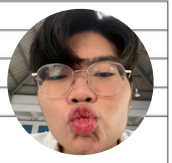
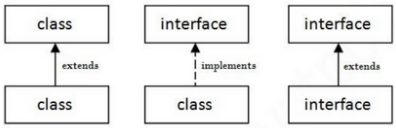
คำอธิบายเครื่องหมายดังแสดงในตารางต่อไปนี้

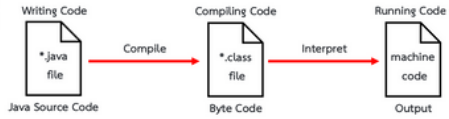
ลำดับ	สัญลักษณ์	ความหมาย
1	+	เป็นการกำหนด access mode เป็น public
2	-	เป็นการกำหนด access mode เป็น private
3	#	เป็นการกำหนด access mode เป็น protected
4	Class	บ่งบอกว่าเป็นคลาสปกติ
5	AbstractClass*	***ชื่อคลาสเป็นตัวเอียงหนา*** บ่งบอกว่าเป็นคลาสไม่สมบูรณ์
6	<<Interface>>	บ่งบอกว่าเป็นอินเตอร์เฟส
7		บ่งบอกการสืบทอด extends
8		บ่งบอกการ implements
9		บ่งบอกการเป็นส่วนหนึ่ง composite
10	*methodname()*	*ตัวเอียง* บ่งบอกว่าเป็นเมธอดไม่สมบูรณ์
11	**attribute** หรือ **methodname()**	**ตัวหนา** บ่งบอกความเป็น final
12	attribute หรือ methodname()	ตัวขีดเส้นใต้ บ่งบอกความเป็น static



byte → short → int → long → float → double

char → int





- "Compile-Time Polymorphism" เรียกอีกอย่างว่า "Static Polymorphism" ซึ่งการเรียกใช้เมธอดจะได้รับการกำหนดช่วงเวลาคอมไพล์ การ Compile-Time Polymorphism สามารถทำได้โดยอาศัยหลักการ Overloading เมธอด
- "Runtime Polymorphism" เรียกอีกอย่างว่า "Dynamic Binding" หรือ "Dynamic Method Dispatch" ในกระบวนการนี้การเรียกใช้เมธอดจะได้รับการกำหนดหรือปรับเปลี่ยนแบบ Dynamic ขณะรันโทม4 การ "Runtime Polymorphism" สามารถทำได้โดยอาศัยหลักการ Overriding เมธอด

- คีย์เวิร์ด **final** สามารถจะใช้ได้กับคลาส ตัวแปร และเมธอด

ตำแหน่ง	ความหมาย
คลาส	ทำให้คลาสอื่นไม่สามารถสืบทอดคลาสนี้ได้
เมธอด	เมธอดที่จะไม่สามารถมีเมธอดแบบ overriden ได้
ตัวแปร	ค่าคงที่ ซึ่งจะทำให้สามารถกำหนดค่าได้เพียงครั้งเดียวเท่านั้น

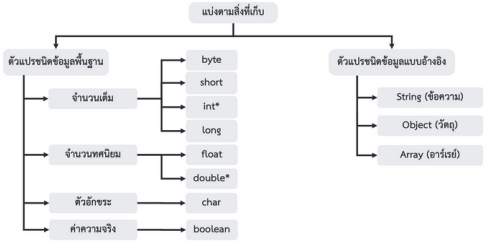
โดยทั่วไปแล้ว **ตัวแปรชนิดอ้างอิง** สามารถเป็นได้เพียงชนิดเดียว (Data type) และไม่สามารถปรับเปลี่ยนได้ ซึ่งเป็นลักษณะการทำงานแบบ Static Binding

- Static Binding คือ ชนิดข้อมูล (Data type) ของตัวแปรชนิดอ้างอิง (Object) จะถูกกำหนดก็ต่อเมื่อโปรแกรมถูก Compile

อย่างไรก็ตาม การที่ตัวแปรชนิดอ้างอิงสามารถย้ายการอ้างอิงจากวัตถุเดิมไปสู่วัตถุใหม่ได้ กรณีเป็นคลาสแม่ลูกกัน

- Dynamic Binding คือ ชนิดข้อมูล (Data type) ของตัวแปรชนิดอ้างอิง (Object) จะถูกกำหนดก็ต่อเมื่อโปรแกรมประมวลผล (runtime)

