เขียนโปรแกรมด้วยภาษา Java เบื้องต้น ฉบับปรับปรุง (2020)

เหมาะสำหรับ

- ผู้ที่สนใจเรียนรู้การเขียนโปรแกรมด้วยตนเอง
- ไม่มีความรู้เรื่องการเขียนโปรแกรมก็เรียนได้
- เรียนฟรี!!

ขอบเขต

- ทฤษฎีการเขียนโปรแกรมด้วยภาษา Java
- แก้โจทย์ปัญหาเพื่อนำความรู้ไปใช้งานจริงได้

ระยะเวลาในการสอน

- ทยอยลงคลิปเพลลิสต์

  การเขียนโปรแกรมด้วยภาษา Java เบื้องต้น [2020]

Phase.1

- ดาวน์โหลดและติดตั้ง JDK
- การกำหนด Path สำหรับ Windows
- ดาวน์โหลดและติดตั้ง TextEditor

การแสดงผลทางจอภาพ

- แสดงผลทางจอภาพ
- Comment
- Single Line / Multiple Line

Phase.1

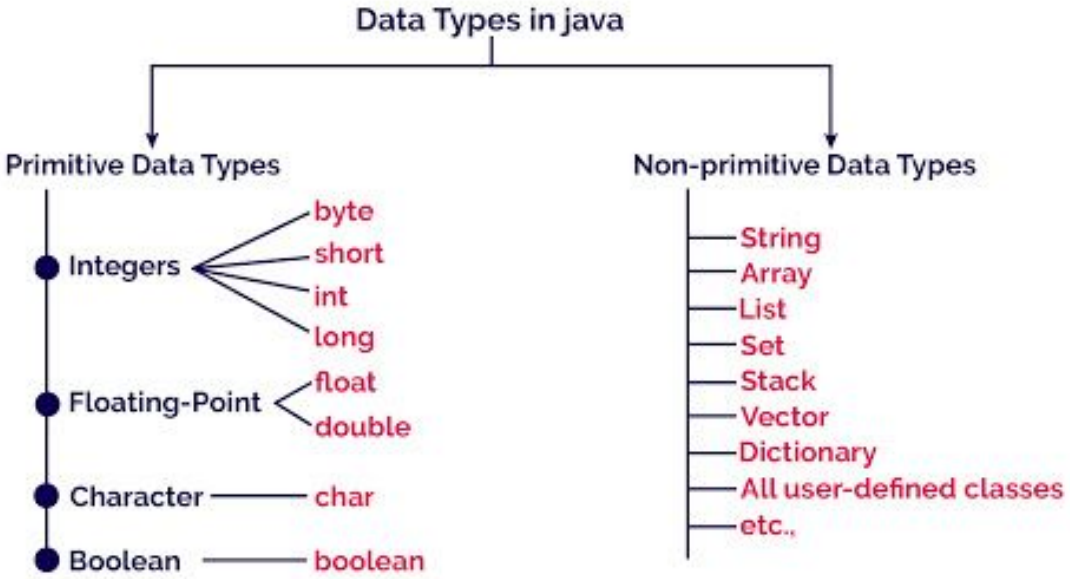
- ตัวแปรและชนิดข้อมูล
- กฎการตั้งชื่อ
- Keywords

ตัวแปรและชนิดข้อมูล

ตัวแปร คือ ชื่อที่ถูกนิยามขึ้นมาเพื่อใช้เก็บค่าข้อมูลลงไปในหน่วยความจำ สำหรับนำไปใช้งานในโปรแกรม โดยข้อมูลอาจจะประกอบด้วย ข้อความ ตัวเลข ตัวอักษร หรือผลลัพธ์จากการประมวลผลข้อมูล

รูปแบบการตั้งชื่อ

1. Class กำหนดให้ตัวอักษรตัวแรกเป็นตัวพิมพ์ใหญ่ที่เหลือเป็นพิมพ์เล็ก เช่น MyClass , HelloWorld
2. Data / ตัวแปร ทุกตัวเป็นตัวพิมพ์เล็ก เช่น color , name , age
3. ค่าคงที่ ตัวพิมพ์ใหญ่ทั้งหมด เช่น SIZE , WIDTH, HEIGHT


Data Types in java
Primitive Data Types
Integers
byte
short
int
long
Floating-Point
float
double
Character
char
Boolean
boolean
Non-primitive Data Types
String
Array
List
Set
Stack
Vector
Dictionary
All user-defined classes
etc.,

Data Type	คำอธิบาย	ขนาด (Bit)
boolean	ค่าทางตรรกศาสตร์	8 (เก็บค่า True /False)
byte	ตัวเลขที่ไม่มีจุดทศนิยม	8
short	ตัวเลขที่ไม่มีจุดทศนิยม	16
int	ตัวเลขที่ไม่มีจุดทศนิยม	32
long	ตัวเลขที่ไม่มีจุดทศนิยม	64
float	ตัวเลขที่มีจุดทศนิยม	32
double	ตัวเลขที่มีจุดทศนิยม	64
char	ตัวอักษร	16

ชนิดข้อมูลจะเป็นตัวกำหนดค่าที่สามารถเก็บได้ในตัวแปร
ยิ่งจำนวนของ bit มากเท่าไร แสดงว่าเราสามารถเก็บค่าได้มากเท่านั้น

Data Type	ค่าต่ำสุด	ค่าสูงสุด
boolean	ค่าทางตรรกศาสตร์	8 (เก็บค่า True /False)
byte	-128	127
short	-32768	32767
int	-2147483648	2147483647
long	-9223372036854775808	9223372036854775807
float	1.4E-45	3.4028235E38
double	4.9E-324	1.79769313486231 57E308
char	-	-

ชนิดข้อมูลจะเป็นตัวกำหนดค่าที่สามารถเก็บได้ในตัวแปร
ยิ่งจำนวนของ bit มากเท่าไร แสดงว่าเราสามารถเก็บค่าได้มากเท่านั้น

การนิยามตัวแปร

ชนิดข้อมูล ชื่อตัวแปร ;

ชนิดข้อมูล ชื่อตัวแปร = ค่าเริ่มต้น;

แบบหลายตัวแปรในบรรทัดเดียว

ชนิดข้อมูล ชื่อตัวแปร = ค่าเริ่มต้น,ชื่อตัวแปร = ค่าเริ่มต้น

ให้นำค่าทางขวามือของเครื่องหมาย = ไปเก็บไว้ในตัวแปรที่อยู่ด้านซ้ายมือ

การนิยามค่าคงที่

final ชนิดข้อมูล ชื่อตัวแปร ;

final ชนิดข้อมูล ชื่อตัวแปร = ค่าเริ่มต้น;

แบบหลายตัวแปรในบรรทัดเดียว

final ชนิดข้อมูล ชื่อตัวแปร = ค่าเริ่มต้น,ชื่อตัวแปร = ค่าเริ่มต้น

ให้นำค่าทางขวามือของเครื่องหมาย = ไปเก็บไวในตัวแปรที่อยู่ด้านซ้ายมือ

กฎการตั้งชื่อตัวแปร

- ประกอบด้วยตัวเลข ตัวอักษร เครื่องหมาย
- อักษรตัวแรกห้ามขึ้นต้นด้วยตัวเลขและสัญลักษณ์พิเศษ ยกเว้น _ (Underscore)
- ห้ามซ้ำกับคำสงวน (Keyword)
- Case Sensitive

