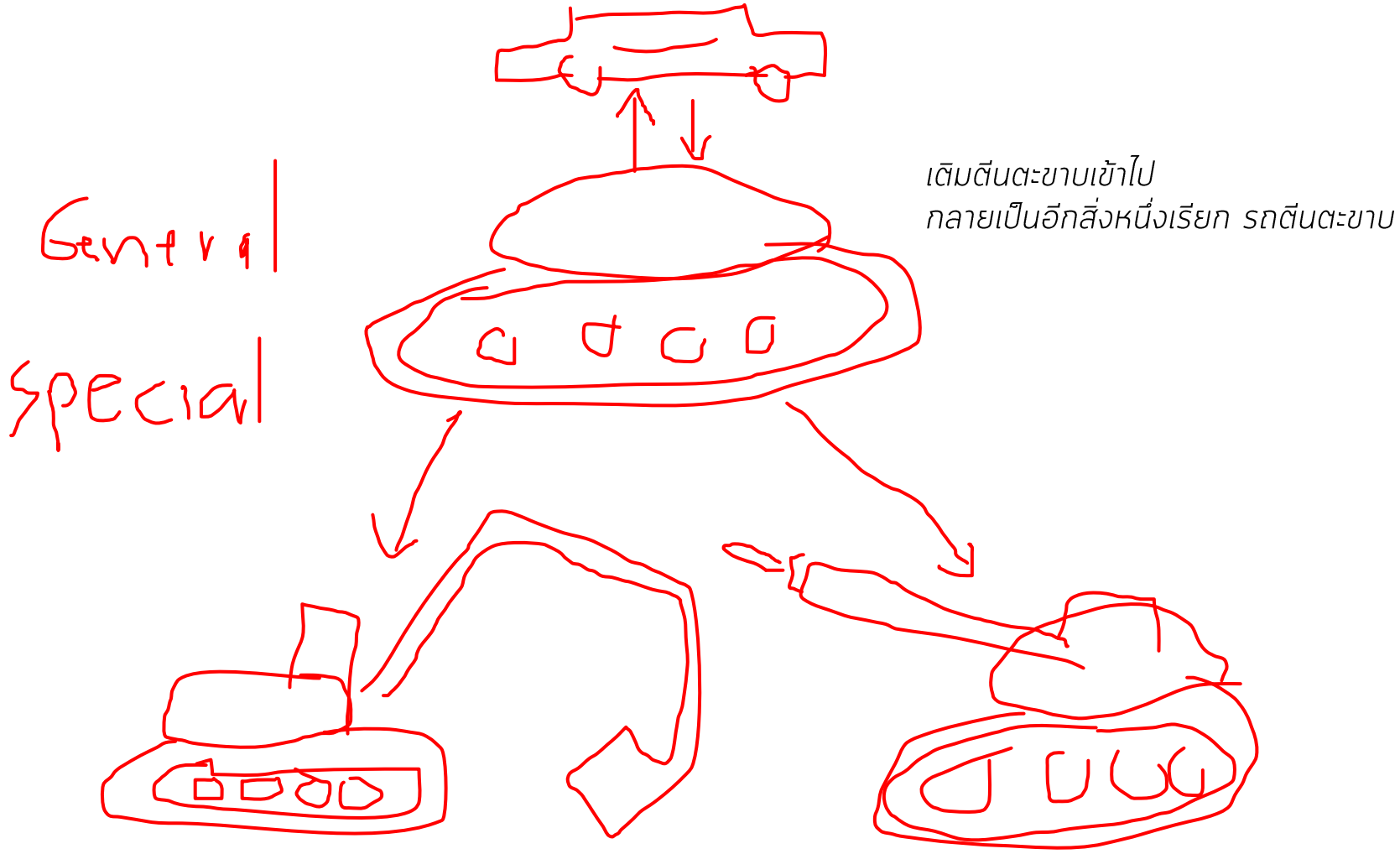
นั่นแปลว่าสองสิ่งนี้มีลักษณะร่วมกันเป็นการมองในลักษณะของ Generalization Abstraction

Generalization รวมคลาส
Specialization แยกคลาส

Generalization/Specialization

- การเติม/เพิ่ม คุณสมบัติหรือฟังก์ชัน คือ การทำ >> special
- ลดทอน คุณสมบัติหรือฟังก์ชัน คือ การทำ >> general


Parent class
Class A
inheritance
Class A + attribute or function added
Child class

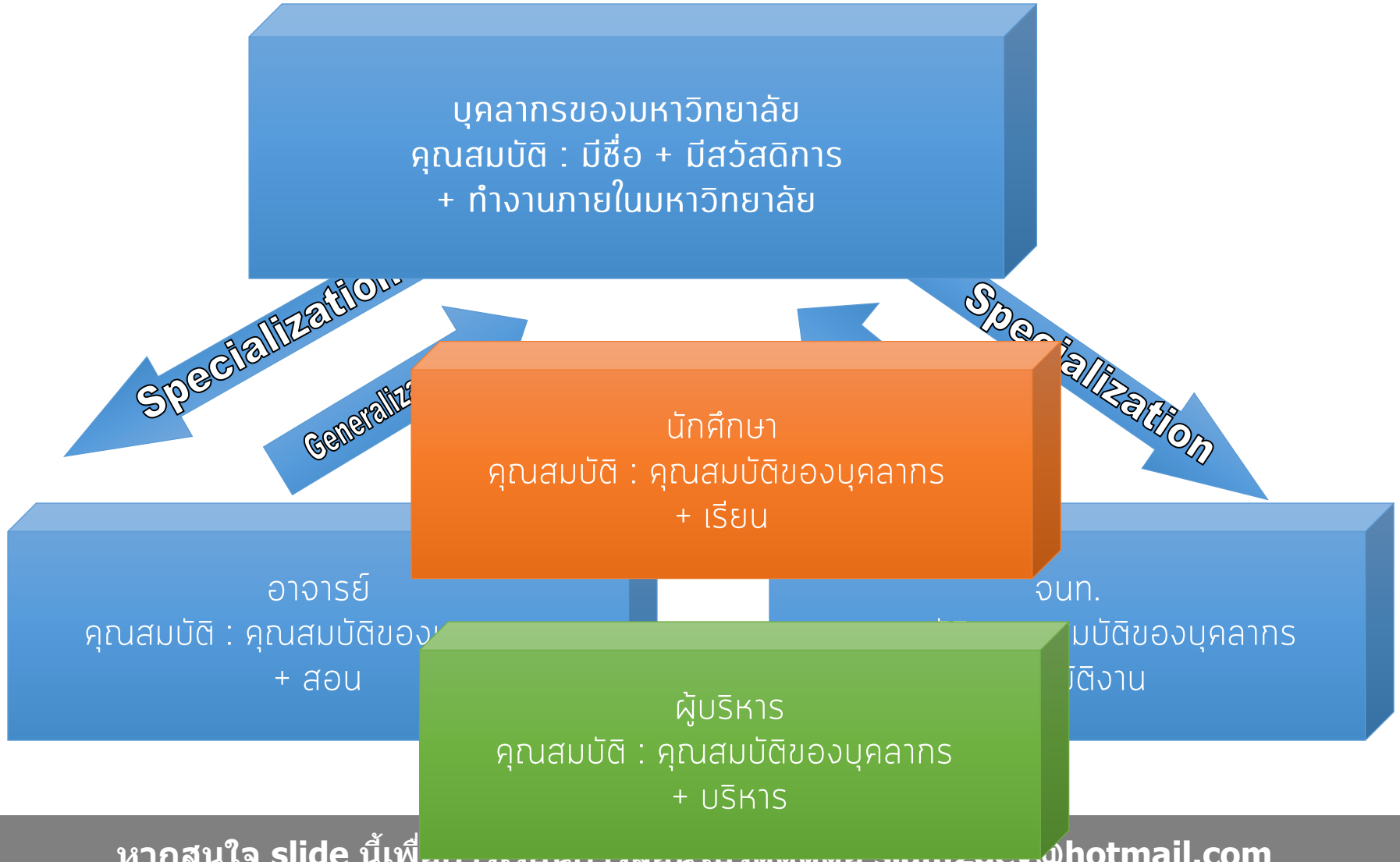


เติมตีนตะขาบเข้าไป
กลายเป็นอีกสิ่งหนึ่งเรียก รถตีนตะขาบ

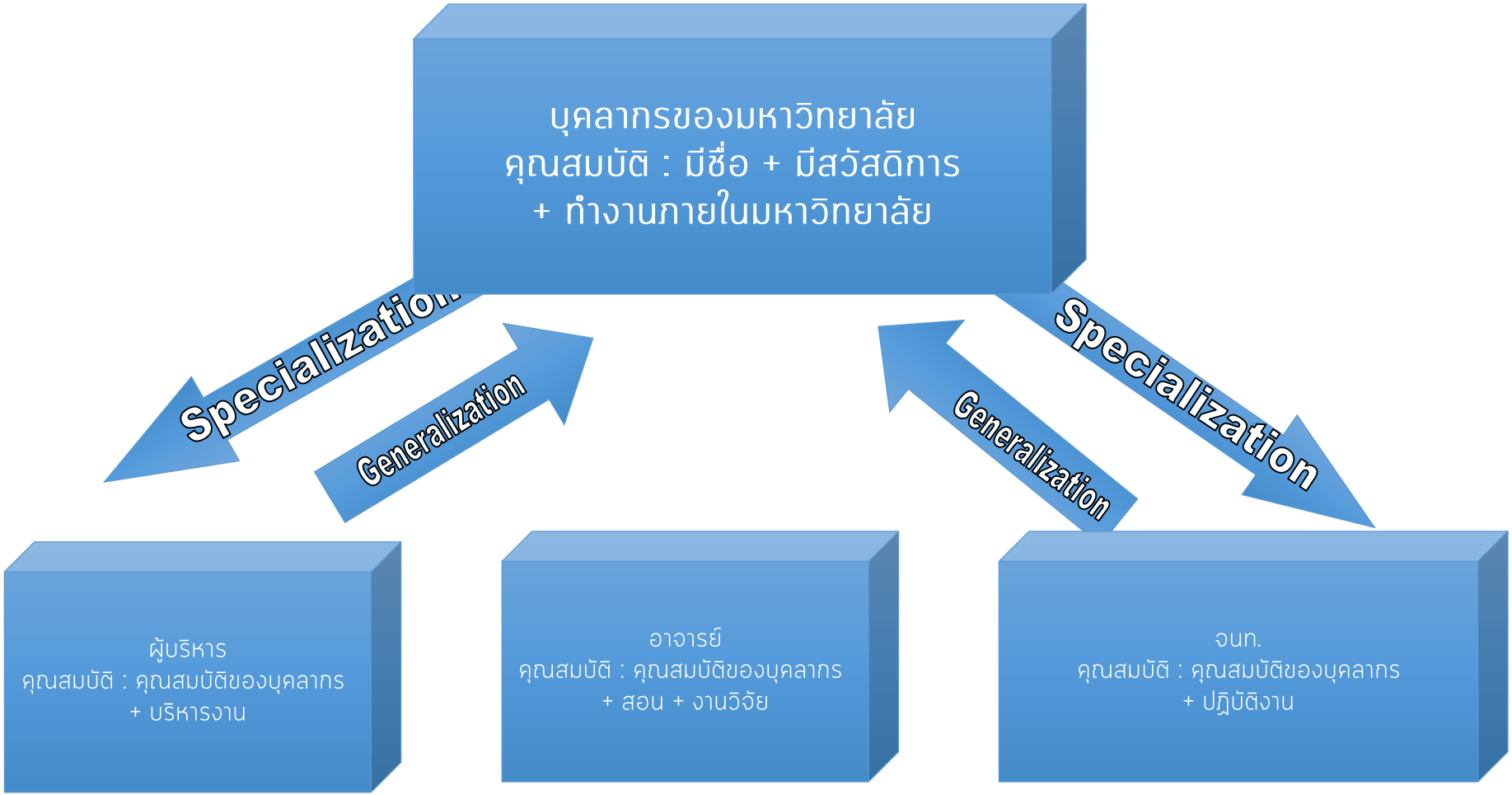
เติมตีนตะขาบเข้าไป
กลายเป็นอีกสิ่งหนึ่งเรียก รถตีนตะขาบ
+ เติมที่ตักดิน / ขุดดินเข้าไป

เติมตีนตะขาบเข้าไป
กลายเป็นอีกสิ่งหนึ่งเรียก รถตีนตะขาบ
+ เติมปืนใหญ่ เข้าไป กลาย เป็น รถถัง

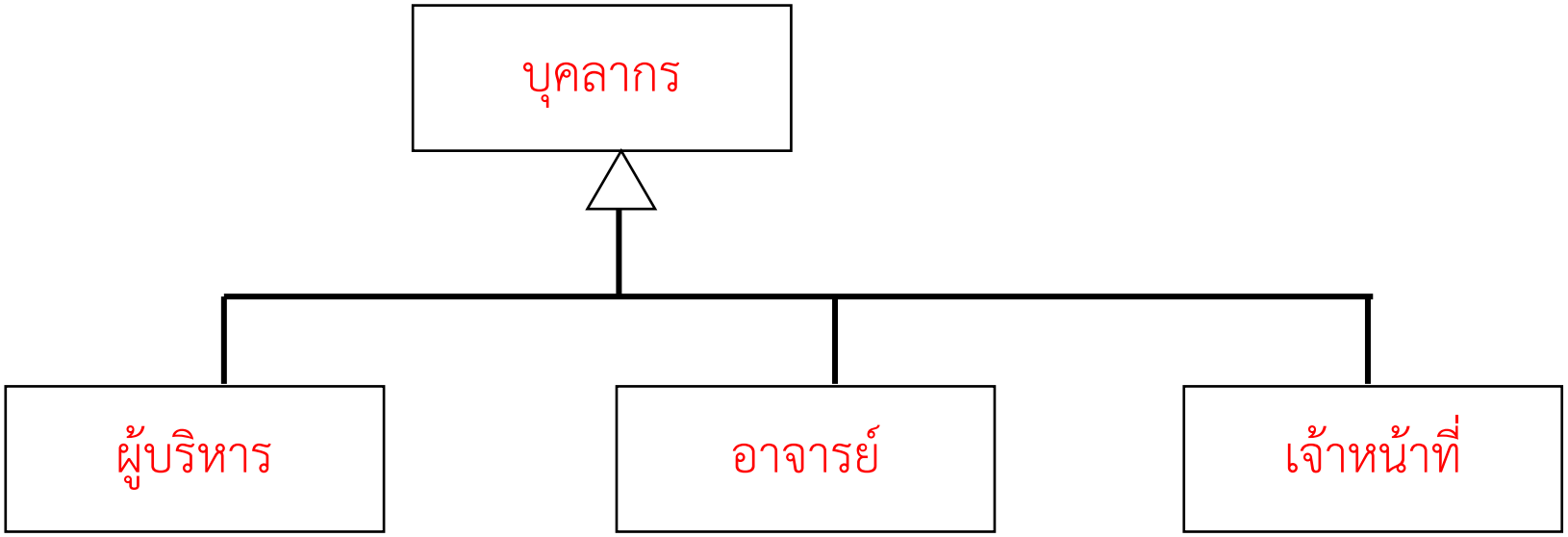
จงอธิบายความหมายของภาพที่กำหนดให้ในเชิงของ Generalization Abstraction



จงอธิบายความหมายของภาพที่กำหนดให้ในเชิงของ Generalization Abstraction

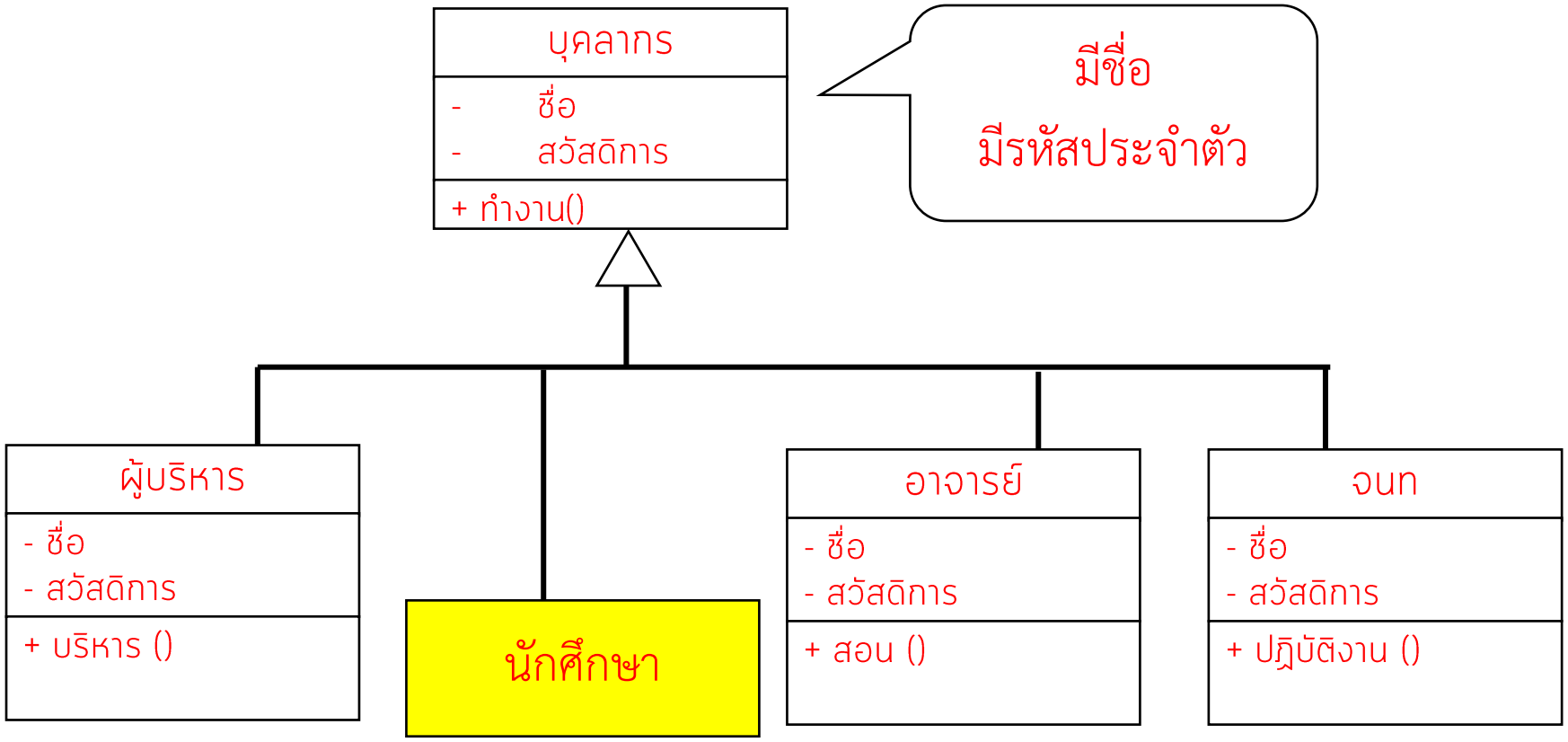


แผนภาพ Generalization Abstraction

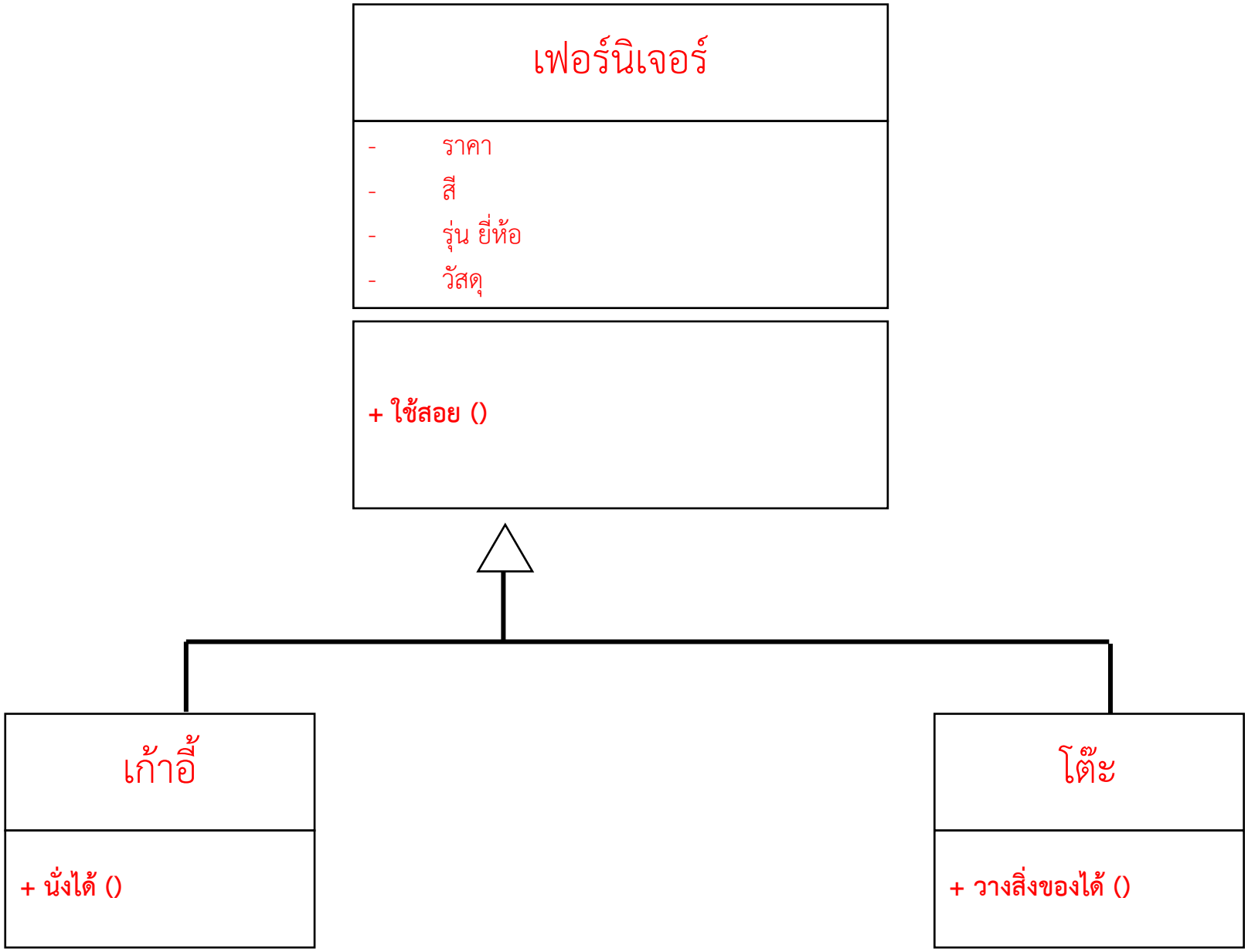


อธิบายได้ว่า บุคลากรของมหาวิทยาลัยแบ่งออกเป็น 3 ประเภท คือ อาจารย์ ผู้บราหร และเจ้าหน้าที่

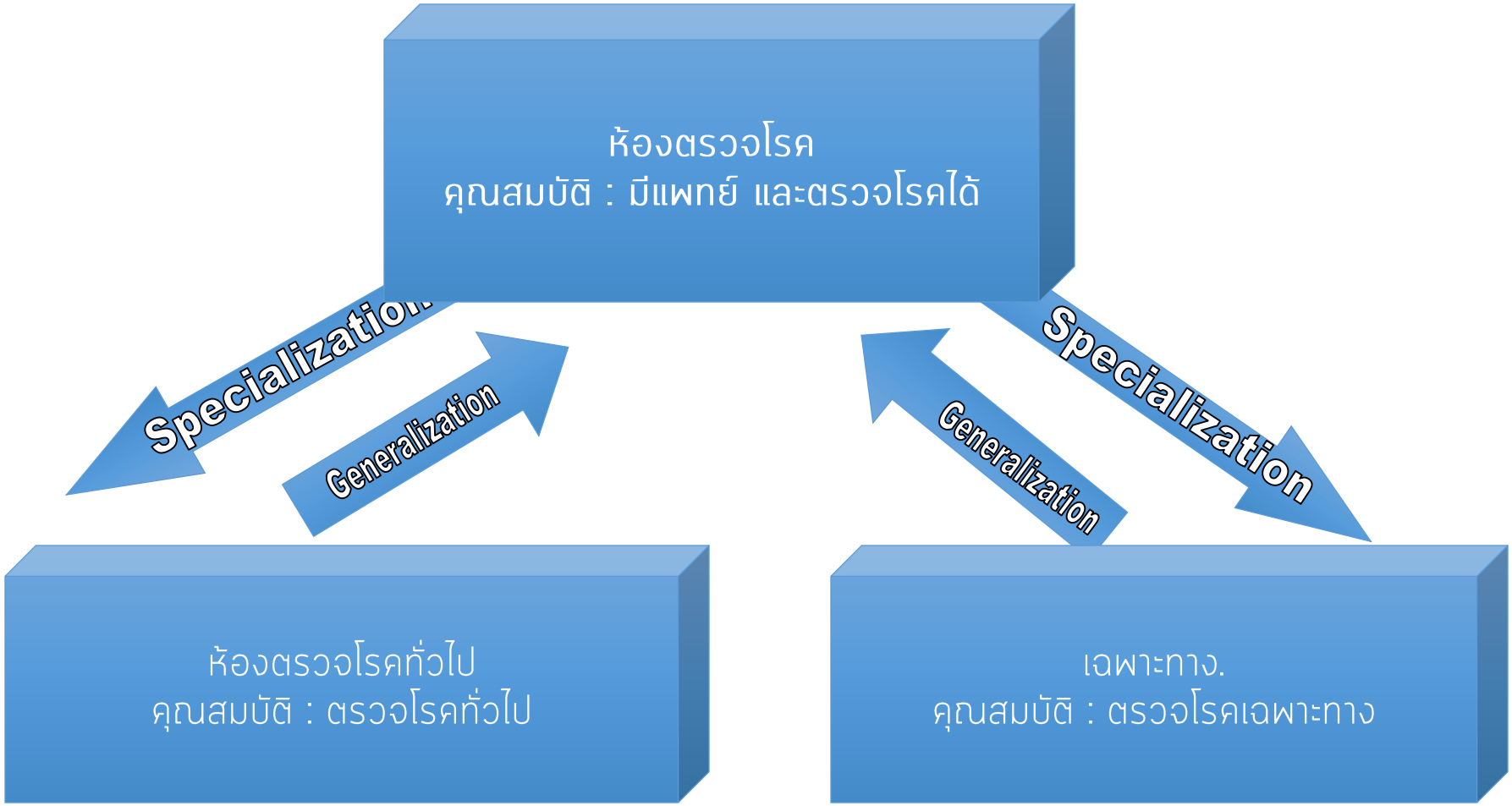
แผนภาพ Generalization Abstraction



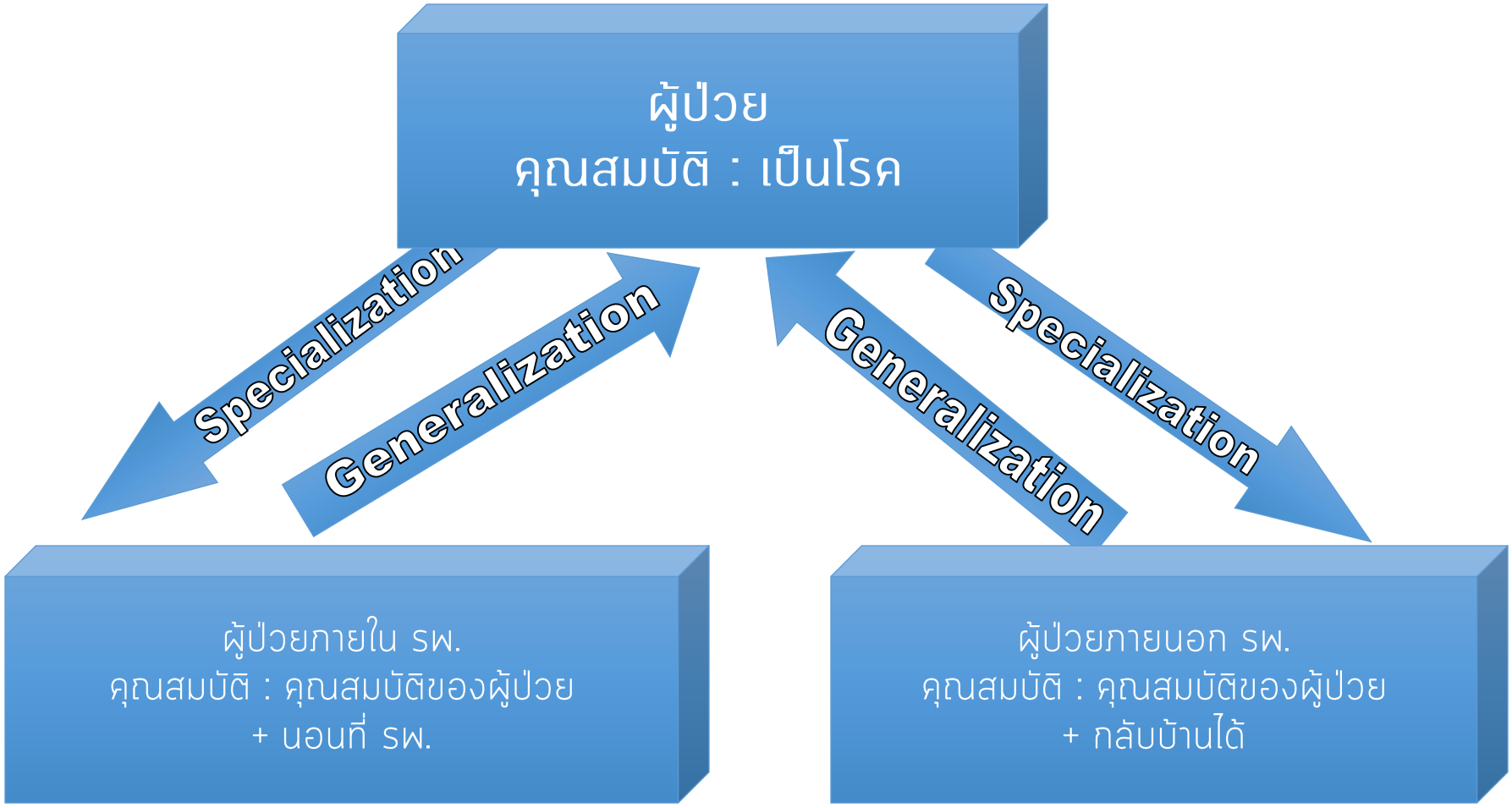
อธิบายได้ว่า บุคลากรของมหาวิทยาลัยแบ่งออกเป็น 3 ประเภท คือ อาจารย์ ผู้บราหร และเจ้าหน้าที่


เฟอร์นิเจอร์
- ราคา
- สี
- รุ่น ยี่ห้อ
- วัสดุ
+ ใช้สอย ()
เก้าอี้
+ นั่งได้ ()
โต๊ะ
+ วางสิ่งของได้ ()

จงอธิบายความหมายของภาพที่กำหนดให้ในเชิงของ Generalization Abstraction



จงอธิบายความหมายของภาพที่กำหนดให้ในเชิงของ Generalization Abstraction

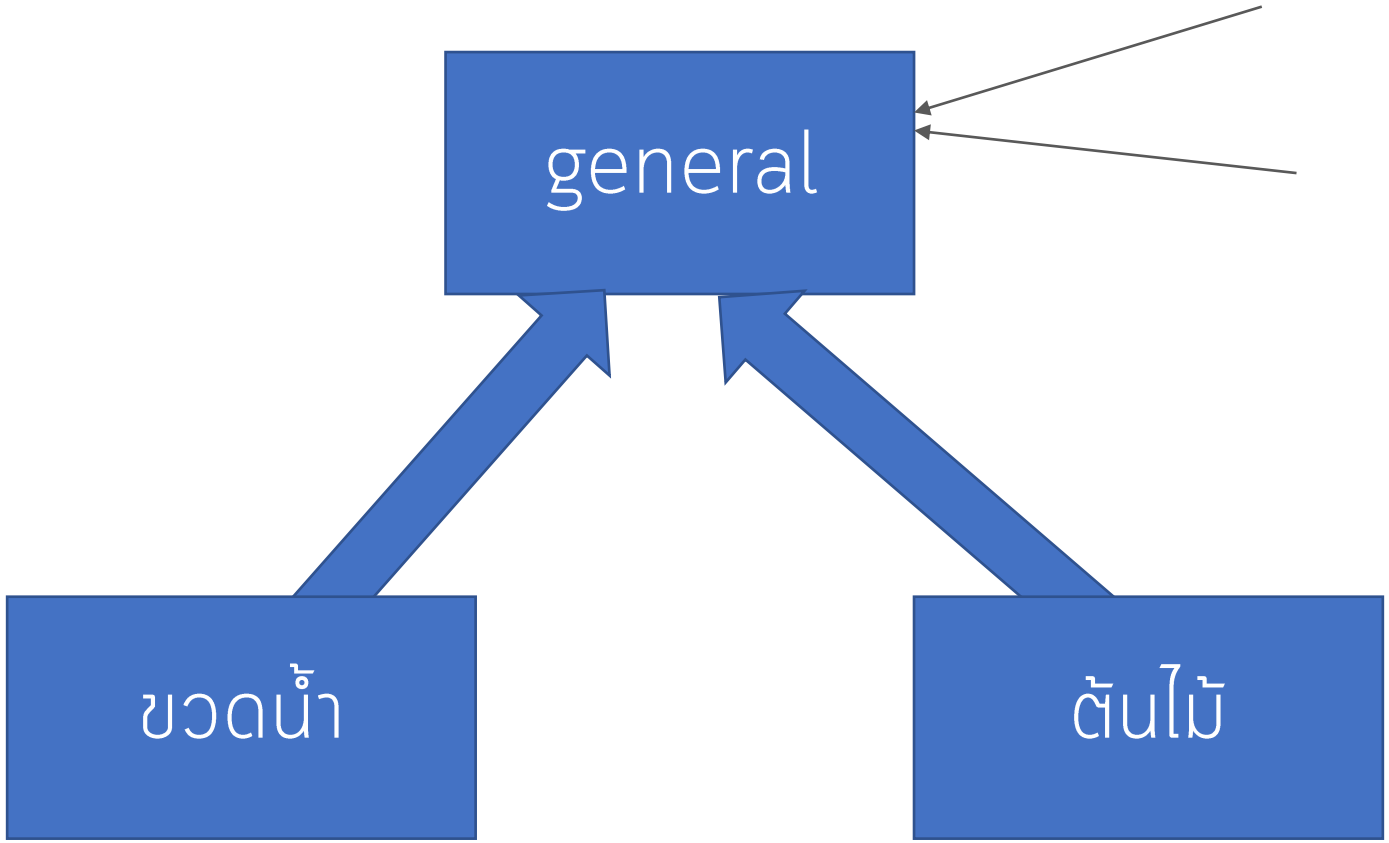


Generalization Abstraction

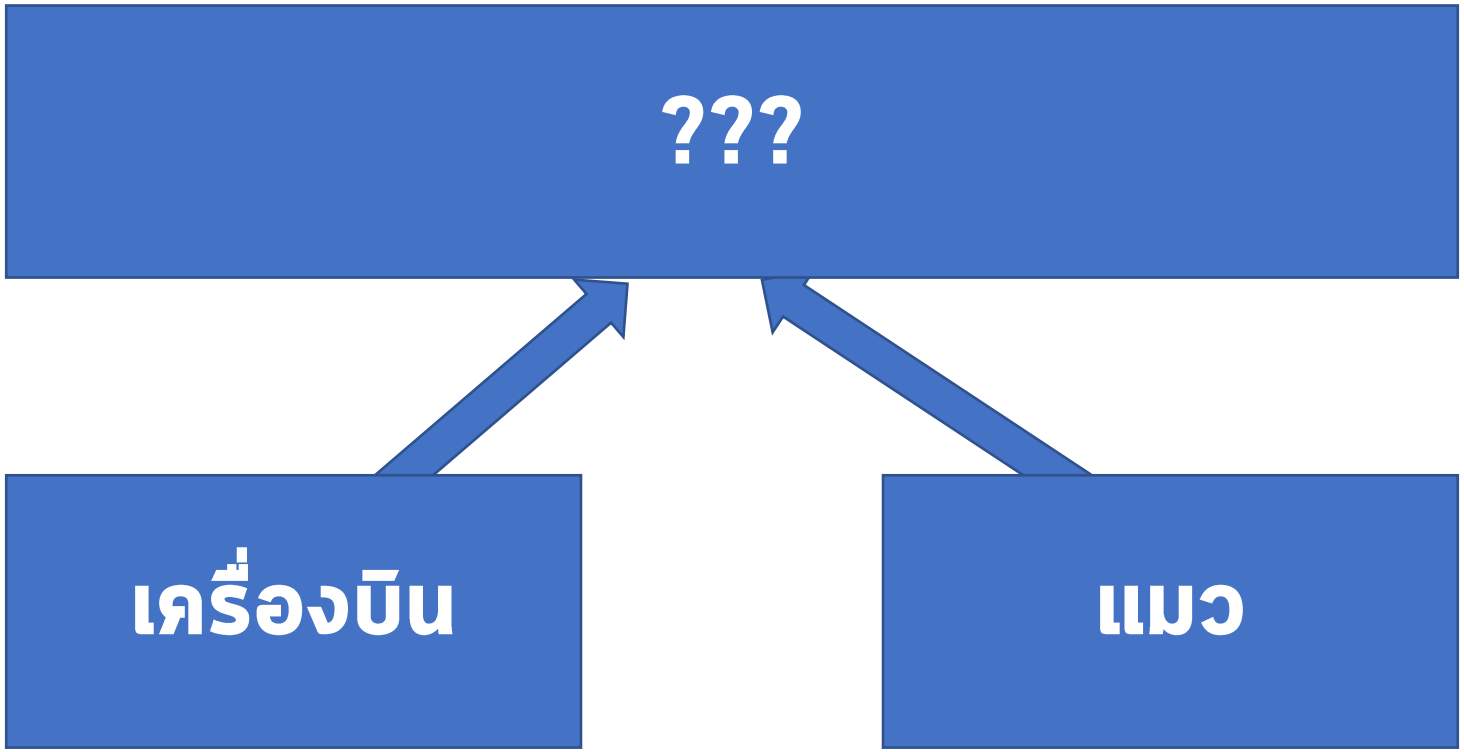
- **ข้อสังเกต**
  - **ถ้าเรานำคลาสย่อย หลายๆ คลาสแล้วมีคุณสมบัติร่วมกันได้ แล้วทำให้เกิดความคิดรวบยอดใหม่(concept) ได้ และจัดอยู่ในคลาสเดียวกัน เรียกว่า Generalization Abstraction**
  - **แต่ถ้ามีคุณสมบัติร่วมกัน แต่ไม่เกิดความคิดรวบยอดใหม่ ก็ไม่จำเป็นจะต้องรวมกันเป็นคลาสเดียว**

หากสนใจ slide นี้เพื่อการเรียนการสอนโปรดติดต่อ siam2dev@hotmail.com

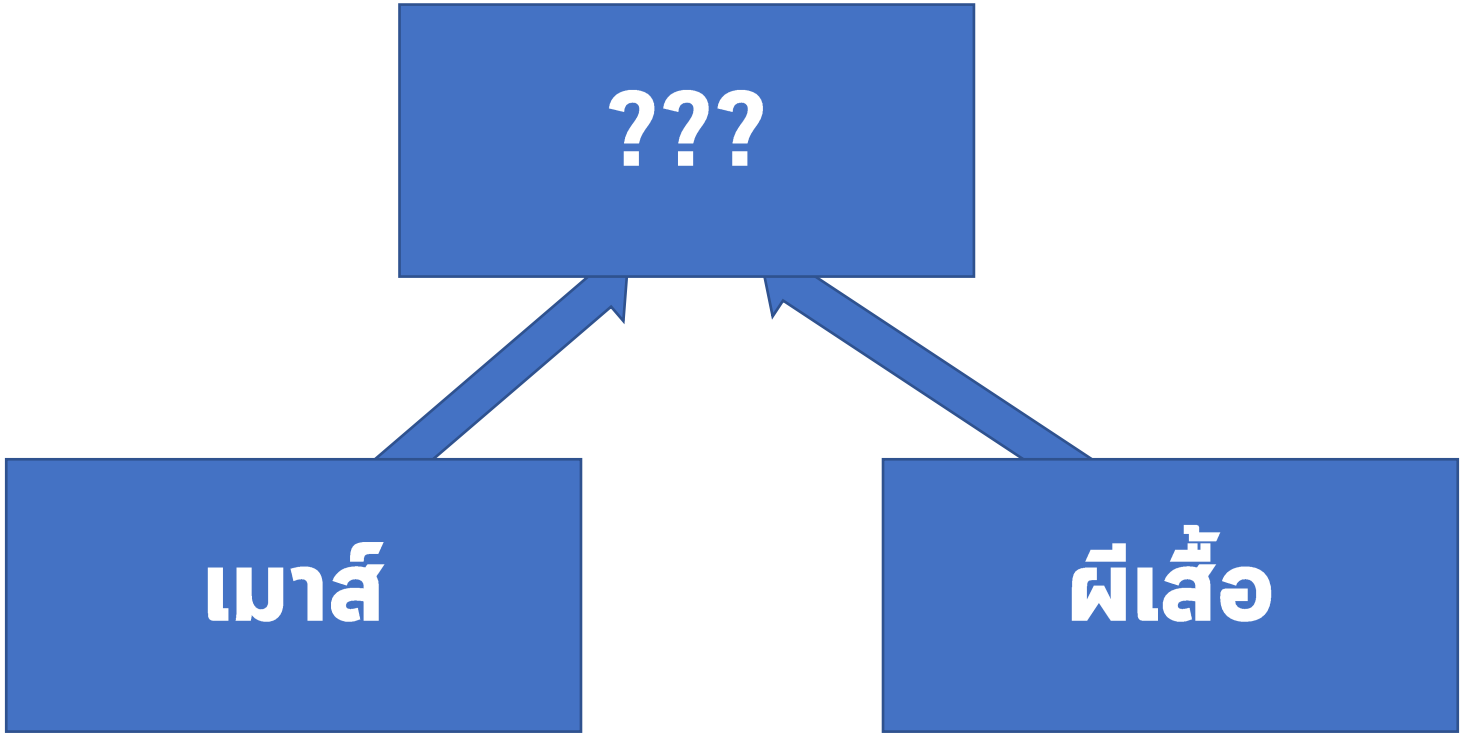
ไม่สามารถรวมหรือจัดให้เป็นประเภทได้



ควรเติมอะไรเข้าไป ใน ???

