

# CHAPTER 2

## การเขียนโปรแกรมภาษาจาวาขั้นพื้นฐาน : ตอนที่ 2 (Basic Java Programming : Part II)

### 1 อาเรย์หนึ่งมิติ (One Dimensional Arrays)

#### 1. นิยามของอาเรย์

- 1) **อาเรย์ (Arrays)** คือ โครงสร้างข้อมูลประเภทหนึ่งที่ใช้เก็บรายการของข้อมูลประเภทเดียวกัน โดยเก็บข้อมูลทั้งหมดเรียงต่อกันไปในหน่วยความจำ เสมือนนำข้อมูลเก็บไว้ใน "ช่อง" ที่วางเรียงกัน
- 2) **ตำแหน่งของอาเรย์** อาเรย์แต่ละชุดจะมีชื่อของอาเรย์และหมายเลขช่องที่ระบุตำแหน่ง (Index) ของข้อมูลในอาเรย์แต่ละตัว ซึ่งจะเริ่มต้นที่ตำแหน่งที่ 0 (Zero Index) เสมอ (เหมือนกับสตริง)

0	1	2	3
6	1	7	9

ตำแหน่ง (Index) เริ่มที่ 0 แต่ลำดับ (Order) เริ่มที่ 1

- 3) **ขนาดหรือความยาวของอาเรย์** คือ จำนวนสมาชิกหรือจำนวนข้อมูลที่เก็บอยู่ในอาเรย์ชุดนั้น

0	1	2	3
6	1	7	9

อาเรย์ยาว 4 (มีสมาชิก 4 ตัว)

- 4) **มิติของอาเรย์** สามารถมีได้ไม่จำกัด เช่นอาเรย์หนึ่งมิติ สองมิติ สามมิติ หรือ n มิติ แต่โดยทั่วไปแล้วเรานิยมใช้อาเรย์หนึ่งมิติและอาเรย์สองมิติ ซึ่งในบทนี้จะนำเสนออาเรย์หนึ่งมิติ

6	1
---	---

หนึ่งมิติ

2	1
7	0

สองมิติ

9	4
5	3

สามมิติ

#### 2. การประกาศและสร้างอาเรย์หนึ่งมิติโดยใช้ Initialized List

การประกาศและสร้างอาเรย์หนึ่งมิติโดยใช้ Initialized List เป็นการสร้างอาเรย์แบบง่าย ซึ่งจะประกาศใช้ได้ก็ต่อเมื่อทราบค่าทุกค่าที่จะเก็บลงไปในอาเรย์แล้ว โดยมีรูปแบบคำสั่งดังนี้

```
<ประเภทข้อมูล> <ชื่ออาเรย์>[] = { สมาชิก , ..., สมาชิก };
```

หลังชื่ออาเรย์ต้องมีสัญลักษณ์ []

สมาชิกแต่ละตัวคั่นด้วย Comma

เช่น `int a[] = {34, 56, 52, 12, 90, 0, 75, 23, 45, 8};` ซึ่งจะได้เป็นโครงสร้างดังนี้

0	1	2	3	4	5	6	7	8	9
34	56	52	12	90	0	75	23	45	8

มีสมาชิก 10 ตัว แต่ตำแหน่งสูงสุดคือ 9

ค่าสมาชิกแต่ละตัวของอาเรย์สามารถอ้างอิงหรือเรียกชื่อได้โดยใช้คำสั่ง `<ชื่ออาเรย์>[<ตำแหน่ง>]` เช่น

0	1	2	3	4	5	6	7	8	9
a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]	a[9]

แต่ละช่องเทียบเท่ากับตัวแปรแต่ละตัวนั่นเอง

### 3. การประกาศและสร้างอาร์เรย์หนึ่งมิติโดยการ new

การประกาศและการสร้างอาร์เรย์โดยการ new นี้จะเป็นวิธีแบบทั่วไปที่ใช้ในการเขียนโปรแกรมภาษาจาวาซึ่งมีขั้นตอนดังต่อไปนี้

#### 1) การประกาศตัวแปรอาร์เรย์หนึ่งมิติ

```
<ประเภทข้อมูล> <ชื่ออาร์เรย์>[];
```

หลังชื่ออาร์เรย์ต้องมีเครื่องหมาย []