Keywords In Java

1. abstract
2. assert
3. boolean
4. break
5. byte
6. case
7. catch
8. char
9. class
10. continue
11. default
12. do
13. double
14. else
15. enum
16. extends
17. final
18. finally
19. float
20. for
21. if
22. implements
23. import
24. instanceof
25. int
26. interface
27. long
28. native
29. new
30. package
31. private
32. protected
33. public
34. return
35. short
36. static
37. strictfp
38. super
39. switch
40. synchronized
41. this
42. throw
43. throws
44. transient
45. try
46. void
47. volatile
48. while

https://www.pixeltrice.com/keywords-in-java/

Global Variable / Local Variable

- การแปลงชนิดข้อมูล
- เช็คชนิดข้อมูล
- รับ Input ผ่านทาง Keyboard

การแปลงชนิดข้อมูล (Type Casting)

1.Widening Casting
คือการแปลงข้อมูลที่มีขนาดเล็กไปสู่ข้อมูลขนาดใหญ่ (แบบอัตโนมัติ)

byte -> short -> char -> int -> long -> float -> double

2.Narrowing Casting
คือการแปลงข้อมูลที่มีขนาดใหญไปสู่ข้อมูลที่มีขนาดเล็ก (ทำเอง)

double -> float -> long -> int -> char -> short -> byte

ตัวดำเนินการ (Operator)

กลุ่มของเครื่องหมายหรือสัญลักษณ์ที่ใช้ในการเขียนโปรแกรม

A+B

1. ตัวดำเนินการ (Operator)
2. ตัวถูกดำเนินการ (Operand)

ตัวดำเนินการทางคณิตศาสตร์

Operator	คำอธิบาย
+	บวก
-	ลบ
*	คูณ
/	หาร
%	หารเอาเศษ

ตัวดำเนินการเปรียบเทียบ

**** ชนิดข้อมูล boolean

Operator	คำอธิบาย
==	เท่ากับ
!=	ไม่เท่ากับ
>	มากกว่า
<	น้อยกว่า
>=	มากกว่าเท่ากับ
<=	น้อยกว่าเท่ากับ

ตัวดำเนินการทางตรรกศาสตร์

Operator	คำอธิบาย
&&	AND
||	OR
!	NOT

ตัวดำเนินการทางตรรกศาสตร์

a	!a	a	b	a && b	a || b
true	false	false	false	false	false
false	true	false	true	false	true
		true	false	false	true
		true	true	true	true

ตัวดำเนินการเพิ่มค่า - ลดค่า

Operator	รูปแบบการเขียน	ความหมาย
++ (Prefix)	++a	เพิ่มค่าให้ a ก่อน 1 ค่าแล้วนำไปใช้
++ (Postfix)	a++	นำค่าปัจจุบันใน a ไปใช้ก่อนแล้วค่อยเพิ่มค่า
-- (Prefix)	--b	ลดค่าให้ b ก่อน 1 ค่าแล้วนำไปใช้
-- (Postfix)	b--	นำค่าปัจจุบันใน b ไปใช้ก่อนแล้วค่อยลดค่า

Compound Assignment

Assignment	รูปแบบการเขียน	ความหมาย
+=	x+=y	x=x+y
-=	x-=y	x=x-y
*=	x*=y	x=x*y
/=	x/=y	x=x/y
%=	x%=y	x=x%y

ลำดับความสำคัญของตัวดำเนินการ

ลำดับที่	เครื่องหมาย	ลำดับการทำงาน
1	( )	
2	++ , --	ซ้ายไปขวา
3	* , / , %	ซ้ายไปขวา
4	+ , -	ซ้ายไปขวา
5	< , <= , > , >=	ซ้ายไปขวา
6	== , !=	ซ้ายไปขวา
7	&&	ซ้ายไปขวา
8	||	ซ้ายไปขวา
9	= , += , -= , *= , /= , %=	ขวาไปซ้าย

กรณีศึกษา

1. 5+8 *9
2. 10 - 4+2
3. 10 - (2+1)
4. 5 * 2 – 40 / 5
5. 7+8/2+25

Assignment 1: โปรแกรมคำนวณค่าดัชนีมวลกาย (BMI)

$$\text{ดัชนีมวลกาย (BMI)} = \frac{\text{น้ำหนักตัว (กิโลกรัม)}}{\text{ส่วนสูง (เมตร)}^2}$$

ยกตัวอย่าง เช่น ถ้ามีน้ำหนัก 60 กิโลกรัม และสูง 155 ซม.
ดัชนีมวลกาย (BMI) = 24.97

โครงสร้างควบคุม (Control Structure)

คือ กลุ่มคำสั่งที่ใช้ควบคุมการทำงานของโปรแกรม

- แบบลำดับ (Sequence)
- แบบมีเงื่อนไข (Condition)
- แบบทำซ้ำ (Loop)

แบบมีเงื่อนไข (Condition)

กลุ่มคำสั่งที่ใช้ตัดสินใจในการเลือกเงื่อนไขต่างๆ ภายในโปรแกรมมาทำงาน

- if
- Switch..Case

รูปแบบคำสั่งแบบเงื่อนไขเดียว

- **if statement**

เป็นคำสั่งที่ใช้กำหนดเงื่อนไขในการตัดสินใจทำงานของโปรแกรม

ถ้าเงื่อนไขเป็นจริงจะทำตามคำสั่งต่างๆ ที่กำหนดภายใต้เงื่อนไขนั้นๆ

```
if(เงื่อนไข){
    คำสั่งเมื่อเงื่อนไขเป็นจริง ;
}
```

รูปแบบคำสั่งแบบ 2 เงื่อนไข

```
if(เงื่อนไข){
    คำสั่งเมื่อเงื่อนไขเป็นจริง ;
}else{
    คำสั่งเมื่อเงื่อนไขเป็นเท็จ ;
}
```

ข้อควรระวังการเขียน if เพื่อตรวจสอบเงื่อนไข

```
if(เงื่อนไข){
    คำสั่งเมื่อเงื่อนไขเป็นจริง ;
}if(เงื่อนไข){
    คำสั่งเมื่อเงื่อนไขเป็นจริง ;
}
```

รูปแบบคำสั่งแบบหลายเงื่อนไข

```
if(เงื่อนไขที่ 1){
        คำสั่งเมื่อเงื่อนไขที่ 1 เป็นจริง ;
}elseif(เงื่อนไขที่ 2){
        คำสั่งเมื่อเงื่อนไขที่ 2 เป็นจริง ;
}elseif(เงื่อนไขที่ 3){
        คำสั่งเมื่อเงื่อนไขที่ 3 เป็นจริง ;
}else{
        คำสั่งเมื่อทุกเงื่อนไขเป็นเท็จ ;
}
```

if..else แบบลดรูป (Ternary Operator)

ตัวแปร = (เงื่อนไข) ? คำสั่งเมื่อเงื่อนไขเป็นจริง : คำสั่งเมื่อเงื่อนไขเป็นเท็จ;

```
if(เงื่อนไข){
    คำสั่งเมื่อเงื่อนไขเป็นจริง
}else{
    คำสั่งเมื่อเงื่อนไขเป็นเท็จ
}
```