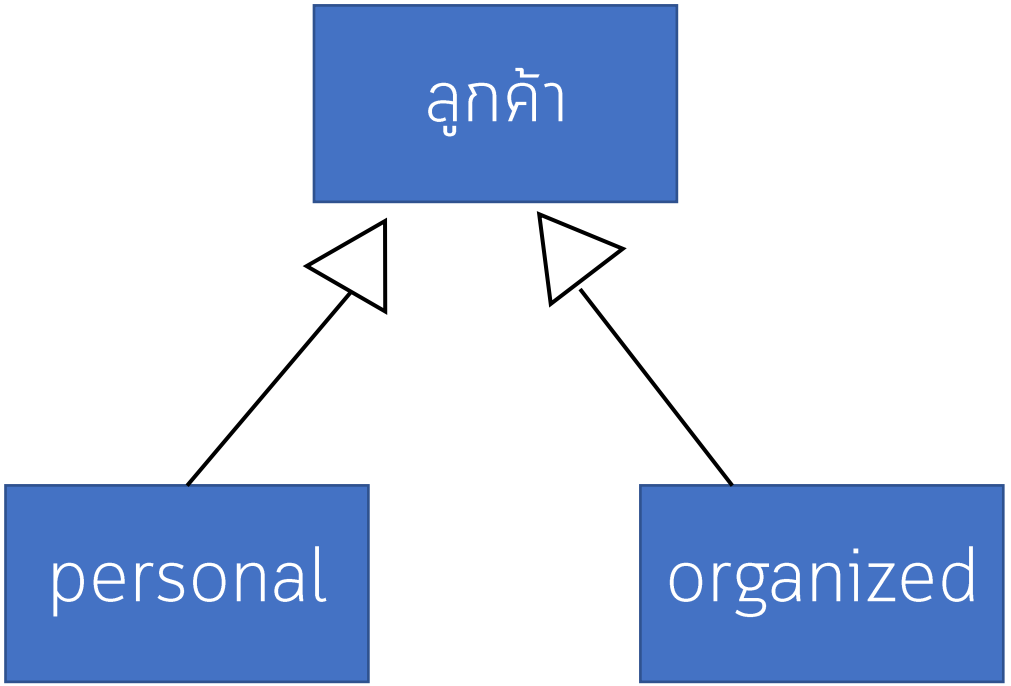


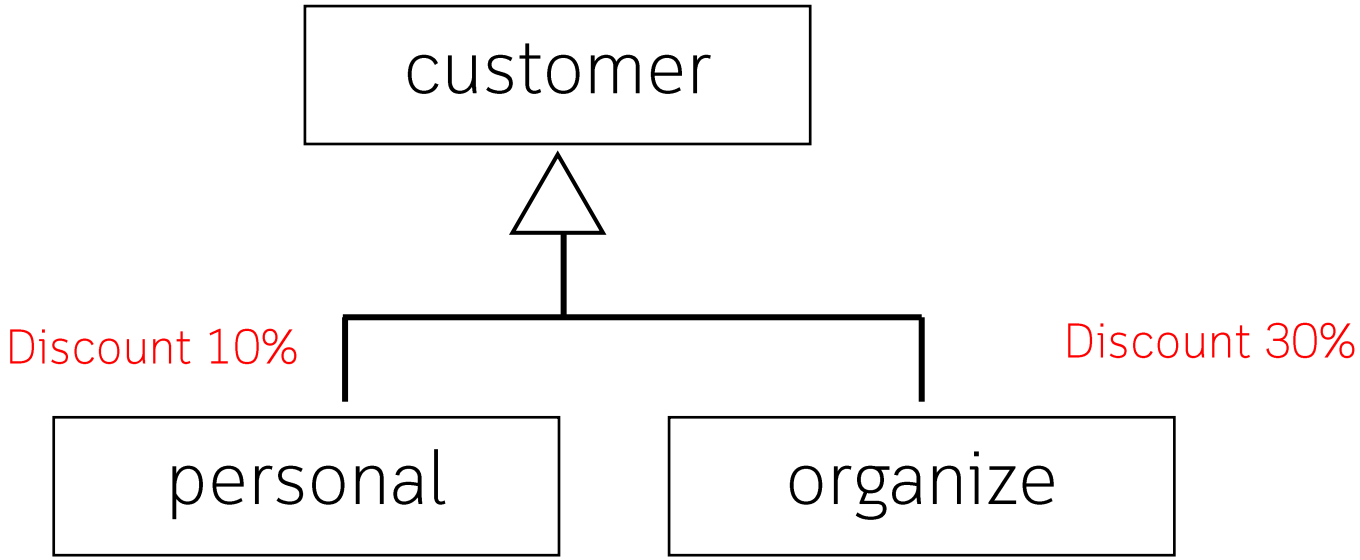
อย่าหาทำ ☺



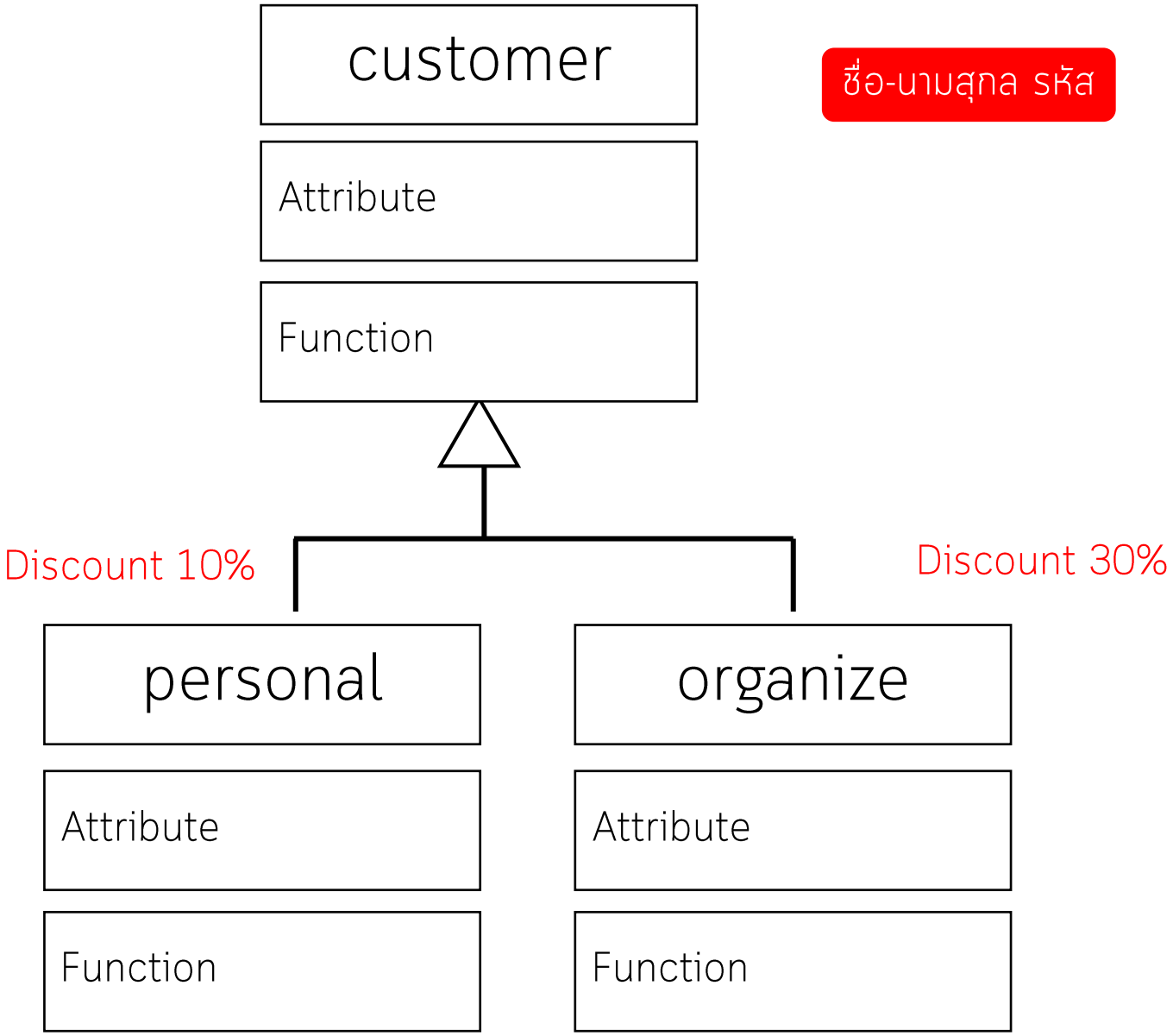
จากรูปนี้ถ้าให้อธิบายโดยใช้ generalization/specialization
จะทำได้หรือไม่อย่างไร
ถ้าทำได้จะเป็นคลาสอะไร จงอธิบายพร้อมให้เหตุผล

แผนภาพ Generalization Abstraction


ลูกค้า
personal
organized

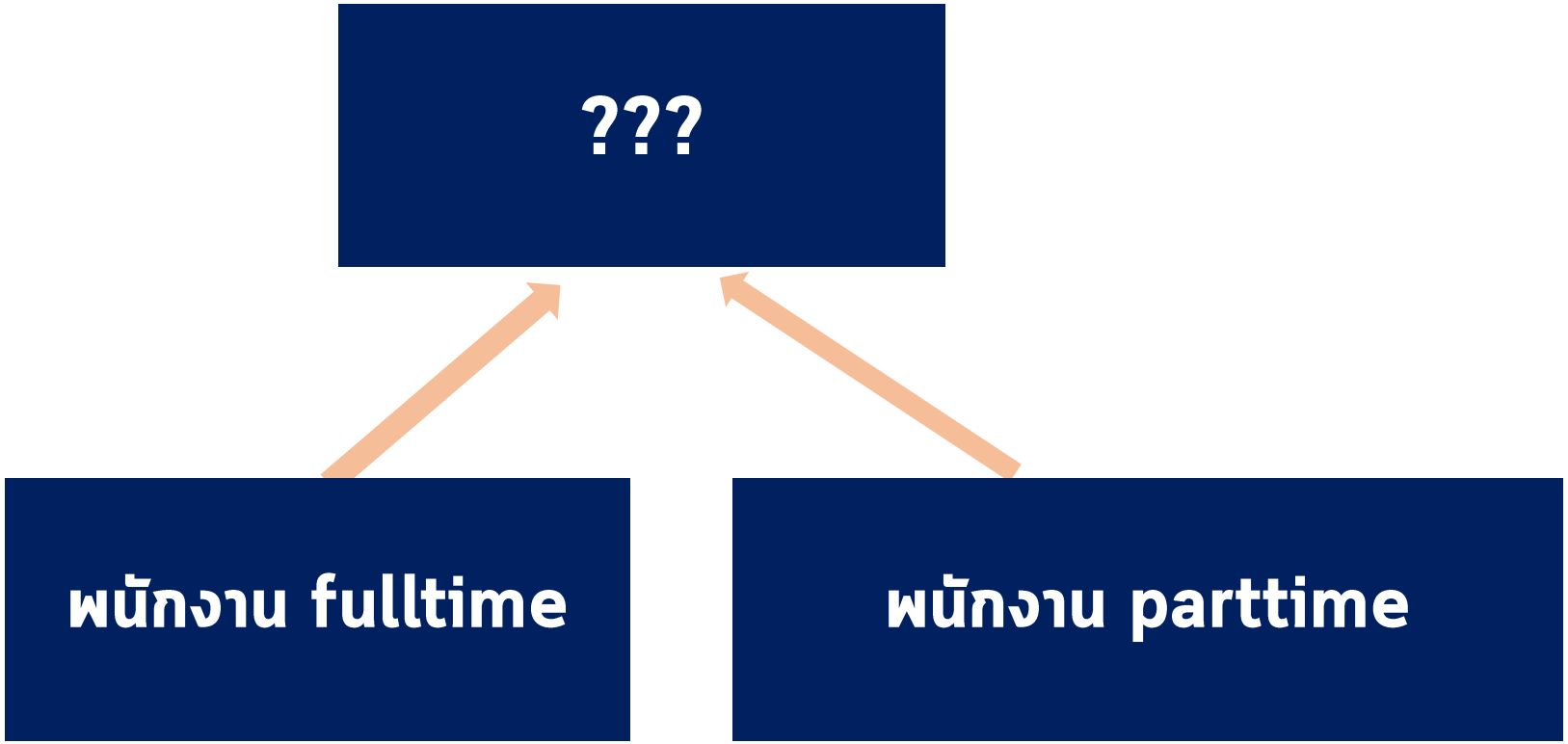

customer
Discount 10%
Discount 30%
personal
organize

จงเติม attribute และ Function ให้ครบทุกคลาส



จงเติมคำในช่องว่าง ???

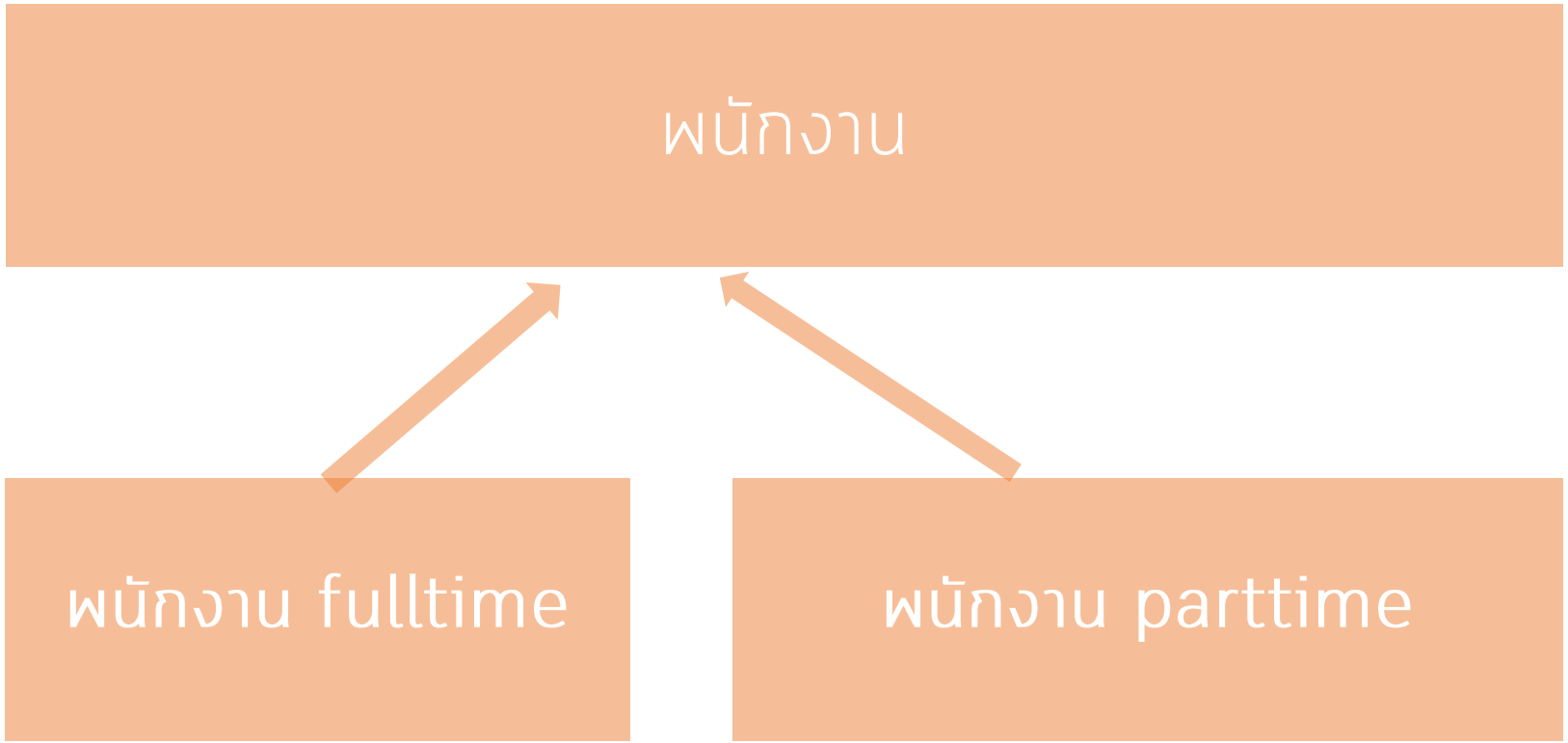
อธิบายแผนภาพนี้



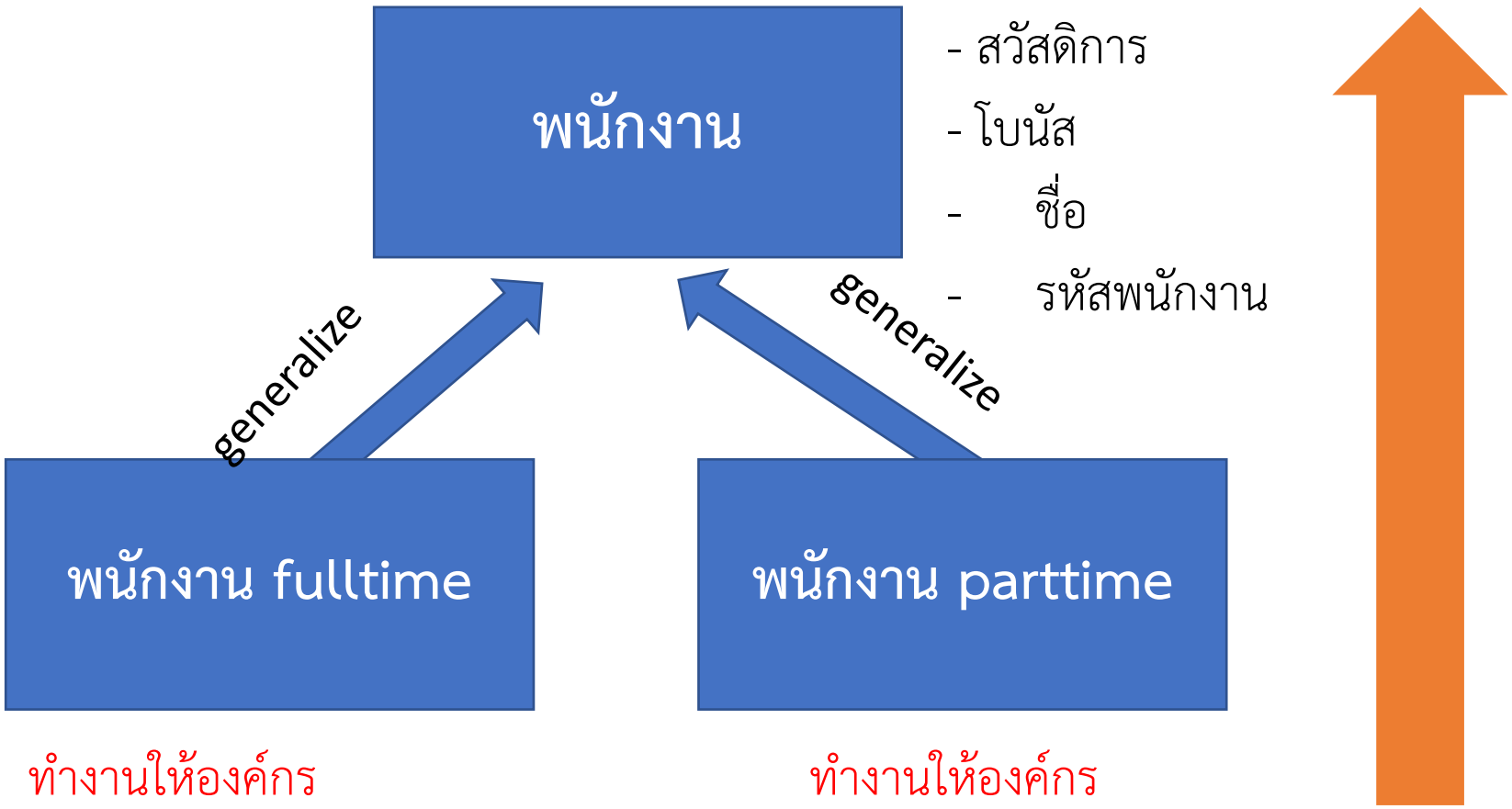
จากรูปนี้ถ้าให้อธิบายโดยใช้ generalization/specialization
จะทำได้หรือไม่อย่างไร
ถ้าทำได้จะเป็นคลาสอะไร จงอธิบายพร้อมให้เหตุผล

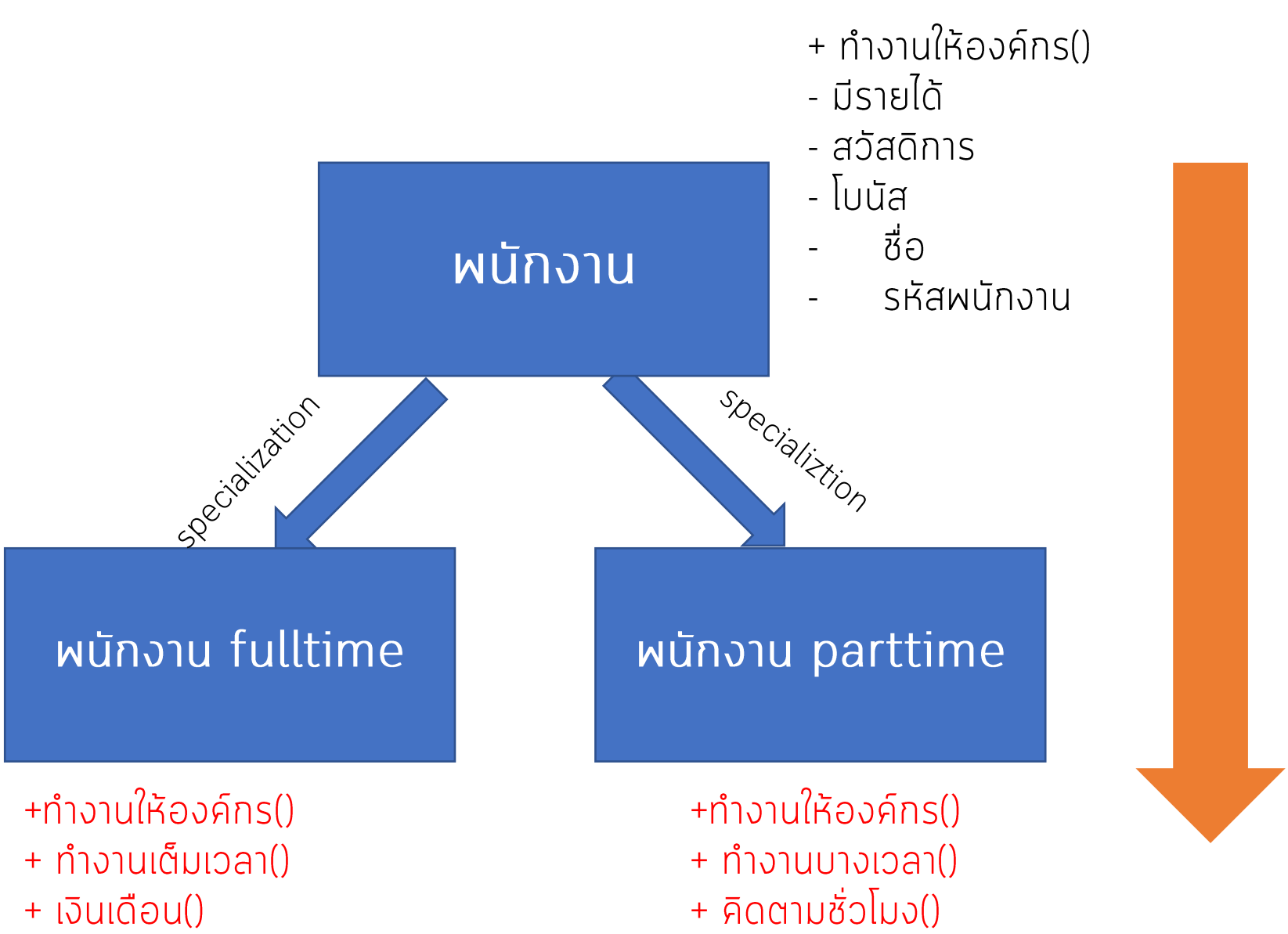
จงเติมคำในช่องว่าง ???

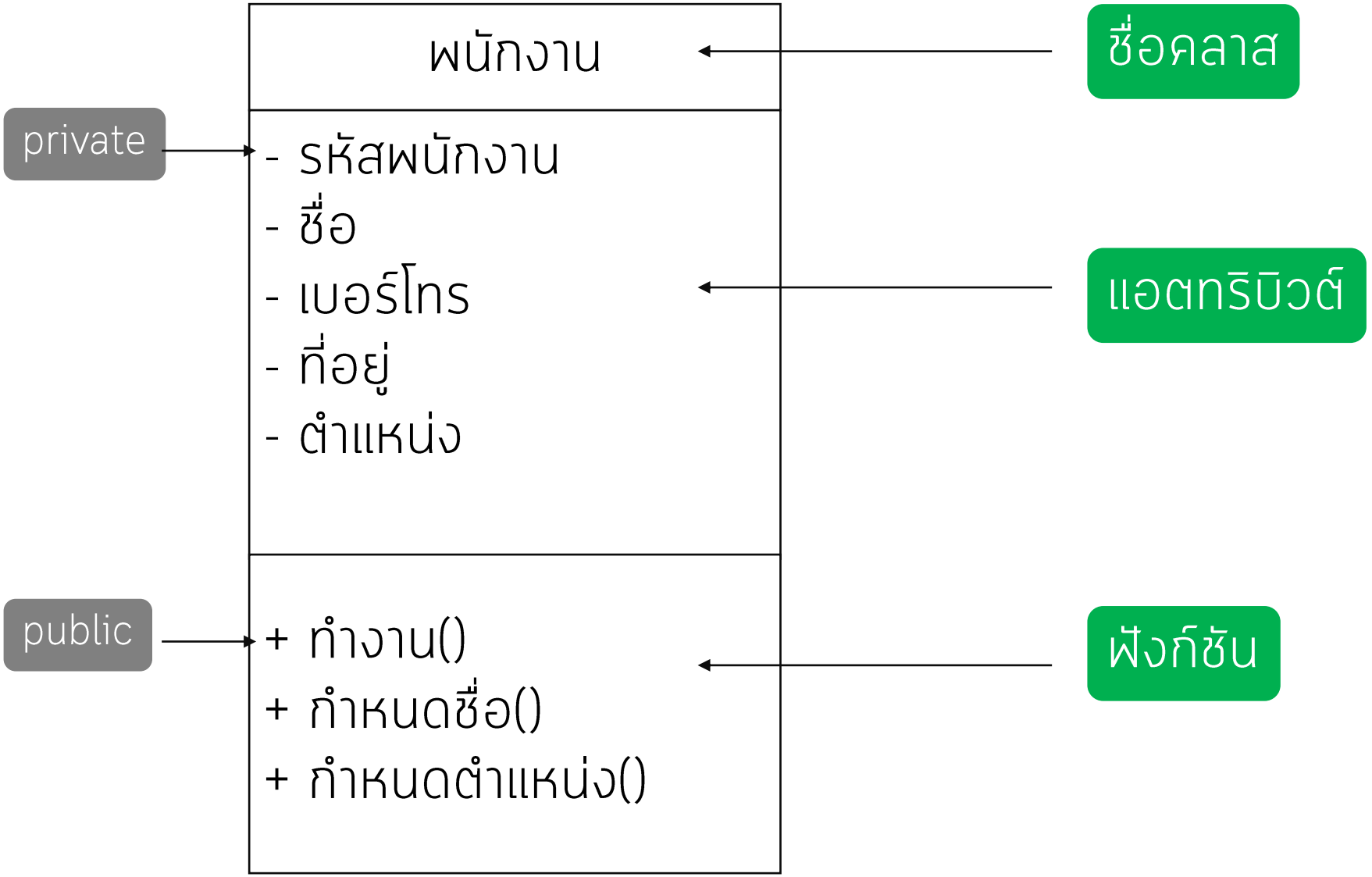
จากแผนภาพนี้ อธิบายได้ว่า พนักงานแบ่งออกเป็น 2 ประเภทคือพนักงานเต็มเวลา กับพนักงานพาร์ทไทม์ โดยทั้งพนักงานเต็มเวลาและพาร์ทไทม์ ต่างก็มีชื่อ มีเพศ มีอายุ และรับเงินและทำงานให้ บ. เหมือนกัน ถ้ากันตรงที่พนักงานเต็มเวลาจะได้รับเงินเดือน แต่พนักงานพาร์ทไทม์ ได้รับเงินตาม ชม. ที่ทำงาน



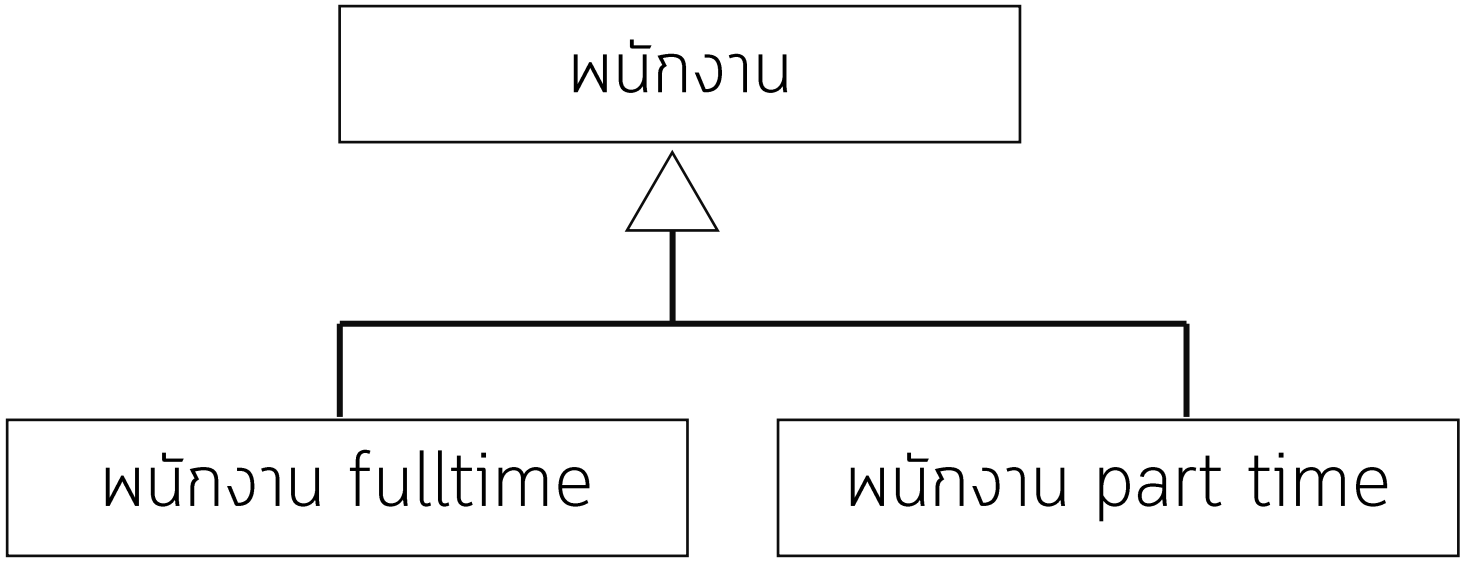
จากรูปนี้ถ้าให้อธิบายโดยใช้ generalization/specialization
จะทำได้หรือไม่อย่างไร
ถ้าทำได้จะเป็นคลาสอะไร จงอธิบายพร้อมให้เหตุผล


+ ทำงานให้องค์กร()
- มีรายได้
- สวัสดิการ
- โบนัส
- ชื่อ
- รหัสพนักงาน
พนักงาน
generalize
generalize
พนักงาน fulltime
พนักงาน parttime
ทำงานให้องค์กร
ทำงานเต็มเวลา
เงินเดือน
ทำงานให้องค์กร
ทำงานบางเวลา
คิดตามชั่วโมง

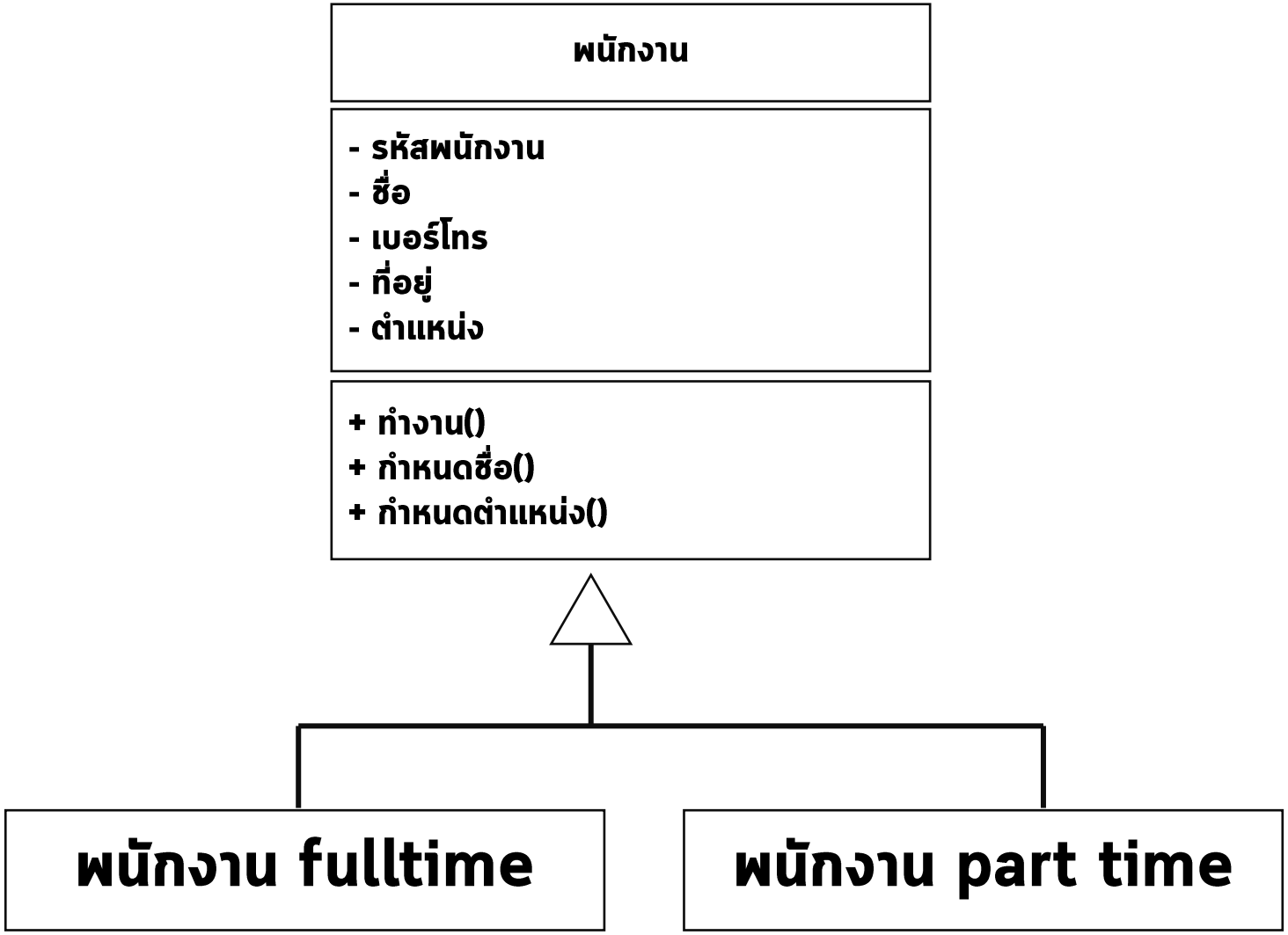

+ ทำงานให้องค์กร()
- มีรายได้
- สวัสดิการ
- โบนัส
- ชื่อ
- รหัสพนักงาน
พนักงาน
specialization
specializtion
พนักงาน fulltime
พนักงาน parttime
+ทำงานให้องค์กร()
+ ทำงานเต็มเวลา()
+ เงินเดือน()
+ทำงานให้องค์กร()
+ ทำงานบางเวลา()
+ คิดตามชั่วโมง()


พนักงาน
ชื่อคลาส
private
- รหัสพนักงาน
- ชื่อ
- เบอร์โทร
- ที่อยู่
- ตำแหน่ง
แอตทริบิวต์
public
+ ทำงาน()
+ กำหนดชื่อ()
+ กำหนดตำแหน่ง()
ฟังก์ชัน

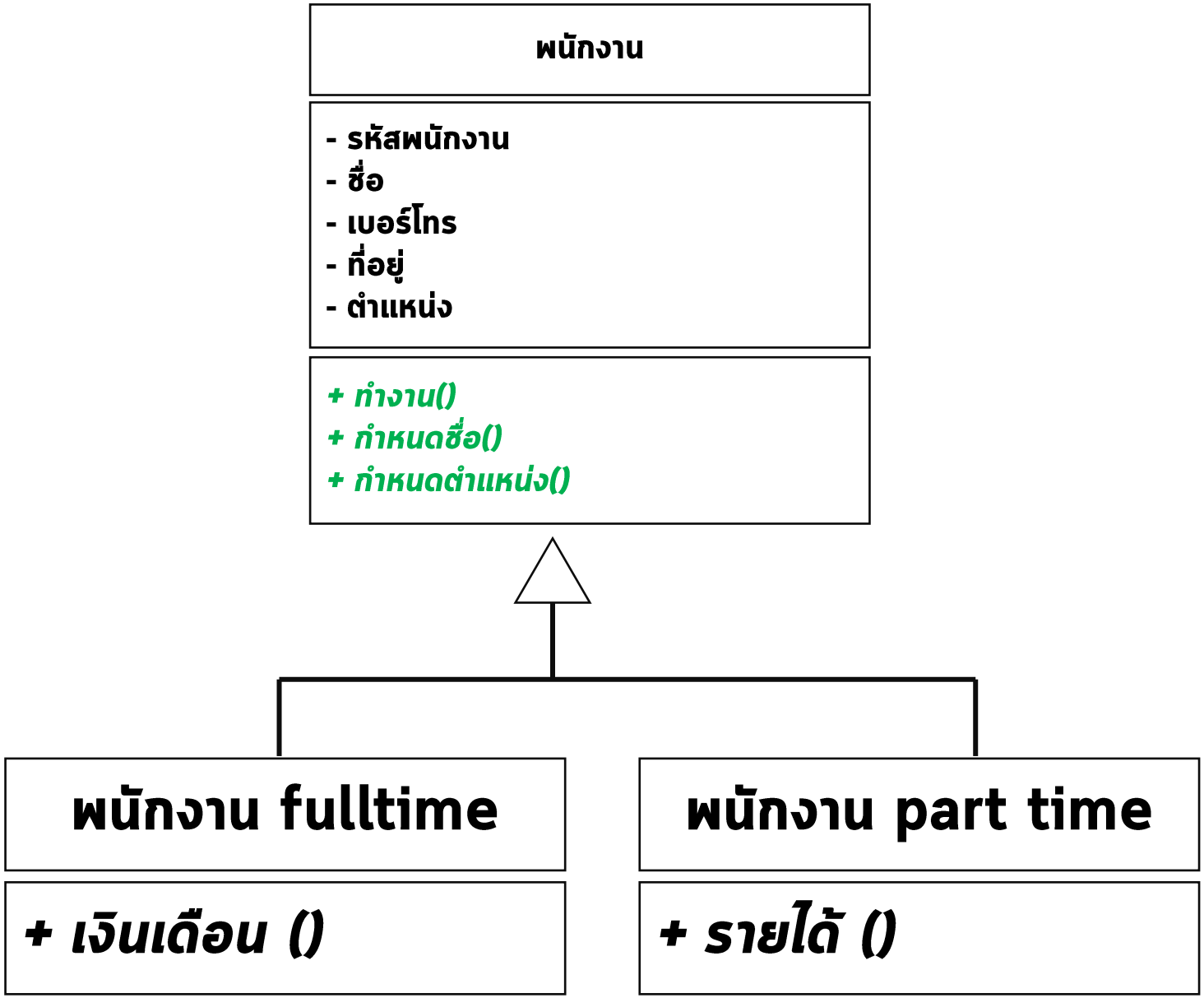
แผนภาพ Generalization Abstraction



แผนภาพ Generalization Abstraction



แผนภาพ Generalization Abstraction



จากตัวอย่างที่กำหนดให้ต่อไปนี้ จงพิจารณาว่าคลาส อะไรสามารถรวมกันได้ และถ้ารวมกันได้จะได้คลาสอะไร และคลาสใดที่รวมกันไม่ได้

1. คน , สัตว์ , พืช → _____
2. ผู้หญิง , ผู้ชาย → _____
3. สามกีบ , สลิ่ม → _____
4. แมว , สุนัข , ไก่ → _____
5. โทรศัพท์มือถือ , คอมพิวเตอร์ , โทรทัศน์ , ตู้เย็น , หม้อหุงข้าว → _____

6. เสื้อ , กางเกง , ผ้าขาวม้า , หมวก → _____
7. โต๊ะ , เก้าอี้ , กระดานดำ , อาจารย์ , นักศึกษา → _____

สิ่งมีชีวิต

คน

คนไทย

สัตว์

สัตว์เลี้ยง

พืช

แมว

เครื่องใช้ไฟฟ้า

เครื่องแต่งกาย

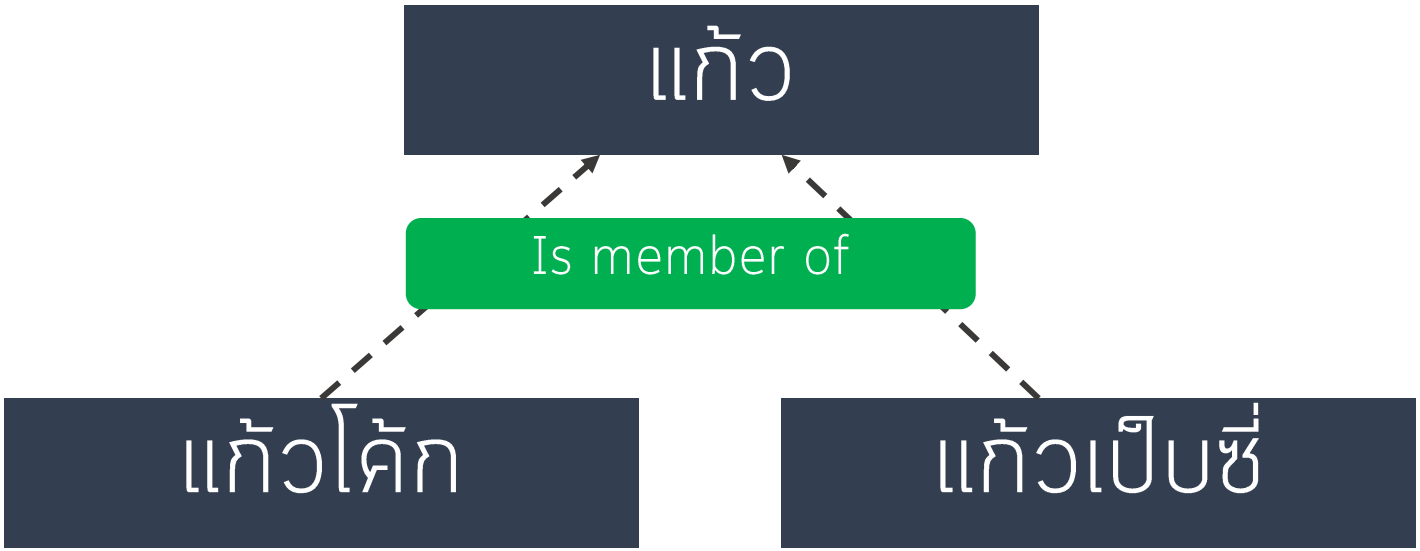
ห้องเรียน

ตัวอย่าง

- แก้วกาแฟ แก้วโค้ก .....แก้ว , ภาชนะ
- กาแฟ โค้ก เป็บซี่ .....
- ข้าวขาหมู แกงไตปลา แกงเขียวหวาน ..... อาหาร

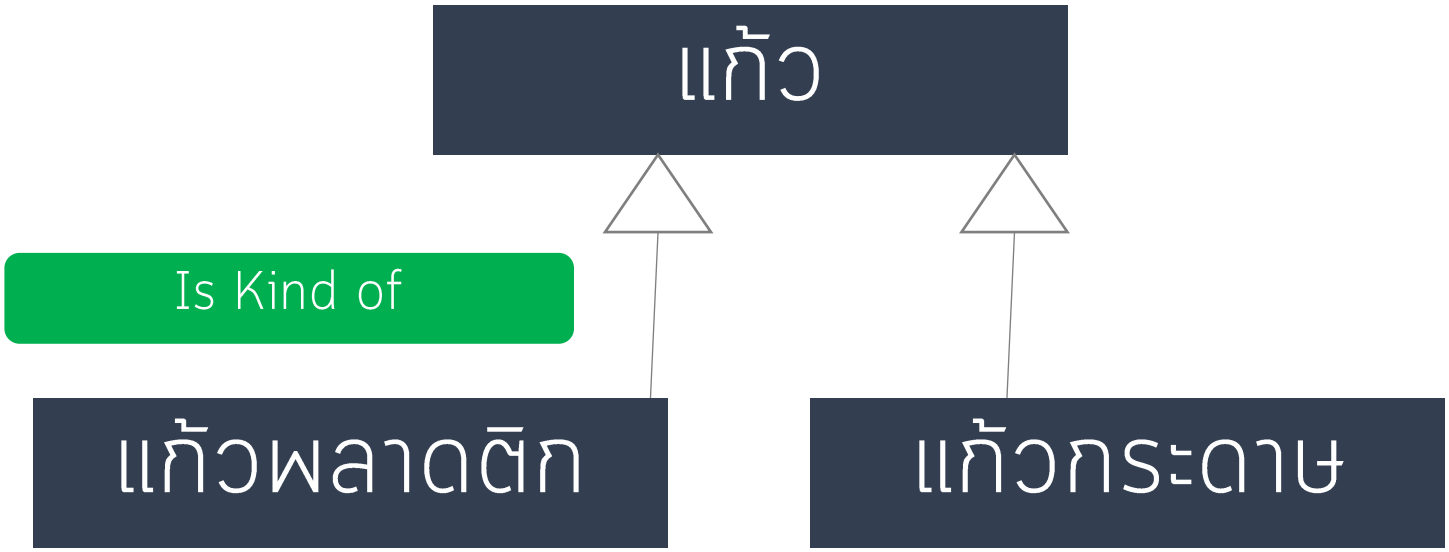
ข้อแตกต่างระหว่าง Classification กับ Generalization