เช่น `int a[];`



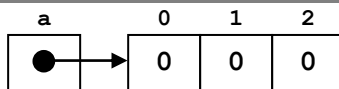
#### 2) การสร้างอาร์เรย์หนึ่งมิติ

```
<ชื่ออาร์เรย์> = new <ประเภทข้อมูล>[<ขนาด>];
```

หลังชื่ออาร์เรย์ไม่ต้องมีเครื่องหมาย []

ระบุขนาดหรือจำนวนช่องของอาร์เรย์ (ไม่ใช่ตำแหน่ง)

เช่น `a = new int[3];`



มีค่าเริ่มต้นของทุกช่องเป็น 0 (ประเภท int)

#### 3) อาร์เรย์หนึ่งมิติที่ได้หลังจากการประกาศและการสร้าง

a[0]	a[1]	a[2]
------	------	------

#### 4) การประกาศและการสร้างอาร์เรย์หนึ่งมิติโดยรวมขั้นตอนที่ 1 และ 2 เข้าด้วยกัน (มี 2 วิธีให้เลือกใช้)

```
<ประเภทข้อมูล> <ชื่ออาร์เรย์>[] = new <ประเภทข้อมูล>[<ขนาด>];
```

```
<ประเภทข้อมูล> [] <ชื่ออาร์เรย์> = new <ประเภทข้อมูล>[<ขนาด>];
```

เครื่องหมาย [] จะอยู่หน้าหรือหลังชื่ออาร์เรย์ก็ได้

เช่น `int a[] = new int[3];`

#### 5) ข้อสังเกตในการประกาศและสร้างอาร์เรย์หนึ่งมิติ

- ลำดับของอาร์เรย์เริ่มนับที่ 1 แต่ตำแหน่งของอาร์เรย์เริ่มนับที่ 0
- ขนาดของอาร์เรย์เป็นค่าของนิพจน์ได้ เช่น `int data[] = new int[x / 2 * y - 10];` เป็นต้น
- ขนาดของอาร์เรย์ต้องระบุเป็นตัวเลขจำนวนเต็มเท่านั้น (ไม่สามารถใช้ตัวเลขจำนวนจริงระบุขนาดของอาร์เรย์ได้)
- ระบบจะตั้งค่าเริ่มต้นของข้อมูลให้กับอาร์เรย์ทุกช่องโดยอัตโนมัติเมื่อเริ่มสร้างอาร์เรย์ตามชนิดของอาร์เรย์  
เช่น `double a[] = new double[5];` จะได้อาร์เรย์ 5 ช่องโดยมีค่าเริ่มต้นของทุกช่องเป็น 0.0

#### 6) การหาขนาดและความยาวของอาร์เรย์หนึ่งมิติ

การหาขนาดของอาร์เรย์จะใช้คำสั่ง `.length` ระบุข้างหลังชื่ออาร์เรย์โดยไม่ต้องใส่วงเล็บ ดังรูปแบบต่อไปนี้

```
<ชื่ออาร์เรย์>.length
```

อย่าจำสับสนกับเมทอด `length()` ของสตริง

```
1 int a[] = new int[500];
2 System.out.println(a.length);
3 String s = "JAVA!";
4 System.out.println(s.length());
5 System.out.println(a.length());
```

หลัง `length` ไม่ใส่วงเล็บ

หลัง `length` ใส่วงเล็บ

Error

#### 4. การประมวลผลกับอาร์เรย์หนึ่งมิติ

##### 1) การอ้างอิงค่าจากอาร์เรย์ มีรูปแบบคำสั่งดังนี้

```
<ประเภทข้อมูล> <ชื่อตัวแปร> = <ชื่ออาร์เรย์>[<ตำแหน่ง>];
```

เช่น `int n = num[0];` (ให้ตัวแปร `n` ประเภทจำนวนเต็มเก็บค่าจากอาร์เรย์ `num` ตำแหน่งที่ 0)

##### 2) การกำหนดค่าลงไปในอาร์เรย์ มีรูปแบบคำสั่งดังนี้

```
<ชื่ออาร์เรย์>[<ตำแหน่ง>] = <ค่าข้อมูล>;
```

เช่น `num[0] = 13;` (ให้อาร์เรย์ `num` ตำแหน่งที่ 0 มีค่าเท่ากับ 13)