การเขียน if ซ้อน if

```
if(เงื่อนไขที่ 1){
    if(เงื่อนไขที่ 2 ){
        คำสั่งเมื่อเงื่อนไขที่ 2 เป็นจริง ;
    }
}
```

แบบมีเงื่อนไข (Condition)

กลุ่มคำสั่งที่ใช้ตัดสินใจในการเลือกเงื่อนไขต่างๆ ภายในโปรแกรมมาทำงาน

- **Switch..Case**

Switch เป็นคำสั่งที่ใช้กำหนดเงื่อนไขคล้ายๆกับ if แต่จะเลือกเพียงหนึ่งทางเลือกออกมาทำงานโดยนำค่าในตัวแปรมากำหนดเป็นทางเลือกผ่านคำสั่ง case

รูปแบบคำสั่ง

```
switch(สิ่งที่ต้องการตรวจสอบ) {

        case ค่าที่ 1 : คำสั่งที่ 1;
                break;
         case ค่าที่ 2 : คำสั่งที่ 2;
                break;
                .....
         case ค่าที่ N : คำสั่งที่ N;
                break;

        default : คำสั่งเมื่อไม่มีค่าที่ตรงกับที่ระบุใน case
}
```

***คำสั่ง break จะทำให้โปรแกรมกระโดดออกไปทำงานนอกคำสั่ง switch ถ้าไม่มีคำสั่ง break โปรแกรมจะทำคำสั่งต่อไปเรื่อยๆ จนจบการทำงาน

รูปแบบคำสั่ง

```
switch(month) {

        case 1 : System.out.println(“มกราคม”);
            break;
        case 2: System.out.println(“กุมภาพันธ์”);
        break;
                .....
         case ค่าที่ N : คำสั่งที่ N;
                  break;
        default : System.out.println(“ไม่พบเดือน”);
}
```

กำหนดให้ตัวแปร month เก็บตัวเลข

รูปแบบคำสั่ง

```
switch(panic) {

        case “ปวดหัว”: System.out.println(“พาราเซ
 ตามอล”);
            break;
        case “ปวดท้อง”: System.out.println(“แอนตาซิล”);
       break;
            .....
        case ค่าที่ N : คำสั่งที่ N;
            break;
        default : System.out.println(“ยาอื่นๆ”);
}
```

กำหนดให้ตัวแปร panic เก็บข้อความ

Switch..Case VS if Statement

```
switch(month) {
                case 1 :
  System.out.println(“มกราคม”);
                    break;
                case 2:
  System.out.println(“กุมภาพันธ์”);
            break;
                        .....
                 case ค่าที่ N : คำสั่งที่ N;
                        break;
                default : System.out.println(“ไม่พบ
  เดือน”);
}
```

```
if(month==1){
        System.out.println(“มกราคม”);
}elseif(month==2){
        System.out.println(“กุมภาพันธ์”);
}elseif(เงื่อนไขที่ 3){
        คำสั่งเมื่อเงื่อนไขที่ 3 เป็นจริง ;
}else{
        System.out.println(“ไม่พบเดือน”);
}
```

แบบทำซ้ำ (Loop)

กลุ่มคำสั่งที่ใช้ในการวนรอบ (loop) โปรแกรมจะทำงานไปเรื่อยๆจนกว่าเงื่อนไขที่กำหนดไว้จะเป็นเท็จ จึงจะหยุดทำงาน

- While
- For
- Do..While

คำสั่งที่เกี่ยวข้องกับ Loop

- **break** ถ้าโปรแกรมพบคำสั่งนี้จะหลุดจากการทำงานในลูปทันที เพื่อไปทำคำสั่งอื่นที่อยู่นอกลูป

- **continue** คำสั่งนี้จะทำให้หยุดการทำงานแล้วย้อนกลับไปเริ่มต้นการทำงานที่ต้นลูปใหม่

คำสั่ง While

- **While Loop**

จะทำงานตามคำสั่งภายใน while ไปเรื่อยๆเมื่อเงื่อนไขที่กำหนดเป็นจริง

```
while(เงื่อนไข){
    คำสั่งที่จะทำซ้ำเมื่อเงื่อนไขเป็นจริง ;
}
```

คำสั่ง For

- **For Loop**

เป็นรูปแบบที่ใช้ในการตรวจสอบเงื่อนไข มีการกำหนดค่าเริ่มต้น และเปลี่ยนค่าไปพร้อมๆกัน เมื่อเงื่อนไขในคำสั่ง for เป็นจริงก็จะทำงานตามคำสั่งที่แสดงไว้ภายในคำสั่ง for ไปเรื่อยๆ

โครงสร้างคำสั่ง

```
for(ค่าเริ่มต้นของตัวแปร; เงื่อนไข; เปลี่ยนแปลงค่าตัวแปร) {
        คำสั่งเมื่อเงื่อนไขเป็นจริง;
}
```

```
for(int i = 1;i<=10;i++) {
    คำสั่งเมื่อเงื่อนไขเป็นจริง;
}
```

คำสั่ง Do..While

- **Do..While**

โปรแกรมจะทำงานตามคำสั่งอย่างน้อย 1 รอบ เมื่อทำงานเสร็จจะมาตรวจสอบเงื่อนไขที่คำสั่ง while ถ้าเงื่อนไขเป็นจริงจะวนกลับขึ้นไปทำงานที่คำสั่งใหม่อีกรอบ แต่ถ้าเป็นเท็จจะหลุดออกจากลูป

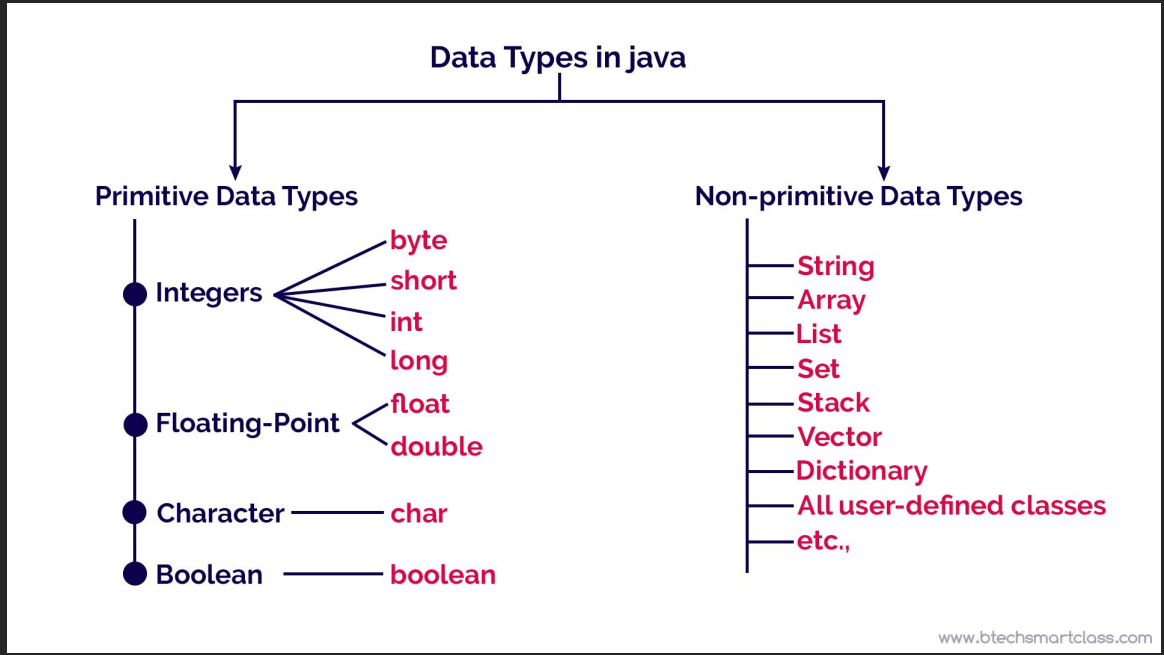
โครงสร้างคำสั่ง

```
do {
        คำสั่งต่างๆ เมื่อเงื่อนไขเป็นจริง;
 } while(เงื่อนไข);
```