Classification Abstraction

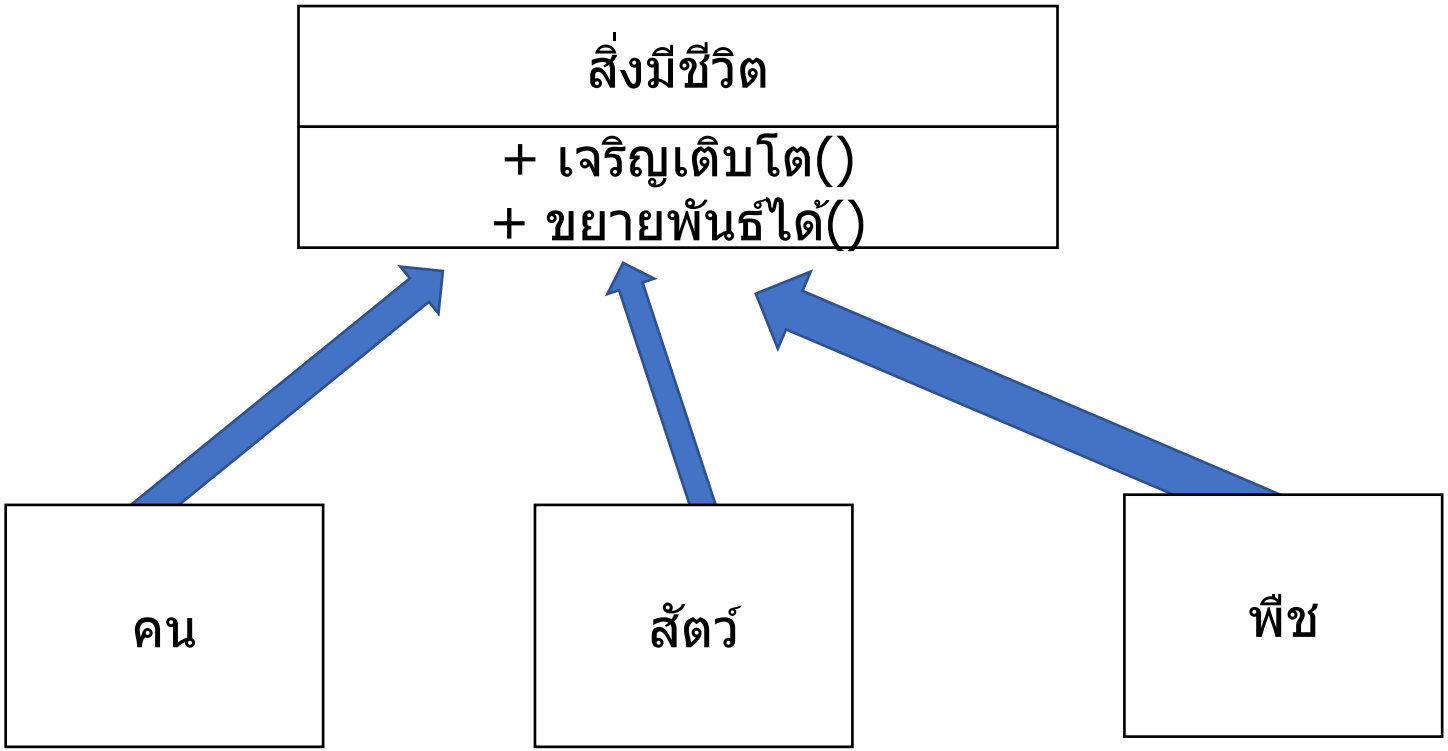


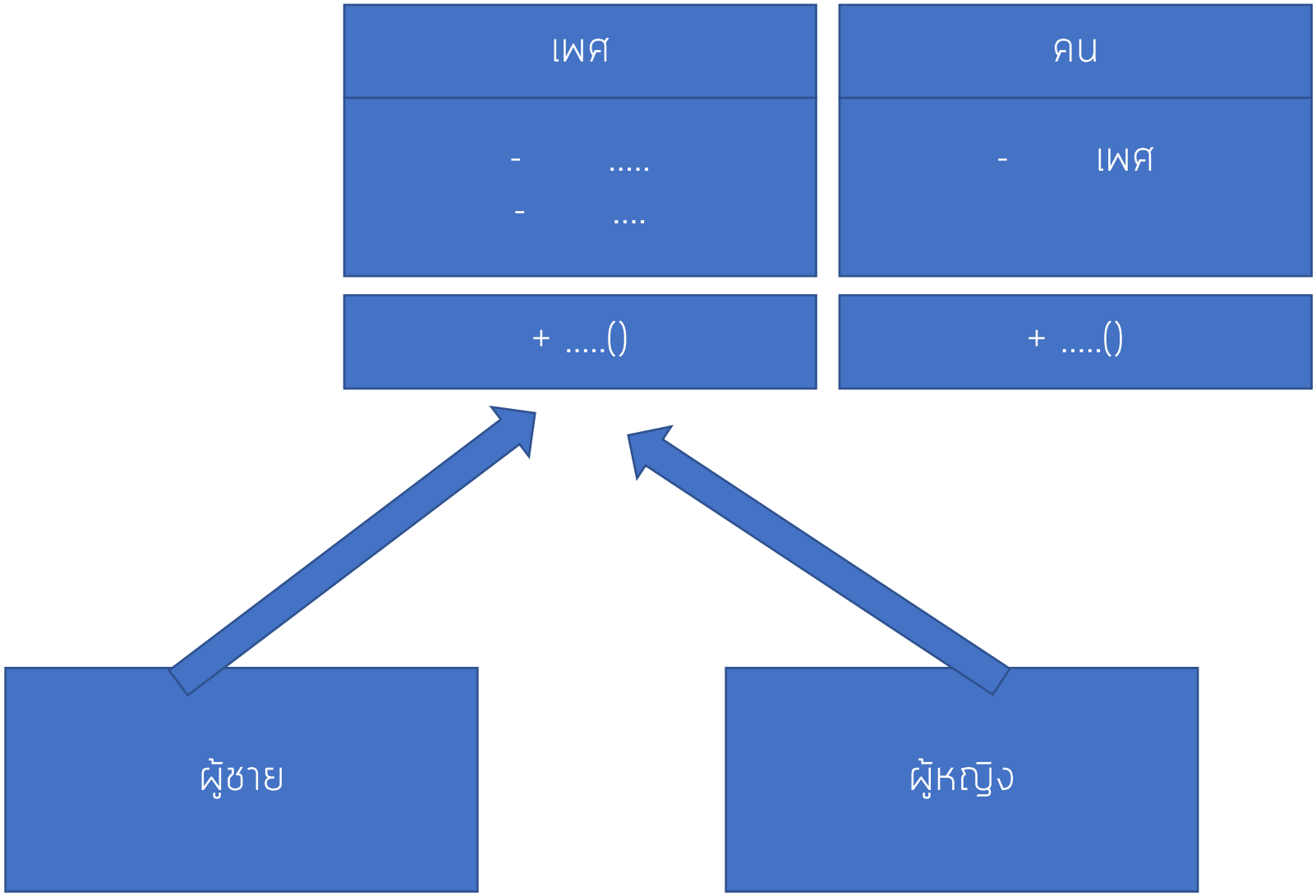
ข้อแตกต่างระหว่าง Classification กับ Generalization

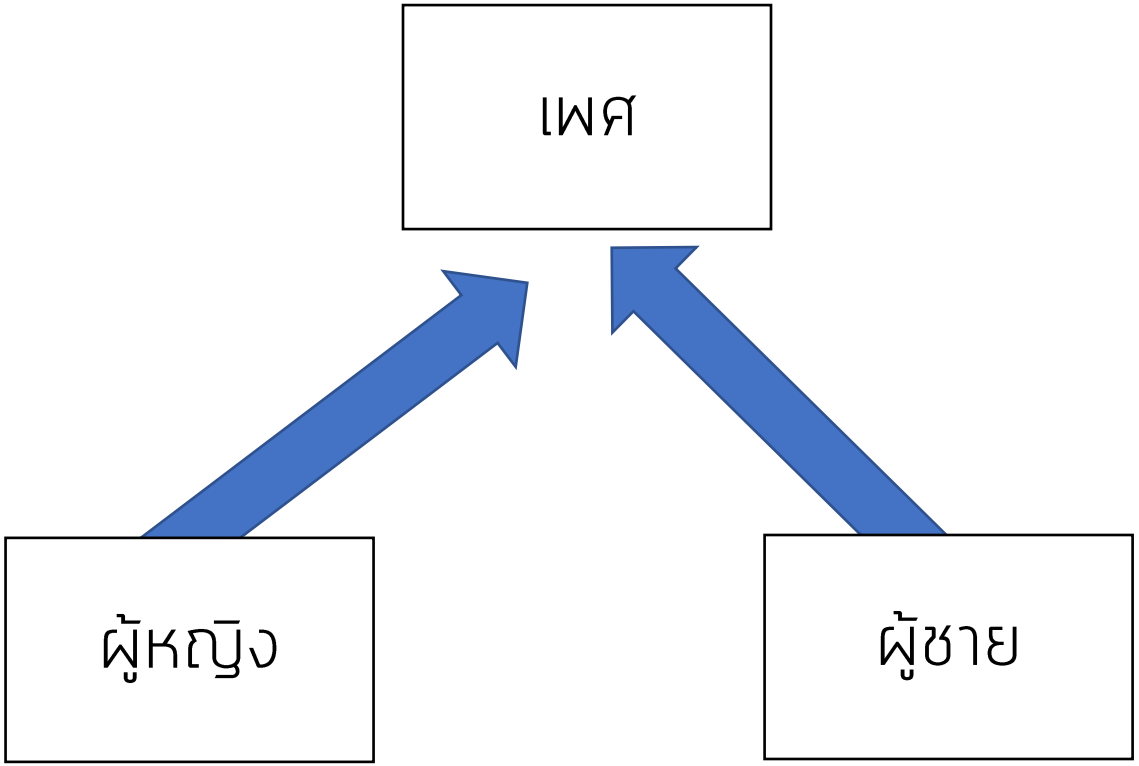
Generalization Abstraction



Generalization Abstraction

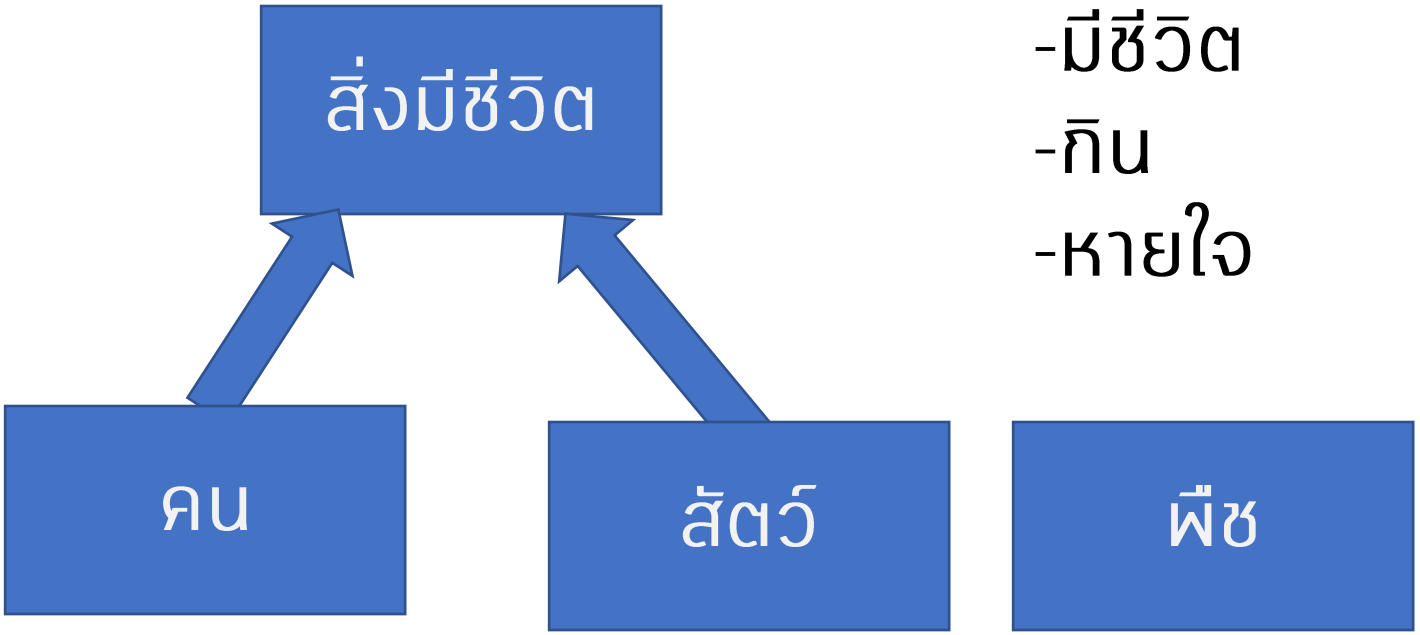



เพศ
- .....
- ....
+ .....()
คน
- เพศ
+ .....()
ผู้ชาย
ผู้หญิง

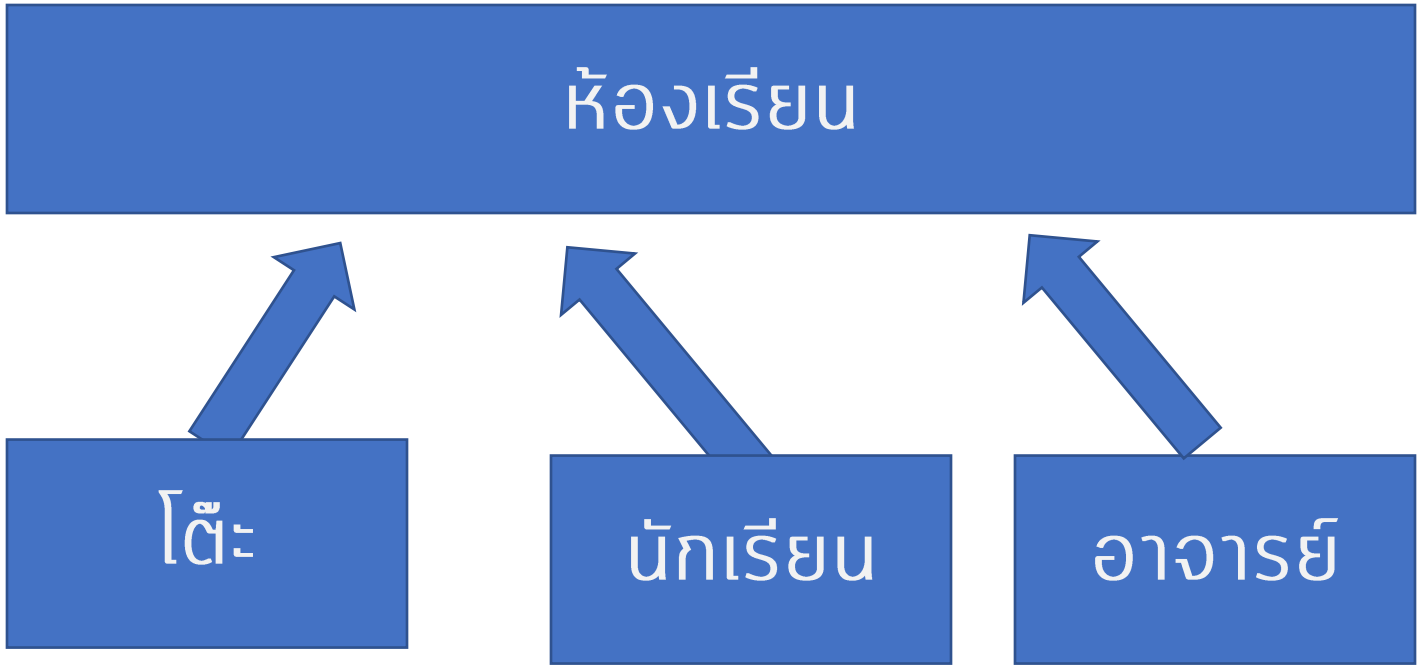


***** ไม่ถูกต้อง เพราะเราจะไม่สามารถระบุ Attribute และ function ของเพศได้ เราจึงจัดให้กลุ่มผู้ชายและกลุ่มผู้หญิงอยู่ในเพศ เพราะเพศ เป็น attribute ไม่ใช่คลาส**

Generalization Abstraction



ตัวอย่างการne Generalization Abstraction ที่ไม่ถูกต้อง

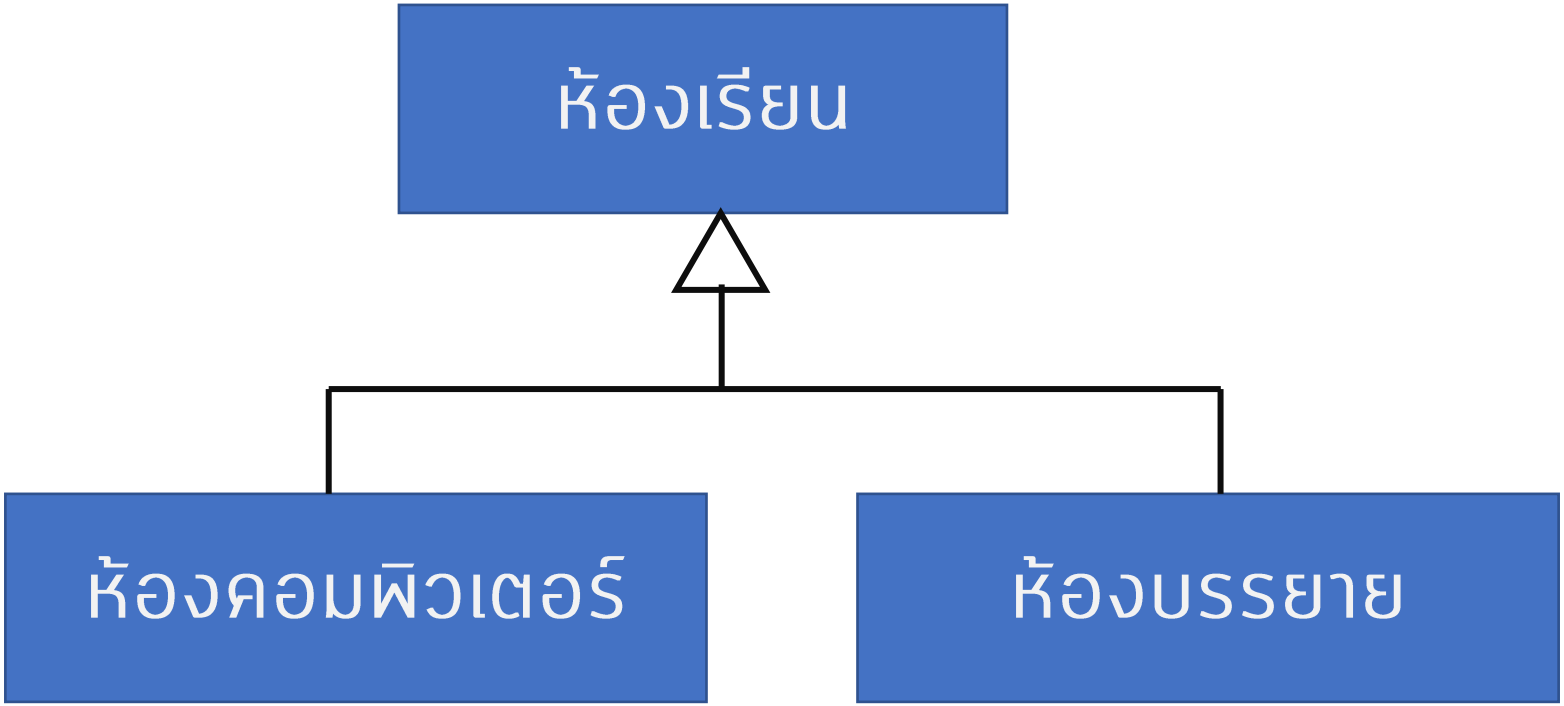


*** ไม่ถูกต้อง เพราะ นักเรียน อาจารย์ โต๊ะ ไม่จัดเป็นห้องเรียน

1. จงเติม attribute และฟังก์ชันให้ครบทุกคลาส

2. จากแผนภาพนี้ อธิบายได้ว่า.....

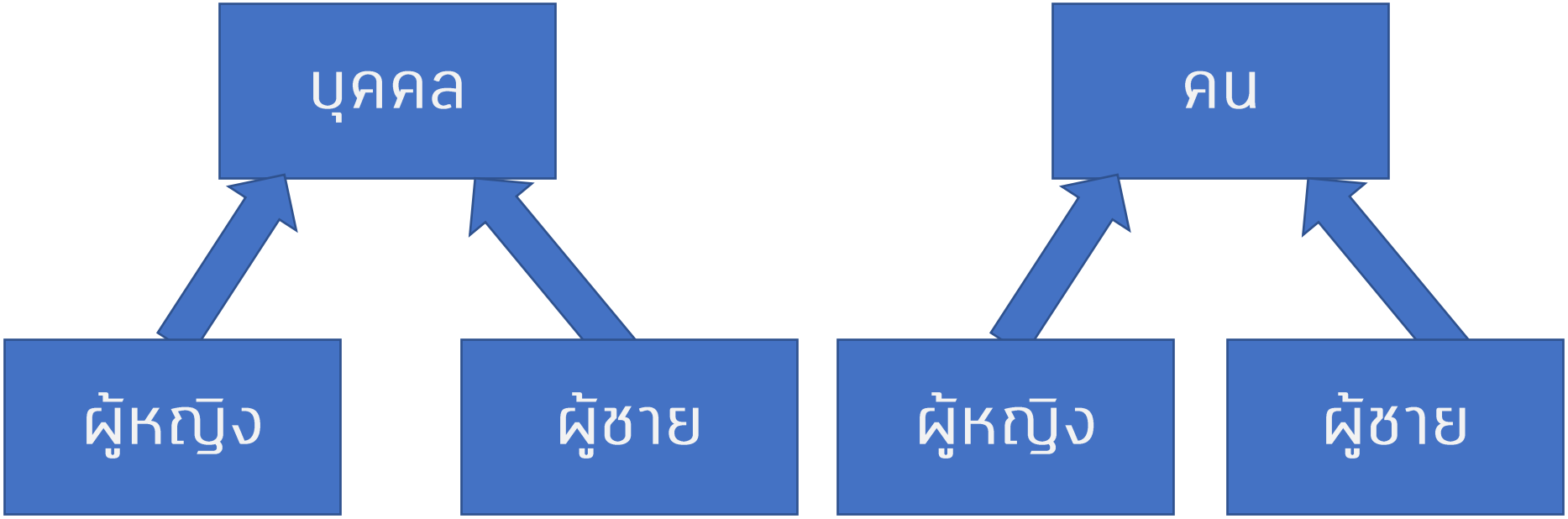
Generalization Abstraction



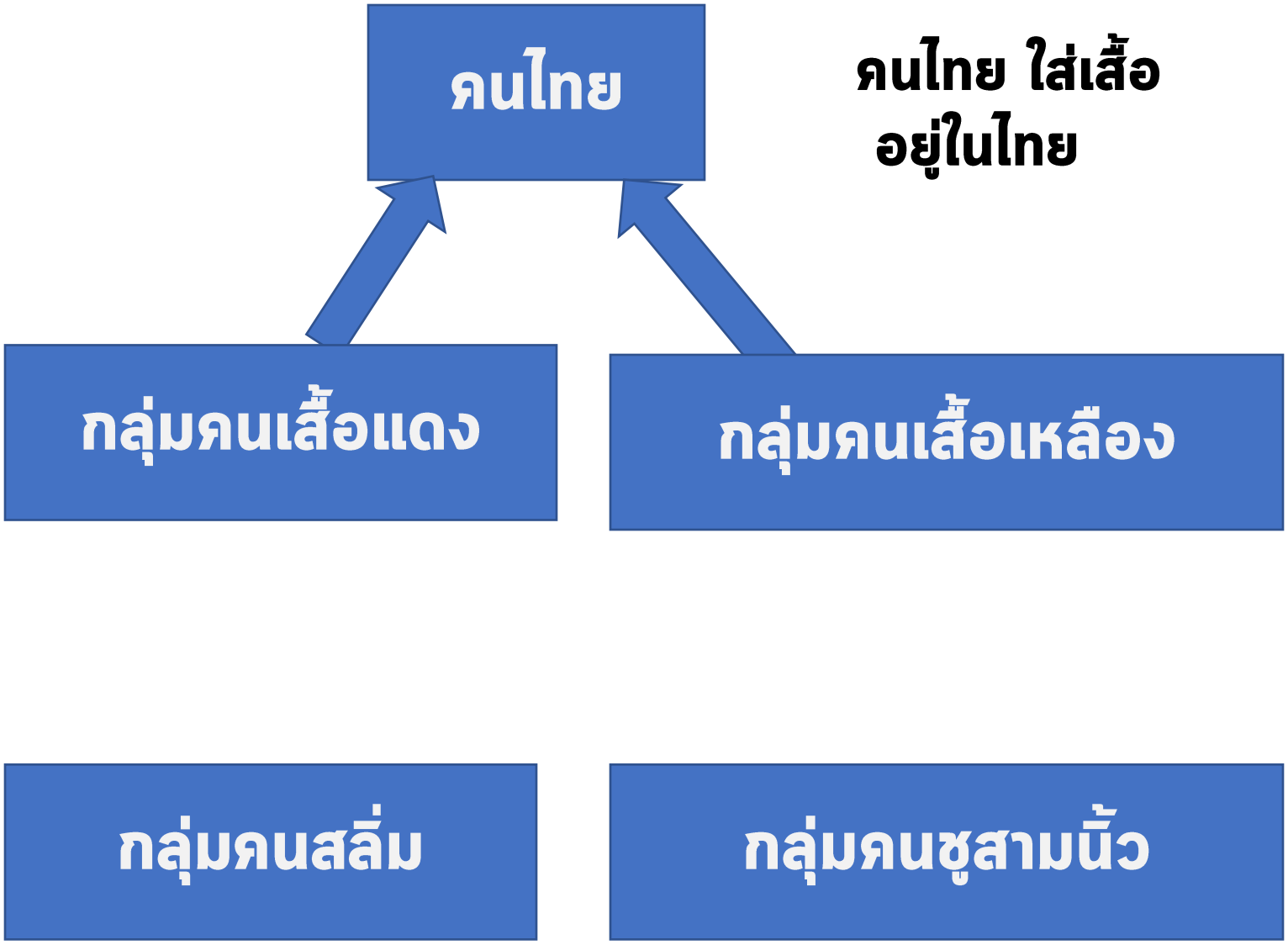
1. จงเติม attribute และฟังก์ชันให้ครบทุกคลาส

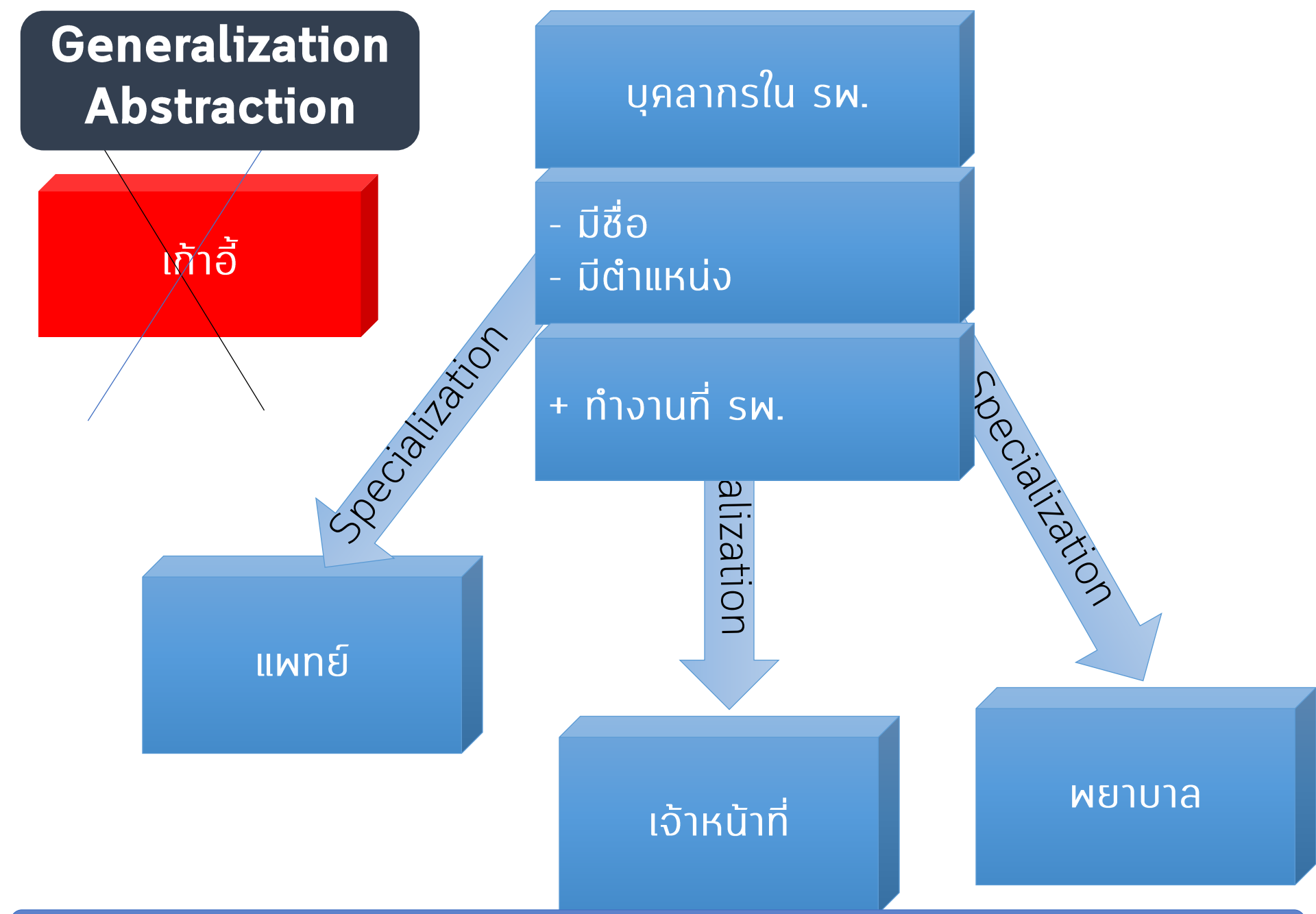
2. จากแผนภาพนี้ อธิบายได้ว่า.....

Generalization Abstraction



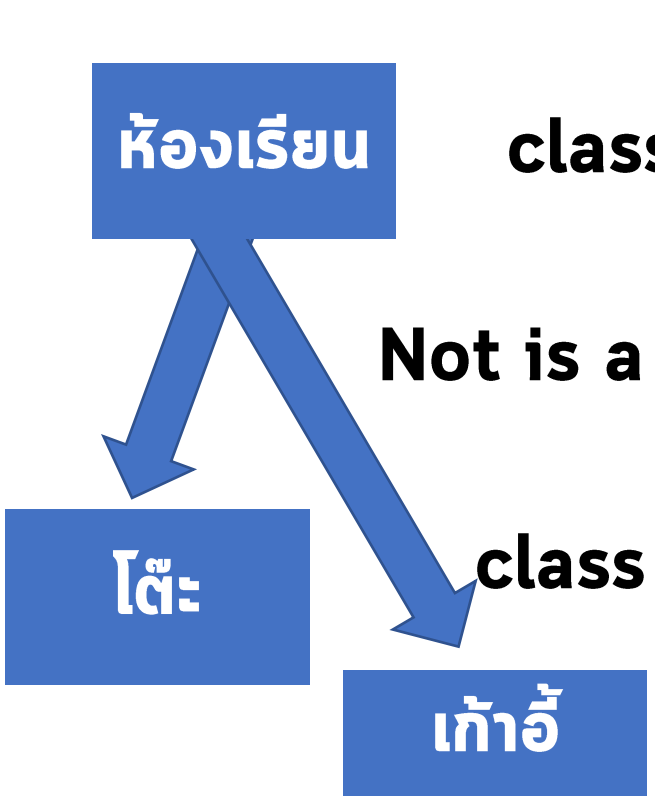
Generalization Abstraction

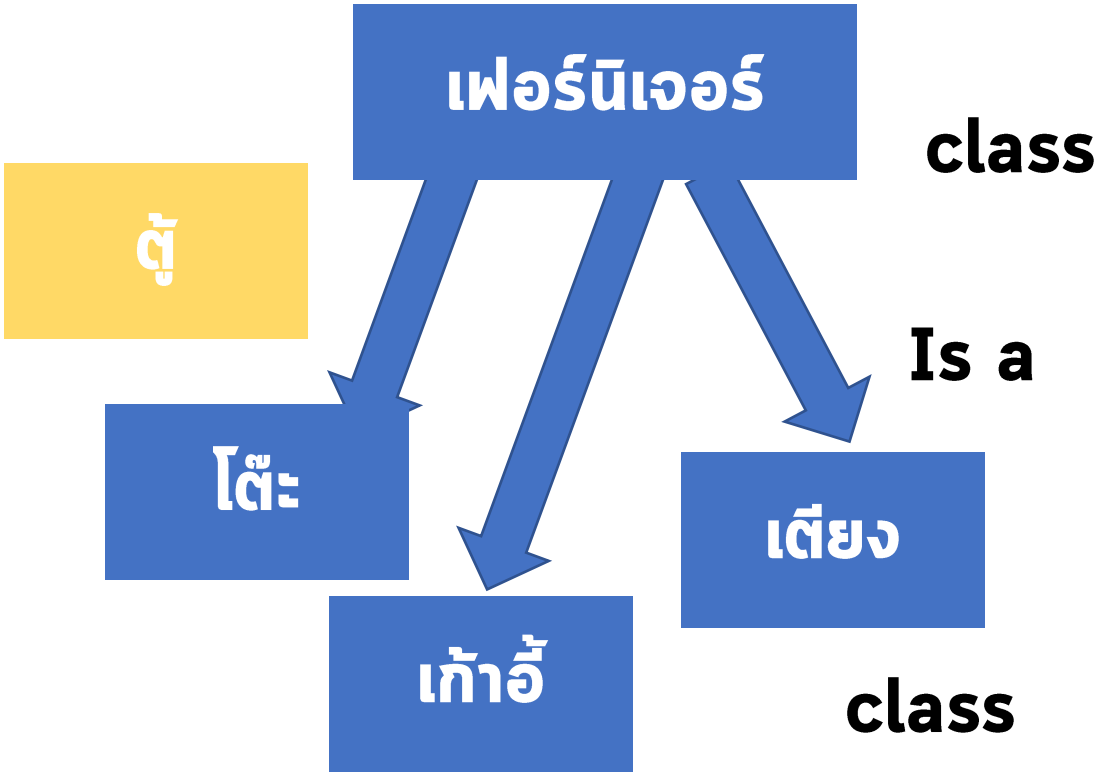




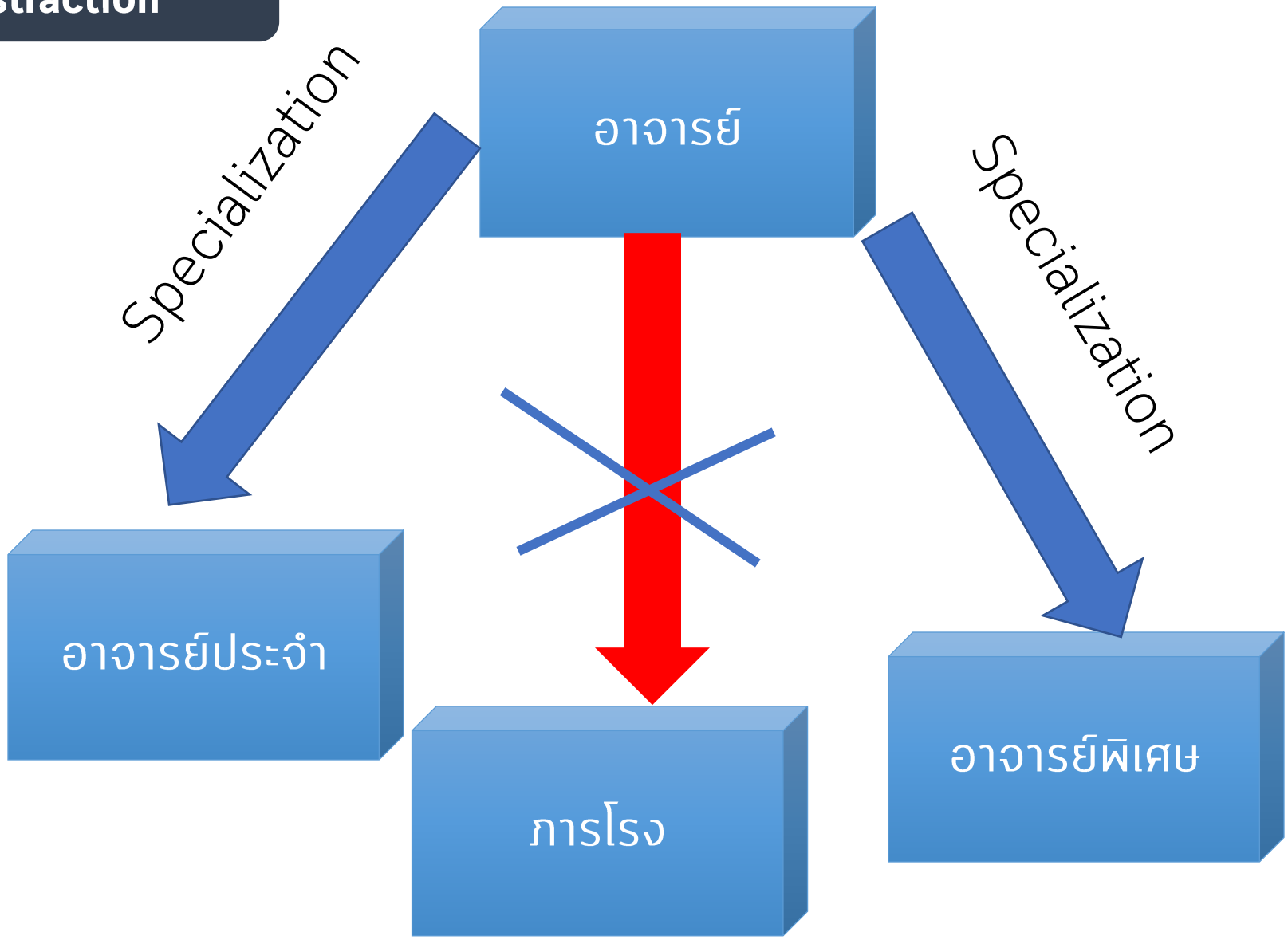
ข้อแตกต่างระหว่าง aggregation กับ specialization

- Aggregate >> ประกอบกัน
- Specialization >> แบ่ง/แยกย่อย จำแนกออกเป็นประเภท

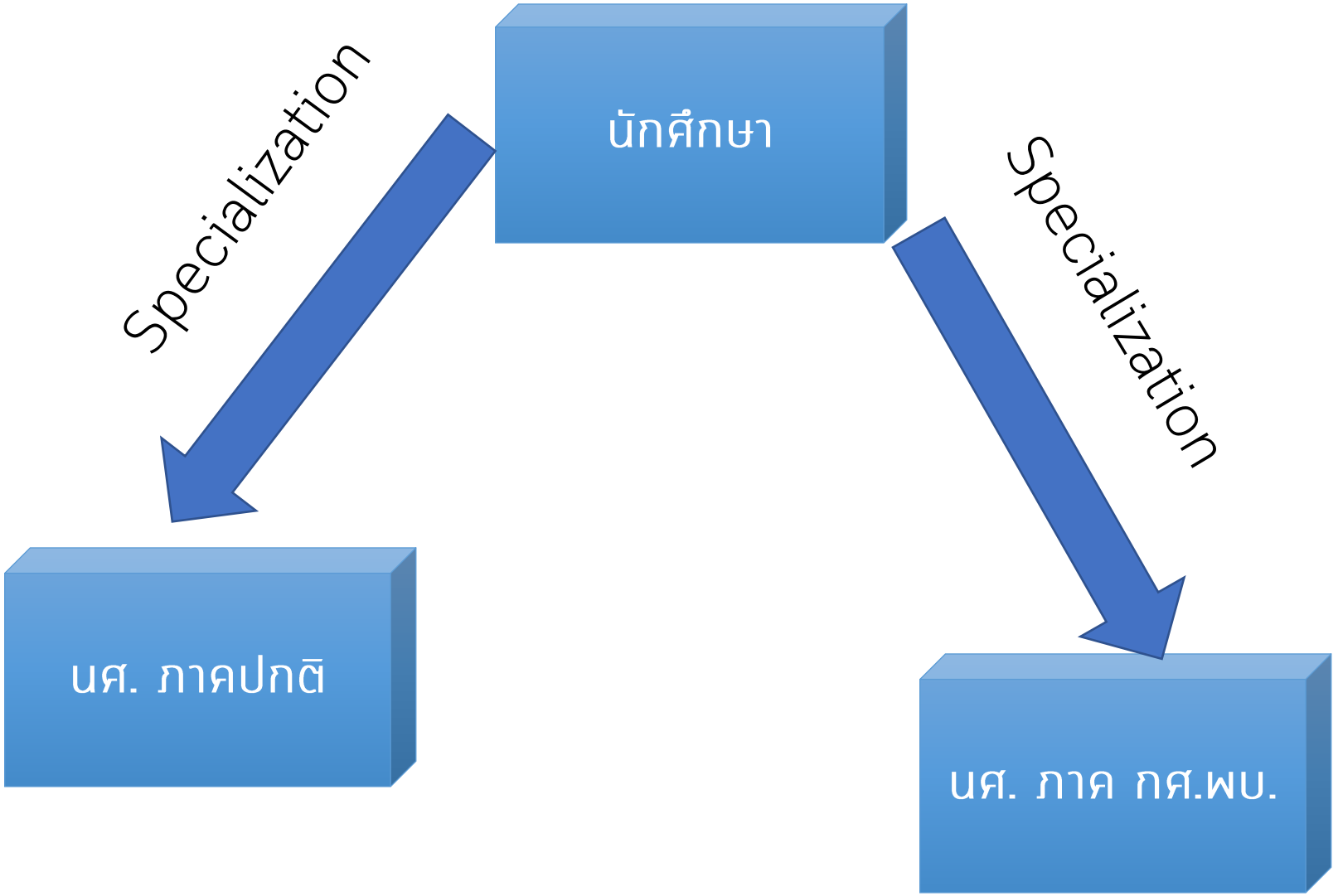




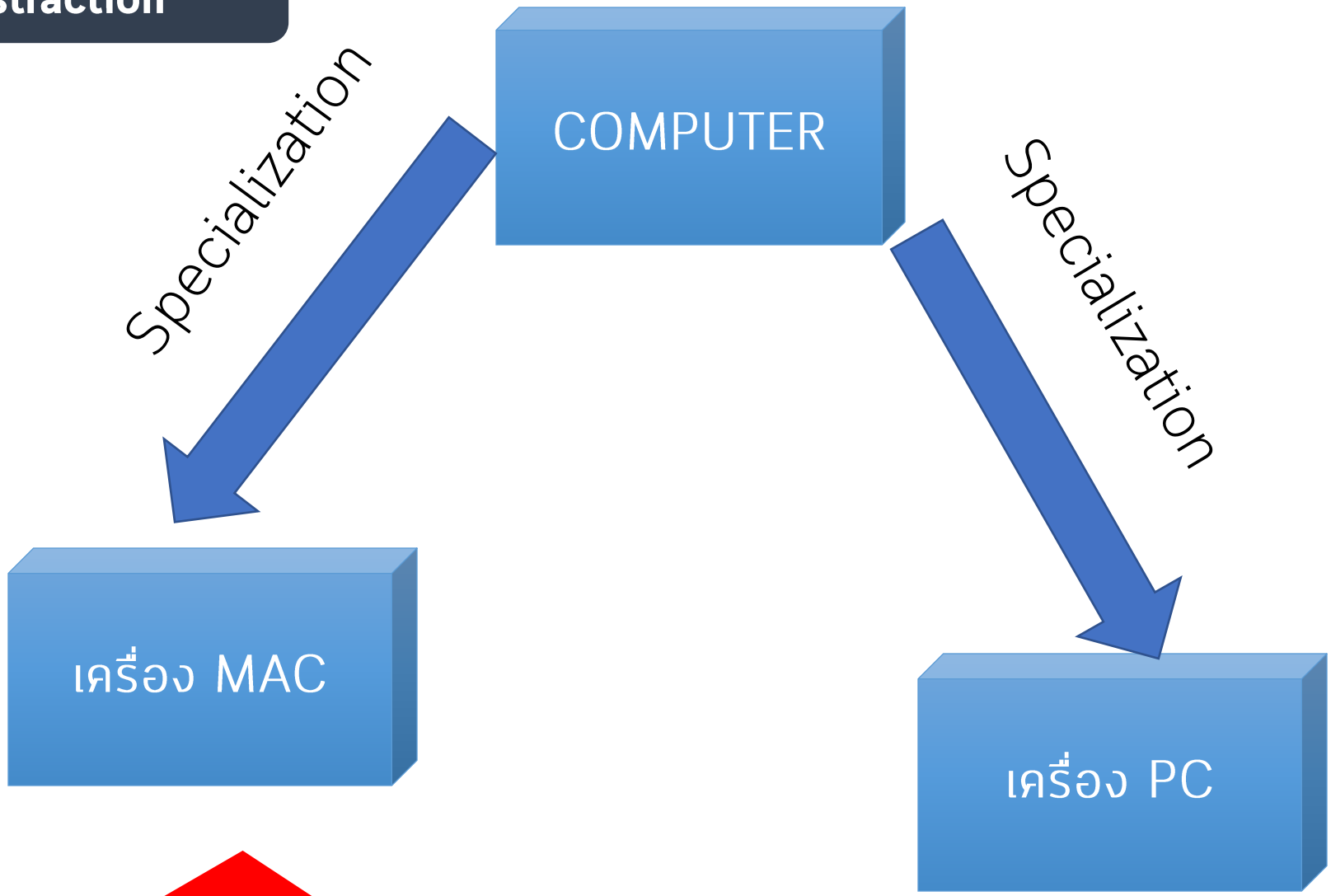
Generalization Abstraction



Generalization Abstraction

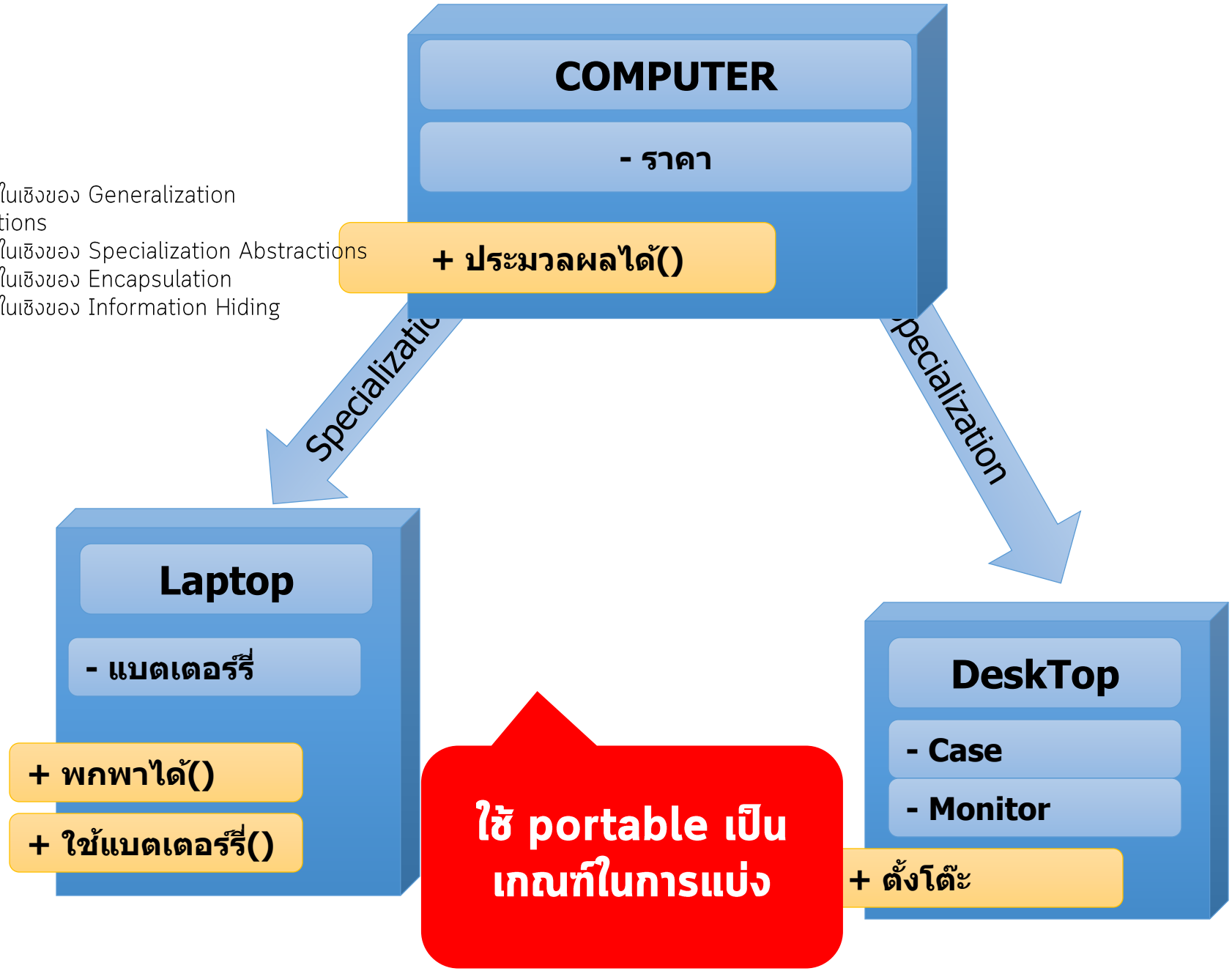


Generalization
Abstraction



ใช้ระบบปฏิบัติการ (OS) เป็นเกณฑ์ในการแบ่ง

จงอธิบายในเชิงของ Generalization Abstractions
จงอธิบายในเชิงของ Specialization Abstractions
จงอธิบายในเชิงของ Encapsulation
จงอธิบายในเชิงของ Information Hiding