##### 3) การประมวลผลกับสมาชิกแต่ละตัวของอาร์เรย์

ไม่ว่าจะเป็นการกำหนดค่า อ้างอิงค่า หรือแสดงผลในอาร์เรย์หนึ่งมิติก็จะต้องใช้คำสั่ง `for` ในการเข้าถึงสมาชิกแต่ละตัวในอาร์เรย์ตั้งแต่ตัวแรก (ตำแหน่งที่ 0) จนถึงตัวสุดท้าย (ตำแหน่งที่ `length - 1`) โดยมีหลักการดังตัวอย่างต่อไปนี้

```
1 int a[] = new int[500];
2 for(int i = 0; i < a.length; i++) {
3     a[i] = 9;
4 }
```

ต้องเขียน `a.length` ไม่ควรเขียนเป็นตัวเลข และต้องเป็นเครื่องหมาย `<` ไม่ใช่ `<=`

`i` ต้องเริ่มที่ 0 ไม่ใช่ 1 (วนตำแหน่งของอาร์เรย์ ไม่ใช่วนลำดับ)

**โจทย์ข้อที่ 1 [ระดับง่าย]** จงเขียนคำสั่งภาษาจาวาเพื่อประกาศและสร้างอาร์เรย์ต่อไปนี้ (12 คะแนน)

- สร้างอาร์เรย์ชื่อ `num` เก็บข้อมูลตัวเลขจำนวนเต็มที่ยาว 15 หลักขึ้นไป จำนวน 200 ค่า (2 คะแนน)
- สร้างอาร์เรย์ชื่อ `dice` เก็บหมายเลขหน้าของลูกเต๋า (2 คะแนน)
- สร้างอาร์เรย์ชื่อ `avgGrade` เก็บผลการเรียนเฉลี่ยของนิสิตจำนวน 451 คน (2 คะแนน)
- สร้างอาร์เรย์ชื่อ `grade` เก็บผลการเรียนรายวิชา 2190101 ของนิสิตจำนวน  $x^2 + 1$  คน (2 คะแนน)
- สร้างอาร์เรย์ชื่อ `x` เก็บค่าความจริง ซึ่งมีตำแหน่งสูงสุดของช่องสมาชิกคือตำแหน่งที่ 10 (2 คะแนน)
- สร้างอาร์เรย์ชื่อ `merge` เก็บข้อมูลที่ได้จากการนำอาร์เรย์ในข้อที่ 1 มาต่อกับอาร์เรย์ในข้อที่ 2 (2 คะแนน)

**โจทย์ข้อที่ 2 [ระดับง่าย]** จงเขียนโปรแกรมภาษาจาวาที่สมบูรณ์เพื่อสร้างอาร์เรย์ `n` แบบ Initialized List โดยเก็บเลขคี่ที่อยู่ในช่วง 1 ถึง 20 พร้อมแสดงค่าของตัวเลขทุกตัวในอาร์เรย์ `n` ขึ้นบนจอภาพที่ละบรรทัด (10 คะแนน)

**โจทย์ข้อที่ 3 [ระดับง่าย]** จงเขียนโปรแกรมภาษาจาวาที่สมบูรณ์เพื่อสร้างอาเรย์ชื่อ *a* โดยจะเก็บตัวเลขจำนวนจริงใด ๆ ที่รับเข้ามาจากแป้นพิมพ์จำนวน 10 ตัว (10 คะแนน)

**โจทย์ข้อที่ 4 [ระดับง่าย]** จงเขียนโปรแกรมภาษาจาวาที่สมบูรณ์เพื่อรับตัวเลขจำนวนเต็มจากแป้นพิมพ์เข้ามาเก็บไว้ในตัวแปร *n* แล้วกำหนดค่าสมาชิกทุกตัวของอาเรย์ *x* (ขนาด 20) ให้มีค่าเท่ากับค่า *n* นั้น (10 คะแนน)

**โจทย์ข้อที่ 5 [ระดับง่าย]** จงเขียนโปรแกรมภาษาจาวาที่สมบูรณ์เพื่อหาผลบวกของสมาชิกทุกตัวในอาเรย์ *x* แล้วแสดงผลลัพธ์ที่ได้ขึ้นบนจอภาพ (10 คะแนน)

```
int x[] = { 7, 9, -1, 4, 12, 9, 3, 2, -7, 2, 1, 9, -15, 24, 11, 13 };
```