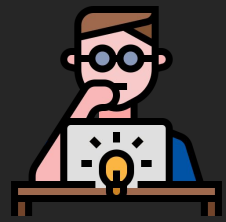
ข้อแตกต่างและการใช้งาน Loop

- For ใช้ในกรณีรู้จำนวนรอบที่ชัดเจน
- While ใช้ในกรณีที่ไม่รู้จำนวนรอบ
- Do..while ใช้ในกรณีที่อยากให้ลองทำก่อน 1 รอบ แล้วทำซ้ำไปเรื่อยๆทราบเท่าที่เงื่อนไขเป็นจริง

Primitive Data Type & Non Primitive Data Type



http://www.btechsmartclass.com/java/java_images/java-data-types.jpg



ข้อจำกัดของชนิดข้อมูลพื้นฐาน

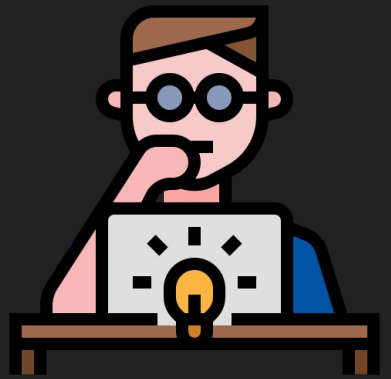
การประกาศตัวแปรแต่ละครั้ง

ตัวแปร 1 ตัวสามารถเก็บข้อมูลได้แค่ 1 ค่าเท่านั้น เช่น

```
int number = 1;
```

ถ้าอยากเก็บเลข 10 ค่าต้องทำอย่างไร ?

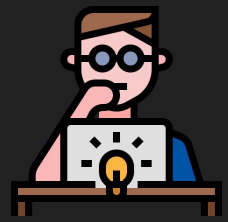
ต้องประกาศตัวแปร 10 ตัวแปร หรือไม่ ?



Array

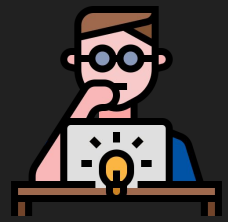
Array คืออะไร

ความหมายที่ 1 ชุดของตัวแปรที่อยู่ในรูปลำดับใช้เก็บค่าข้อมูลให้อยู่ในกลุ่มเดียวกัน ข้อมูลภายในอาร์เรย์จะถูกเก็บบนหน่วยความจำในตำแหน่งที่ต่อเนื่องกัน โดยขนาดของอาร์เรย์จะเล็กหรือใหญ่ขึ้นกับจำนวนมิติที่กำหนดขึ้น



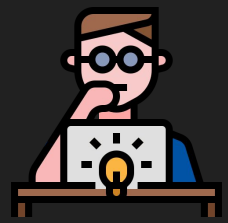
Array คืออะไร

ความหมายที่ 2 เป็นตัวแปรที่ใช้ในการเก็บข้อมูลที่มีลำดับที่ต่อเนื่อง ซึ่งข้อมูลมีค่าได้หลายค่าโดยใช้ชื่ออ้างอิงได้เพียงชื่อเดียว และใช้หมายเลขกำกับ (index) ให้กับตัวแปรเพื่อจำแนกความแตกต่างของค่าตัวแปรแต่ละตัว



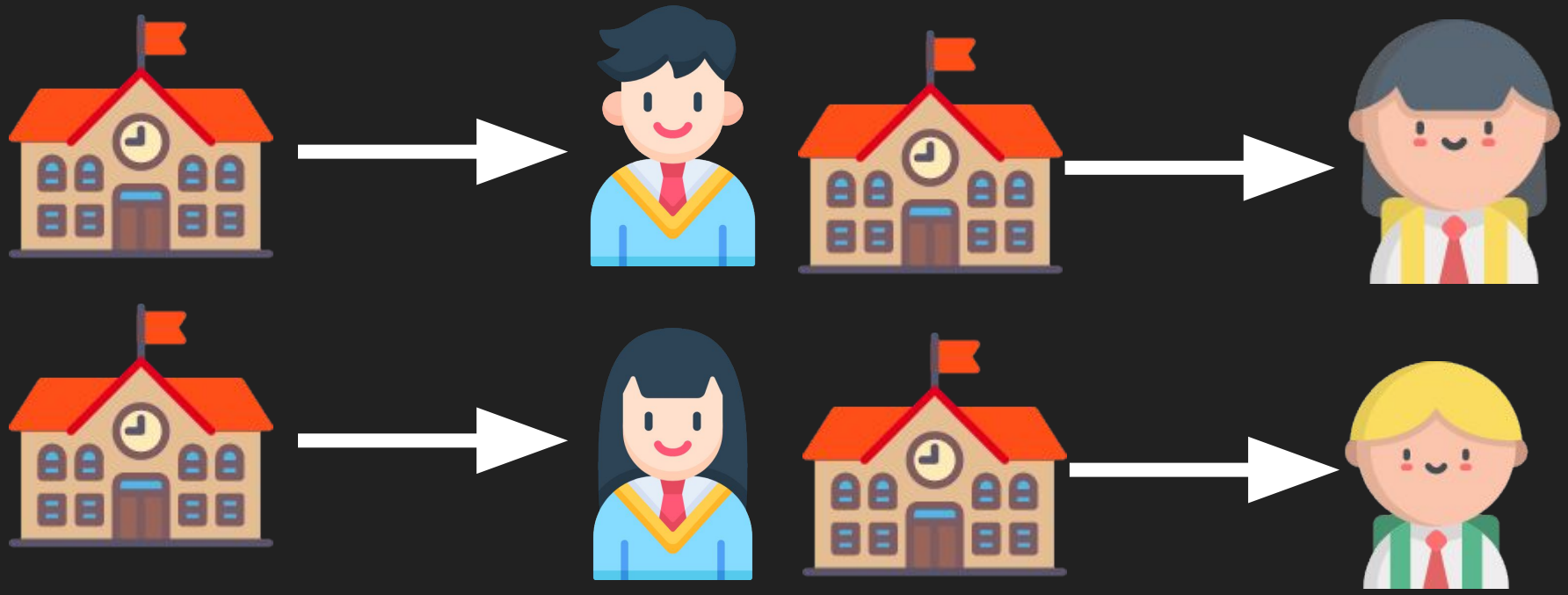
คุณสมบัติของ Array

1. ใช้เก็บกลุ่มของข้อมูล
2. ข้อมูลที่อยู่ในอาร์เรย์จะเรียกว่าสมาชิก หรือ อิลิเมนต์ (element)
3. แต่ละอิลิเมนต์ (element) จะเก็บค่าข้อมูล (value) และอินเด็กซ์ (Index) เอาไว้
4. Index หมายถึงคีย์ของอาร์เรย์ใช้อ้างอิงตำแหน่งของ element **เริ่มต้นที่ 0**
5. สมาชิกใน array ต้องมี**ชนิดข้อมูลเหมือนกัน**
6. สมาชิกใน array จะถูกคั่นด้วยเครื่องหมาย comma