ชนิดข้อมูล	ขนาด (Bit/Byte)	ช่วงค่า	ค่าเริ่มต้น
boolean	1 / Not Precisely define	true หรือ false	false
char	16 / 2	16 bit Unicode character '\u0000' ถึง '\uFFFF'	\u0000
byte	8 / 1	-128 ถึง +127	0
short	16 / 2	-32,768 ถึง +32,767	0
int	32 / 4	-2^{31} ถึง $+2^{31}$-1	0
long	64 / 8	-2^{63} ถึง $+2^{63}$-1	0L
float	32 / 4	IEEE 754 floating point standard -3.40E+38 ถึง +3.40E+38	0.0f
double	64 / 8	IEEE 754 floating point standard -1.80E+308 ถึง +1.80E+308	0.0d หรือ 0.0



Component		Interface	Abstract Class
	Constructor	×	√
แอททริบิวท์	Static	√	√
	Non-Static	×	√
	Final	√	√
	Non-final	×	√
เมธอด	Abstract	√	√
	Static	√	√
	Non-static	√	√
	Final	×	√
	Non-final	√	√
Access Modifier	private	×	√
	default	√	×
	protected	×	√
	public	√	√

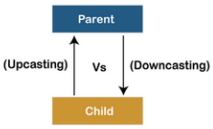
modifier	คลาสเดียวกัน	คลาสที่อยู่ในแพคเก็จเดียวกัน	คลาสที่เป็น subclass	คลาสใด ๆ
public	✓	✓	✓	✓
protected	✓	✓	✓	✗
default	✓	✓	✗	✗
private	✓	✗	✗	✗

SIMPLE TYPE	WRAPPER CLASSES
boolean	Boolean
char	Character
double	Double
float	Float
int	Integer
long	Long

	ประเภท	บ่งบอกถึง	ความหมาย
this	คีย์เวิร์ด	คลาสตนเอง	ใช้เพื่อเรียกใช้งานแอททริบิวท์หรือเมธอด**ภายในคลาส** ซึ่งนิยมเรียกใช้งานเมื่อเกิดความคลุมเครือเท่านั้น และควรใช้กับแอททริบิวท์
super	คีย์เวิร์ด	คลาสแม่	ใช้เพื่อเรียกใช้งานแอททริวบิวท์และเมธอดของ**คลาสแม่**
this(...)	เมธอด	คลาสตนเอง	เมธอดที่เรียกใช้งานเพื่อเรียกคอนสตรัคเตอร์**ภายในคลาส**
super(...)	เมธอด	คลาสแม่	เมธอดที่เรียกใช้งานเพื่อเรียกคอนสตรัคเตอร์ของ**คลาสแม่**

เป็นการอนุญาตให้คลาสลูก (Subclass) สามารถเพิ่มเติมการทำงานจำเพาะ (เพิ่มโค้ด) ในเมธอดที่คลาสแม่ (Superclass) มีไว้แล้วได้ โดยมีเงื่อนไขดังนี้

- ข้อที่ 1: จำนวนและชนิดของ argument จะต้องเหมือนเดิม
- ข้อที่ 2: ชนิดข้อมูลของค่าที่ส่งกลับจะต้องเหมือนเดิม
- ข้อที่ 3: Access Modifier จะต้องมีระดับไม่ต่ำกว่าเดิม




```
Parent a = new Child();
```

Upcasting คือการแปลงชนิดข้อมูล (typecasting) รูปแบบหนึ่งที่พยายามแปลงออบเจ็คจากคลาสลูก (Child Object) ไปเป็นออบเจ็คของคลาสแม่ (Parent Class Object) ซึ่งการทำ Upcasting นิยมนำมาใช้เมื่อต้องการเข้าถึงแอททริบิวหรือและเมธอดของคลาสลูก อย่างไรก็ตาม นักศึกษาไม่สามารถเข้าถึงแอททริบิวหรือและเมธอดทั้งหมดได้ มีสิทธิ์เข้าถึงเพียงแอททริบิวหรือและเมธอดของคลาสลูกเท่านั้น

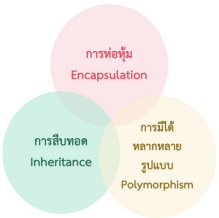
Overloading

- เปลี่ยนจำนวน Parameter ที่รับเข้ามา
- เปลี่ยน DataType ของ Parameter

DataType - Casting (x = ตัวแปร)

- ..... > String : String.valueOf(x);

  String > Double : Double.parseDouble(x)
  String > Int : Integer.parseInt(x);
  String > Float : Float.parseFloat(x);
  String > char : "x".charAt(0)