**โจทย์ข้อที่ 6 [ระดับปานกลาง]** จงเขียนโปรแกรมภาษาจาวาที่สมบูรณ์เพื่อนับจำนวนเลขคู่และคู่ที่เก็บอยู่ในอาเรย์ *x* ว่ามีอย่างละกี่จำนวนแล้วแสดงผลขึ้นบนจอภาพ (10 คะแนน)

```
int x[] = { 7, 9, -1, 4, 12, 9, 3, 2, -7, 2, 1, 9, -15, 24, 11, 13 };
```

**โจทย์ข้อที่ 7 [ระดับปานกลาง]** จงเขียนโปรแกรมภาษาจาวาที่สมบูรณ์เพื่อสร้างอาเรย์ชื่อ *a* ยาว 80 ช่อง โดยเก็บตัวเลขจำนวนเต็มที่ประกอบไปด้วยค่าสมาชิก 1, 1, 2, 2, ..., 39, 39, 40, 40 ตามลำดับ (10 คะแนน)

**โจทย์ข้อที่ 8 [ระดับปานกลาง]** จงเขียนโปรแกรมภาษาจาวาที่สมบูรณ์เพื่อสร้างอาเรย์ชื่อ *y* โดยถ้าค่าสมาชิกช่องใดในอาเรย์ *x* มากกว่า 10 ให้ค่าสมาชิกของอาเรย์ *y* เป็น true ไม่เช่นนั้นให้ค่าเป็น false (10 คะแนน)

```
int x[] = { 7, 9, -1, 4, 12, 9, 3, 2, -7, 2, 1, 9, -15, 24, 11, 13 };
```

**โจทย์ข้อที่ 9 [ระดับปานกลาง]** จงเขียนโปรแกรมภาษาจาวาที่สมบูรณ์เพื่อขยายขนาดของอาร์เรย์  $x$  ขึ้นเป็นสองเท่าของขนาดเดิม โดยให้สร้างเป็นอาร์เรย์ใหม่ชื่อ  $y$  พร้อมทั้งคัดลอกค่าสมาชิกเก่าในอาร์เรย์  $x$  ทุกตัวมายังอาร์เรย์  $y$  ส่วนช่องสมาชิกที่เหลือของอาร์เรย์  $y$  ให้ปล่อยว่างเอาไว้ (ยังคงมีค่าเป็น 0) (10 คะแนน)

```
int x[] = { 7, 9, -1, 4, 12, 9, 3, 2, -7, 2, 1, 9, -15, 24, 11, 13 };
```

**โจทย์ข้อที่ 10 [ระดับปานกลาง]** จงเขียนโปรแกรมภาษาจาวาที่สมบูรณ์เพื่อสร้างอาร์เรย์  $y$  ที่มีสมาชิกประกอบไปด้วยค่าในอาร์เรย์  $x$  ทั้งหมด และค่าของตัวแปร  $n$  อีกหนึ่งค่าวางเรียงกันตามลำดับตามตัวอย่าง (10 คะแนน)

```
int x[] = { 3, -2, 1, 10, 2, -8, 3, 2, -1, 1, 2, 1, 9, -15 };
int n = 10;
```

จะได้อาร์เรย์  $y[] = \{ 3, -2, \dots, 9, -15, 10 \}$

อาร์เรย์  $x$       ตัวแปร  $n$

**โจทย์ข้อที่ 11 [ระดับปานกลาง]** จงเขียนโปรแกรมภาษาจาวาที่สมบูรณ์เพื่อรับตัวเลขจำนวนเต็มจากแป้นพิมพ์เก็บในตัวแปร  $x$  แล้วนับว่ามีสมาชิกกี่ตัวในอาร์เรย์  $num$  ที่มีค่าเท่ากับ  $x$  พร้อมแสดงผลบนจอภาพ (10 คะแนน)

```
int num[] = { 3, 2, 1, 10, 2, 8, 3, 2, 1, 1, 8, 5, 10, 11, 7, 6, 10 };
```