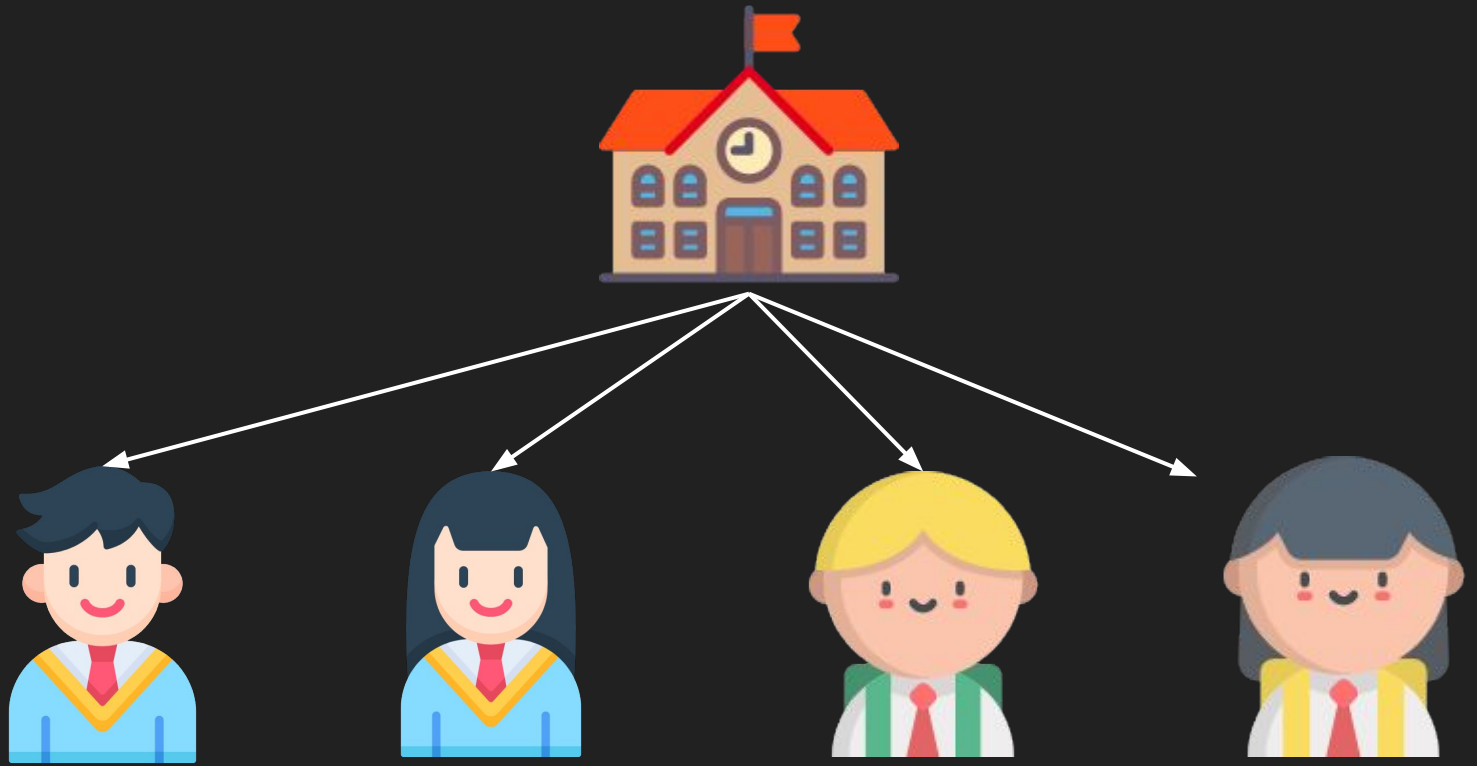


ตัวแปร = โรงเรียน

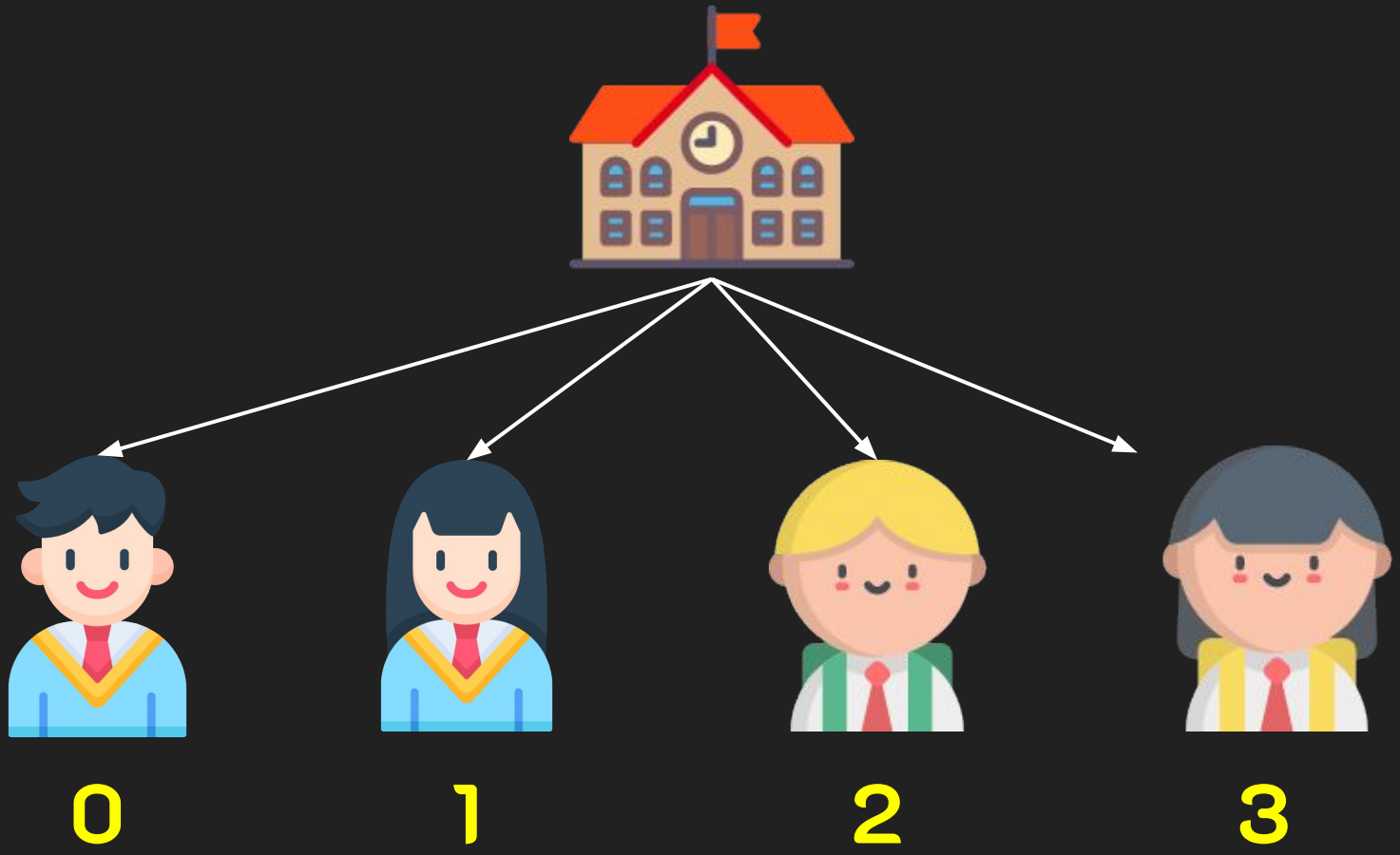
ค่าที่เก็บในตัวแปร = นักเรียน

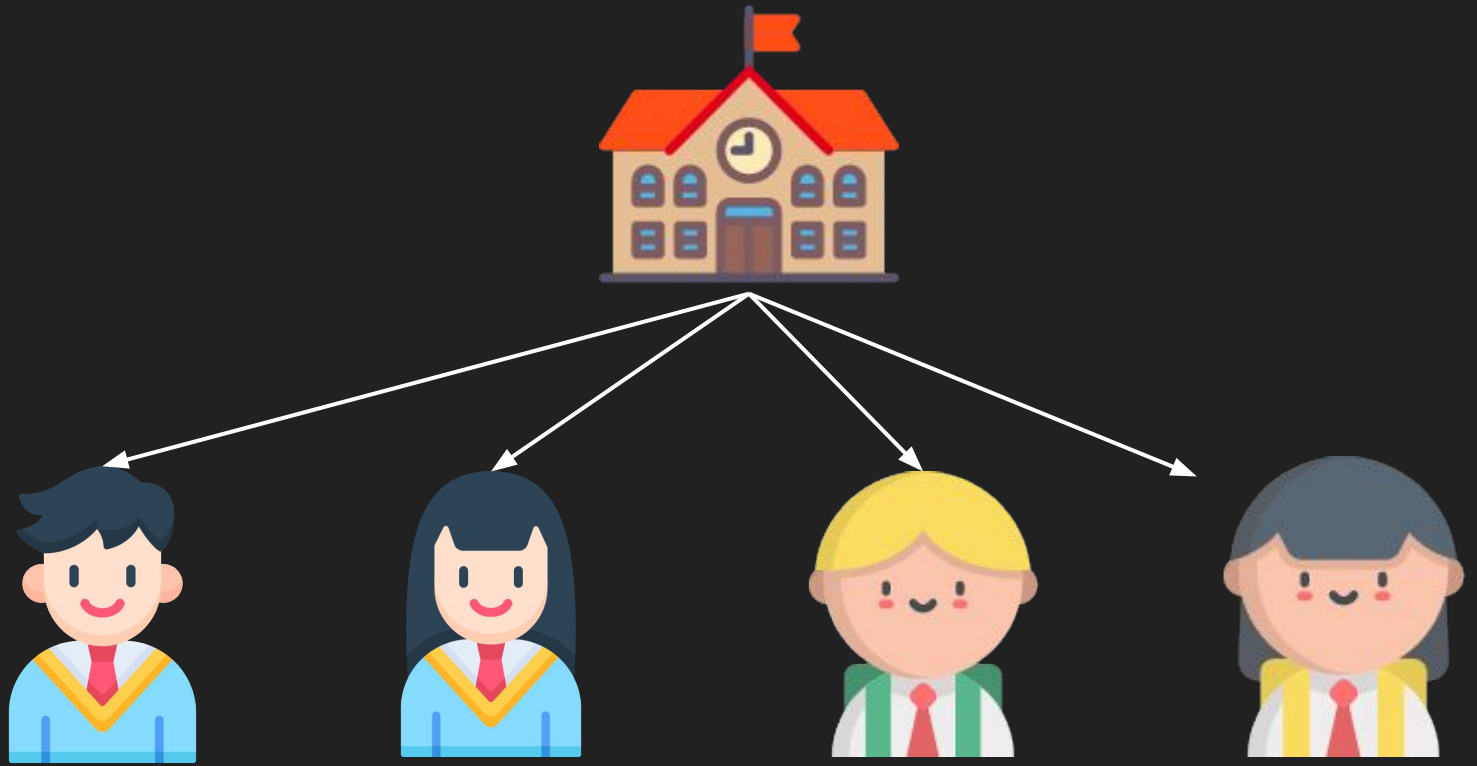


การสร้างตัวแปรแบบปกติ



การสร้างตัวแปรแบบ Array


0
1
2
3



ข้อจำกัด คือ มีขนาดที่แน่นอน

ข้อจำกัด คือ ต้องเป็นนักเรียนเท่านั้น!!

การประกาศใช้ Array แบบ Primitive Data Type

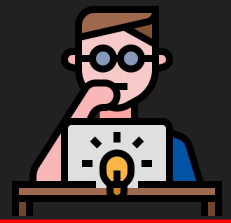
การประกาศ

ชนิดข้อมูล [] ชื่อตัวแปร = new ชนิดข้อมูล [ขนาด];

int[] number = new int[4];

หรือ

ชนิดข้อมูล [] ชื่อตัวแปร = {สมาชิก,....};



การกำหนดค่า

```
number[0] = 10;
number[1] = 20;
number[2] = 30;
number[3] = 40;
```

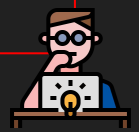
การสร้าง Array แบบ Primitive Data Type

```
int[] number = {10, 20, 30, 40};
```

10	20	30	40

```
String [] pets = {“แมว”,”กระต่าย”};
```

แมว	กระต่าย