จากรูปในสไลด์ก่อนหน้า

- จงอธิบายในเชิงของ Generalization Abstractions
- จงอธิบายในเชิงของ Specialization Abstractions
- จงอธิบายในเชิงของ Encapsulation
- จงอธิบายในเชิงของ Information Hiding

Specialization abstraction

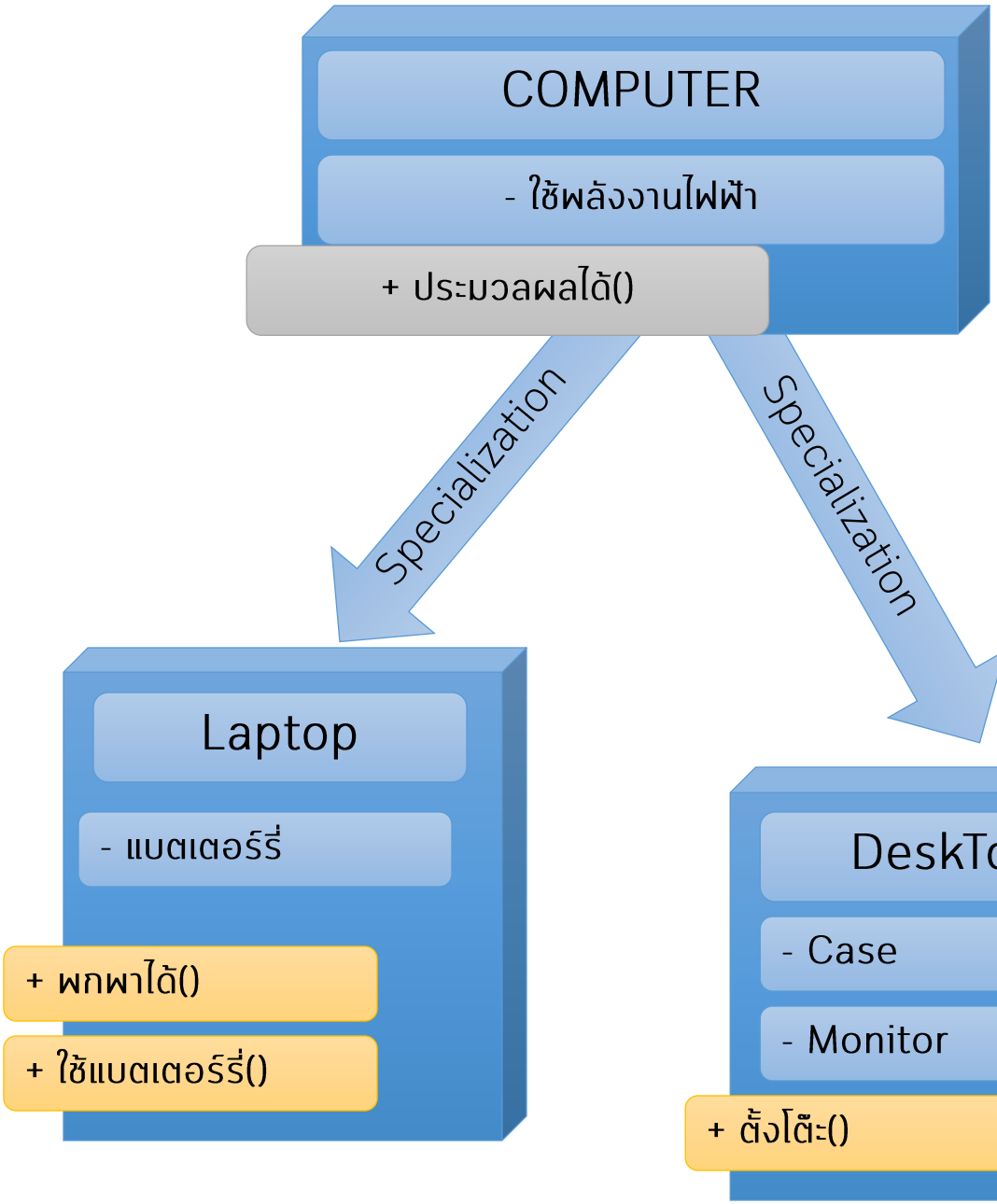
- จากรูปก่อนหน้านี้ สามารถอธิบายในเชิงของ specialization abstraction ได้ว่า “เราสามารถแบ่งประเภทของคอมพิวเตอร์ตามลักษณะการใช้งานได้ 2 ประเภท คือ 1. Desktop 2. Laptop ”

Intensive

ทั้ง Desktop และ LabTop ต่างก็เป็นเครื่องคอมพิวเตอร์ และประมวลผลได้ แต่ต่างกันตรงที่ LabTop สามารถพกพาไปได้ ในขณะ desktop ไม่สะดวกที่จะพกพา

ข้อสรุป สำหรับ specialization

1. เราพิจารณาดูว่าคลาสหนึ่ง ๆ แบ่งออกเป็นคลาสย่อยอะไรได้บ้าง
2. ในการแบ่งออกเป็นคลาสสามารถทำได้หลายประเภท/วิธีขึ้นอยู่กับว่าเราจะใช้เกณฑ์อะไร เป็นตัวแบ่ง
3. ถ้าแบ่งแล้วมีลักษณะพิเศษขึ้นมา ก็ควรแบ่ง
4. การแบ่งก็คือการเจาะจงลงไป ว่ามีลักษณะพิเศษอะไรบ้างจงเรียกว่า special
5. การแบ่งก็คือการเอาจุดที่แตกต่างกันของแต่ละคลาสมาแบ่ง



จากรูปนี้

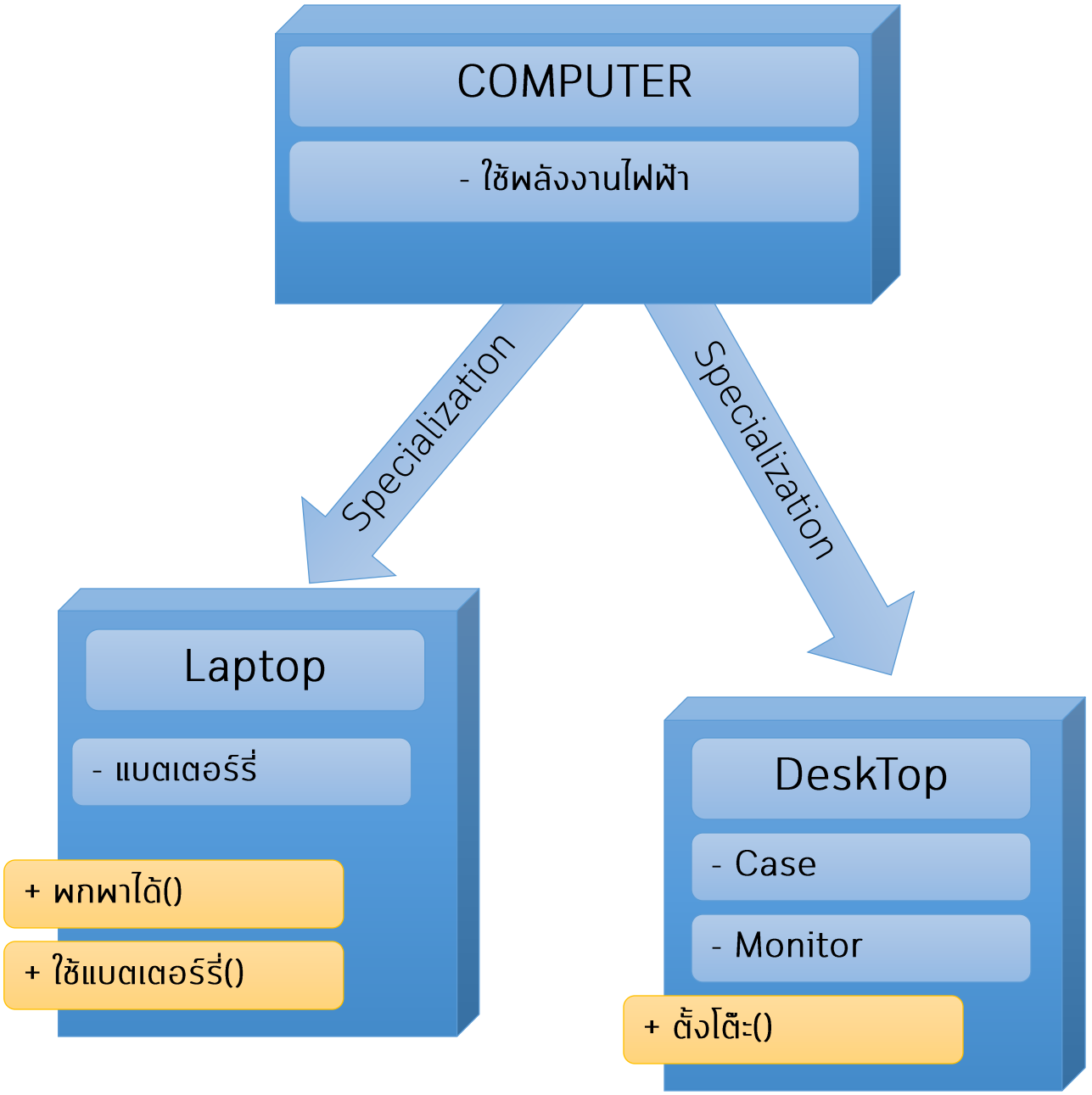
ตู้เย็นเป็นคอมพิวเตอร์หรือไม่?

ทีวีเป็นคอมพิวเตอร์หรือไม่?

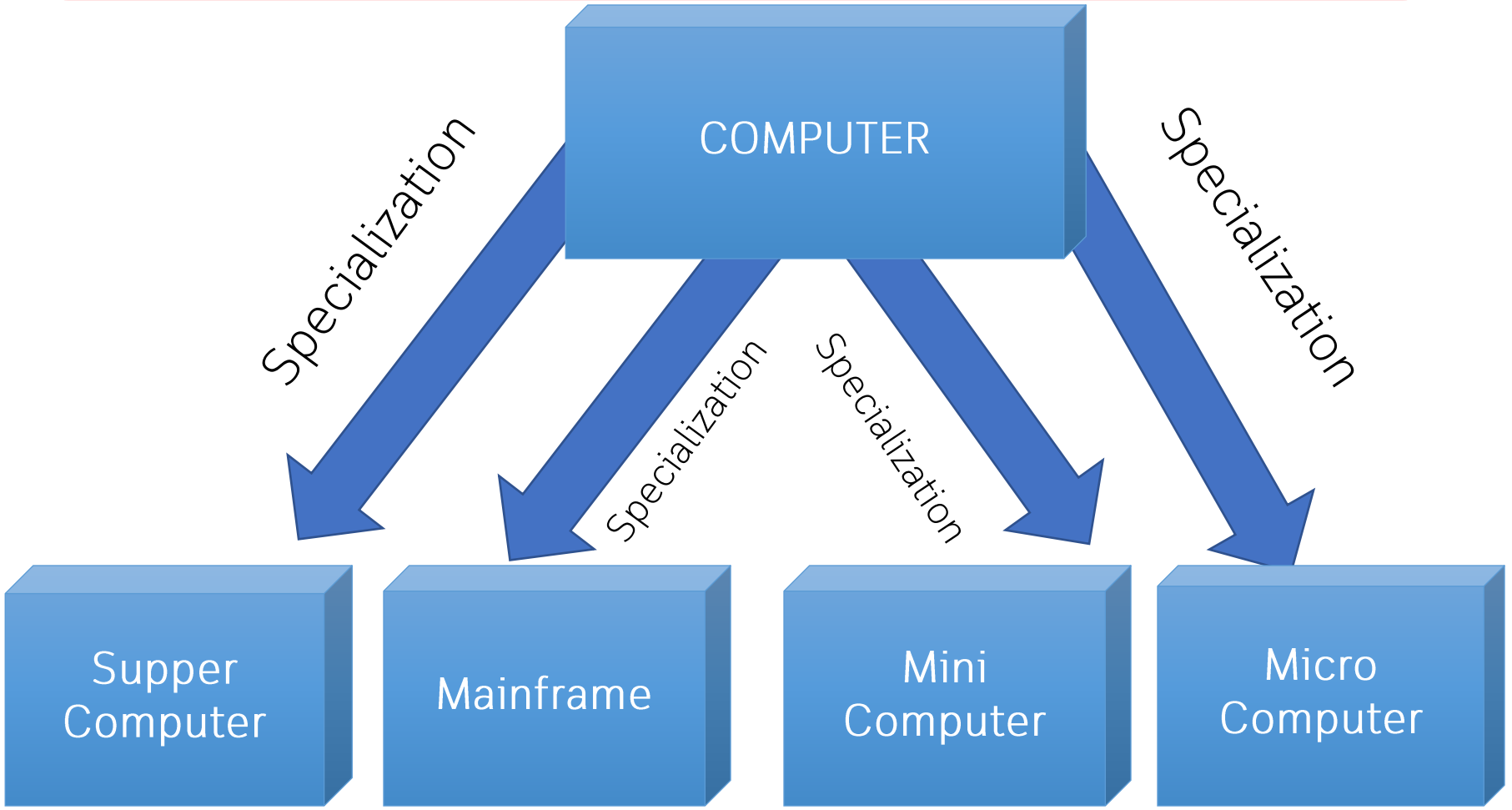
พัดลมเป็นคอมพิวเตอร์หรือไม่?

เครื่องคิดเลขเป็นคอมพิวเตอร์หรือไม่?

โทรศัพท์มือถือเป็นคอมพิวเตอร์หรือไม่?

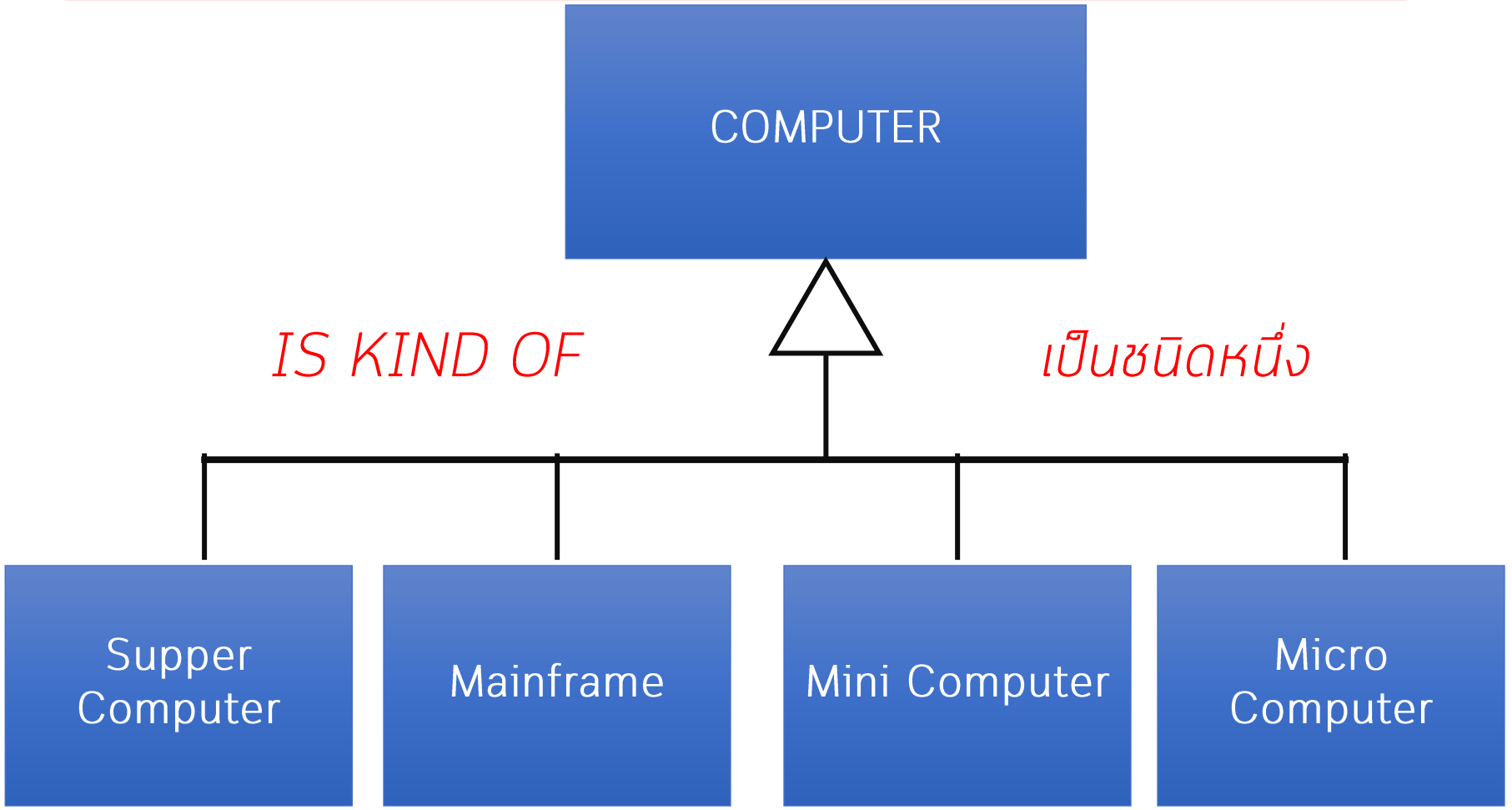

COMPUTER
- ใช้พลังงานไฟฟ้า
Specialization
Specialization
Laptop
- แบตเตอร์รี่
+ พกพาได้()
+ ใช้แบตเตอร์รี่()
DeskTop
- Case
- Monitor
+ ตั้งโต๊ะ()

วิธีแบ่งโดยใช้ขนาดเป็นเกณฑ์

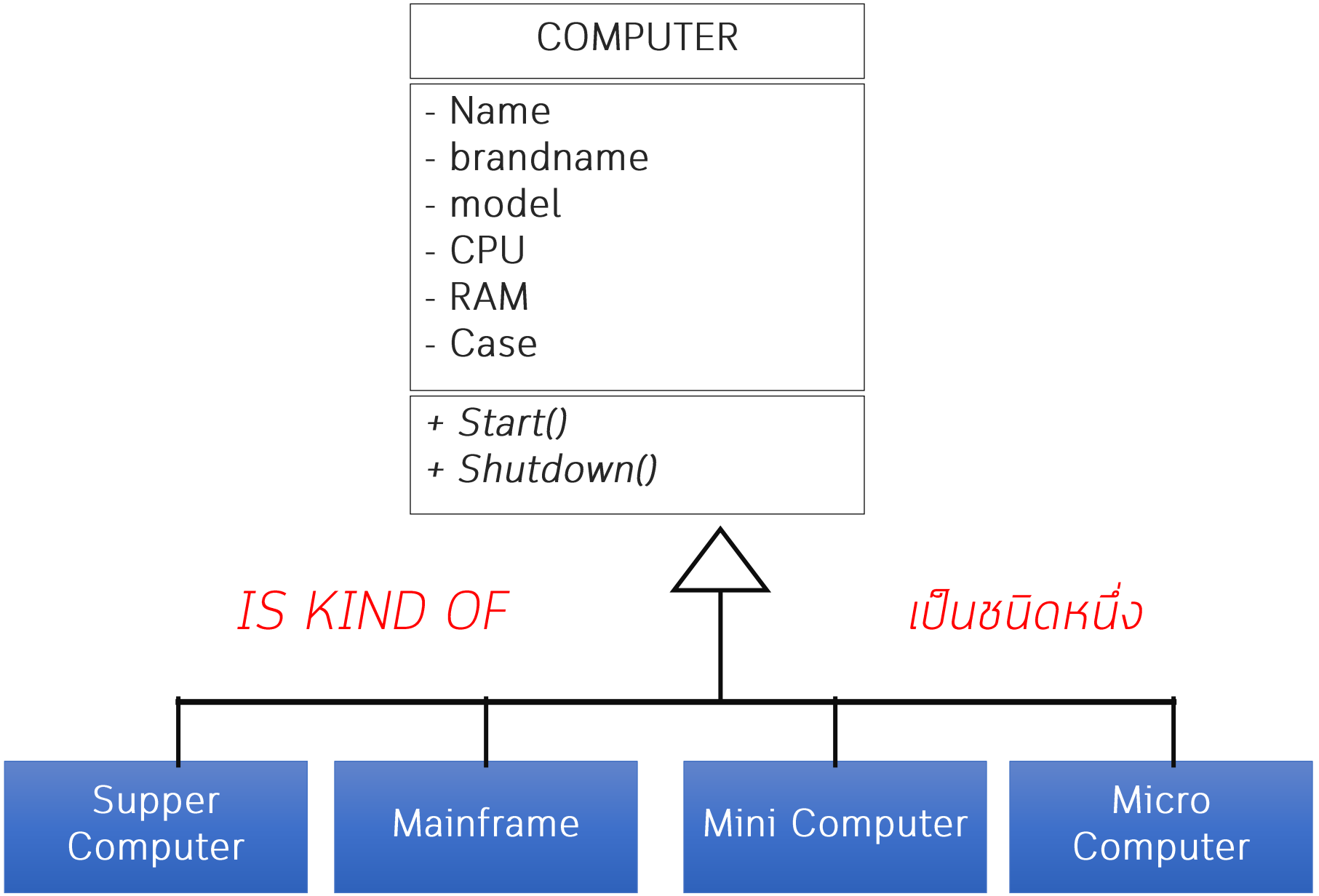


*** ใช้เกณฑ์ ขนาด และความสามารถในการประมวลผล

วิธีแบ่งโดยใช้ขนาดเป็นเกณฑ์



*** ใช้เกณฑ์ ขนาด และความสามารถในการประมวลผล



จากภาพก่อนหน้า สามารถอธิบายในเชิงของ specialization abstraction ได้ว่า

- อธิบายได้ว่า คอมพิวเตอร์แบ่งออกเป็น 4 ประเภทคือ SuperComputer , MainFrame , Mini , Micro โดยแบ่งตามขนาด และความสามารถในประมวลผล

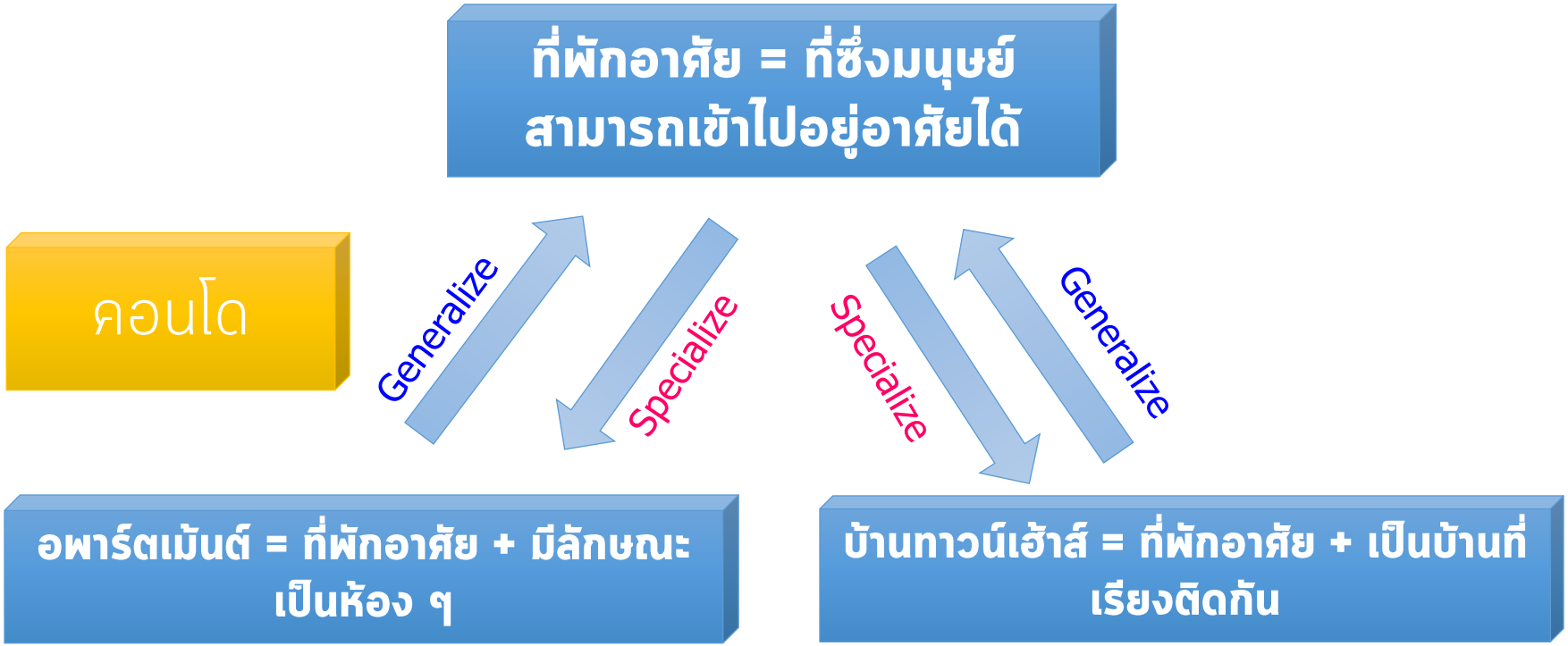
Intensive

ทั้ง 4 ประเภท ต่างก็เป็นคอมพิวเตอร์ ที่ใช้ไฟฟ้าและประมวลผลได้ แต่ต่างกันตรงที่ ขนาด ประวิทธิภาพในการทำงาน

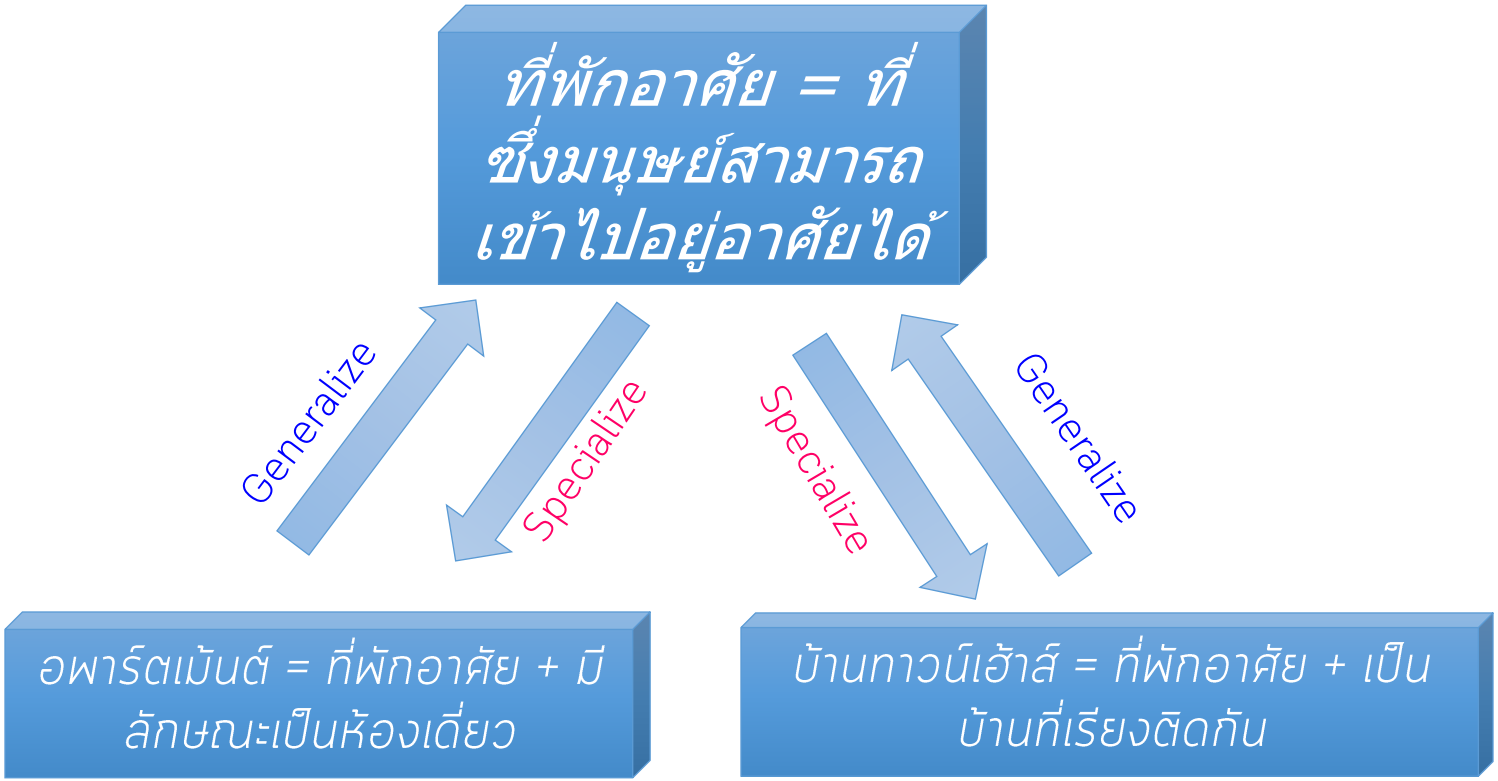
หากสนใจ slide นี้เพื่อการเรียนการสอนโปรดติดต่อ siam2dev@hotmail.com

Generalization/Specialization

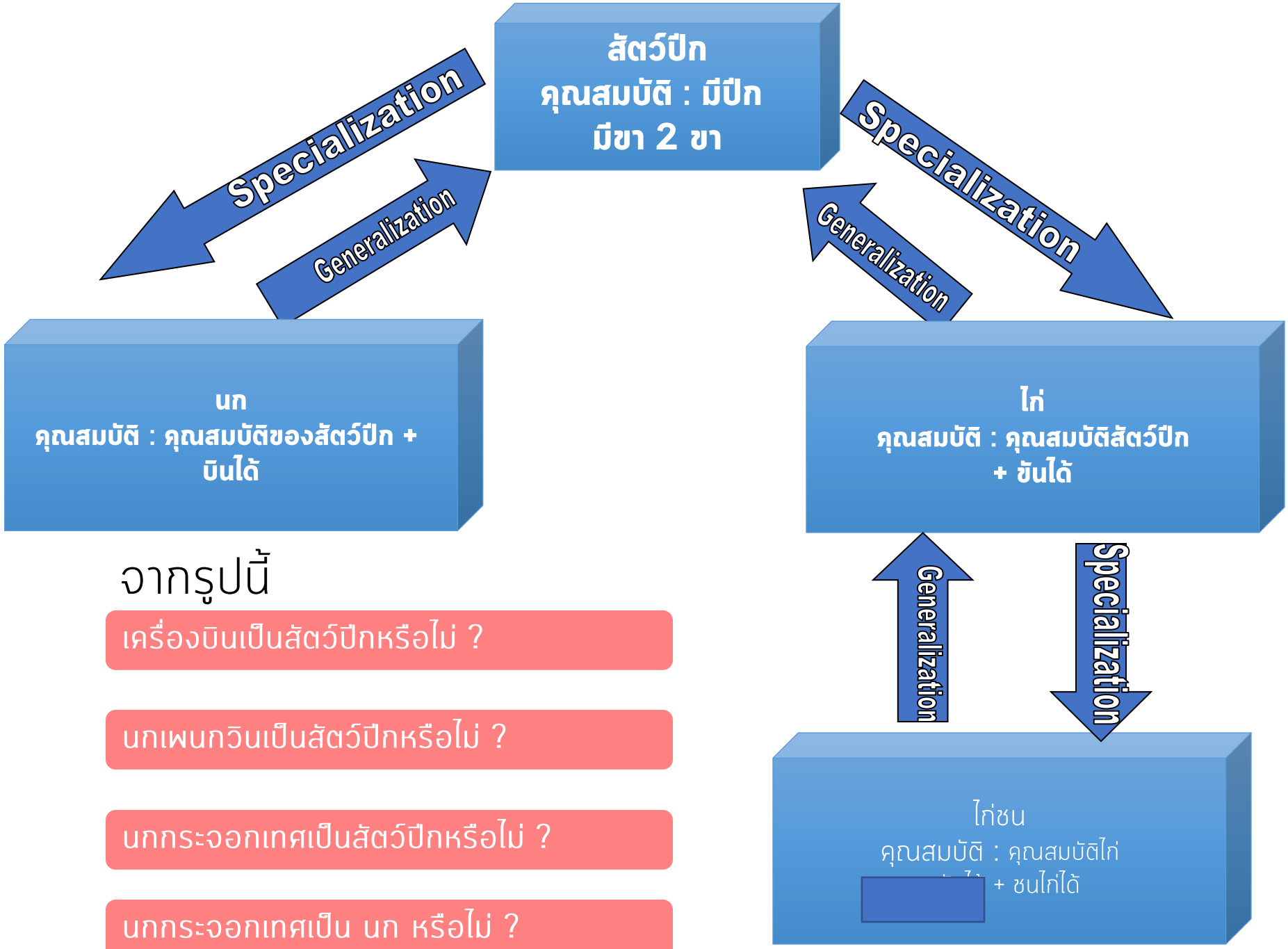
จากรูปก่อนหน้านี้อธิบายในเชิงของ Generalization Abstraction ได้ว่า **“super computer mainframe mini computer และ micro computer ต่างก็คุณสมบัติร่วมกันคือ ใช้ไฟฟ้า และ สามารถคำนวณได้เราจึงจัดรวมกันเป็นคลาสเดียวกันนั่นก็คือ คลาส Computer”** และได้ทางกลับกัน (Flip side) อธิบายในเชิงของ specialization abstraction ได้ว่า **“เราสามารถแบ่งคอมพิวเตอร์ออกเป็นประเภทต่าง ๆ ได้ 4 ประเภทคือ 1. super computer 2. Mainframe 3. Mini computer และ 4. Micro Computer” โดยเกณฑ์ที่ใช้แบ่งขึ้นอยู่กับขนาดและประสิทธิภาพในการประมวลผล**



จากรูป เป็นการอธิบาย “ที่พักอาศัย จำแนกเป็น อพาร์ตเม้นต์ และบ้านทาวน์เฮาส์” จะเห็นว่า เราใช้ Specialize เพื่อสร้างอพาร์ตเม้นต์ และทาวน์เฮาส์ขึ้น ในขณะเดียวกัน เราใช้ Generalize เพื่อทำให้อพาร์ตเม้นต์ และทาวน์เฮาส์มี Concept ร่วมเดียวกัน นั่นคือ ทั้งอพาร์ตเม้นต์และทาวน์เฮาส์ต่างก็ใช้เพื่อเป็นที่อยู่อาศัยของมนุษย์ ซึ่งนี่คือ Concept ของที่อยู่อาศัย

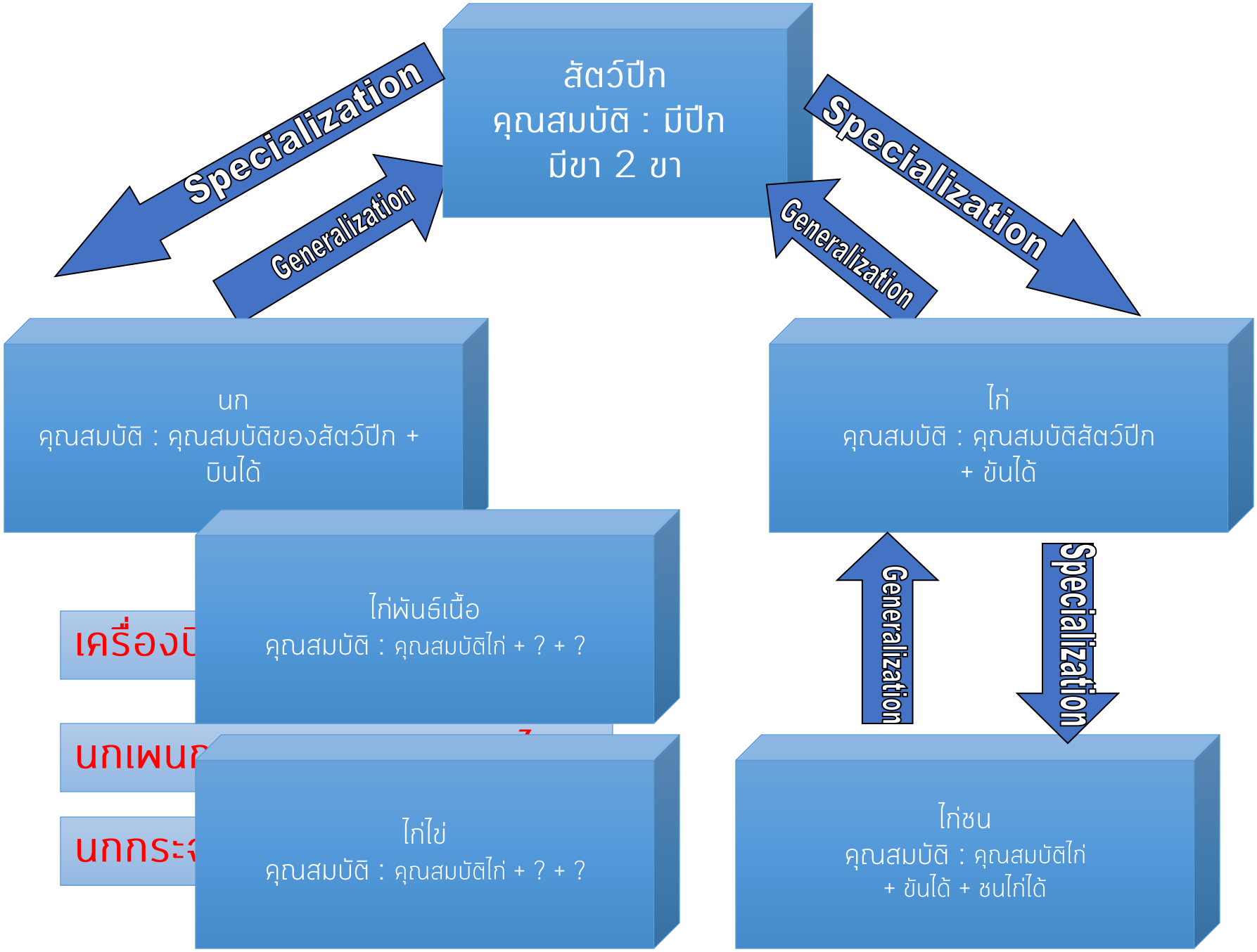


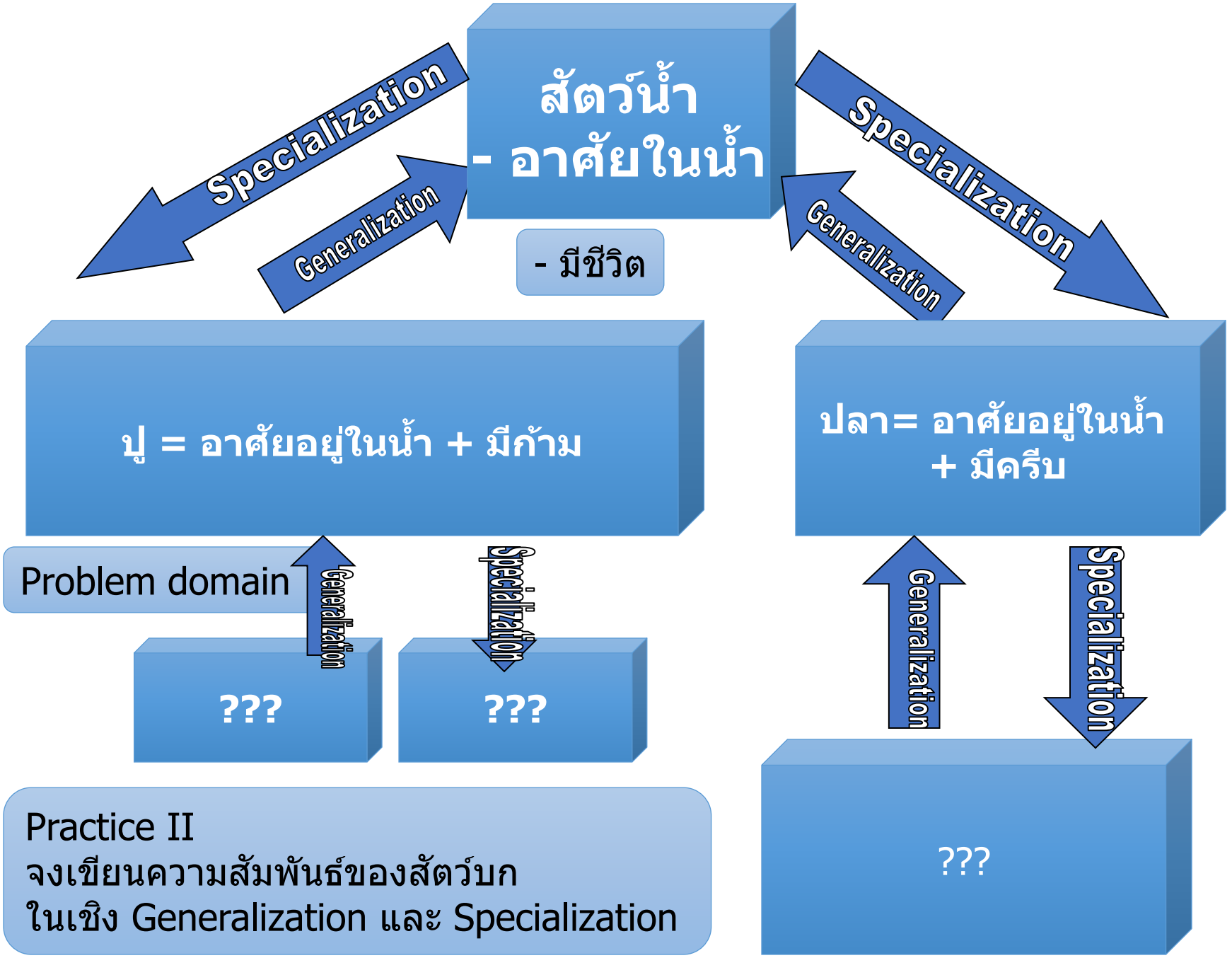
จากรูป เป็นการอธิบาย “ที่พักอาศัย จำแนกเป็น อพาร์ตเม้นต์ และบ้านทาวน์เฮาส์” จะเห็นว่า เราใช้ Specialize เพื่อสร้างอพาร์ตเม้นต์ และทาวน์เฮาส์ขึ้น ในขณะเดียวกัน เราใช้ Generalize เพื่อทำให้อพาร์ตเม้นต์ และทาวน์เฮาส์มี Concept ร่วมเดียวกัน นั่นคือ ทั้งอพาร์ตเม้นต์และทาวน์เฮาส์ต่างก็ใช้เพื่อเป็นที่อยู่อาศัยของมนุษย์ ซึ่งนี่คือ Concept ของที่อยู่อาศัย



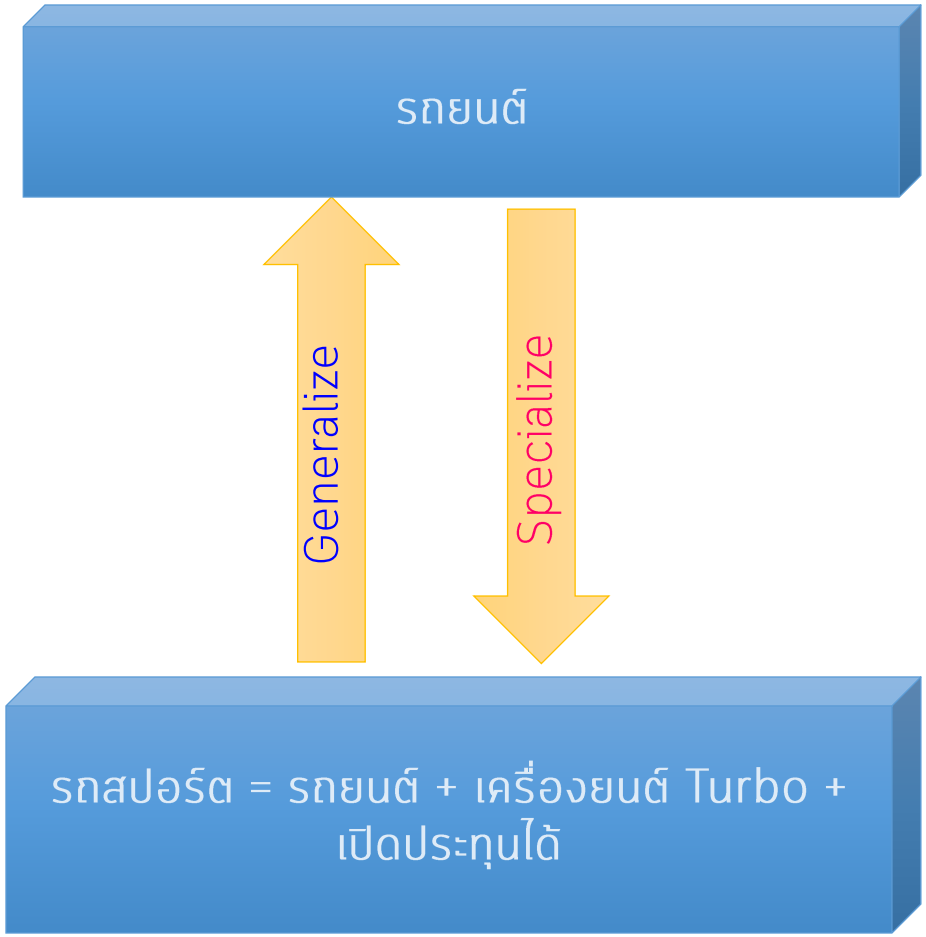
จากรูปนี้

- เครื่องบินเป็นสัตว์ปีกหรือไม่ ?
- นกเพนกวินเป็นสัตว์ปีกหรือไม่ ?
- นกกระจอกเทศเป็นสัตว์ปีกหรือไม่ ?
- นกกระจอกเทศเป็น นก หรือไม่ ?