**โจทย์ข้อที่ 12 [ระดับยาก]** จงเขียนโปรแกรมภาษาจาวาที่สมบูรณ์เพื่อรับค่าตัวเลขจำนวนเต็มเก็บไว้ในตัวแปร  $x$  และ  $y$  แล้วแบ่งช่วง (Sub Array) สมาชิกจากอาร์เรย์  $num$  ตั้งแต่ตำแหน่งที่  $x$  จนถึงตำแหน่งที่  $y$  เก็บไว้ในอาร์เรย์ใหม่ชื่อ  $snum$  โดยสมมติให้  $x$  มีค่าน้อยกว่า  $y$  และมีค่าตำแหน่งที่ไม่เกินความยาวของอาร์เรย์ (10 คะแนน)

```
int num[] = { 3, 2, 1, 10, 2, 8, 3, 2, 1, 1, 8, 5, 10, 11, 7, 6, 10 };
```

**โจทย์ข้อที่ 13 [ระดับยาก]** จงเขียนโปรแกรมภาษาจาวาที่สมบูรณ์เพื่อทำการกลับค่าสมาชิกในอาร์เรย์ (Reverse Array) ที่ชื่อ  $num$  จากหลังมาหน้า แล้วเก็บค่าไว้ในอาร์เรย์ตัวเดิมโดยห้ามประกาศอาร์เรย์ตัวใหม่ (10 คะแนน)

```
int num[] = { 3, 2, 1, 10, 2, 8, 3, 2, 1, 1, 8, 5, 10, 11, 7, 6, 10 };
```

**โจทย์ข้อที่ 14 [ระดับยาก]** จงเขียนโปรแกรมภาษาจาวาที่สมบูรณ์เพื่อทำการผสาน (Merge/Append Array) หรือเชื่อมต่อสมาชิกในอาร์เรย์ a และอาร์เรย์ b เข้าไว้เป็นอาร์เรย์เดียวกัน ซึ่งเก็บไว้ในอาร์เรย์ใหม่ที่ชื่อ ab โดยกำหนดให้อาร์เรย์ b ต่อท้ายอาร์เรย์ a (10 คะแนน)

```
int a[] = { 1, 2, 3, 4, 5, 6, 7, 8, 9 }, b[] = { 1, 4, 6, 8, 10, 12 };
```

**โจทย์ข้อที่ 15 [ระดับยาก]** จงเขียนโปรแกรมภาษาจาวาที่สมบูรณ์เพื่อรับตัวเลขจำนวนเต็มจากแป้นพิมพ์เก็บไว้ในตัวแปร x แล้วทำการแบ่ง (Split) ค่าสมาชิกในอาร์เรย์ num ออกเป็นอาร์เรย์ใหม่ โดยถ้าค่าสมาชิกมีค่ามากกว่าหรือเท่ากับ x ให้เก็บไว้ในอาร์เรย์ upper ไม่เช่นนั้นให้เก็บไว้ในอาร์เรย์ lower (10 คะแนน)

```
int num[] = { 3, 2, 1, 10, 2, 8, 3, 2, 1, 1, 8, 5, 10, 11, 7, 6, 10 };
```

## 2 อาร์เรย์สองมิติ (Two Dimensional Arrays)

### 1. การประกาศและสร้างอาร์เรย์สองมิติโดยใช้ Initialized List

การประกาศและสร้างอาร์เรย์สองมิติโดยใช้ Initialized List เป็นการสร้างอาร์เรย์สองมิติแบบง่าย ซึ่งจะประกาศ ใช้ได้ก็ต่อเมื่อทราบค่าทุกค่าที่จะเก็บลงไปในการอาร์เรย์แล้ว โดยมีรูปแบบคำสั่งดังนี้

```
<ประเภทข้อมูล> <ชื่ออาร์เรย์> [][] = {{ สมาชิก , ... , สมาชิก } , ... };
```

หลังชื่ออาร์เรย์ต้องมี  
สัญลักษณ์ [] []

เช่น `int num[][] = {{1, 0}, {2, 1}, {2, 4}};` สามารถวาดเป็นโครงสร้างตารางได้ดังนี้