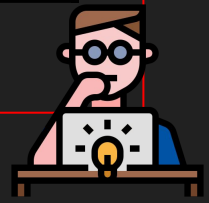
การเข้าถึงสมาชิก Array

int[] number = {10, 20, 30, 40};

10 (0)	20 (1)	30 (2)	40 (3)

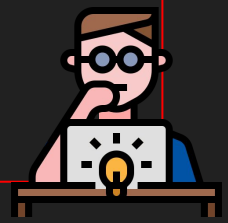
String [] pets = {“แมว”,”กระต่าย”};

แมว (0)	กระต่าย (1)



การเปลี่ยนแปลงข้อมูลสมาชิก Array

```
int[] number = {10, 20, 30, 40};
number[1] = 100;
String [] pets = {“แมว”,”กระต่าย”};
pets [1] = “เต่า”;
```



นับจำนวนสมาชิกใน Array

```
int[] number = {10, 20, 30, 40};
number.length;
String [] pets = {“แมว”,”กระต่าย”};
pets.length;
```

การเข้าถึงสมาชิกด้วย For Loop

```
String [] pets = {“แมว”,”กระต่าย”};

for (int i = 0; i < pets.length; i++) {
    System.out.println(pets[i]);
}
```

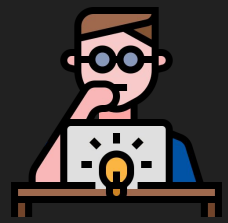
การเข้าถึงสมาชิกด้วย ForEach

```
String [] pets = {“แมว”,”กระต่าย”};

    for (String name : pets) {
        System.out.println(name);
    }
```