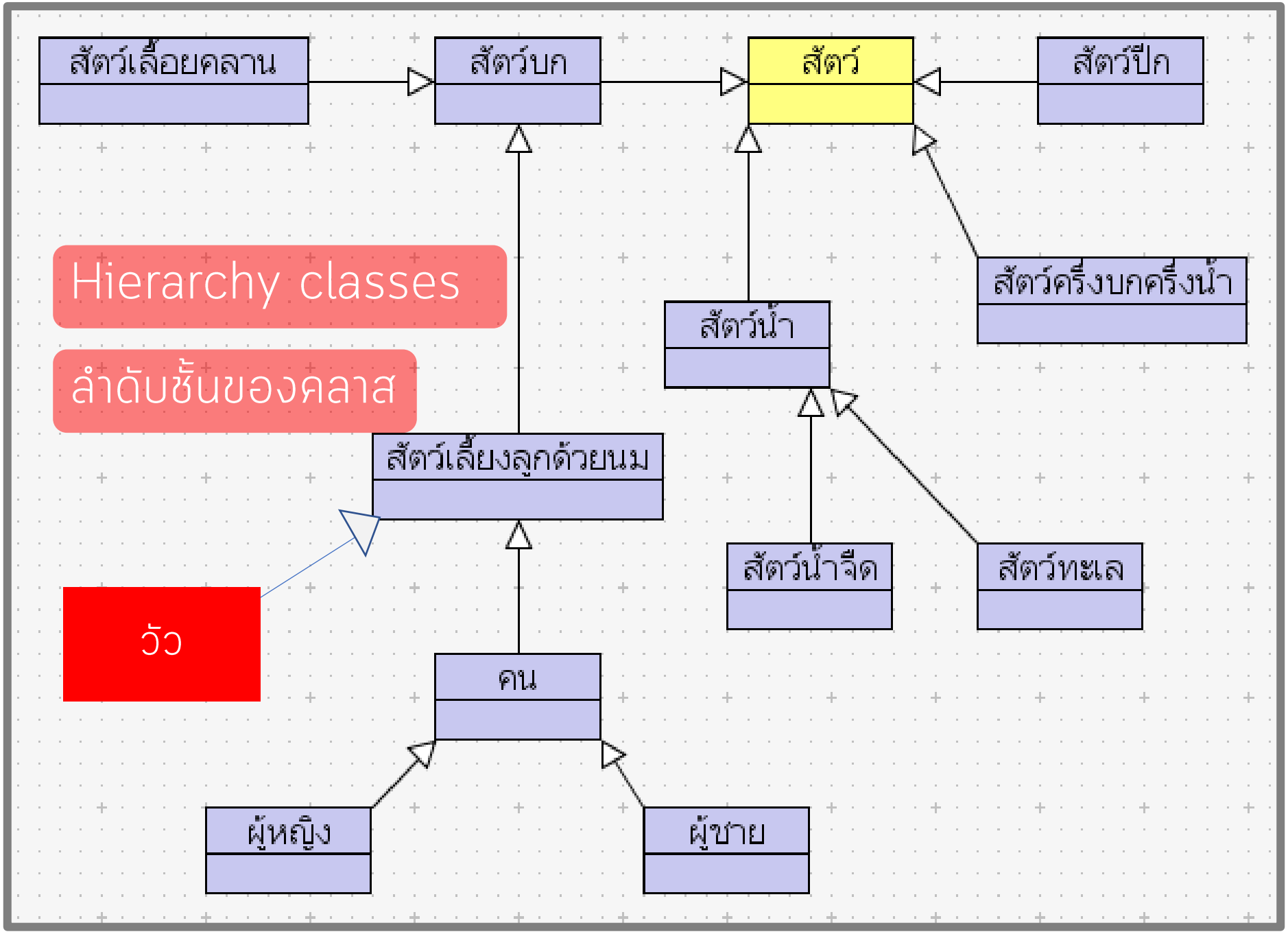

สัตว์ปีก
คุณสมบัติ : มีปีก
มีขา 2 ขา
Specialization
Generalization
Specialization
Generalization
นก
คุณสมบัติ : คุณสมบัติของสัตว์ปีก +
บินได้
ไก่
คุณสมบัติ : คุณสมบัติสัตว์ปีก
+ ขันได้
ไก่พันธ์เนื้อ
คุณสมบัติ : คุณสมบัติไก่ + ? + ?
Generalization
Specialization
ไก่ไข่
คุณสมบัติ : คุณสมบัติไก่ + ? + ?
ไก่ชน
คุณสมบัติ : คุณสมบัติไก่
+ ขันได้ + ชนไก่ได้

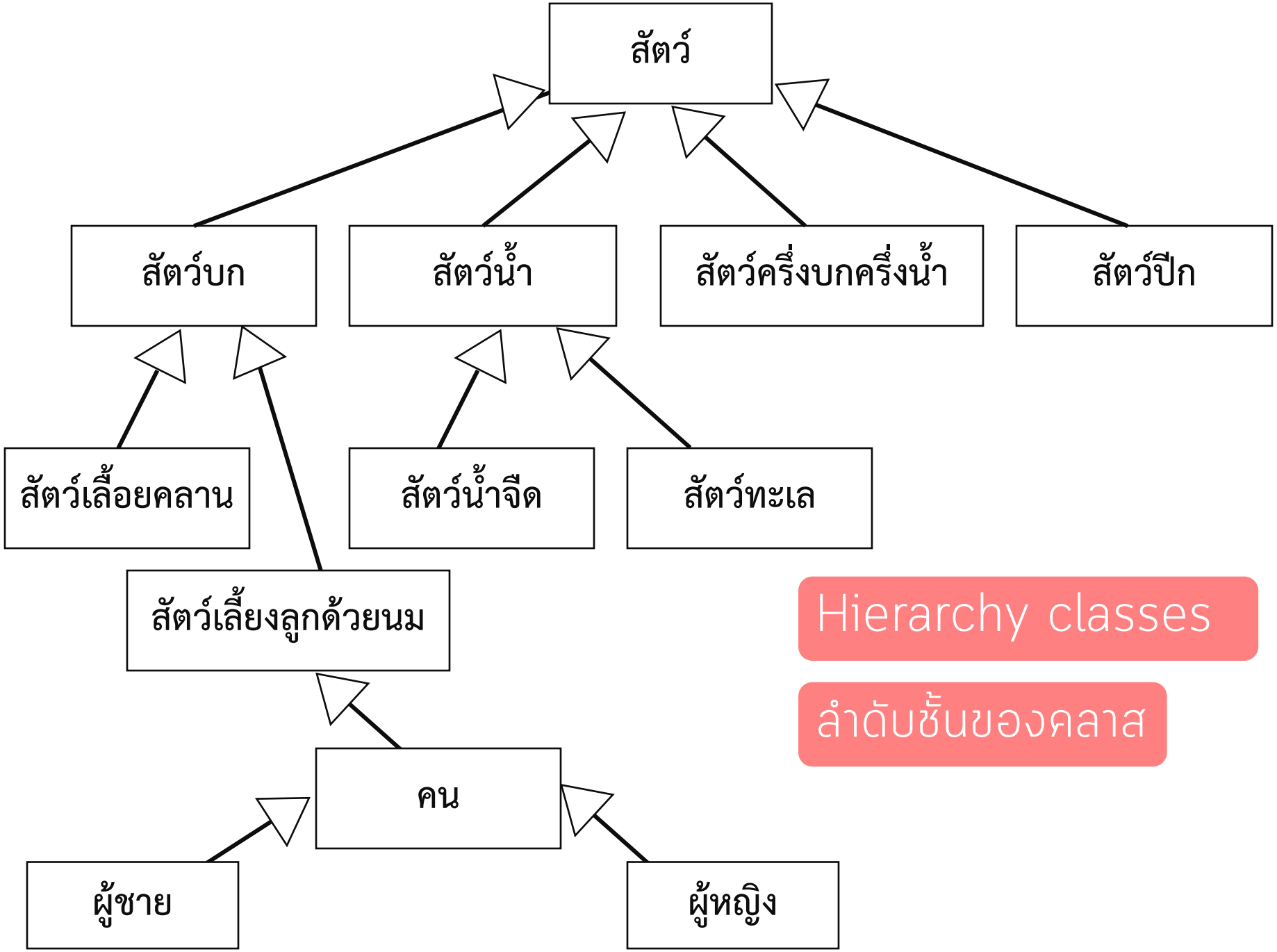

สัตว์น้ำ
- อาศัยในน้ำ
- มีชีวิต
Specialization
Generalization
Specialization
Generalization
ปู = อาศัยอยู่ในน้ำ + มีก้าม
ปลา= อาศัยอยู่ในน้ำ
+ มีครีบ
Problem domain
Generalization
Specialization
???
???
Generalization
Specialization
???
Practice II
จงเขียนความสัมพันธ์ของสัตว์บก
ในเชิง Generalization และ Specialization

ตัวอย่าง ของการทำ Specialize เพื่อทำให้ รถยนต์กลายเป็นรถสปอร์ต และการทำ Generalize เพื่อให้รถสปอร์ตกลายมาเป็นรถยนต์



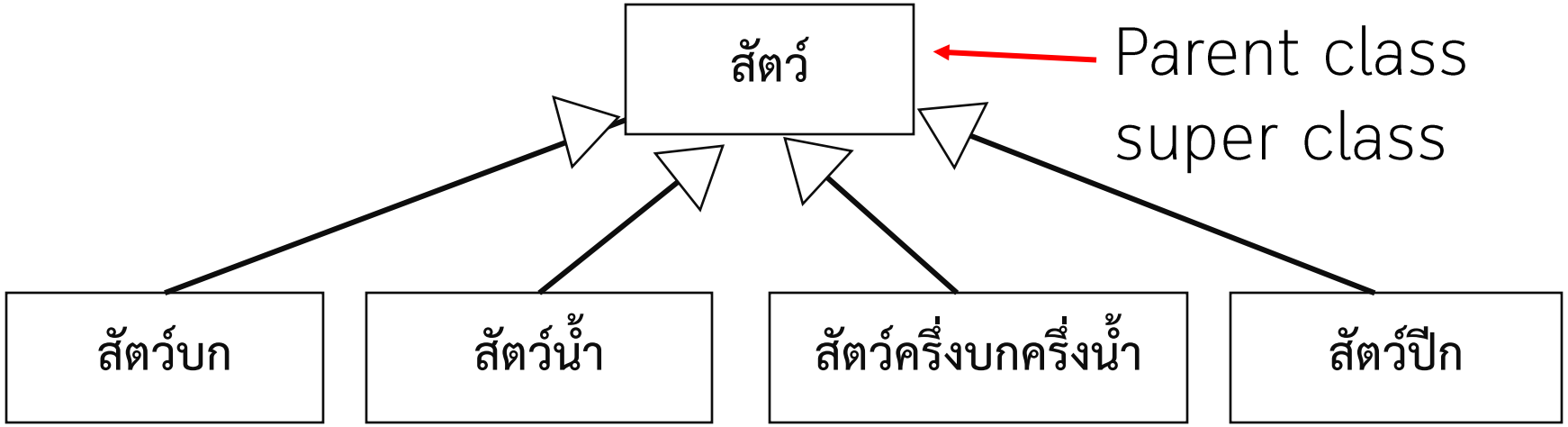
จากรูปจะพบว่า การทำ Specialize เพื่อทำให้ รถยนต์กลายเป็นรถสปอร์ต ทำได้โดยการเพิ่มเติมส่วนของเครื่องยนต์ Turbo และความสามารถในการเปิดประทุนได้เข้าสู่รถยนต์ปกติ และในทางกลับกันการทำ Generalize เพื่อให้รถสปอร์ตกลายเป็นรถยนต์ก็ทำได้โดยการเอาเครื่องยนต์ Turbo และหลังคาเปิดประทุนได้ออกจากรถสปอร์ตนั่นเอง


สัตว์เลื้อยคลาน
สัตว์บก
สัตว์
สัตว์ปีก
Hierarchy classes
ลำดับชั้นของคลาส
สัตว์ครึ่งบกครึ่งน้ำ
สัตว์น้ำ
สัตว์เลี้ยงลูกด้วยนม
สัตว์น้ำจืด
สัตว์ทะเล
วัว
คน
ผู้หญิง
ผู้ชาย



Hierarchy classes

ลำดับชั้นของคลาส



child class
sub class

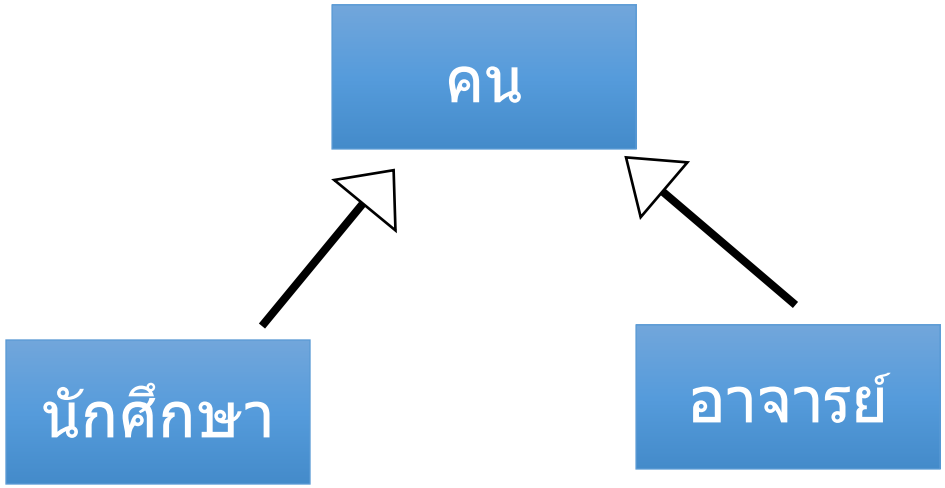
Child class คลาสลูกจะได้รับการถ่ายทอดคุณสมบัติมาจากคลาสพ่อแม่ และเพิ่มเติมความสามารถของคลาสตัวเอง เรียกว่าการทำ inheritance

child class

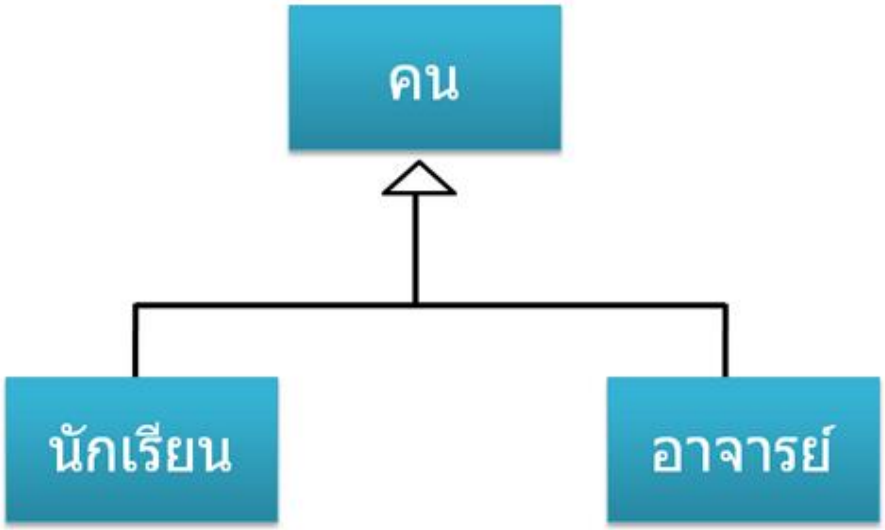
sub class

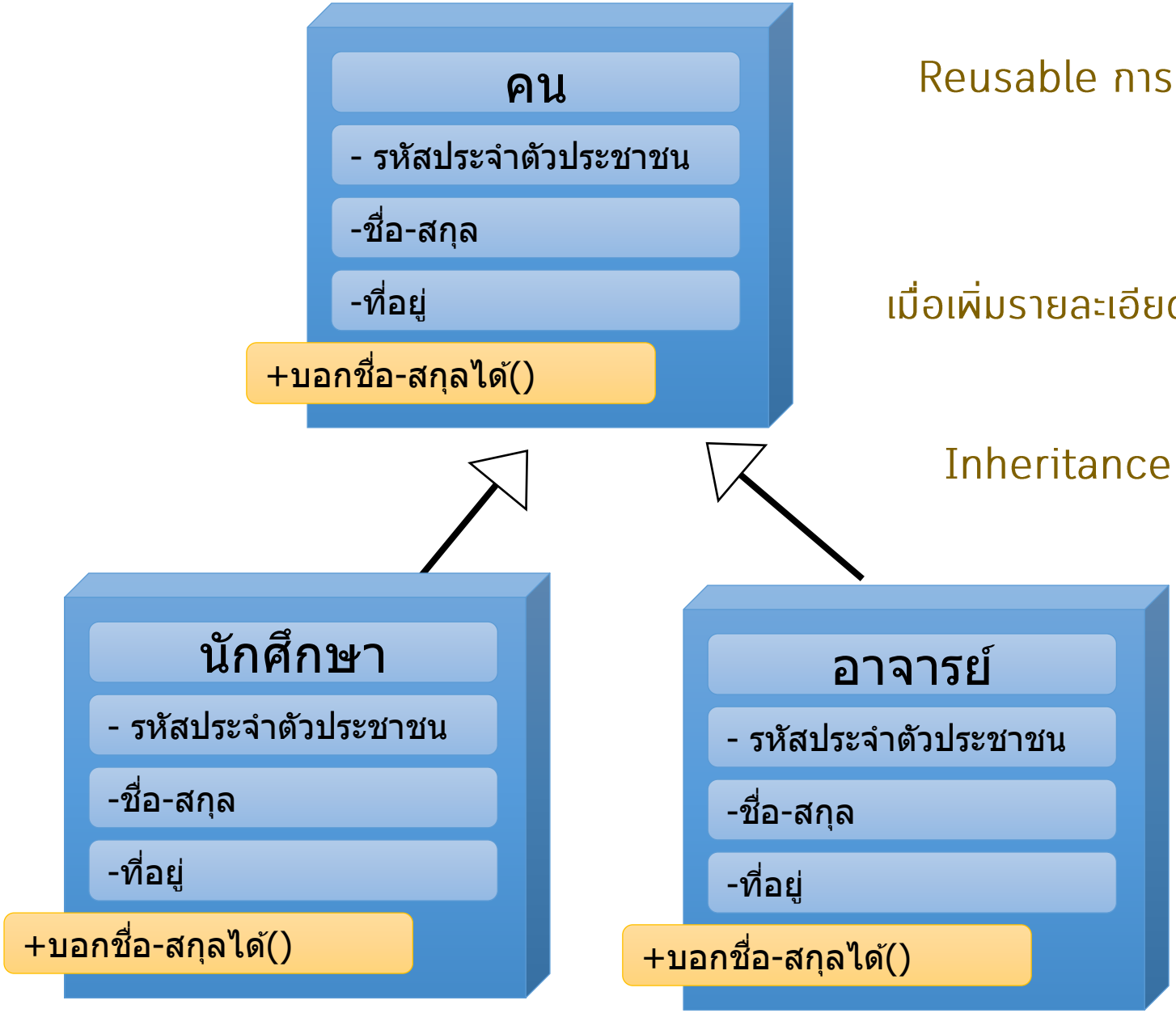
derive class

1



2

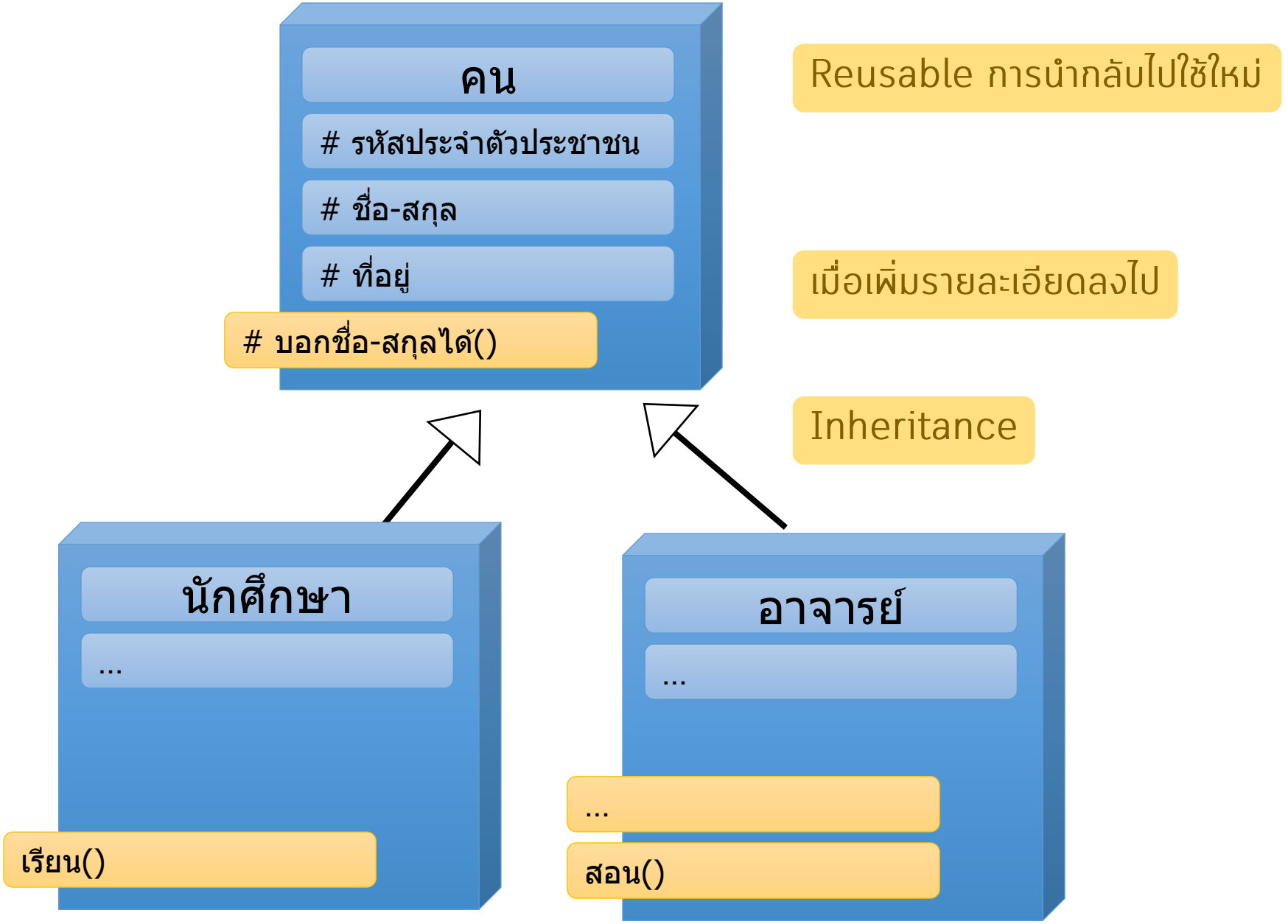




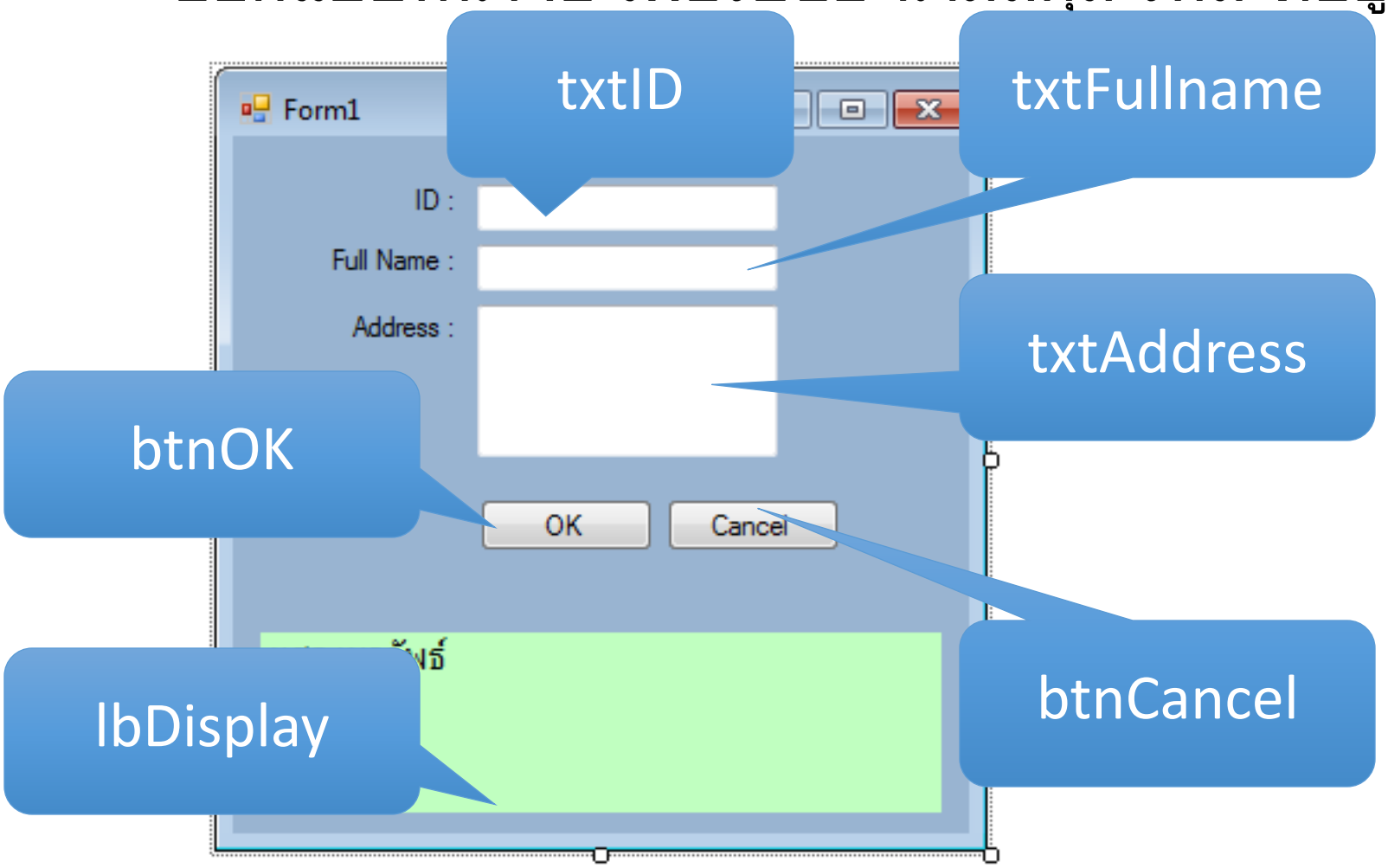
Reusable การนำกลับไปใช้ใหม่

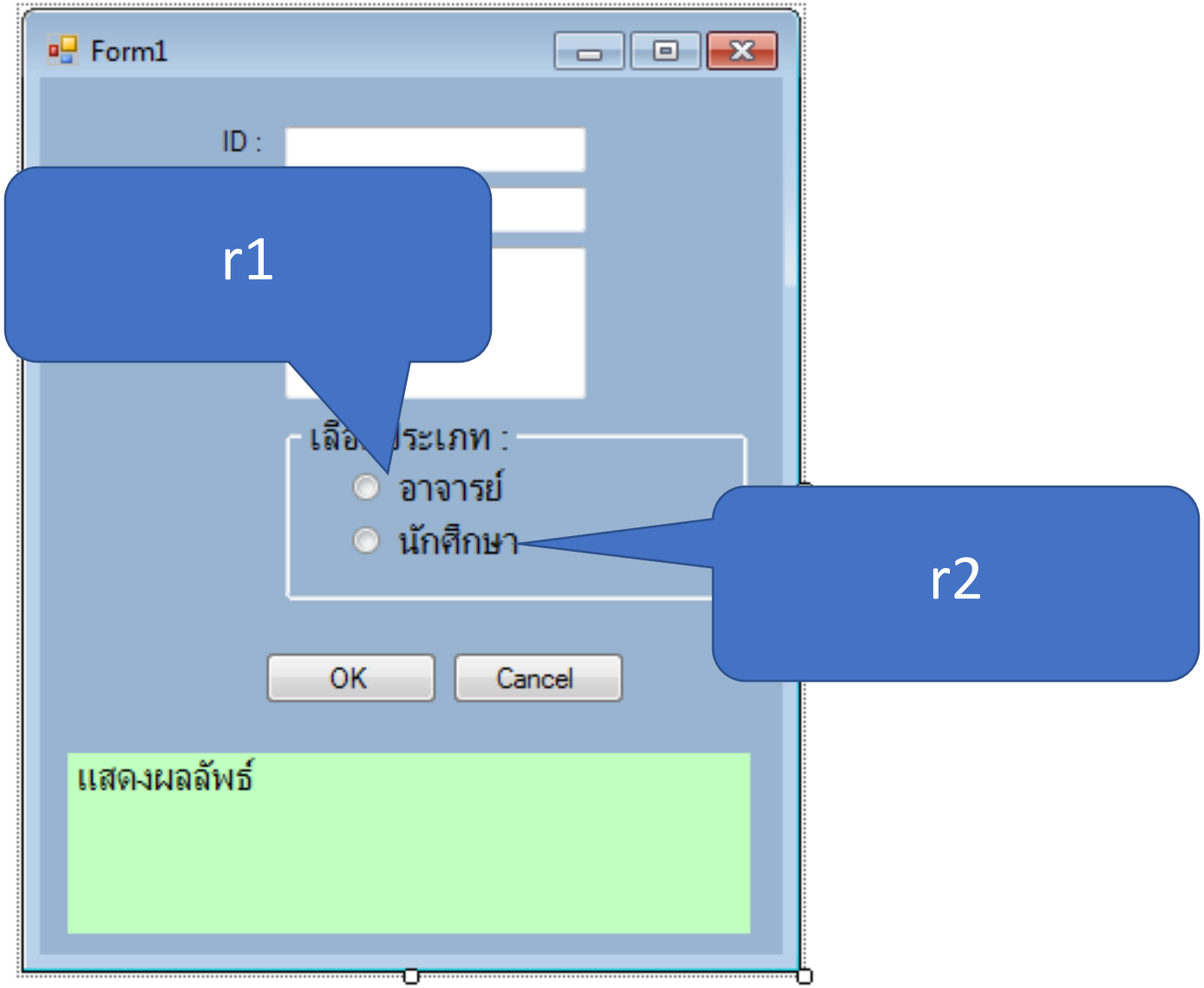
เมื่อเพิ่มรายละเอียดลงไป

Inheritance


คน
รหัสประจำตัวประชาชน
ชื่อ-สกุล
ที่อยู่
บอกชื่อ-สกุลได้()
Reusable การนำกลับไปใช้ใหม่
เมื่อเพิ่มรายละเอียดลงไป
Inheritance
นักศึกษา
...
เรียน()
อาจารย์
...
...
สอน()

ออกแบบหน้าจอ เพื่อรับชื่อ นามสกุล รหัส ที่อยู่

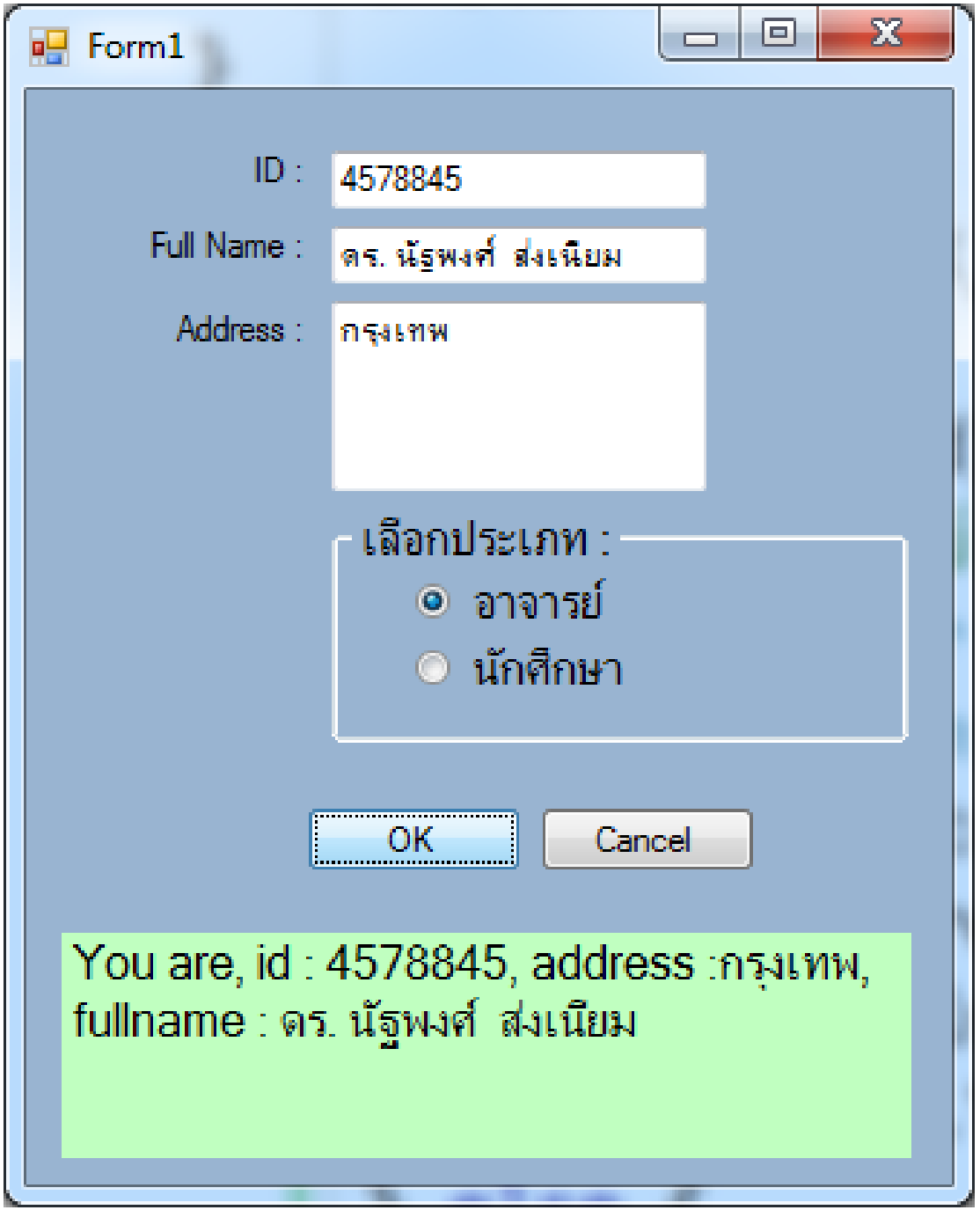




```
19          private void btnOK_Click(object sender, EventArgs e)
20          {
21              if (r1.Checked) {
22
23
24              } else {
25
26
27              }
28          }
29      }
30  }
```

```
19          private void btnOK_Click(object sender, EventArgs e)
20          {
21              if (r1.Checked) {
22              // ถ้าเลือก อาจารย์
23                  Instructor i;
24                  i = new Instructor();
25                  i.set_id(txtID.Text);
26                  i.set_address(txtAddress.Text);
27                  i.set_fullname(txtFullname.Text);
28                  lbDisplay.Text = i.get_fullname();
29              } else {
30
31              }
32          }
33      }
34  }
```

จะเห็นได้ว่า คลาสอาจารย์ สามารถเข้าถึงหรือเรียกใช้ attribute / function จากคลาสพ่อแม่ หรือคลาส person ได้โดยไม่ต้องมาสร้างใหม่

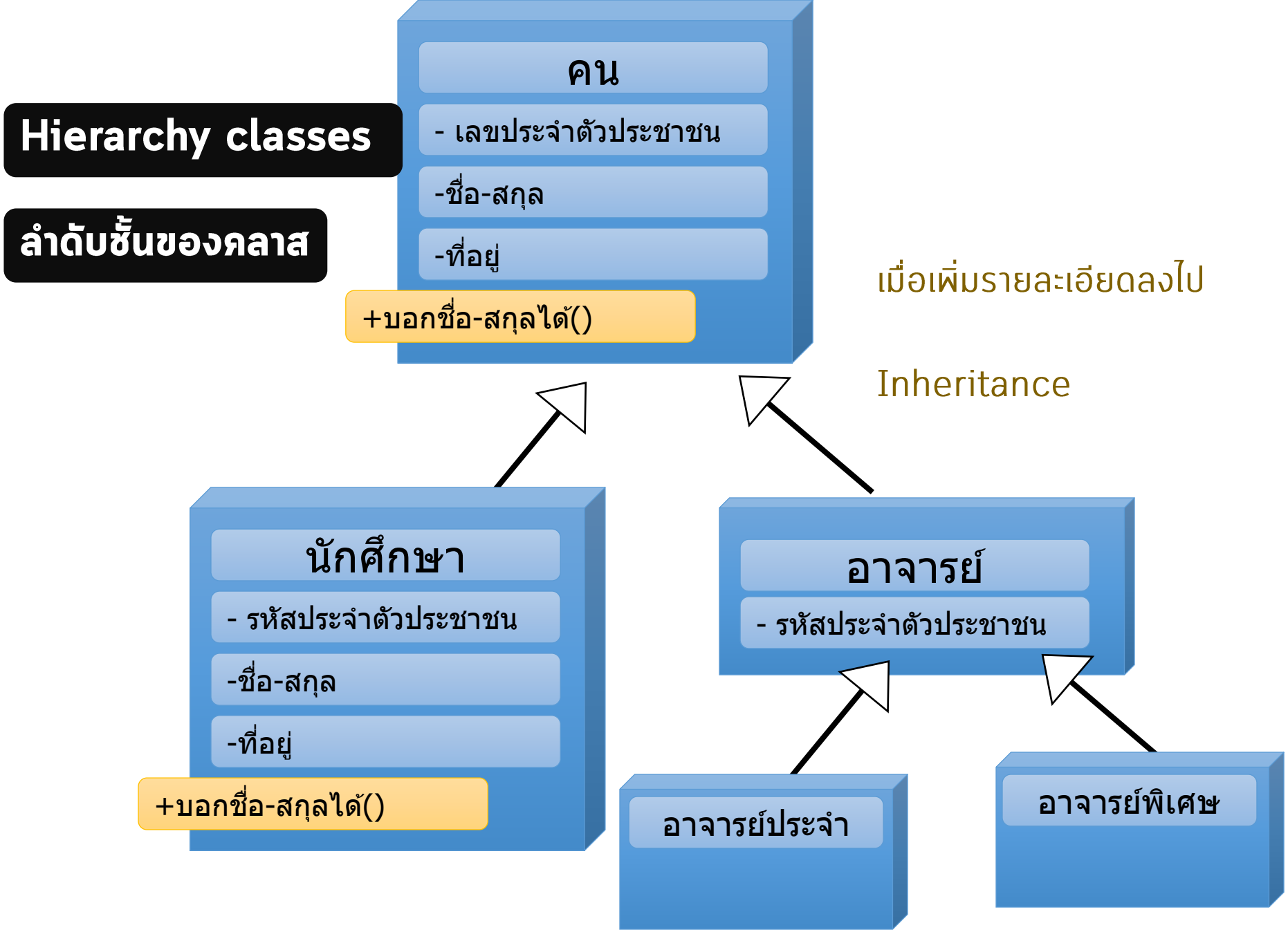

Form1
ID :
4578845
Full Name :
ดร. นัฐพงศ์ ส่งเนียม
Address :
กรุงเทพ
เลือกประเภท :
อาจารย์
นักศึกษา
OK
Cancel
You are, id : 4578845, address :กรุงเทพ,
fullname : ดร. นัฐพงศ์ ส่งเนียม

ถ้าเลือกนักศึกษา

```
28                  lbDisplay.Text = i.get_fullname();
29              } else {
30                  // ถ้าเลือก นักศึกษา
31                  student i;
32                  i = new student();
33                  i.set_id(txtID.Text);
34                  i.set_address(txtAddress.Text);
35                  i.set_fullname(txtFullname.Text);
36                  lbDisplay.Text = i.get_fullname();
37              }
38          }
39      }
```