Encapsulation

- การใช้ Access Modifier
- การใช้ getter , setter

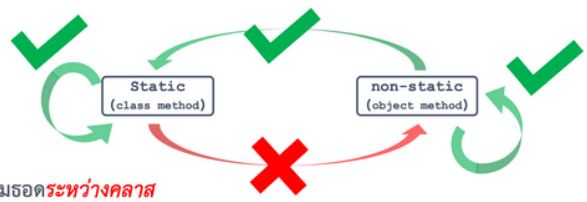
Inheritance

- (Class) การสืบทอดโดยใช้ Extends
- (Interface) สืบทอดโดยใช้ Implenents

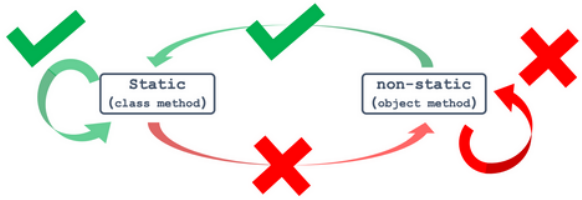
Polymorphism

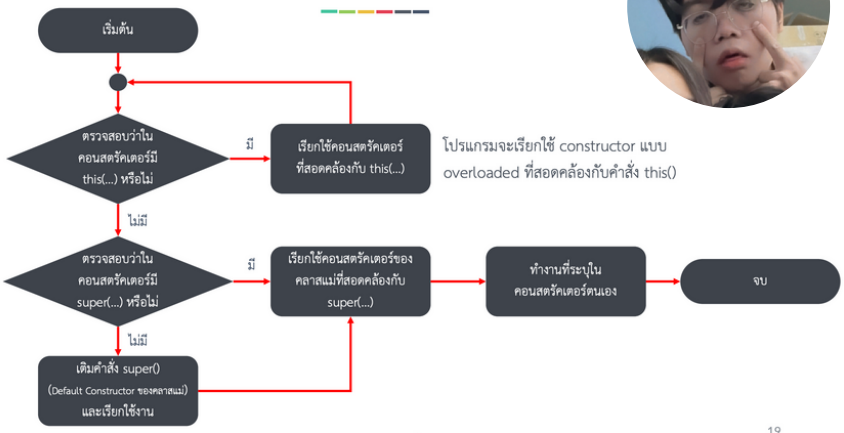
- การ Overloading / Overriding

- การเรียกใช้เมธอด*ภายในคลาสเดียวกัน*

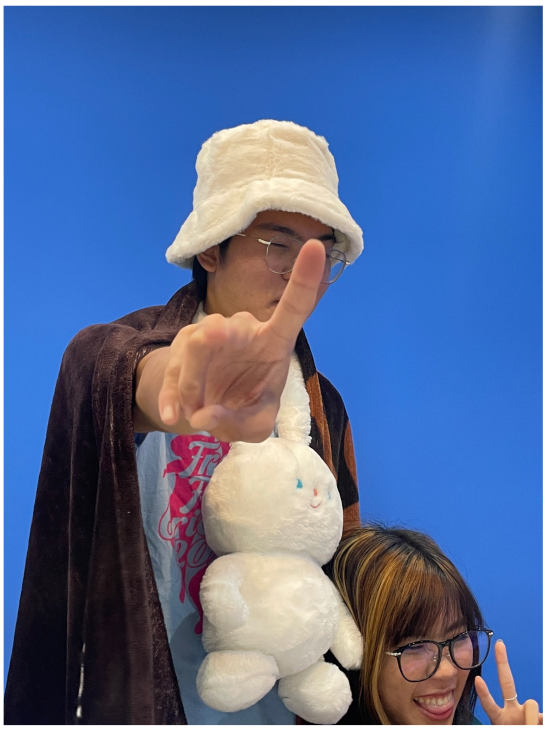


- การเรียกใช้เมธอด*ระหว่างคลาส*



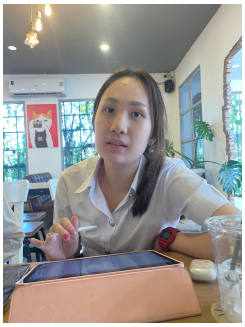


โปรแกรมจะเรียกใช้ constructor แบบ overloaded ที่สอดคล้องกับคำสั่ง this()



ใจเกเร

บอลลูนโอเค
กับสิ่งนี้

มึงกล้าเอากู
มาทำ MEME หรอ

พอเถอะอานนท์