Array 2 มิติ

- Array ที่มีข้อมูลสมาชิกภายในเป็น array (array ซ้อน array) เปรียบเสมือนกัน matrix
- มีโครงสร้างเป็นรูปแบบแถว (แนวนอน) และคอลัมน์ (แนวตั้ง)



Array 2 มิติ

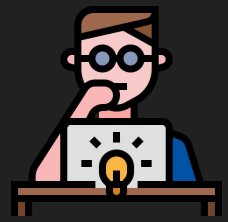
ชนิดข้อมูล [][] ชื่อตัวแปร = {

{Array ตัวที่ 1},

{Array ตัวที่ 2},

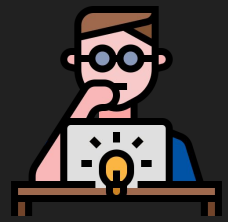
{Array ตัวที่ 3}

}



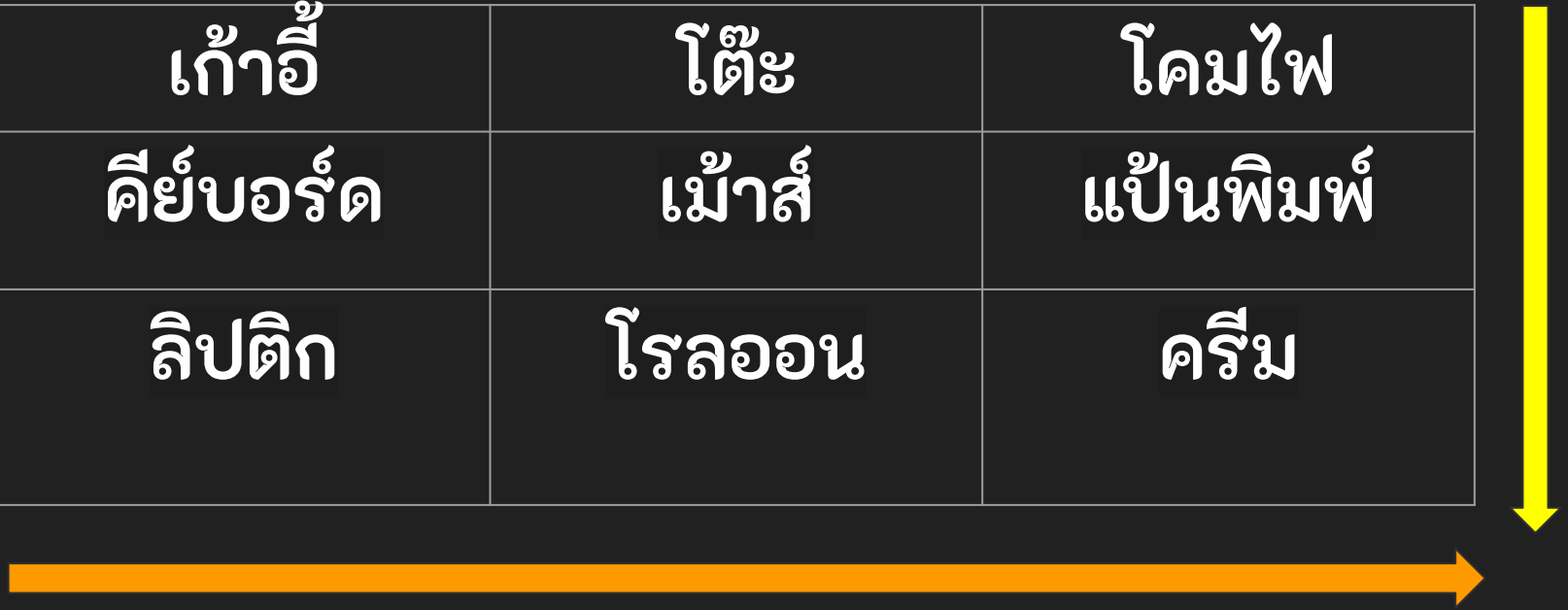
Array 2 มิติ

```
String [][] products = {
    {“เก้าอี้”,”โต๊ะ”,”โคมไฟ”},
    {“คีย์บอร์ด”,”เม้าส์”,”แป้นพิมพ์”},
    {“ลิปติก”,”โรลออน”,”ครีม”}
}
```



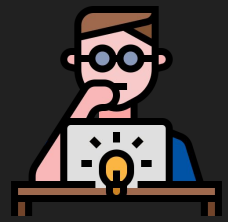
Array 2 มิติ

	คอลัมน์ที่ 0	คอลัมน์ที่ 1	คอลัมน์ที่ 2
แถวที่ 0	เก้าอี้	โต๊ะ	โคมไฟ
แถวที่ 1	คีย์บอร์ด	เม้าส์	แป้นพิมพ์
แถวที่ 2	ลิปติก	โรลออน	ครีม



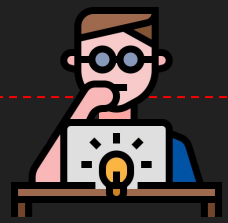
การเข้าถึงข้อมูลใน Array 2 มิติ

- ชื่อตัวแปร [เลขแถว][เลขคอลัมน์];
- เช่น products [0][1];



การเข้าถึงข้อมูลด้วย Loop

```
for (int rows = 0; rows < products.length; rows++) {
        for(int column = 0; column < products[rows].length; column++) {
                System.out.println(products[rows][column]);
        }
}
```



เมธอด
(Method)

เมธอด (Method) คืออะไร

ชุดคำสั่งที่นำมาเขียนรวมกันเป็นกลุ่มเพื่อให้เรียกใช้งานตามวัตถุประสงค์ที่ต้องการ และลดความซ้ำซ้อนของคำสั่งที่ใช้งานบ่อยๆ เมธอดสามารถนำไปใช้งานได้ทุกที่และแก้ไขได้ในภายหลัง ทำให้โค้ดในโปรแกรมมีระเบียบและใช้งานได้สะดวกมากยิ่งขึ้น

**ใช้ความรู้เรื่อง OOP มาประยุกต์ใช้

การสร้างเมธอด (Method)

เมื่อสร้างเมธอดในภาษา Java สามารถที่จะเรียกใช้งานได้จากส่วนใดๆ ของโปรแกรมก็ได้ขึ้นกับขอบเขตและระดับการเข้าถึงที่ผู้เขียนได้กำหนดขึ้น

```
type name ( parameter1, parameter2, ... ) {
    statements
}

access_modifier type name ( parameter1, parameter2, ... ) {
    statements
}
```

รูปแบบของเมธอด

1.เมธอดที่ไม่มีการรับและส่งค่า

```
modifier void ชื่อเมธอด(){
    // คำสั่งต่างๆ
}
```

การเรียกใช้งานเมธอด

```
ชื่อเมธอด ();
```

รูปแบบของเมธอด

2.เมธอดที่มีการรับค่าเข้ามาทำงาน

modifier void ชื่อเมธอด(parameter1,parameter2,.....){

// กลุ่มคำสั่งต่างๆ

}

อาร์กิวเมนต์ คือ ตัวแปรหรือค่าที่ต้องการส่งมาให้กับเมธอด (ตัวแปรส่ง)
พารามิเตอร์ คือ ตัวแปรที่เมธอดสร้างไว้สำหรับรับค่าที่จะส่งเข้ามาให้กับเมธอด (ตัวแปรรับ)