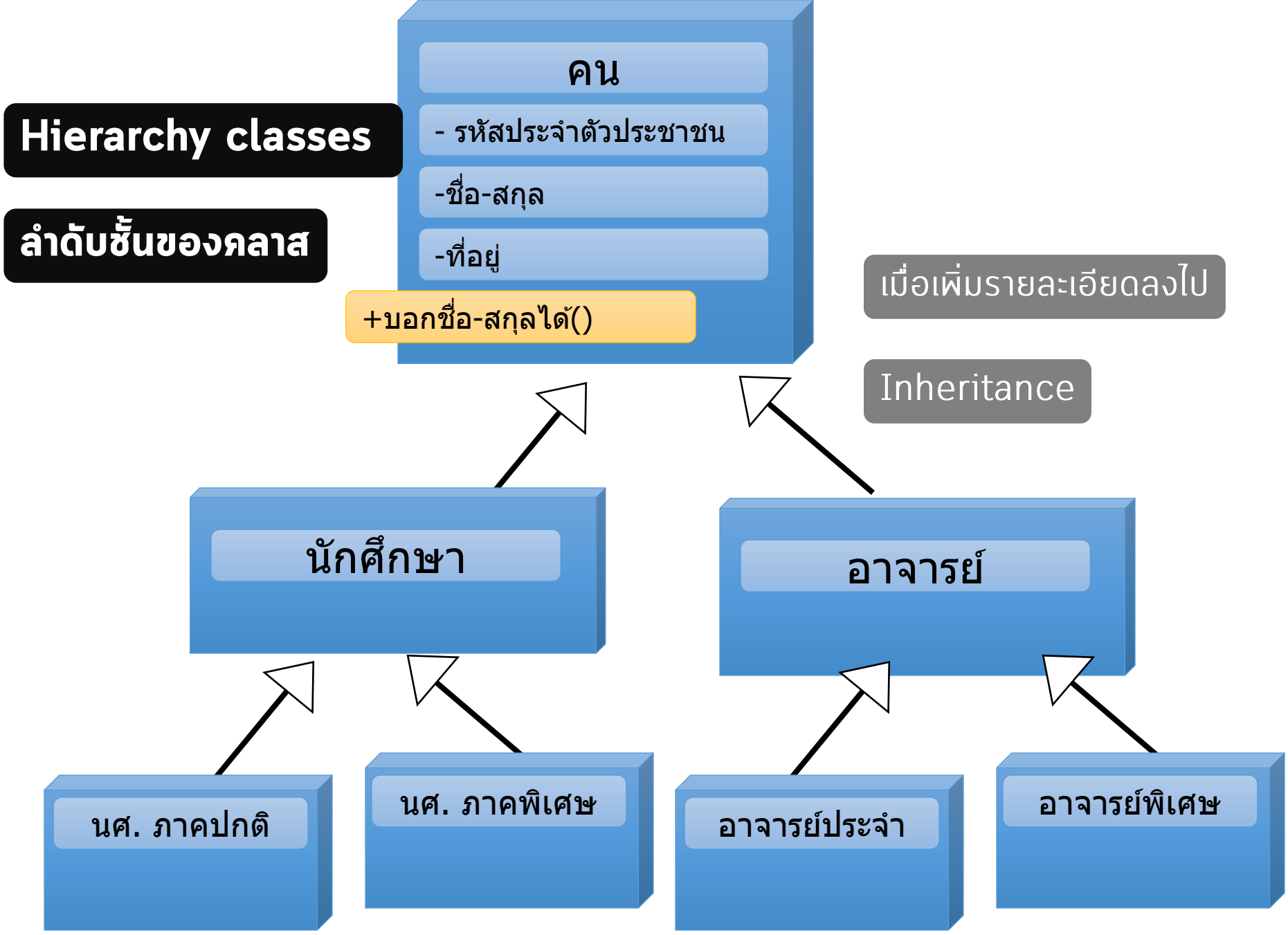
Hierarchy classes

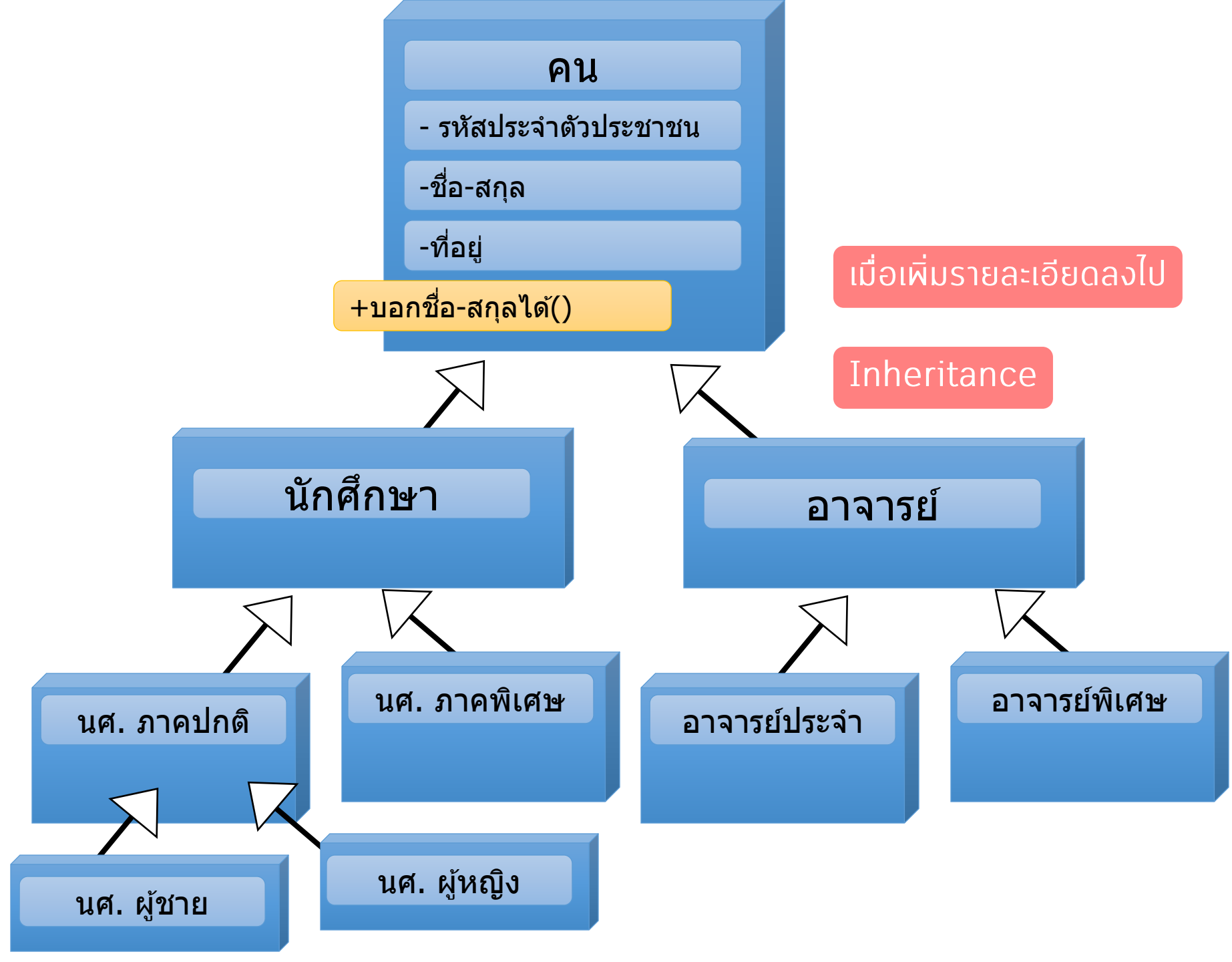
ลำดับชั้นของคลาส



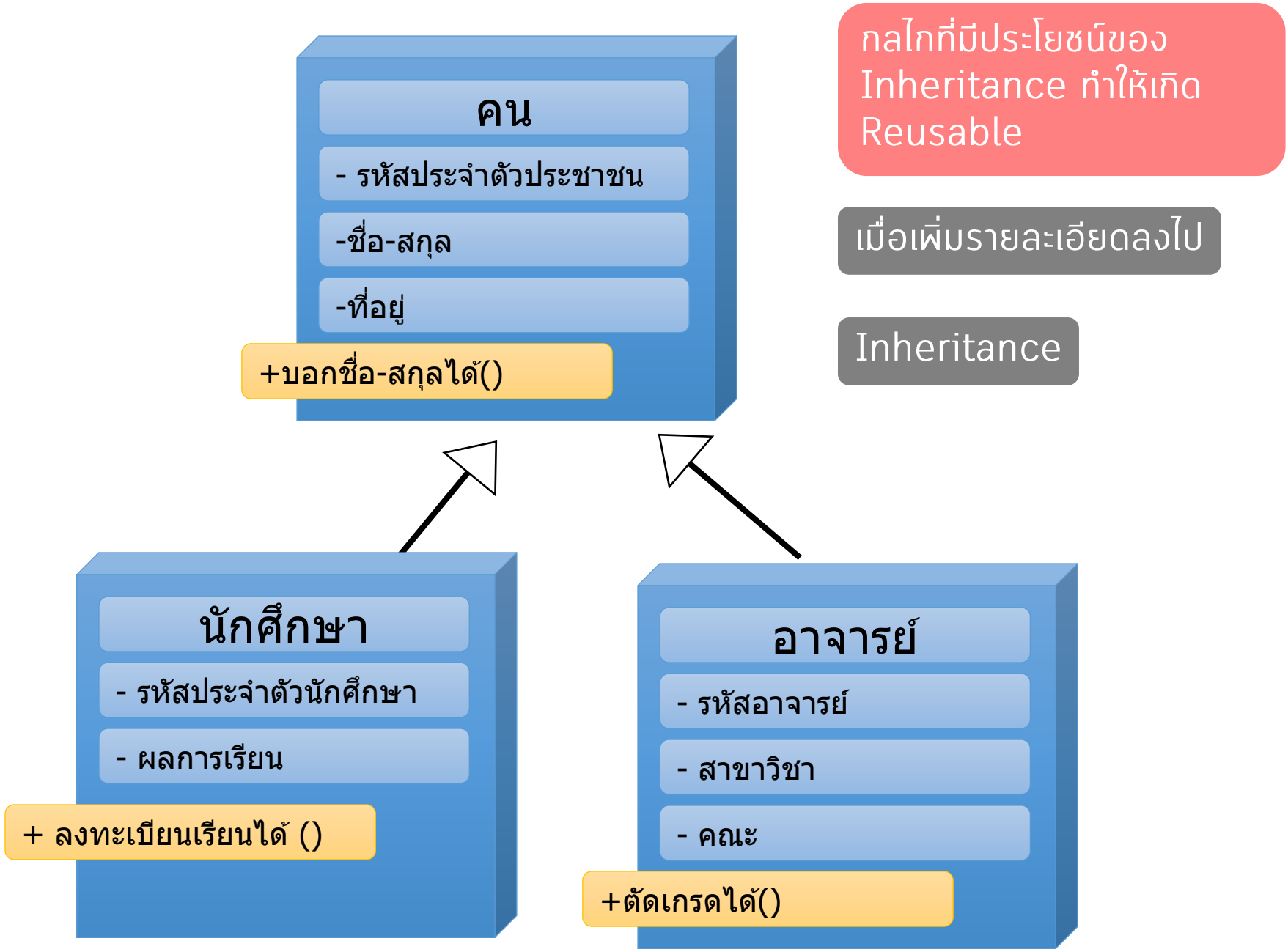
Hierarchy classes

ลำดับชั้นของคลาส




คน
- รหัสประจำตัวประชาชน
-ชื่อ-สกุล
-ที่อยู่
+บอกชื่อ-สกุลได้()
เมื่อเพิ่มรายละเอียดลงไป
Inheritance
นักศึกษา
อาจารย์
นศ. ภาคปกติ
นศ. ภาคพิเศษ
อาจารย์ประจำ
อาจารย์พิเศษ
นศ. ผู้ชาย
นศ. ผู้หญิง

```
25          }
26      }  // จบคลาส person
27
28      class Instructor : person {}  // คลาสอาจารย์
29      class student : person { }  // คลาสนักศึกษา
30
31      class main_Instructor : Instructor { } // อาจารย์ประจำ
32      class special_Instructor : Instructor { } // อาจารย์พิเศษ
33
34  }
35
```


คน
- รหัสประจำตัวประชาชน
-ชื่อ-สกุล
-ที่อยู่
+บอกชื่อ-สกุลได้()
นักศึกษา
- รหัสประจำตัวนักศึกษา
- ผลการเรียน
+ ลงทะเบียนเรียนได้ ()
อาจารย์
- รหัสอาจารย์
- สาขาวิชา
- คณะ
+ตัดเกรดได้()
กลไกที่มีประโยชน์ของ Inheritance ทำให้เกิด Reusable
เมื่อเพิ่มรายละเอียดลงไป
Inheritance

แบบฝึกหัด

จงสร้างแผนภาพเพื่อแสดง Generalization / Specialization จาก Problem Domain ที่กำหนดให้ต่อไปนี้

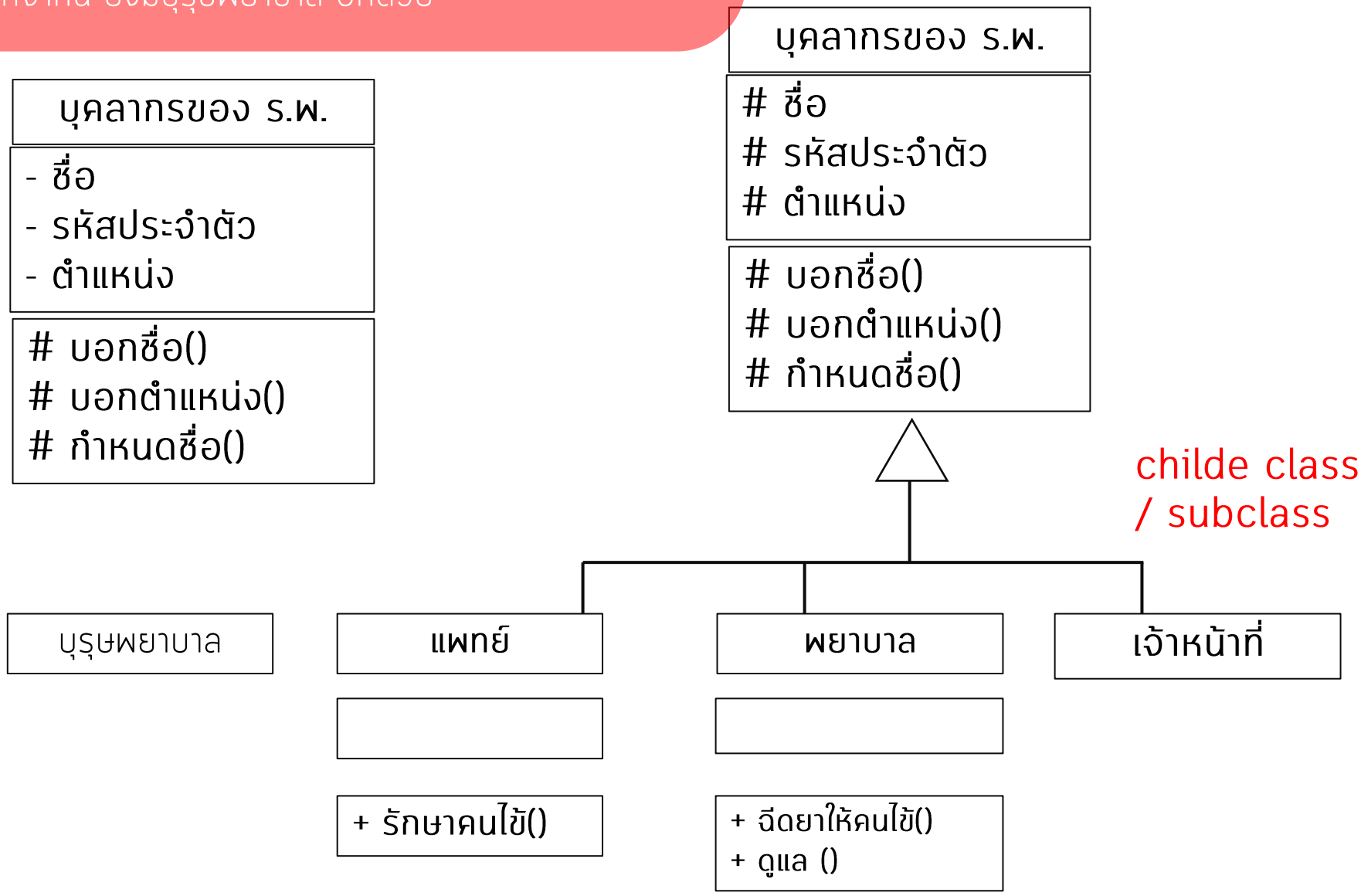
Problem Domain 1
“โรงพยาบาลแห่งหนึ่งมีบุคลากรอยู่ 4 ประเภทดังนี้
1. แพทย์ 2. พยาบาล 3. คนไข้ 4. เจ้าหน้าที่”
นอกจากนี้ ยังมีบุรุษพยาบาล อีกด้วย

Problem Domain 2
“หากเราจะจำแนกประเภทของพนักงานในบริษัทสามารถแบ่งออกเป็น พนักงานเต็มเวลา และพนักงานพาร์ทไทม์”

Problem Domain 1

“โรงพยาบาลแห่งหนึ่งมีบุคลากรอยู่ 4 ประเภทดังนี้
1. แพทย์ 2. พยาบาล 3. คนไข้ 4. เจ้าหน้าที่”
นอกจากนี้ ยังมีบุรุษพยาบาล อีกด้วย

Parent class / superclass

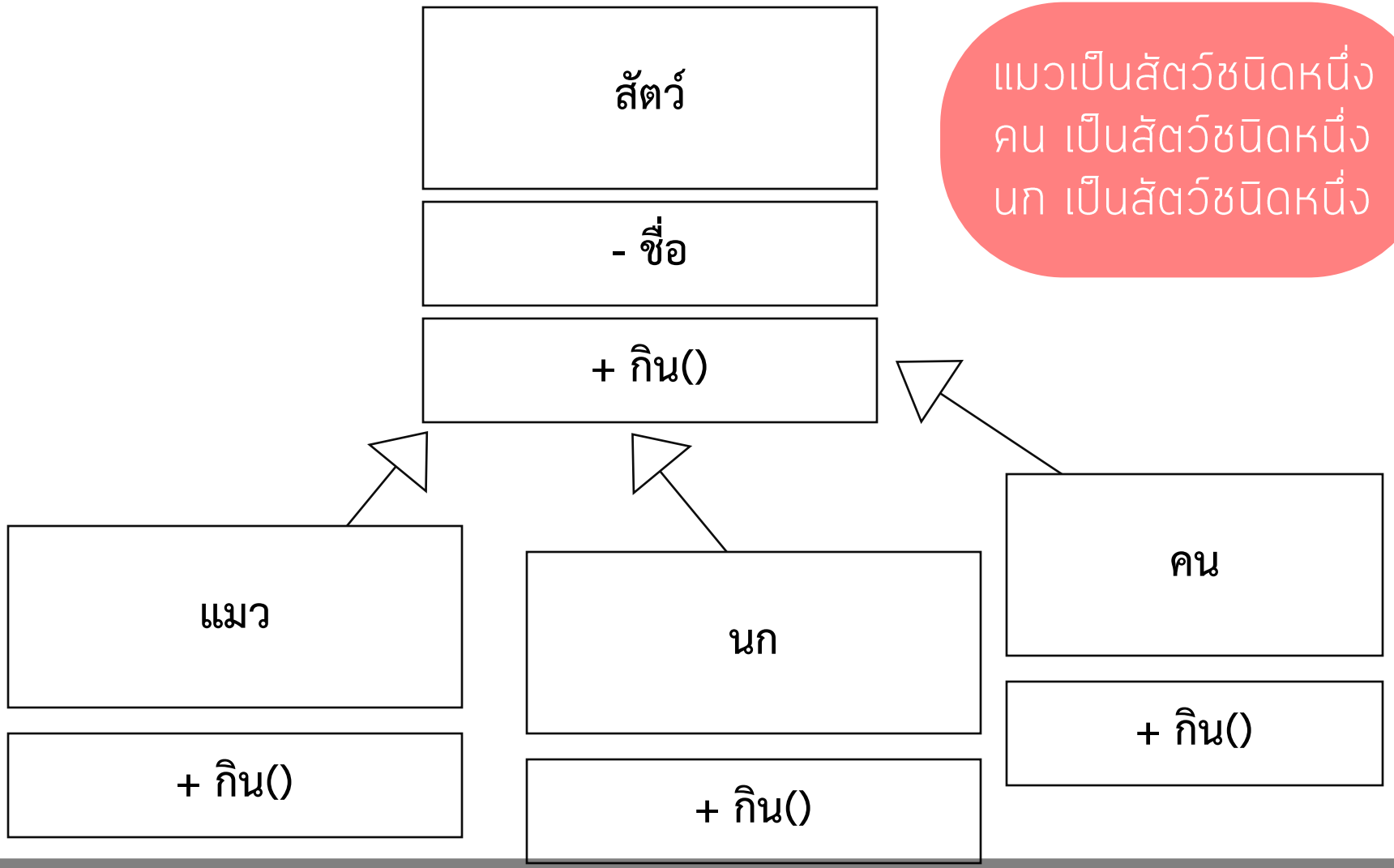


Generalization Implementation in OOP

- แนวคิดตามหลักการเชิงวัตถุ
  - Generalization จัดการ classes ให้อยู่ในรูปของโครงสร้างลำดับชั้น (class hierarchy) ขึ้นกับ similarities และ differences
  - เราเรียกคลาสที่อยู่ในระดับที่สูงกว่าของ classes hierarchy ว่า “superclasses” และที่อยู่ในระดับต่ำกว่าว่า “subclasses”
  - ความสัมพันธ์เป็นแบบ “is kind-of” relationship

Inheritance and Polymorphism

- “subclass” รับถ่ายทอดคุณสมบัติ (inherits) อันได้แก่ attributes, operations และ associations มาจาก “superclass” ของตัวเอง
- แต่อย่างไรก็ตาม ถ้า attribute หรือ operation ของ “superclass” ถูกกำหนดให้ค่าใหม่ใน “subclass” จะเป็นการ “overrides” คำนิยามที่กำหนดไว้ใน “superclass”
- ซึ่งจะนำไปสู่แนวคิดของ polymorphism



*** ดังนั้น คน ต้องมีการปรับวิธีกิน () ให้เหมาะกับ คน โดยใช้ช้อน หรือ ซ้อมในการกิน เรียกว่าการ override method

การทำ Override ทำให้เกิด Polymorphism

Superclasses and Subclasses

- “Superclass” มีคำนิยาม attributes, operations และ associations ร่วมกันกับ “subclasses” ของคลาสนั้นๆ
- “Subclasses” มี attributes, operations และ associations เฉพาะเป็นของตัวเอง โดยเลือกที่จะกำหนดนิยามใหม่ให้กับ attribute, operation หรือ relationship ที่รับมาจาก “superclass” หรือไม่ก็ได้

**** subclass รับคุณสมบัติมาจากพ่อแม่ จะมีการปรับเปลี่ยนหรือไม่ก็ได้*

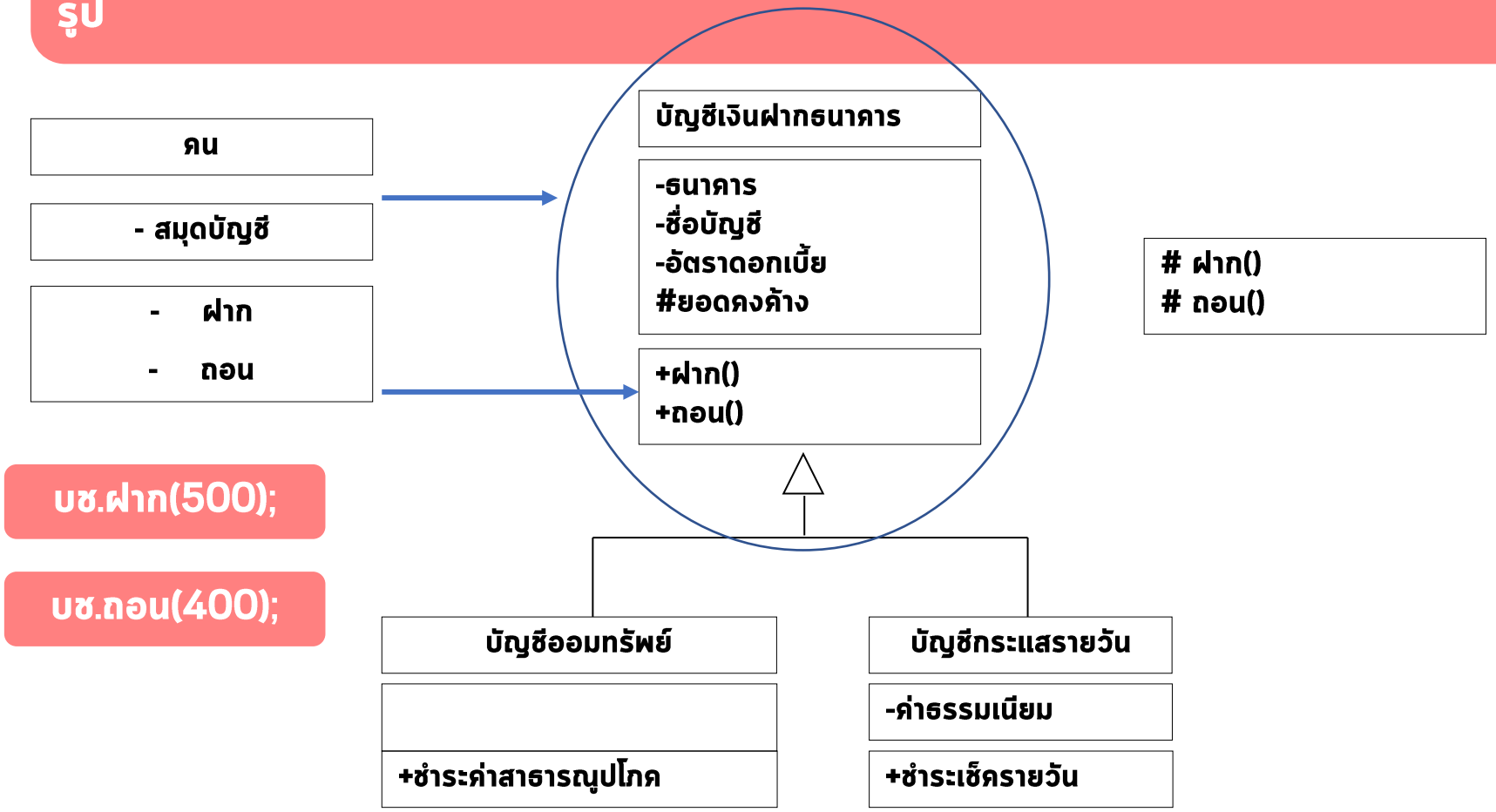
Inheritance - กลไกที่เกิดจาก Specialization Abstraction

Subclass Superclass และ Inheritance

จากหัวข้อที่ผ่านมา การทำ Specialize เกิดจาก Class เดิม หรือเรียกอีกอย่างหนึ่งว่า Class เริ่มต้น หรือ Superclass หรือ Seed Class (Seed หมายถึง เมล็ดพันธุ์) และ Class ที่เกิดจากการทำ Specialize นั้นเรียกว่า Subclass ในทาง Object Orientation เราเรียกกระบวนการ Specialization ว่า Inheritance (มาจาก Inherit หมายถึง การถ่ายทอด เช่น การถ่ายทอดทางพันธุกรรม เป็นต้น)

*** ข้อควรจำ ในการทำ Inheritance คือ Subclass ที่ Inherit มาจาก SuperClass นั้น จะต้องมีคุณสมบัติทุกอย่างของ Superclass (จะขาดคุณสมบัติใดๆ ของ Superclass ไม่ได้) ผนวกกับคุณสมบัติพิเศษที่เพิ่มเข้าไปในแต่ละ Subclass เสมอ

ในทาง Object Orientation เราใช้สัญลักษณ์ลูกศรซึ่งหัวลูกศรเป็นรูปสามเหลี่ยมใสชี้จาก Subclass ไปยัง Superclass เพื่อแสดงการทำ Inheritance ดังรูป



บช.ฝาก(500);

บช.ถอน(400);

กฎเกณฑ์ของการทำ Inheritance

การทำ Inheritance นั้นเป็นการถ่ายทอดคุณสมบัติทุกอย่างจาก Superclass ไม่ว่าจะเป็น Attribute หรือ Function แต่มีข้อควรจำว่า Visibility ของ Attributes หรือ Functions นั้นมีความสัมพันธ์กับการทำ Inheritance เสมอนั่นคือ

1. Private Attributes/Functions จะถ่ายทอดมาเป็น Private Attributes/Functions ของ Subclass แต่ส่วนที่ Inherit มาจาก Private Attributes/Functions มายัง Subclass จะไม่สามารถเข้าถึงได้โดย Function ที่มีอยู่ใน Subclass แต่ไม่ได้มาจากการ Inherit

2. Protected Attributes/Functions ของ Superclass จะถ่ายทอดมาเป็น Protected Attributes/Functions ของ Subclass อนึ่ง การเข้าถึง Attributes และ Functions ของ Subclass ที่เกิดจากการ Inherit ในกรณีนี้จะทำได้ โดยผ่าน Function ใด ๆ ของ Subclass นั้น โดยไม่คำนึงว่าจะเป็น Function ที่ได้มาจากการ Inherit หรือไม่

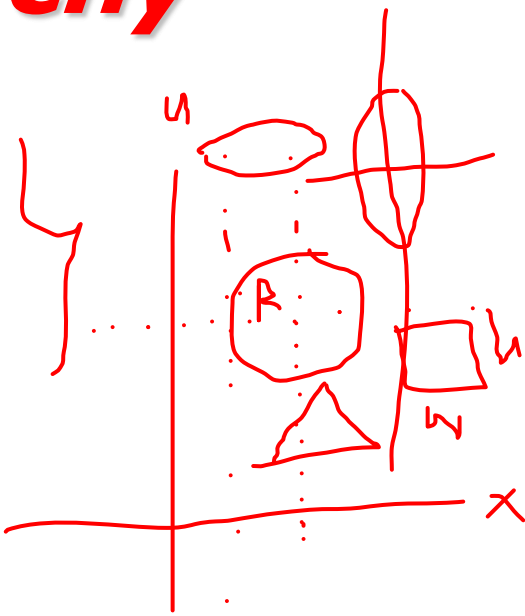
3. Public Attributes/Functions จะถ่ายทอดมาเป็น Public Attributes/Functions ของ Subclass เสมอ

Superclass

Subclass

บัญชีเงินฝากธนาคาร	บัญชีออมทรัพย์	บัญชีกระแสรายวัน
-ธนาคาร -ชื่อบัญชี -อัตราดอกเบี้ย #ยอดคงค้าง	-ธนาคาร -ชื่อบัญชี -อัตราดอกเบี้ย #ยอดคงค้าง	-ธนาคาร -ชื่อบัญชี -อัตราดอกเบี้ย #ยอดคงค้าง -ค่าธรรมเนียม
+ฝาก +ถอน	+ฝาก +ถอน +ชำระค่าสาธารณูปโภค	+ฝาก +ถอน +ชำระค่าสาธารณูปโภค

A Generalization Hierarchy



Superclass

Geometric_Figure

#X_Position
#Y_Position

Ellipse

- Major_Axis
- Minor_Axis

Circle

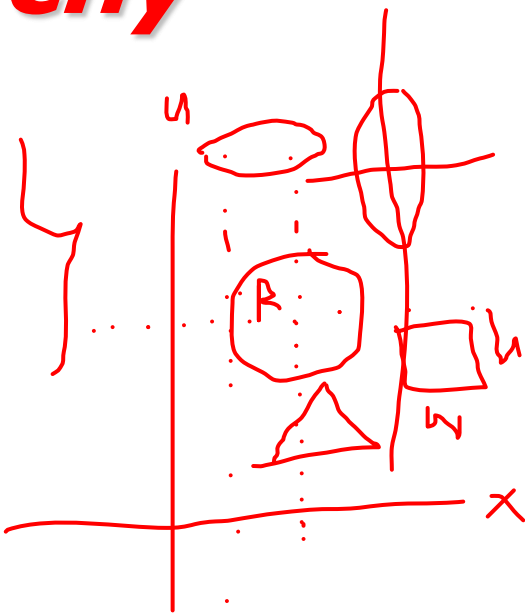
- Radius

Rectangle

- Width
- Height

Subclasses

A Generalization Hierarchy



Superclass

Geometric_Figure

#X_Position
#Y_Position

Ellipse

- Major_Axis
- Minor_Axis

Circle

- Radius

Rectangle

- Width
- Height

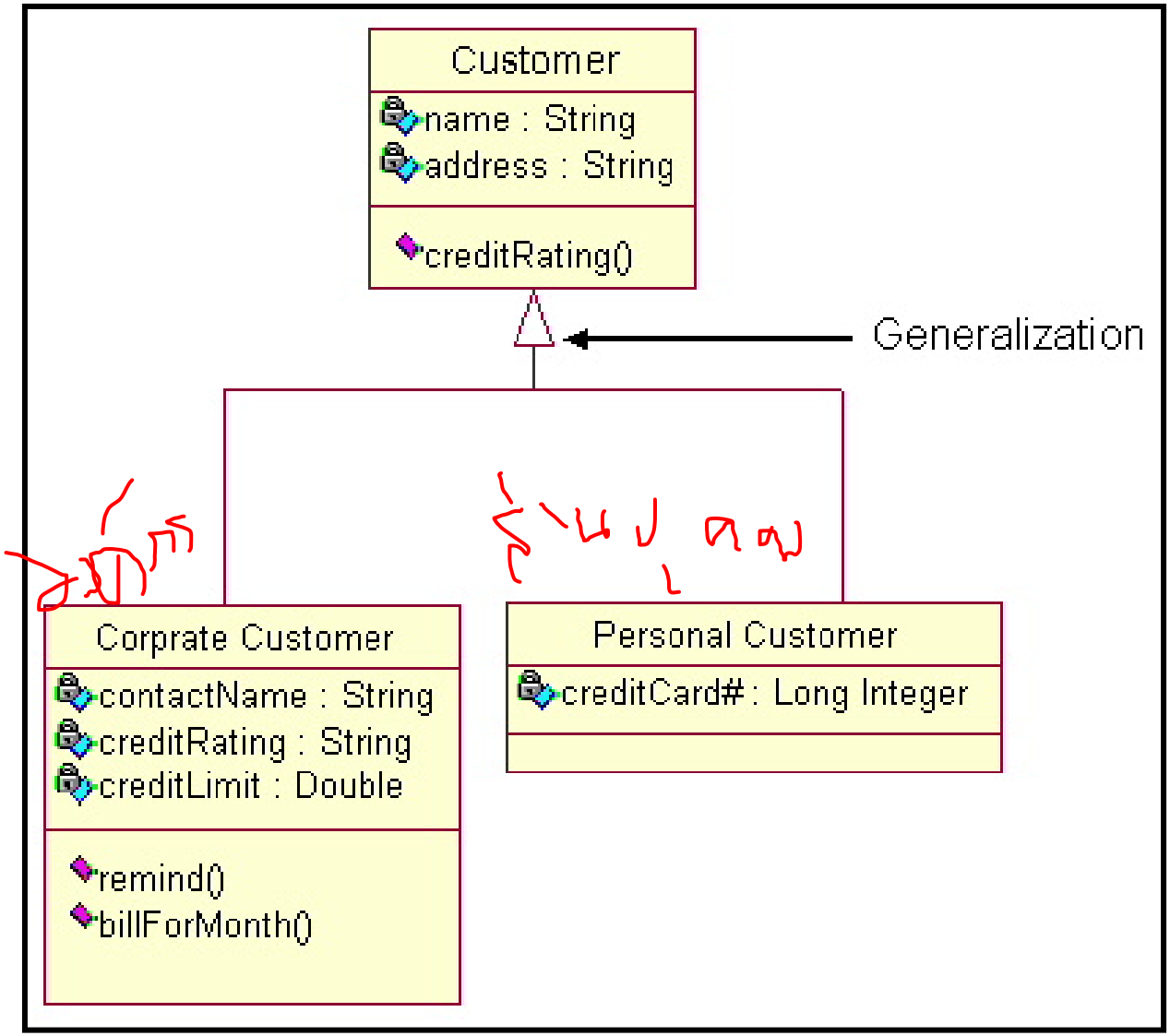
calculateArea()

calculateArea()

calculateArea()

Subclasses

Class Diagrams-inheritance



Class diagrams-inheritance

A simple example of inheritance in Java:

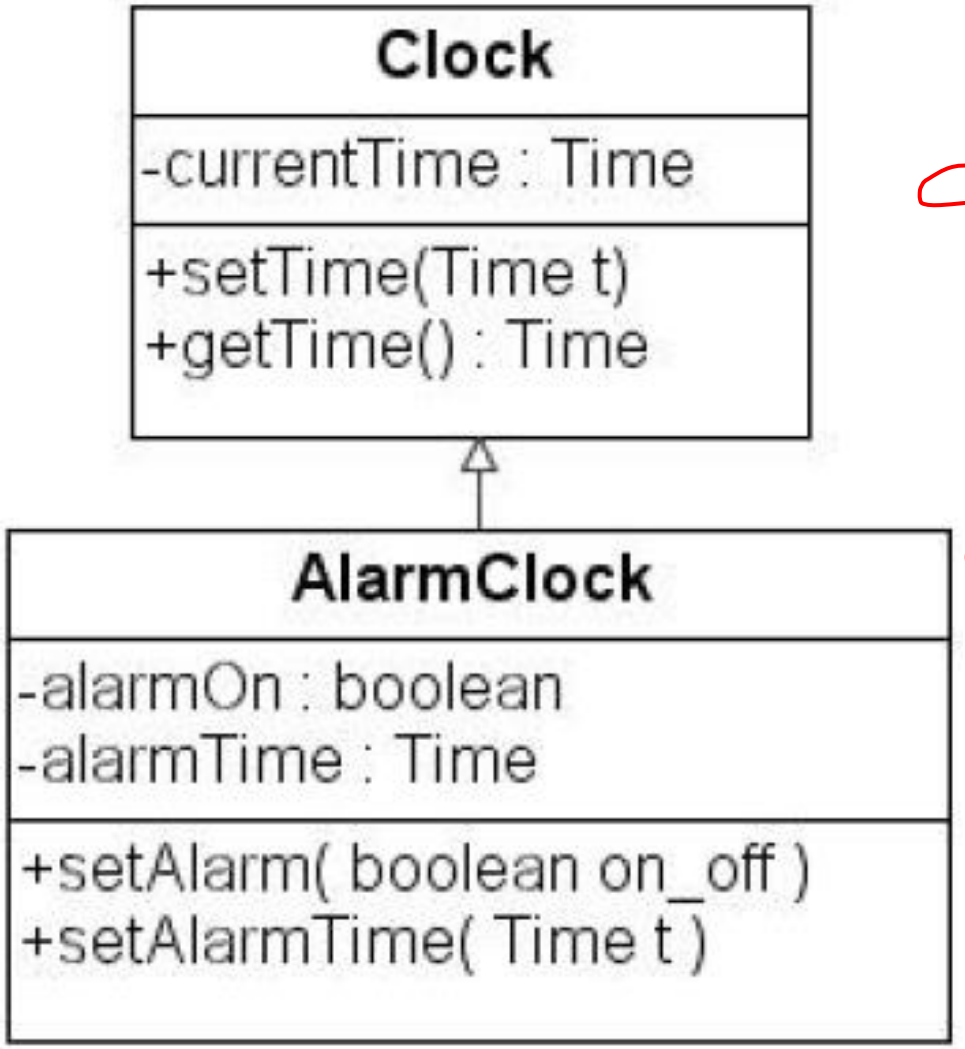
```
class Clock {
    private Time currentTime;
    public void setTime(Time t) { ... }
    public Time getTime() { ... }
    }

class AlarmClock extends Clock {
    private Time alarmTime;
    private boolean alarmOn;
    public void setAlarmTime(Time t) { ... }
    public void setAlarm(boolean on_off) { ... }
    }
```

Clock

Inheritance

Alarm Clock

Class diagrams-inheritance

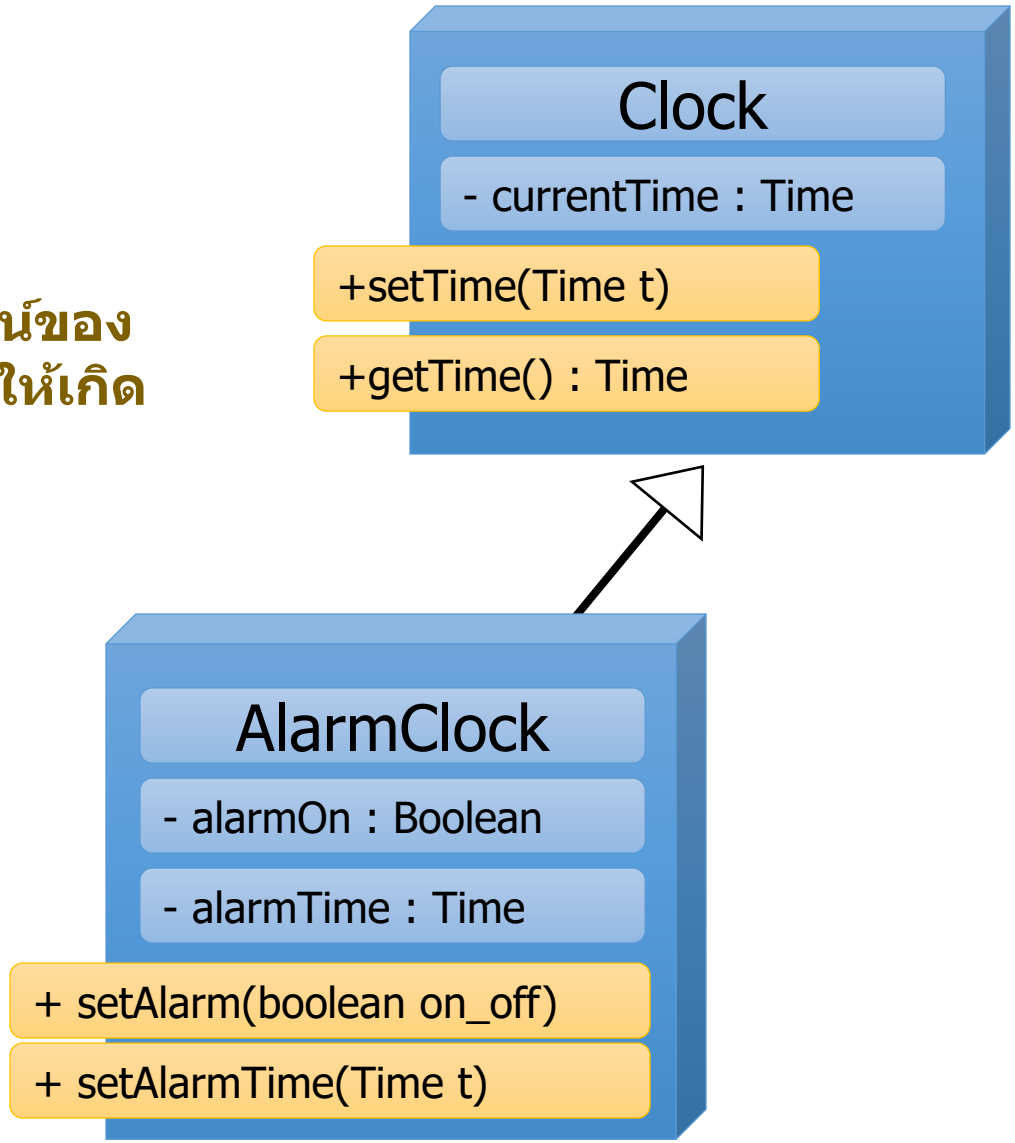


Super Class

sub/ child Class

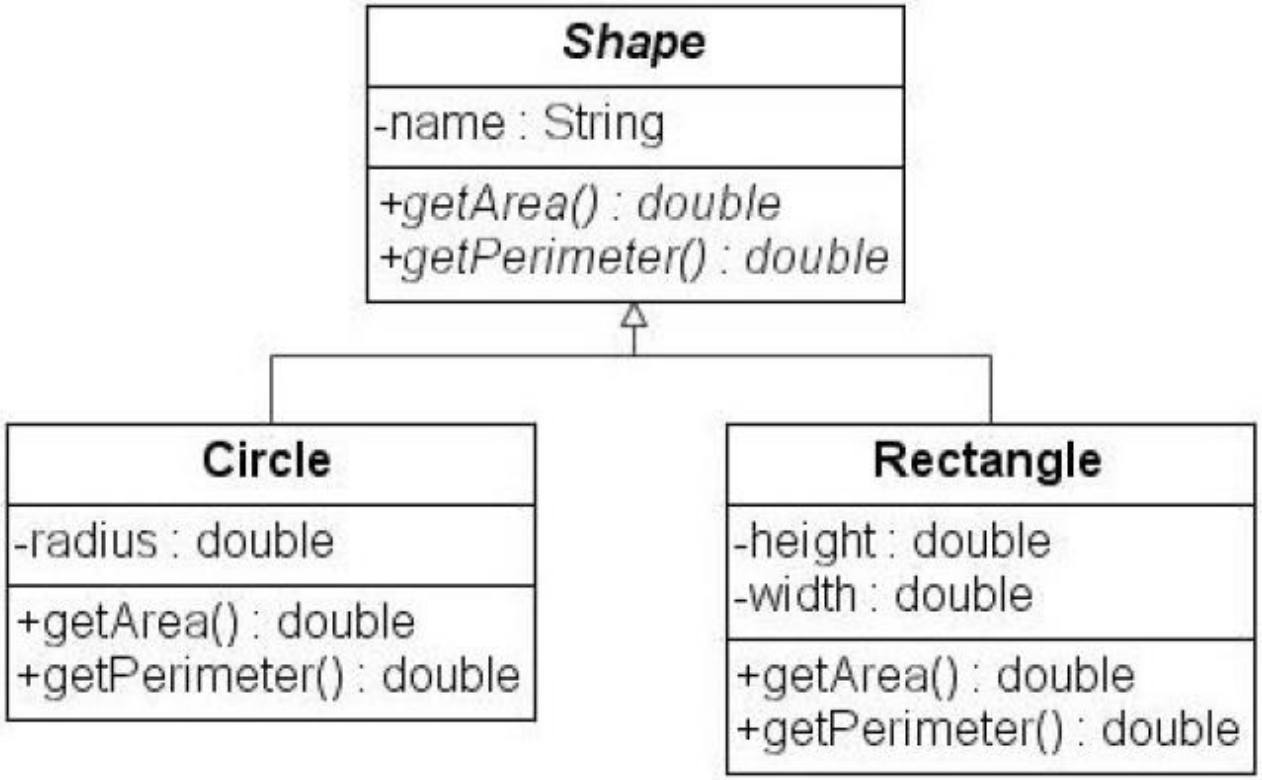
Class diagrams-inheritance

กลไกที่มีประโยชน์ของ Inheritance ทำให้เกิด Reusable



Class diagrams-inheritance

Abstract classes and methods are indicated as being such by italicizing the name of the class or method:



Inheritance

- Class Diagram สามารถแสดงการสืบทอดคลาสได้ เพื่อลดความซ้ำซ้อนในการอธิบายข้อมูล ดังนี้

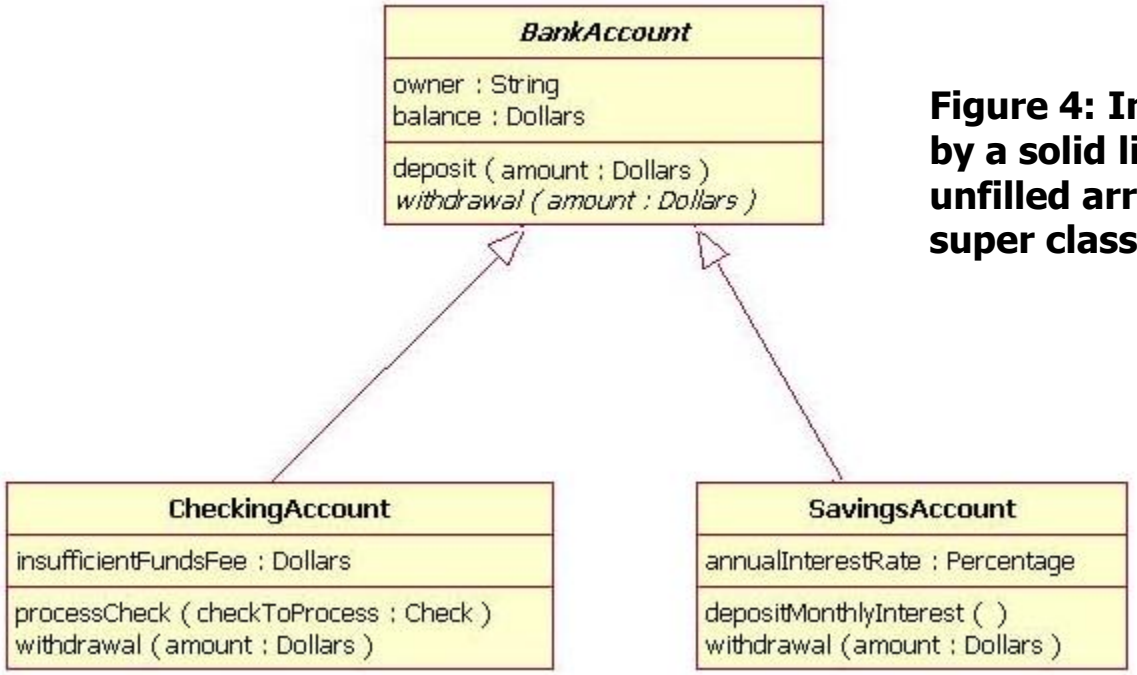
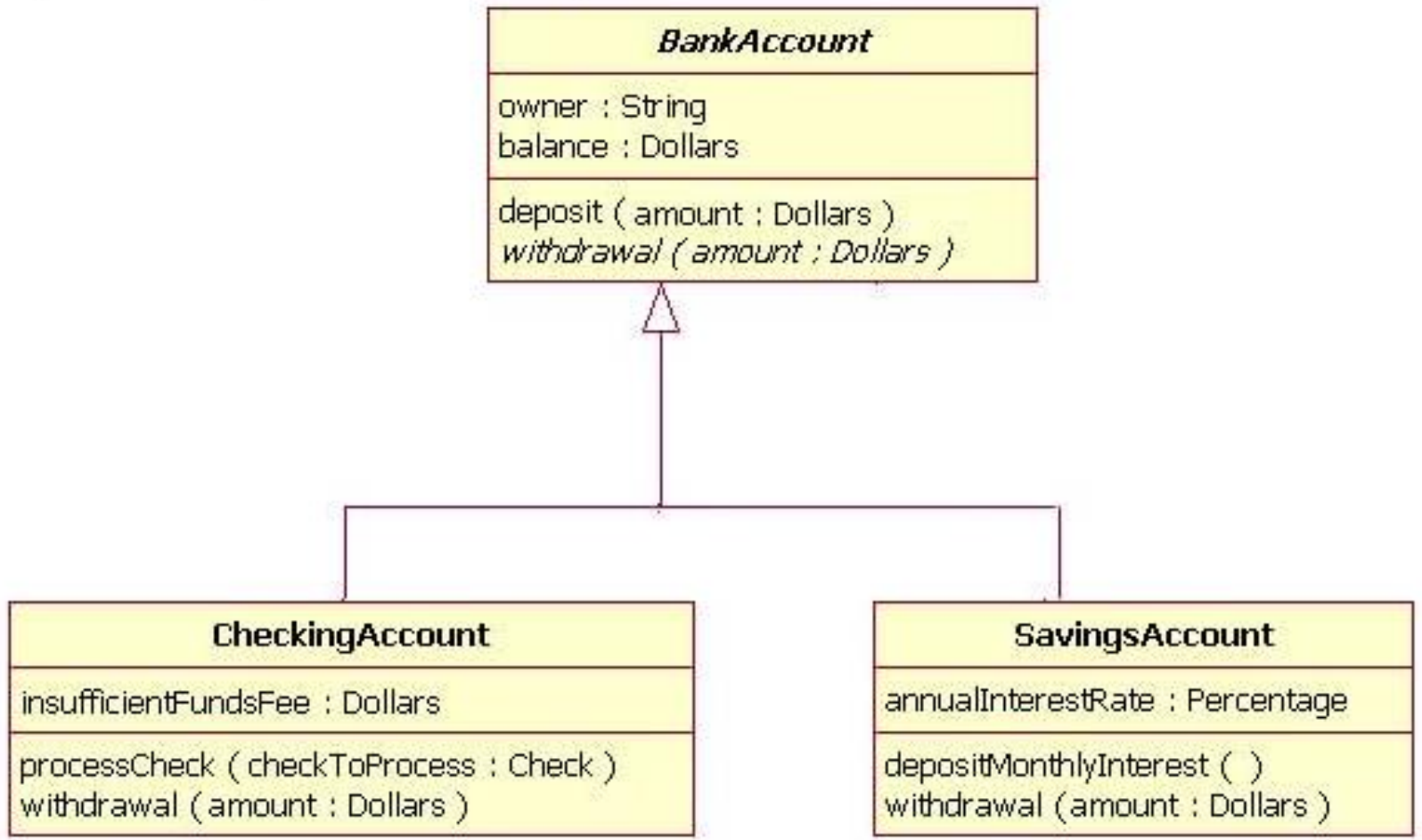
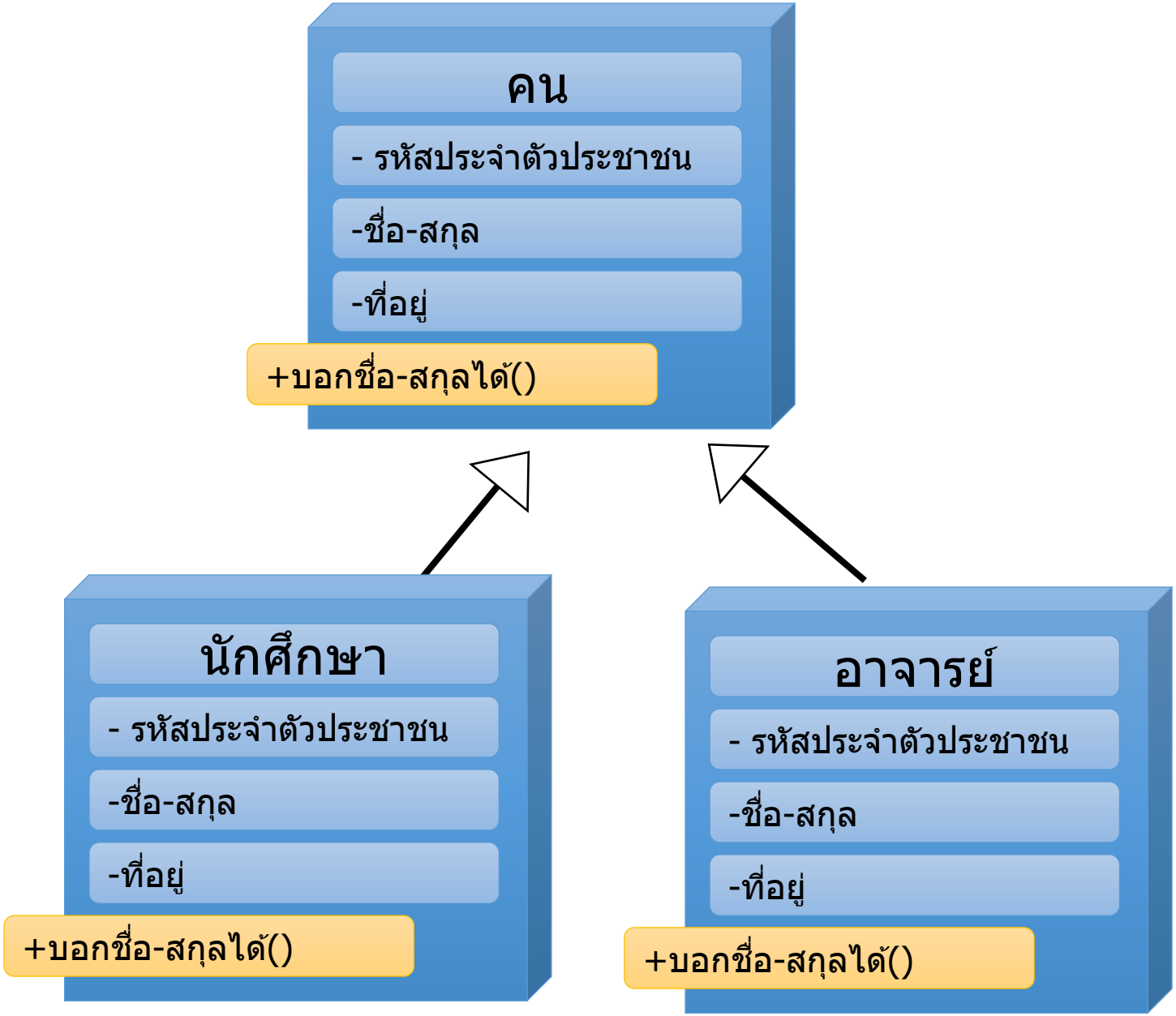
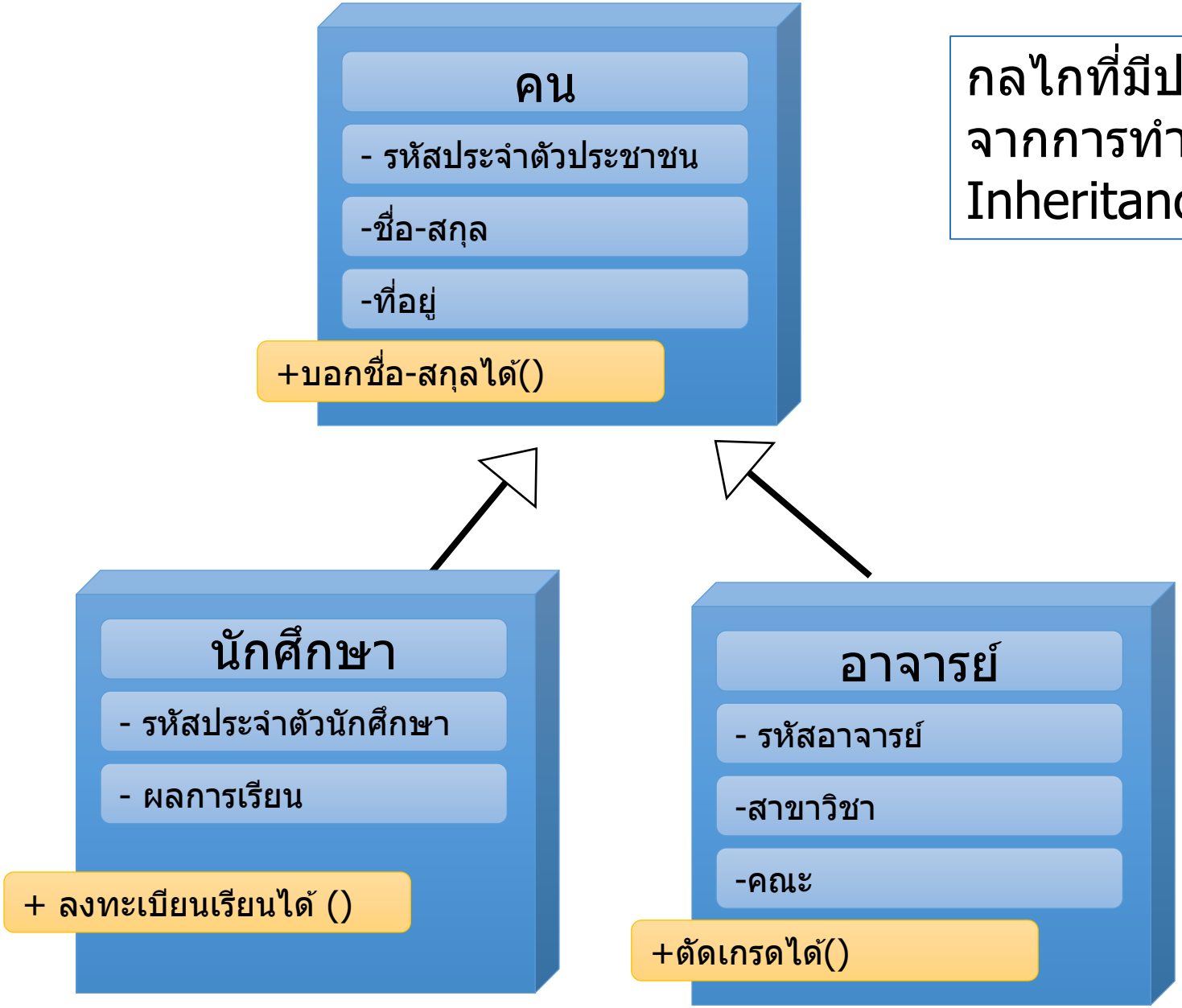


Figure 4: Inheritance is indicated by a solid line with a closed, unfilled arrowhead pointing at the super class

Figure 5: An example of inheritance using tree notation




คน
- รหัสประจำตัวประชาชน
-ชื่อ-สกุล
-ที่อยู่
+บอกชื่อ-สกุลได้()
นักศึกษา
- รหัสประจำตัวประชาชน
-ชื่อ-สกุล
-ที่อยู่
+บอกชื่อ-สกุลได้()
อาจารย์
- รหัสประจำตัวประชาชน
-ชื่อ-สกุล
-ที่อยู่
+บอกชื่อ-สกุลได้()



กลไกที่มีประโยชน์จากการทำ Inheritance

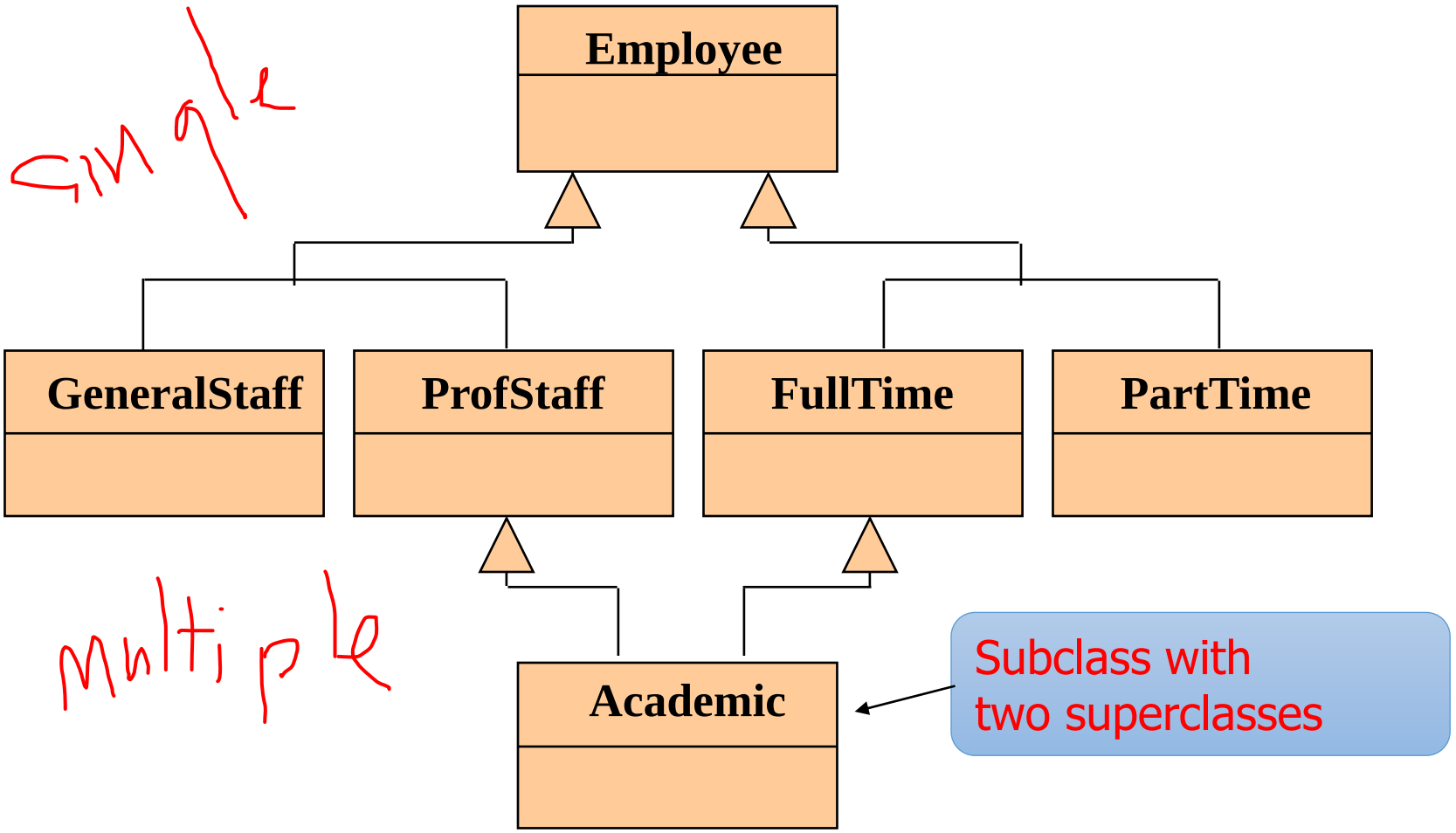
Single & Multiple Inheritance

- **generalizations มักเกิดใน class hierarchies ที่แต่ละ "subclass" มีเพียง 1 "superclass" เรียกว่า "single inheritance"**

- **แต่มีบางสถานการณ์ที่ "subclass" อาจมีได้มากกว่า 1 "superclass" เรียกว่า "multiple inheritance"**

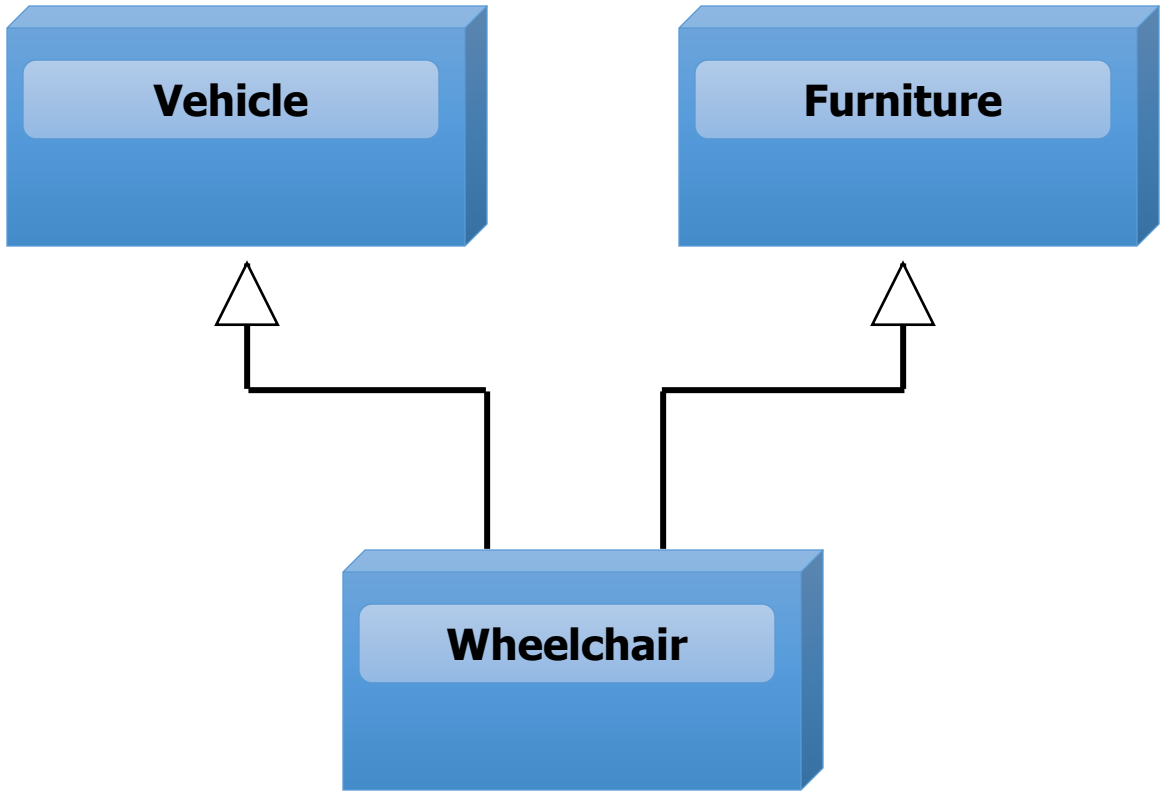
*** นักวิชาการ เป็นทั้งสองอย่าง คือ ได้รับการสืบทอดมาจากสองคลาส
คือเป็นได้ทั้ง พนังงานเต็มเวลา และพนักงานชำนาญการ

Multiple Inheritance

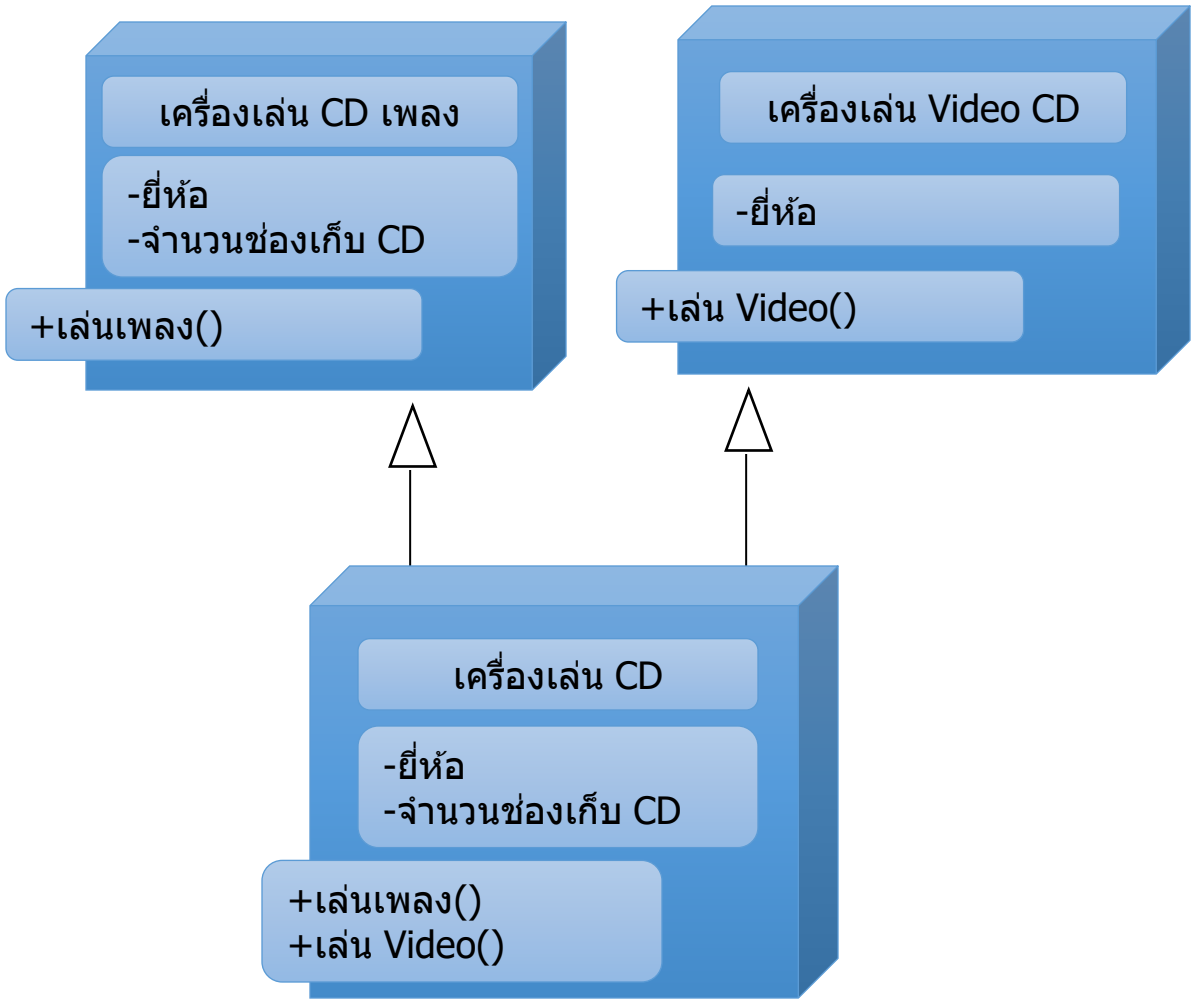


Multiple Inheritance

*** วีลแชร์ จัดว่าเป็นทั้ง เฟอร์นิเจอร์ และ เป็นยานพาหนะ



Multiple Inheritance การทำ Inheritance ยังมีกรณีที่เราทำ Inherit จาก Superclass ที่มากกว่า 1 ตัว เพื่อให้ได้ Subclass ที่มีคุณสมบัติพิเศษเพียงตัวเดียว หรือมากกว่า

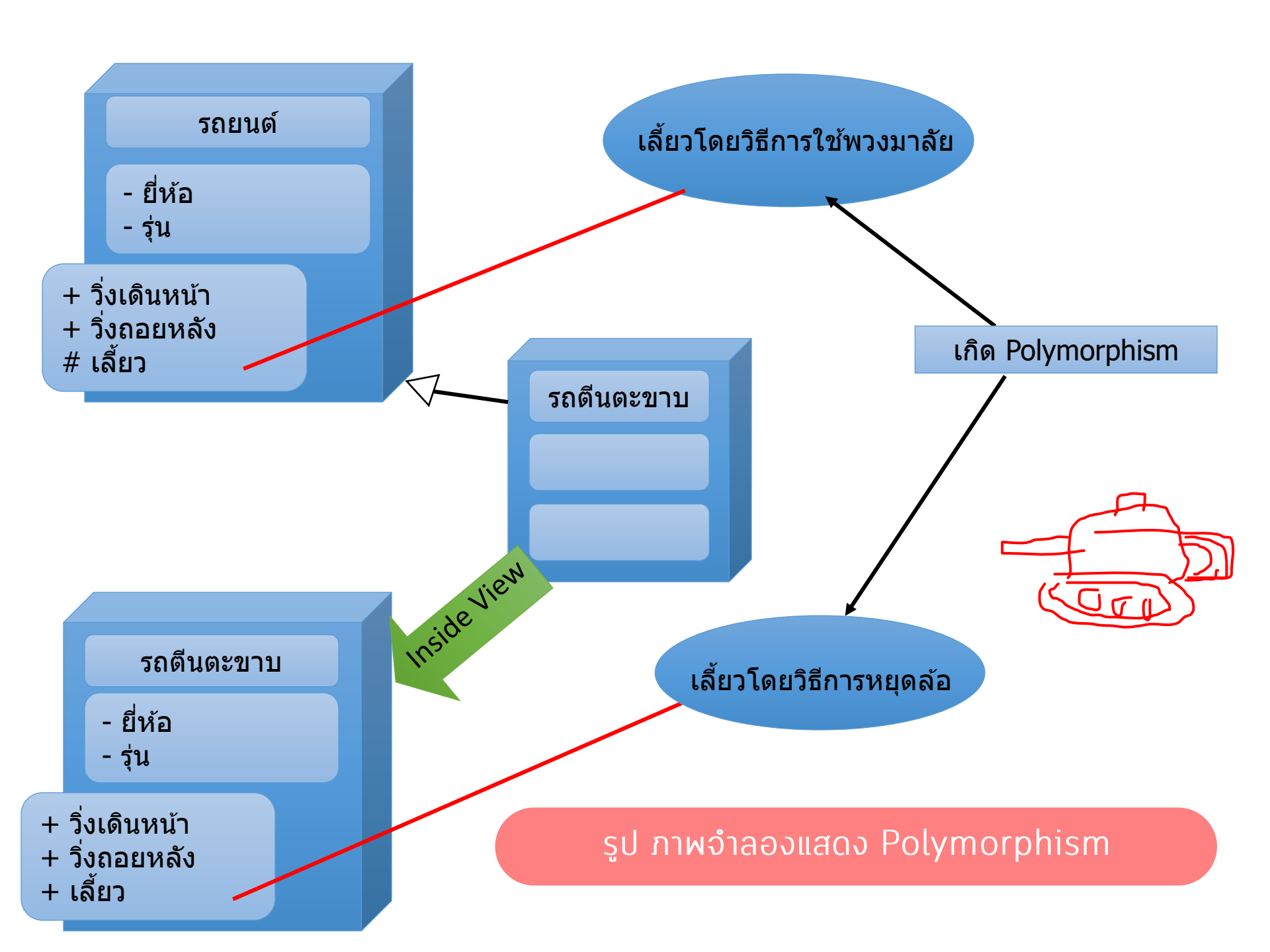


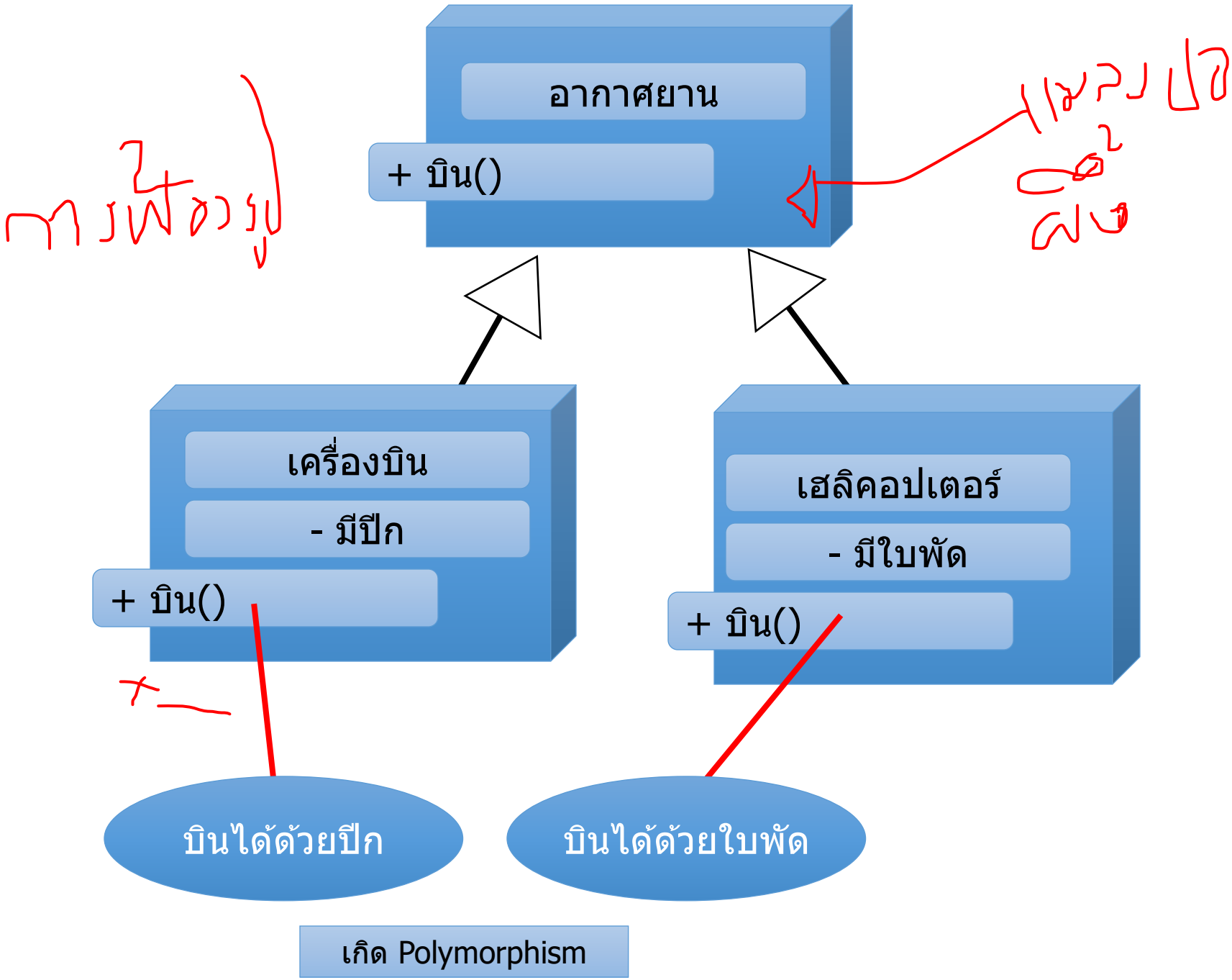
จากรูปจะเห็นว่า เครื่องเล่น CD ได้ดึงเอา Attributes จำนวนช่องเก็บ CD และ Functions เล่นเพลง จากเครื่องเล่น CD เพลงมาเป็น Attributes และ Functions ของตนเอง ในขณะเดียวกันได้ดึงเอา Functions เล่น Video ของเครื่องเล่น Video CD มาเป็น Functions หนึ่งของตนเอง ซึ่งทุกอย่างล้วนเป็นไปตามกฎของ Inheritance ทั้งสิ้น แต่สิ่งที่สนใจก็คือ ทั้งเครื่องเล่น CD เพลง และเครื่องเล่น Video CD ต่างก็มี Attributes ยี่ห้อเหมือนกัน แล้วเครื่องเล่น CD ก็ไม่สามารถมี Attributes 2 ตัวที่มีชื่อเหมือนกันได้ ดังนั้นทางเลือกก็คือ มันต้องเลือกที่จะเอา Attributes ยี่ห้อจาก Class ใด Class หนึ่งเท่านั้น เพื่อการแก้ปัญหาในการเลือก ในกรณีนี้ Subclass ที่เกิดจาก Multiple Inheritance นั้น จะเลือกเอา Attributes หรือ Functions ที่ชื่อซ้ำกันจาก Superclass ที่ได้ทำ Inherit ก่อนเสมอ ซึ่งพิจารณาจากรูป Subclass ที่อยู่ทางซ้าย จะทำ Inherit ก่อน Superclass ที่อยู่ทางขวาเสมอ

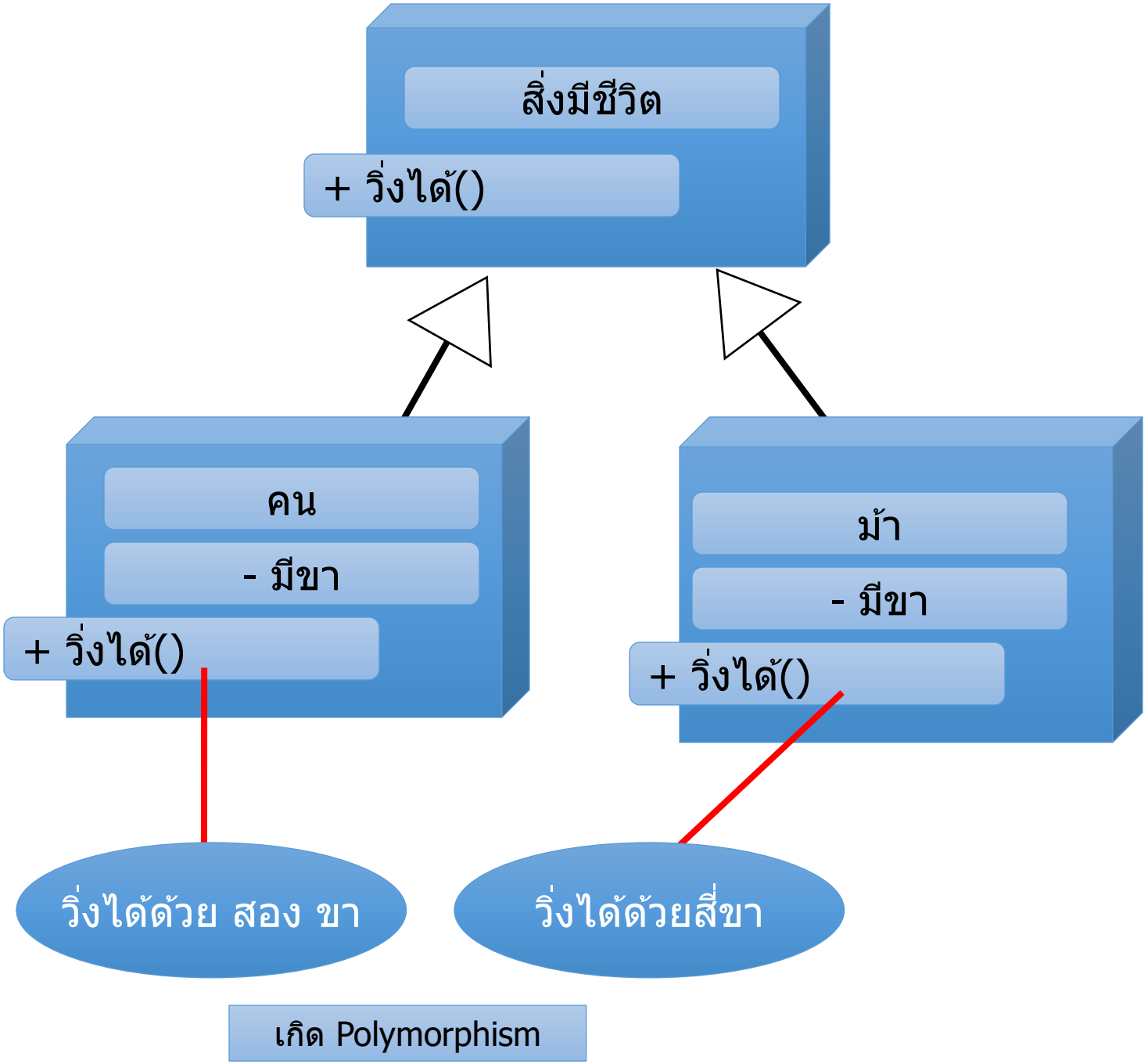
- **Polymorphism**

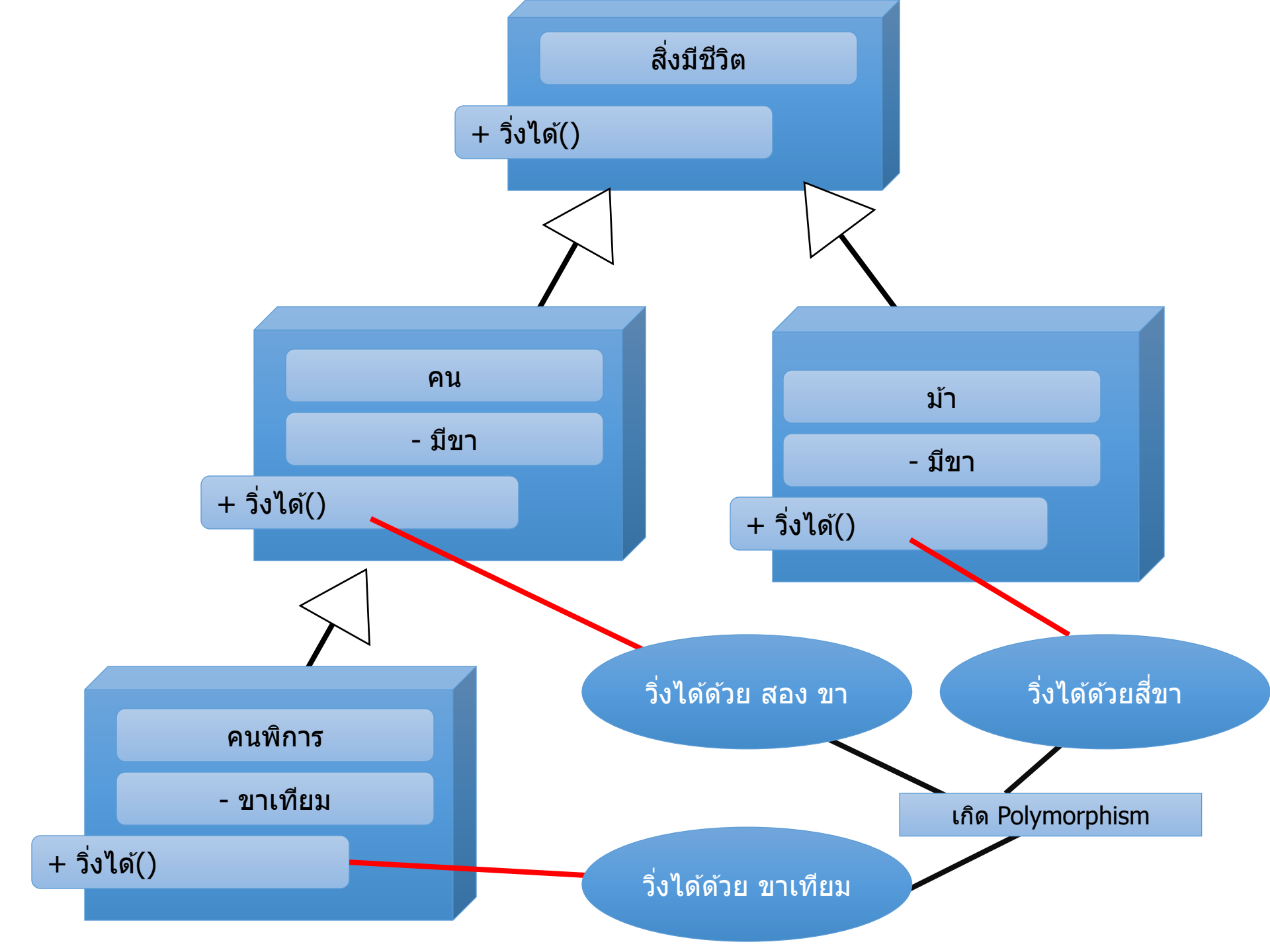
การที่ Subclass ที่เกิดจากการ Inherit จาก Superclass และมีการดัดแปลง Functions บางอย่างไม่ได้ยึดตาม Superclass ทั้งหมด จะเรียก Class นั้นมีคุณสมบัติ Polymorphism

Polymorphism เป็นตัวการที่จะทำให้ Subclass ที่มี Functions เดียวกันกับ Superclass (หรืออีกนัยหนึ่งคือ Subclass ที่มี Functions เป็นชื่อเดียวกันกับ Functions ใน Superclass) มีการทำงานที่แตกต่างกัน


รถยนต์
- ยี่ห้อ
- รุ่น
+ วิ่งเดินหน้า
+ วิ่งถอยหลัง
# เลี้ยว
เลี้ยวโดยวิธีการใช้พวงมาลัย
เกิด Polymorphism
รถตีนตะขาบ
Inside View
รถตีนตะขาบ
- ยี่ห้อ
- รุ่น
+ วิ่งเดินหน้า
+ วิ่งถอยหลัง
+ เลี้ยว
เลี้ยวโดยวิธีการหยุดล้อ
รูป ภาพจำลองแสดง Polymorphism


อากาศยาน
+ บิน()
เครื่องบิน
- มีปีก
+ บิน()
เฮลิคอปเตอร์
- มีใบพัด
+ บิน()
บินได้ด้วยปีก
บินได้ด้วยใบพัด
เกิด Polymorphism


สิ่งมีชีวิต
+ วิ่งได้()
คน
- มีขา
+ วิ่งได้()
ม้า
- มีขา
+ วิ่งได้()
วิ่งได้ด้วย สอง ขา
วิ่งได้ด้วยสี่ขา
เกิด Polymorphism


สิ่งมีชีวิต
+ วิ่งได้()
คน
- มีขา
+ วิ่งได้()
ม้า
- มีขา
+ วิ่งได้()
คนพิการ
- ขาเทียม
+ วิ่งได้()
วิ่งได้ด้วย สอง ขา
วิ่งได้ด้วยสี่ขา
เกิด Polymorphism
วิ่งได้ด้วย ขาเทียม

```
class human {
  private String name;

  public void walk() {
    System.out.println(“Walk with 2 leg”);
  }

}
```

```
class ab_human {
  private String name;

  public  override void walk() {
    System.out.println(“Walk with artificial 2 leg ”);
  }

}
```

Specialization

- Specialization เป็นกระบวนการย้อนกลับของ generalization แต่ให้ผลเช่นเดิม แตกต่างกันที่จุดเริ่มต้นเท่านั้น
- Generalization เป็นการค้นหาคุณลักษณะร่วมกัน ของ (sub)classes
- Specialization เป็นการแยก และค้นหาคุณลักษณะพิเศษของ (super)classes เพื่อให้ได้ subclasses

Generalization and Classification (= is_a?)