การเรียกใช้งานเมธอด

ชื่อเมธอด (argument1,argument2,.....);

รูปแบบของเมธอด (Method)

3.เมธอดที่มีส่งค่าออกมา

```
modifier type ชื่อเมธอด(){
    return ค่าที่จะส่งออกไป (type)
}
```

รูปแบบของเมธอด

4.เมธอดที่มีการรับค่าเข้ามาและส่งค่าออกไป

```
modifier type ชื่อเมธอด(parameter1,parameter2,.....){
    retrun ค่าที่จะส่งออกไป
}
```

เมธอดที่รับค่า Array

```
modifier void ชื่อเมธอด (type [] arr){

    // คำสั่งต่างๆ

}
```

เมธอดที่รับค่า Array และคืนค่าออกไป

```
modifier type [] ชื่อเมธอด (type [] arr){
    // คำสั่งต่างๆ
}
```

เมธอดที่รับค่า Array และคืนค่า Array

```
modifier []type ชื่อเมธอด (type [] arr){
    // คำสั่งต่างๆ
     return []
}
```

Variable Arguments(var-args)

```
modifier type ชื่อเมธอด (type...arr){
    // คำสั่งต่างๆ
}
```

Utility Methods (Array)

การสร้าง Method สำหรับจัดการ Array

- แสดงสมาชิกใน Array
- หาค่าสูงสุดของสมาชิก
- การเปรียบเทียบ Array
- การสลับค่าใน Array
- การ Copy Array
- การค้นหาข้อมูลใน Array

Utility Methods (Character)

เจาะลึก Character

- สร้าง Object Character ด้วย Class
- เช็ครูปแบบตัวอักษรด้วย isletter , isDigit
- isUpperCase , isLowerCase
- toUpperCase , toLowerCase

Utility Methods (String)

เจาะลึก String

- สร้าง Object ด้วย Class
- Concatenation
- หาความยาว String
- หาตำแหน่งตัวอักษร (charAt)
- เปรียบเทียบ String (equals & equalsIgnorecase)
- หาข้อความที่อยู่หน้าสุด (startsWith)
- หาข้อความที่อยู่ท้ายสุด (endsWith)
- หาตำแหน่งคำในข้อความ (IndexOf)

เจาะลึก String

- แทนที่ข้อความ (replace , replaceFirst)
- การหั่น String (split)
- หาข้อความย่อย (subString)
- แปลง String เป็น Character
- แปลง Character เป็น String
- การตัดช่องว่างใน String
- แปลงเป็นตัวพิมพ์เล็ก - พิมพ์ใหญ่
- แปลงตัวเลขเป็น String (valueOf)

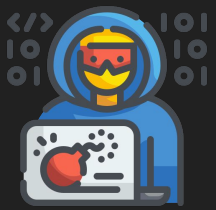
จัดการข้อผิดพลาด (Exception)

Exception

การที่โปรแกรมทำงานบางอย่างแต่เกิดข้อผิดพลาดขึ้นแล้วโปรแกรมไม่สามารถจัดการข้อผิดพลาดนั้นได้ ซึ่งทำให้เกิดสิ่งผิดปกติหรือ Exception ส่งผลทำให้โปรแกรมหยุดทำงาน

ตัวอย่าง Exception

- ArrayIndexOutOfBoundsException
- ArithmeticException
- ZeroDivisionException
- IOException
- FileNotFoundException
- อื่นๆ



จัดการ Exception ด้วย Try...Catch

```
try{
    // ลองทำคำสั่งในนี้
}catch(Exception e){
    // ถ้าเกิดข้อผิดพลาดจะมาทำตรงส่วนนี้
}
```

Try...Catch แบบหลายเหตุการณ์

```
try {
    // ลองทำคำสั่งในนี้
} catch (ExceptionType1 e1) {
   // ถ้าเกิดข้อผิดพลาดที่ 1 จะมาทำตรงส่วนนี้
} catch (ExceptionType2 e2) {
    // ถ้าเกิดข้อผิดพลาดที่ 2 จะมาทำตรงส่วนนี้
}
```

Finally เมื่อเกิดข้อผิดพลาด หรือ ไม่เกิด ก็จะทำงานคำสั่งในส่วนนี้ทุกครั้ง คำสั่งที่ระบุมักจะเป็นคำสั่งที่ทำงานส่วนที่สำคัญของโปรแกรม เช่น ปิดไฟล์ ปิดการเชื่อมต่อฐานข้อมูล หรือ คำสั่ง Disconnect กับ Server

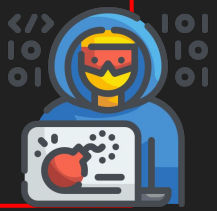
```
try{

   // ลองทำคำสั่งในนี้

}catch(Exception e){

    // ถ้าเกิดข้อผิดพลาดจะมาทำตรงส่วนนี้

}finally {
        // คำสั่งต่างๆ
}
```



จัดการ Exception ด้วย Throw

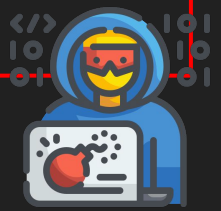
```
if (age <=20) {

    throw new ArithmeticException("ตัวเลขไม่ถูกต้อง");

 }
```

Throw เขียนร่วมกับ Method

```
function setAge(int age) throws ArithmeticException , IOException {
        if (age <=20) {
            throw new ArithmeticException("ตัวเลขไม่ถูกต้อง");
         }
}
```



การสร้าง Exception ขึ้นมาใช้เอง

```
if (age <=20) {

    throw new Exception("อายุไม่ถึงเกณฑ์");

 }
```

จัดการไฟล์ (Java I/O)

การอ่านและเขียนไฟล์เบื้องต้น

- FileWriter
- FileReader
- BufferedWriter
- BufferedReader

โปรแกรมตัดเกรด

อ่านไฟล์ที่เก็บรหัสนักเรียนและคะแนนสอบวิชา Java เบื้องต้น (100 คะแนน)
จากนั้นให้คำนวณเกรดที่จะได้รับผ่านคะแนนของนักเรียนแต่ละคนโดยมีเกณฑ์ ดังนี้

- 80 คะแนนขึ้นไป ได้เกรด A
- 70 คะแนนขึ้นไป ได้เกรด B
- 60 คะแนนขึ้นไป ได้เกรด C
- 50 คะแนนขึ้นไป ได้เกรด D
- ต่ำกว่า 50 คะแนน ได้เกรด F

การแสดงผลด้วย Printf

แสดงผลตัวเลขและตัวอักษรด้วย Format String

- %d เลขฐาน 10
- %o เลขฐาน 8
- %X เลขฐาน 16
- %f แสดงทศนิยม
- %c แสดงตัวอักษร
- %t แสดงวันเดือนปี

Math

จัดการค่าทางคณิตศาสตร์ (Math)

- ค่าคงที่ PI , E
- คำนวณค่าสัมบูรณ์
- การปัดเศษตัวเลข
- คำนวณหาค่ารากที่ 2
- คำนวณหาค่าเลขยกกำลัง
- คำนวณค่าต่ำสุด - สูงสุดของชุดตัวเลข