1. Shep is a Border Collie.
2. A Border Collie is a Dog.
3. Dogs are Animals
4. A Border Collie is a Breed.
5. Dog is a Species

1+2: Shep is a Dog
1+2+3: Shep is a animal
1+4: Shep is a breed?????
2+5: A Border Collie is a Species?????
Generalization is transitive (is kind of)
Classification is not transitive (is instance of)

Polymorphism in VB.NET

```
Public Class Shape
    Public Overridable Sub draw()
        MsgBox("Draw Shape")
    End Sub

End Class
Public Class Rectangle
    Inherits Shape
    Public Overrides Sub draw()
        MsgBox("Draw  Rec ")

    End Sub
End Class
Public Class Circle
    Inherits Rectangle
    Public Overrides Sub draw()
        MsgBox("Draw  Circle ")

    End Sub
End Class
```

-Prefix **Overridable** before Base Method
-Prefix **Overrides** before derived class

Polymorphism in VB.NET

```
Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As
    System.EventArgs) Handles MyBase.Load

    Dim s As Shape = New Shape
    s.draw()
    Dim s1 As Rectangle = New Rectangle
    s1.draw()

    Dim s3 As New Circle
    s3.draw()

End Sub
```

Polymorphism in VB.NET

```
Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As
    System.EventArgs) Handles MyBase.Load
    Dim s As Shape = New Shape
    s.draw()
    Dim s1 As Rectangle = New Rectangle
    s1.draw()

    Dim s3 As New Circle
    s1 = s3
    s1.draw()


End Sub
```

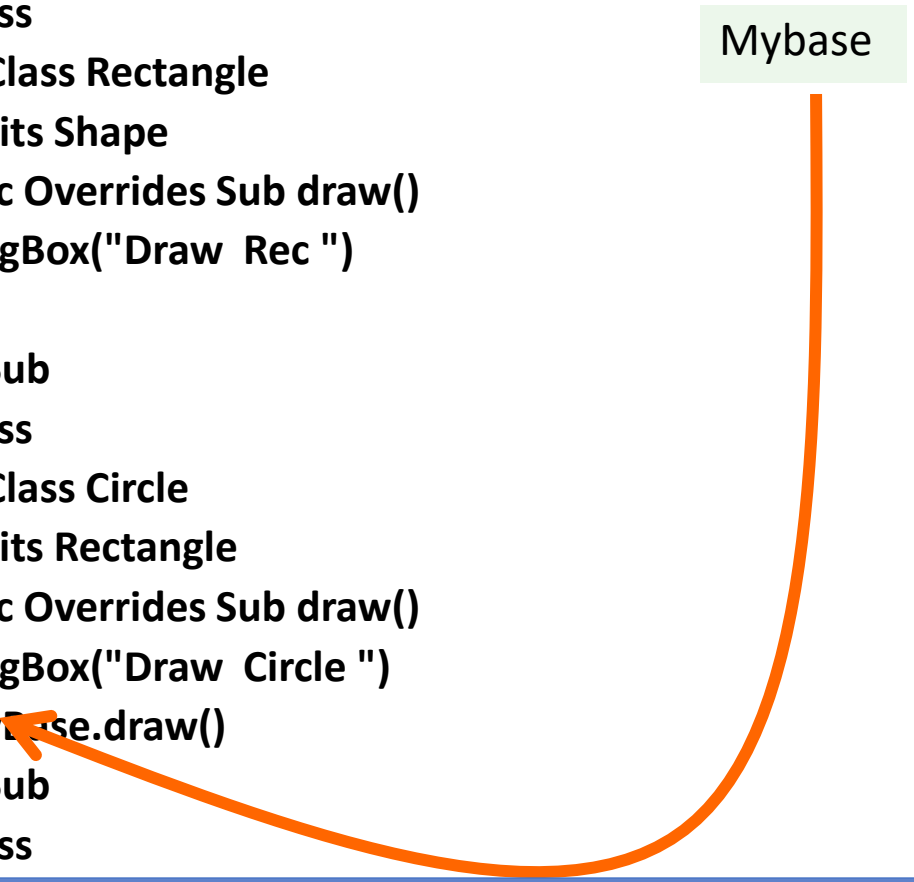
No polymorphism

```
Public Class Shape
    Public Overridable Sub draw()
        MsgBox("Draw Shape")
    End Sub

End Class
Public Class Rectangle
    Inherits Shape
    Public Overrides Sub draw()
        MsgBox("Draw  Rec ")

    End Sub
End Class
Public Class Circle
    Inherits Rectangle
    Public Overrides Sub draw()
        MsgBox("Draw  Circle ")

    End Sub
End Class
Public Class Triangle
    Inherits Shape
    Public Shadows Sub draw()
        MsgBox("Draw  Circle ")
```

```
Dim s As Shape = New Shape
s.draw()
Dim s1 As Rectangle = New Rectangle
s1.draw()

Dim s3 As New Circle
s1 = s3
s1.draw()
```

Calling Base Method

```
Public Class Shape
    Public Overridable Sub draw()
        MsgBox("Draw Shape")
    End Sub

End Class
Public Class Rectangle
    Inherits Shape
    Public Overrides Sub draw()
        MsgBox("Draw  Rec ")

    End Sub
End Class
Public Class Circle
    Inherits Rectangle
    Public Overrides Sub draw()
        MsgBox("Draw  Circle ")
        MyBase.draw()
    End Sub
End Class
```



การสืบทอด
INHERITANCE

(in JAVA)

in java

This References

เราจะทราบถึงสิ่งที่อ้างอิงเมื่อได้ทำการสร้างวัตถุแล้ว แต่ถ้าการเขียนสิ่งที่อยู่ภายในคลาส ยังไม่ได้ถูกสร้างเป็นวัตถุ เราจะอ้างได้อย่างไร คำว่า “this” เป็นวิธีที่ช่วยให้การอ้างอิงตัวมันเองโดยยังไม่ต้องสร้างวัตถุก็สามารถทำได้

- **Using this with a Field**
- **Using this with a Constructor**

in java

Using this with a Field

```
public class Point {
    public int x = 0;
    public int y = 0;
    //constructor
  public Point(int a, int b){
        x = a;
        y = b;
}
}
```

```
public class Point {
    public int x = 0;
    public int y = 0;
    //constructor
  public Point(int x, int y){
        this.x = x;
        this.y = y;
}
}
```

in java

Using this with a Constructor

```
public class Rectangle {
private int x, y;
private int width, height;
public Rectangle() {
    this(0, 0, 0, 0);
}
public Rectangle(int width, int height) {
  this(0, 0, width, height);
}
public Rectangle(int x, int y, int width, int height) {
this.x = x;
this.y = y;
this.width = width;
this.height = height;
}
...
}
```

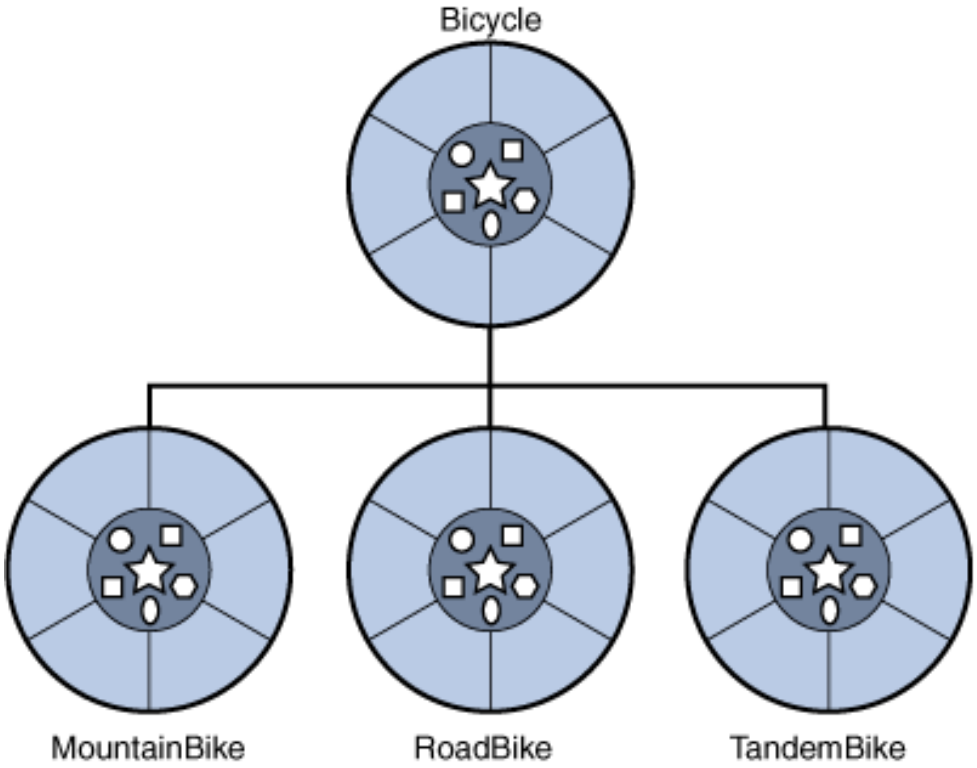
in java

INHERITANCE

- Inheritance เป็นรูปแบบของการนำกลับมาใช้ใหม่ของซอฟต์แวร์ ซึ่งคลาสใหม่จะถูกสร้างจากการรับสิ่งต่างๆจากคลาสที่มีอยู่ ทั้งส่วน Attribute และส่วนของ Method และคลาสใหม่ยังสามารถเพิ่มเติมความสามารถบางประการตามต้องการ
- ดังนั้นคลาสที่ถูกสร้างใหม่จะได้รับคุณสมบัติของคลาสเดิมและเพิ่มคุณสมบัติบางประการ

in java

INHERITANCE



in java

รูปแบบ

สมมุติว่ามี Class A และเราต้องการสร้าง class B ที่ได้รับคุณลักษณะต่างๆ จาก Class A สิ่งที่เราต้องการตอนสร้าง class B คือการเพิ่ม keyword “extends” เข้าไปดังแสดงในรูป

```
Class B extends A
{
    //definition of class B
}
```

in java

Super class & Sub Class

- Class A : parent class/base class/super class
- Class B : child class/extended class/sub class

```
class A {
    void printA() {System.out.println('A');}
}
class B extends A {
     void printB() {System.out.println('B');}
}
```

in java

การสืบทอดจาก Super Class

ซับคลาส (Sub Classes) สืบทอดสมาชิกทุกอย่าง(แอตทิบิวส์และเมดธอร์ด) จากซูเปอร์ Super Class ยกเว้น

- The private member of Super Class
- Constructor of Super Class

```
class InheritTest1 {
    public static void main(String args[]) {
      A x = new A(); x.printA();
      B y = new B(); y.printA(); y.printB();
    }
}
```

in java

KEYWORD “final”

ใส่ final ไว้หน้า Class ถ้าไม่ต้องการให้ Class นั้นสืบทอดได้

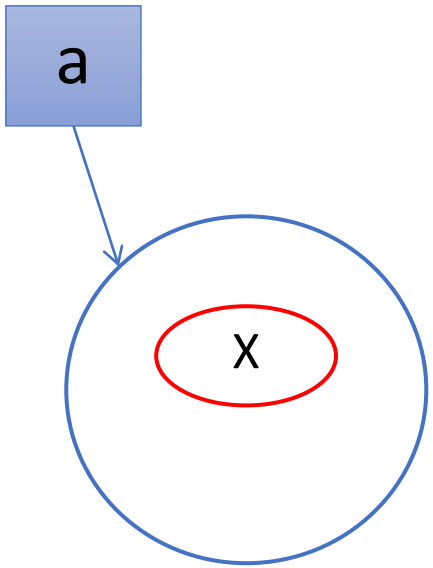
```
class A { int a = 1;}
class B extends A { int b = 2; }
final class C extends B { int c = 3; }
class Inherit2 {
   public static void main(String args[]){
      C z = new C();
      System.out.println(z.a+z.b+z.c);
   }
}
```

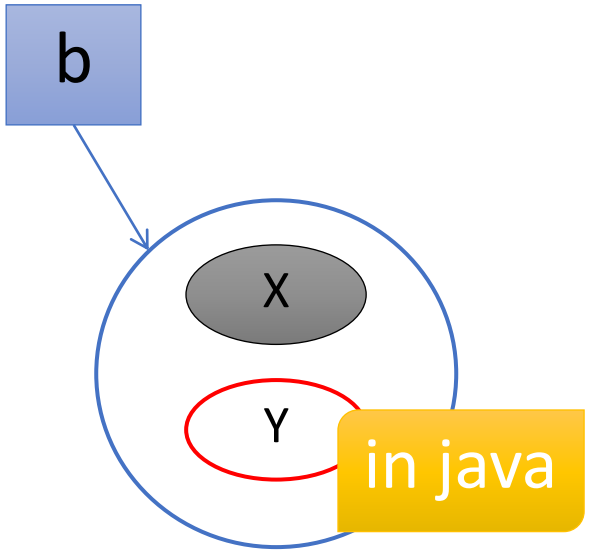
in java

REFERENCE

```
class A {int x;}
class B extends A
{int y;}
```

```
A  a = new A();
B  b = new B();
```





ทดสอบโปรแกรม

```
class A { int x = 1;}
class B extends A { int y = 2; }
class Inherit3 {
    public static void main(String args[]) {
      A a = new A();
      System.out.println(a.x);
      B b = new B();
      System.out.println(b.x + “,” + b.y);
      b.x--;
      // b = a;
      a = b;
      System.out.println(a.x);
    }
}
```

in java

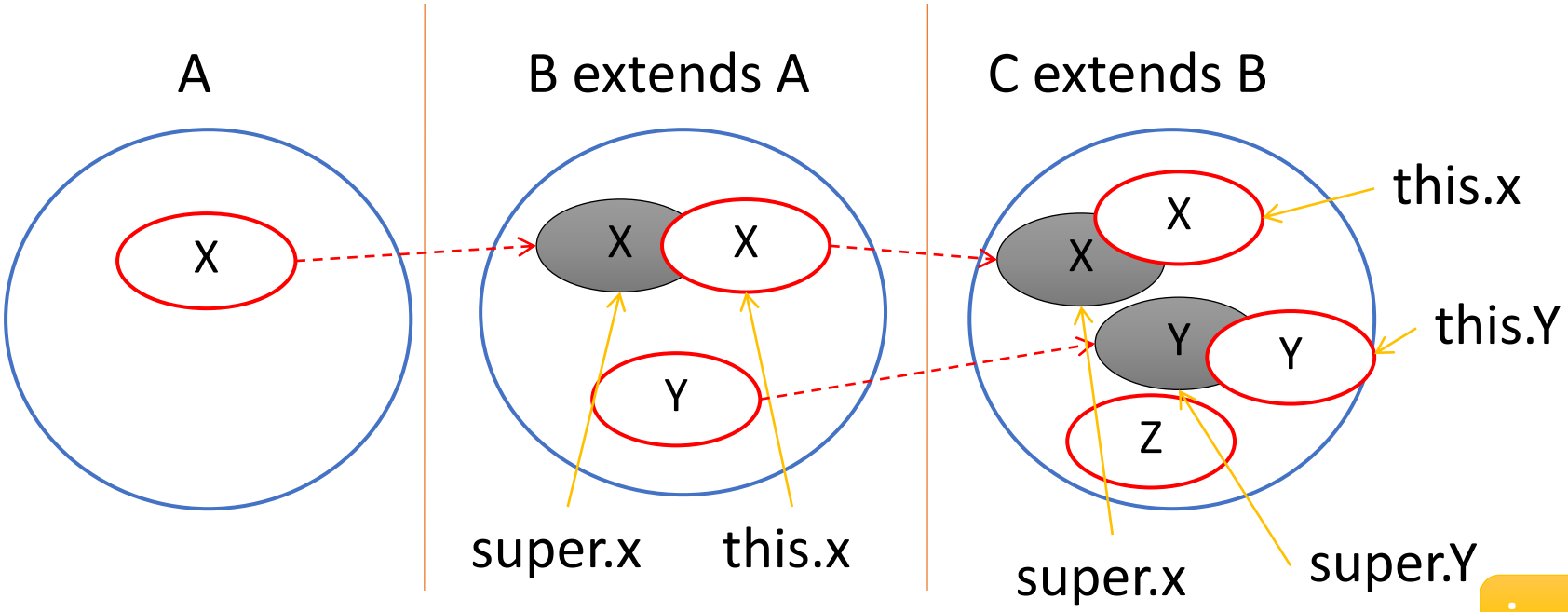
Constructor Chaining

- เมื่อมีการสร้าง instance ของคลาสลูกขึ้น constructors ของคลาสบรรพบุรุษทั้งหมด จะถูกทำงาน

```
class A { A() {System.out.println("A");} }
class B extends A {
   B() {System.out.println("B");}
}
class ConstructorChain {
   public static void main(String args[])
   { new B();}
}
```

Super Reference

- ใช้ keyword ว่า “super” แทน class ที่ inherite เพื่อใช้ในการอ้างถึง member ของ super class
- ในการอ้าง super จะหมายถึงตัว data member ตัวแรกที่เจอในสายของบรรพบุรุษ เช่น



Super Reference

```
class A {
   int a;
   void print() { System.out.println(a);}
}
class B extends A {
   int a;
   B(int x, int y){super.a = x; this.a = y;}
   void print() {
      super.print(); System.out.println(a);}
}
class Super1{
   public static void main(String args[]){
      B b = new B(1,2);
      b.print();
   }
}
```

in java

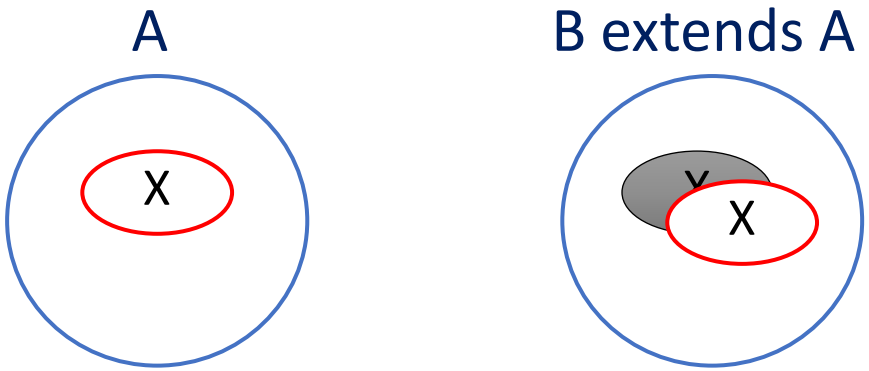
Super Constructors

```
class A {
   A() { System.out.println("A");}
   A(char c) { System.out.println(c); }
}
class B extends A {
   B() {
      //super('a');
      System.out.println("B");
   }
}
class SuperConstructor {
   public static void main(String args[]){new B();}
}
```

in java

Shadowing

ถ้าเรากำหนด data member ในคลาสลูกมีชื่อเหมือนกับ data member ในคลาสแม่ ชื่อของลูกจะบัง(shadow)ชื่อในคลาสแม่

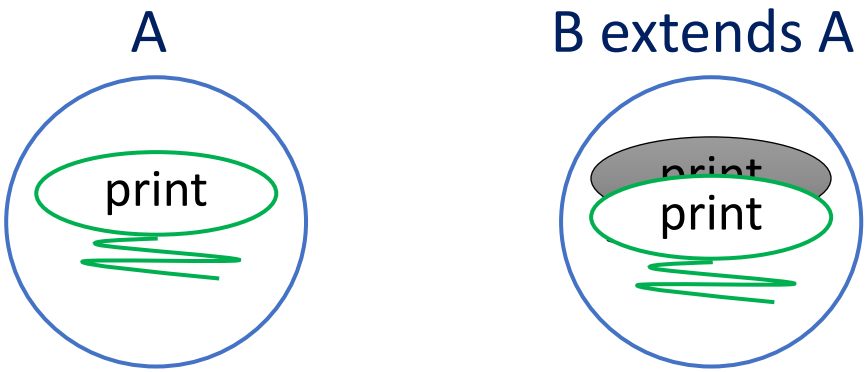



```
class A {int x = 1;}
class B extends A {float x = 2.0f;}
class Shadowing {
   public static void main(String args[]) {
        B b = new B();
        System.out.println(b.x);
   }
}
```

in java

Overriding

ถ้าเรากำหนด method ในคลาสลูกมี signature เหมือนกับ method ในคลาสแม่ พฤติกรรมของลูกจะลบล้าง(override)พฤติกรรมในคลาสแม่

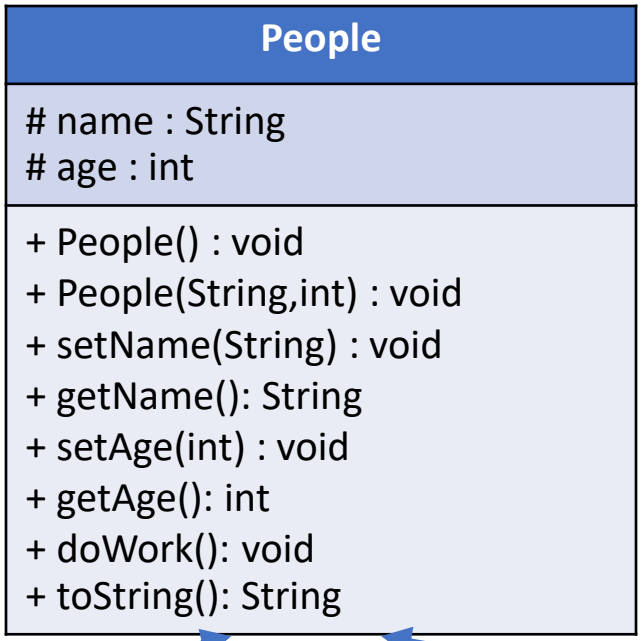



```
class A {void print() {System.out.println("A");}}
class B extends A {
   void print() {System.out.println("B");}
}
class Overriding {
   public static void main(String args[]) {
       new B().print();
   }
}
```

in java

ทดสอบการสืบทอดคุณสมบัติของคลาส

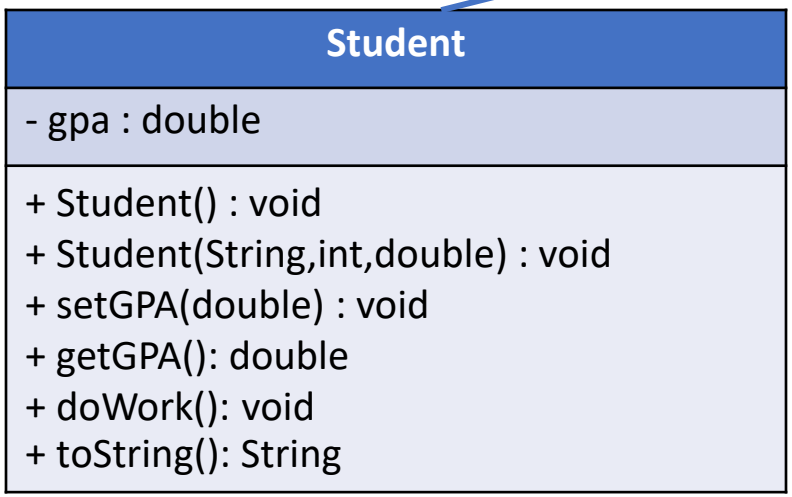
ประชาชน

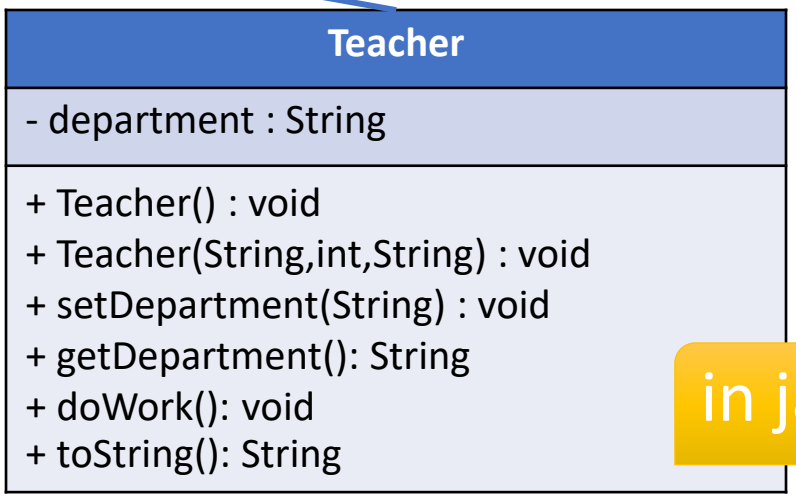


สัญลักษณ์ของ UML
(Unified Modeling Language)
- private
+ public
# protected

นักเรียน

อาจารย์





in java

Class People

```
public class People {
protected String name;
protected int age;

public People(){
  this(null,0);
}
public People(String n,int a){
    name = n;
    age = a;
}
public void setName(String n){
    name = n;
}
public String getName(){
    return name;
}
```

ต่อ

```
public void setAge(int a){
    age =a;
}
public int getAge(){
    return age;
}
public void doWork(){
}
public String toString(){
    return "Name : " + name + "  Age : "
+age;
}
}
```

in java

class Student

```
public class Student extends People {
  private double gpa;
  public Student(){
      this(null,0,0);
  }
  public Student(String n,int a,double g){
      super(n,a);
      gpa = g;
  }
  public void setGPA(double g){ gpa = g; }
  public double getGGA(){ return gpa;}
  public void doWork(){
      System.out.println("Study in school");
  }
  public String toString(){
      return super.toString()+ "  GPA : "+ gpa;
  }
}
```

in java

class Teacher

```
public class Teacher extends People{
private String department;
public Teacher(){
    this(null,0,null);
}
public Teacher(String n,int a,String dep){
 super(n,a);
 department = dep;
}
public void setDepartment(String dep){department = dep;}
public String getDepartment(){return department; }
public void doWork(){
    System.out.println("Teach in school");
}
public String toString(){
    return super.toString()+ "  Department : "+ department;
}
}
```

in java

class Demo ทดสอบการทำงาน

```
public class Demo {

  public static void main(String [] args){
      Student s1 = new Student("Winai",15,3.5);
      Teacher t1 = new Teacher("Pranee",30,"Science");
      System.out.println(s1);
      System.out.println(t1);
      s1.doWork();
      t1.doWork();
  }
}
```

in java

Polymorphism

With java

in java

Polymorphism

เป็นลักษณะการทำงานของ method ชนิดหนึ่งที่ต้องอาศัยกลไกการทำงานของ Inheritance และ Dynamic Binding

method จะสามารถตอบสนองการทำงานได้หลายรูปแบบขึ้นอยู่กับ Object ที่ส่งเข้ามาว่าเป็นของ class ใด ก็จะทำตามคำสั่งที่ได้เขียนไว้ใน method ของ class นั้น

ตัวอย่างซูเปอร์คลาสและซับคลาส

- สมมุติว่าเรามีคลาสเริ่มต้นเป็น คลาสชื่อ Shape บอกลักษณะของรูปเรขาคณิตของวัตถุ 2 มิติ
- มีคลาส Square บอกลักษณะของสี่เหลี่ยม และคลาส Triangle บอกลักษณะของสามเหลี่ยม ที่สืบทอดลักษณะมาจากคลาส Shape

inheritance & overriding

```
class Figure {
   double width, height;
   String name;
   Figure(double w, double h, String n) {
      width = w; height = h; name = n;
   }
   public String getName() { return name;}
   public double getArea() { return 0.0;}
   public void setWidth(double w)  { width = w; }
   public void setHeight(double h) { height = h;}
}
```

inheritance & overriding

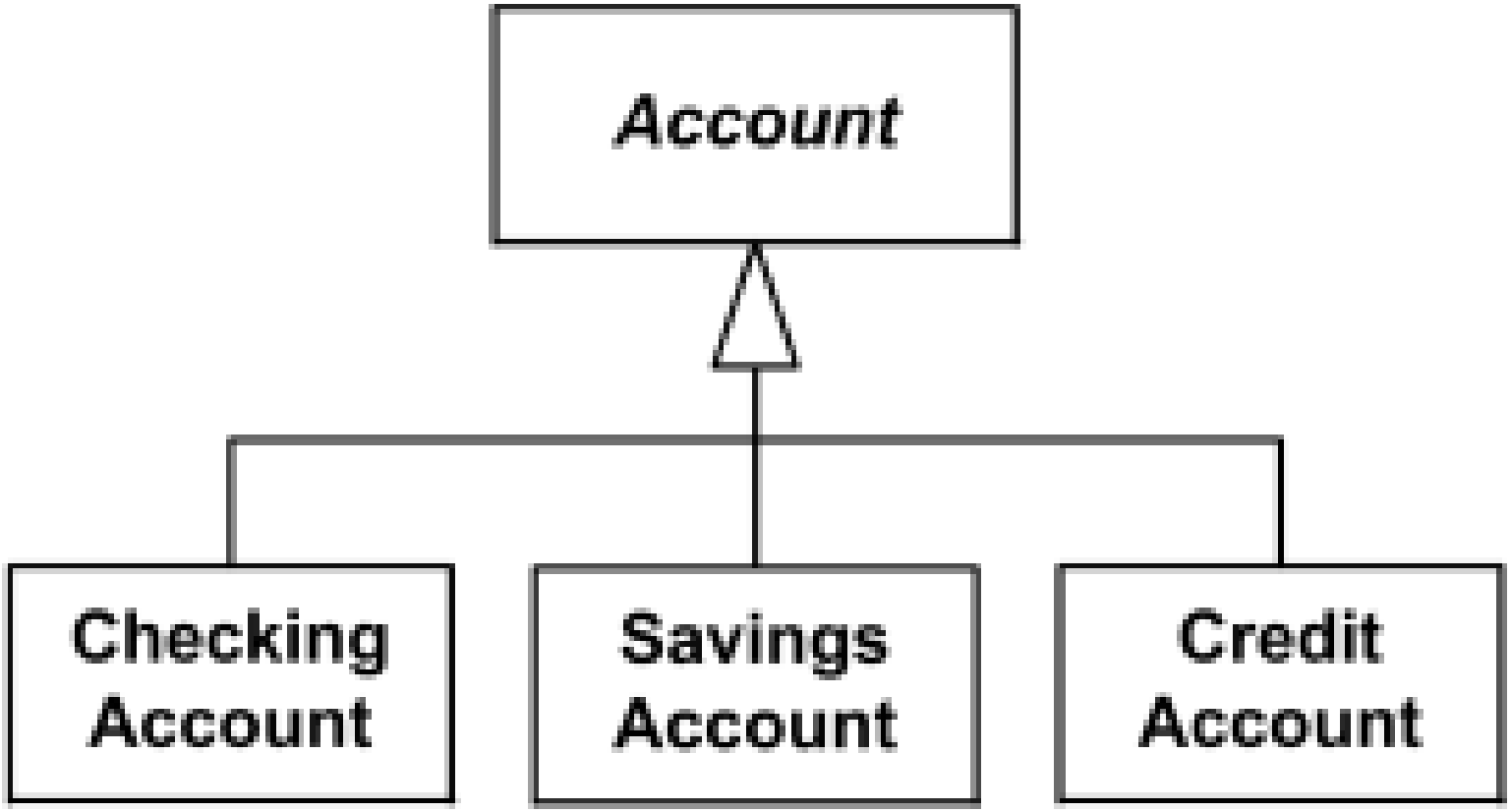
```
class Rectangle extends Figure {
   Rectangle(double w, double h) {
      super(w,h,"rectangle");
   }
   double getArea() { return width*height;}
}

class Triangle extends Figure {
   Triangle(double w, double h) {
      super(w,h,"triangle");
   }
   double getArea() { return 0.5*widht*height;}
}
```

Polymorphism

```
class PolyEx {
   static void compute(Figure x) {
      System.out.println(x.getName()+" is");
      System.out.println(x.getArea());
   }
   public static void main(String args[]){
      compute(new Figure(1, 1, "undefined");
      compute(new Triangle(1, 1));
      compute(new Rectangle(1, 1));
   }
}
```

Polymorphism

```
class A {
  void print() { System.out.println("Im A"); }
}
class B1 extends A {
  void print() { System.out.println("Im B1");}
}
class B2 extends A {
  void print() { System.out.println("Im B2");}
}
class Other {
  void call(A a) { a.print(); }
  public static void main(String args[]){
    call(new A()); call(new B1()); call(new B2());
  }
}
```


Account
Checking Account
Savings Account
Credit Account

คำถามท้ายบท

1. จงเขียนแผนภาพแสดง Generalization จาก Problem Domain ต่อไปนี้

"Problem Domain 1 เมื่อเราพูดถึงฐานะของบุคคล เราจะจำแนกฐานะออกเป็น 2 ส่วนคือ ทรัพย์สิน และหนี้สิน โดยทรัพย์สินจำแนกออกเป็นเงินสดและเงินฝากในบัญชี และเงินในรูปแบบอื่นๆ ซึ่งได้แก่ หุ้น และหลักทรัพย์ หนี้สิน จำแนกออกเป็น หนี้สินระยะสั้น และหนี้สินระยะยาว"

“Problem Domain 2 งานศิลปะแบ่งออกเป็น 3 ประเภทคือ งานจิตรกรรม งานประติมากรรม และงานสถาปัตยกรรม โดยงานจิตรกรรมนั้นจำแนกเป็น ภาพวาด (งานลายเส้น งานสีน้ำ งานสีน้ำมัน งานสีชอล์ค และงานสีฝุ่น) และภาพพิมพ์ งานประติมากรรม แบ่งออกเป็น งานปั้น และงานหล่อ ส่วนงานสถาปัตยกรรมนั้น แบ่งออกเป็น สถาปัตยกรรมไทย และสถาปัตยกรรมประยุกต์”

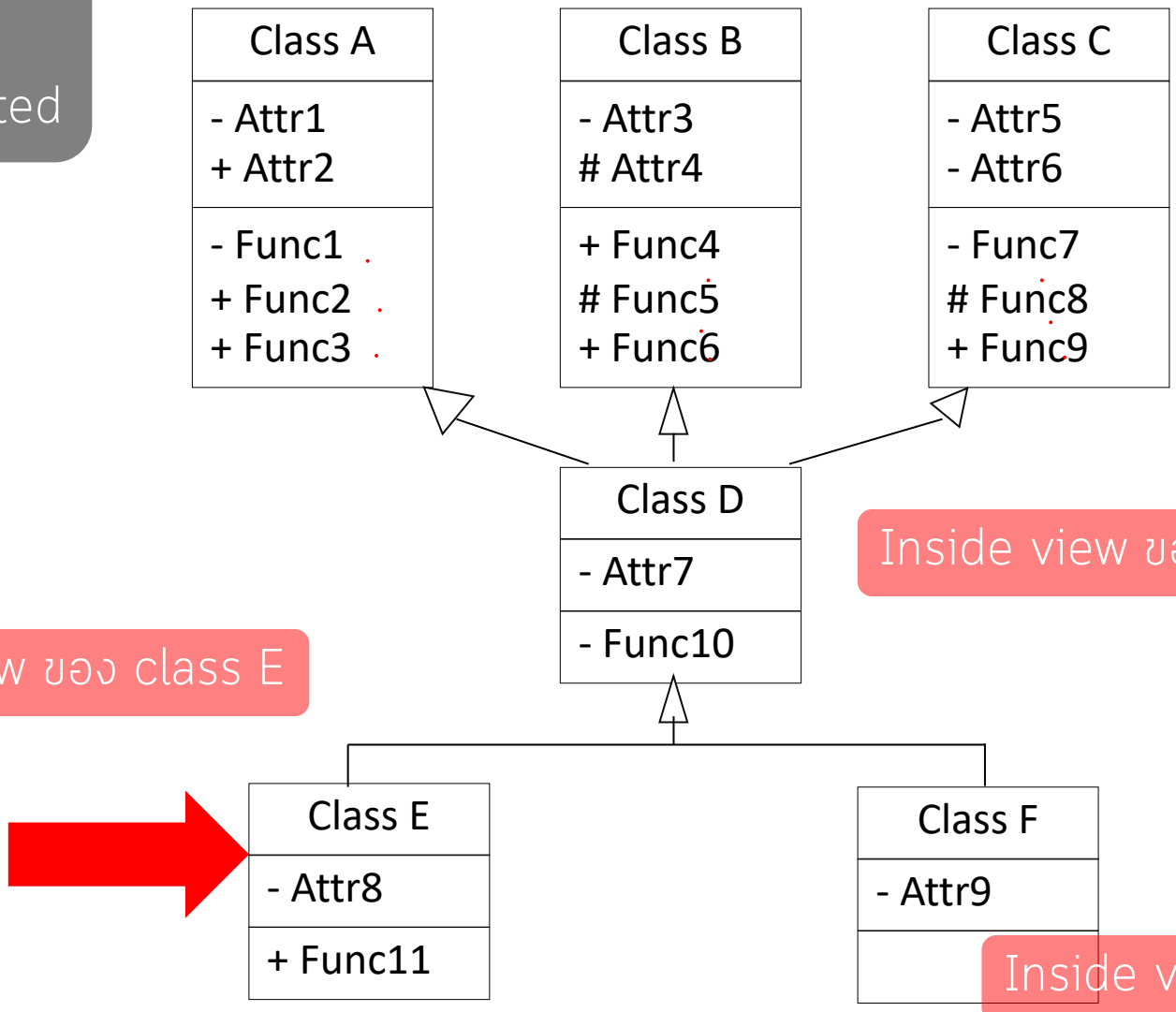
2. ด้วยหลักการ Generalization Abstract ของลูกจ้าง กับหัวหน้างาน จะเปลี่ยนเป็นเช่นไร เมื่อมี Class ต่อไปนี้เพิ่มเข้าไป

Class : ลูกจ้างชั่วคราว
Attributes : เงินเดือน
Functions : ปฏิบัติงาน
ลาพักงาน

Class : บริการระดับสูง
Attributes : เงินเดือน
ตำแหน่ง
เงินประจำตำแหน่ง
Functions : ปฏิบัติงาน
ลาพักงาน
สั่งงาน
วางนโยบาย

3. จากแผนภาพต่อไปนี้ จงตอบคำถามต่อไปนี้

- Private
+ Public
# protected

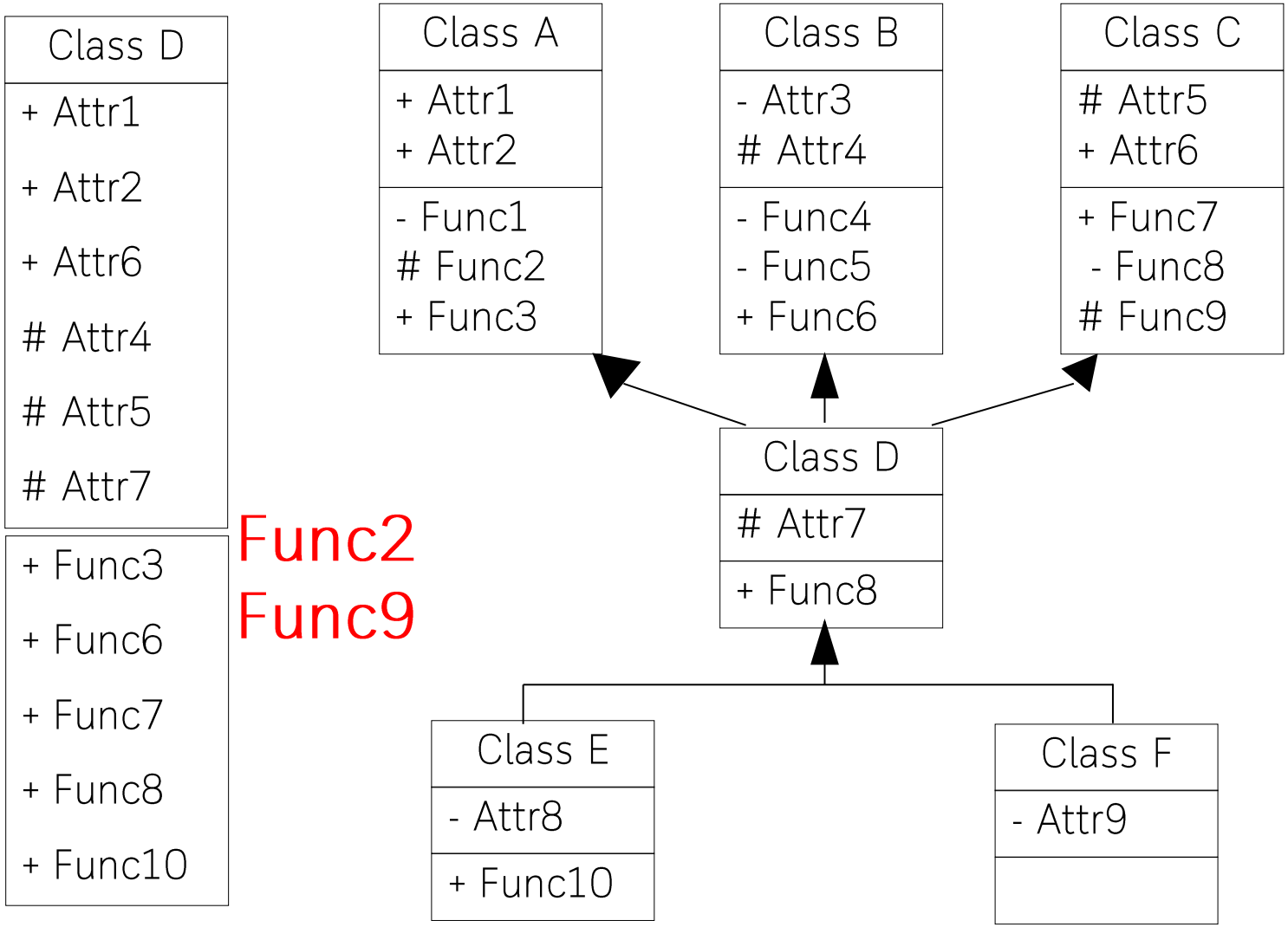


Inside view ของ class D

Inside view ของ class E

Inside view ของ class F

3. จากแผนภาพต่อไปนี้ จงตอบคำถามต่อไปนี้



Func2
Func9

คำถาม

1. Inside View ของ Class D, Class E, และ Class F เป็นอย่างไร
2. ใน Class D มี Attr1 หรือไม่ ถ้ามี Attr1 ได้มาจากการ Inherit จาก Class ใด
3. ใน Class D มี Attr2 หรือไม่ ถ้ามี Attr2 ได้มาจากการ Inherit จาก Class ใด
4. ใน Class D มี Attr3 หรือไม่ ถ้ามี Attr3 ได้มาจากการ Inherit จาก Class ใด
5. ใน Class D มี Func1 หรือไม่ ถ้ามี Func1 ได้มาจากการ Inherit จาก Class ใด
6. ใน Class D มี Func4 หรือไม่ ถ้ามี Func4 ได้มาจากการ Inherit จาก Class ใด
7. ใน Class D มี Attributes ตัวใดบ้างที่ Func8 สามารถเข้าถึงได้
8. ใน Class E มี Attributes ตัวใดบ้างที่ Func9 สามารถเข้าถึงได้
9. ใน Class F มี Func8 หรือไม่ ถ้ามี มี Attributes ตัวใดบ้างที่ Func8 สามารถเข้าถึงได้
10. ใน Class F มี Func4 หรือไม่ ถ้ามี มี Attributes ตัวใดบ้างที่ Func4 สามารถเข้าถึงได้

Inside View ของ Class D