1	0
2	1
2	4

อาร์เรย์ num มีขนาด 3 x 2 ซึ่งประกอบไปด้วย

- แถว (Row) หรือ แนวนอน จำนวน 3 แถว
- หลัก (Column) หรือ แนวตั้ง จำนวน 2 หลัก

- ในชุดใหญ่มีชุดย่อย 3 ชุด นั่นคือมี 3 แถว
- ในแต่ละชุดย่อยมีสมาชิก 2 ค่า นั่นคือมี 2 หลัก (นอนก่อนแล้วค่อยตั้ง)

### 2. การประกาศและสร้างอาร์เรย์สองมิติโดยการ new

การประกาศและการสร้างอาร์เรย์สองมิติโดยการ new นี้จะเป็นวิธีแบบทั่วไปที่ใช้ในภาษาจาวา ซึ่งมีขั้นตอนดังนี้

## 1) การประกาศตัวแปรอาร์เรย์สองมิติ

```
<ประเภทข้อมูล> <ชื่ออาร์เรย์> [ ] [ ] ;
```

จำนวนมิติให้นับที่จำนวนคู่ของเครื่องหมาย [ ] เช่น  
ถ้าเป็นอาร์เรย์สองมิติจะต้องมี [ ] จำนวน 2 คู่

```
เช่น int a [ ] [ ] ;
```

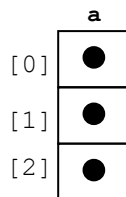


## 2) การสร้างอาร์เรย์สองมิติ

## (1) สร้างแถวของอาร์เรย์

```
<ชื่ออาร์เรย์> = new <ประเภทข้อมูล> [<ขนาดแถว>] [ ] ;
```

```
เช่น a = new int [3] [ ] ;
```



สร้างอาร์เรย์ a ให้มีขนาดเป็น 3 แถว แต่ยังไม่ระบุ  
ขนาดของหลัก (เว้นว่างเอาไว้)

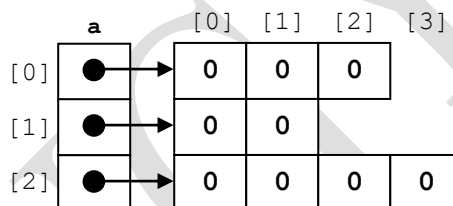
หลังชื่ออาร์เรย์ไม่ต้องมี  
เครื่องหมาย [ ] [ ]

ภายใน [ ] คู่ที่สองไม่ต้องระบุค่า  
ใดๆ (ให้เว้นว่างเอาไว้)

## (2) สร้างหลักหรือคอลัมน์ของอาร์เรย์ (เริ่มสร้างหลักของทุกแถว โดยเริ่มตั้งแต่แถวแรกจนถึงแถวสุดท้าย)

```
<ชื่ออาร์เรย์> [<ตำแหน่งแถว>] = new <ประเภทข้อมูล> [<ขนาดหลัก>] ;
```

```
เช่น a [0] = new int [3] ;  
a [1] = new int [2] ;  
a [2] = new int [4] ;
```



แถวที่ 1 (ตำแหน่ง 0 ) มี 3 หลัก  
แถวที่ 2 ( ตำแหน่ง 1 ) มี 2 หลัก  
แถวที่ 3 ( ตำแหน่ง 2 ) มี 4 หลัก

## 3) อาร์เรย์สองมิติที่ได้หลังจากการประกาศและการสร้างในขั้นตอนที่ 1 และ 2

a[0][0]	a[0][1]	a[0][2]	
a[1][0]	a[1][1]		
a[2][0]	a[2][1]	a[2][2]	a[2][3]

ถ้าประกาศและสร้างอาร์เรย์แบบแยกส่วนจะสามารถสร้างอาร์เรย์สองมิติ  
ที่บิดเบี้ยว โดยที่มีจำนวนช่องของแถวแต่ละแถวไม่เท่ากัน

## 4) การประกาศและการสร้างอาร์เรย์สองมิติโดยรวมขั้นตอนที่ 1 และ 2 เข้าด้วยกัน (มี 2 วิธีให้เลือกใช้)

```
<ประเภทข้อมูล> <ชื่ออาร์เรย์> [ ] [ ] = new <ประเภทข้อมูล> [<ขนาดแถว>] [<ขนาดหลัก>] ;
```

หรือ

```
<ประเภทข้อมูล> [ ] [ ] <ชื่ออาร์เรย์> = new <ประเภทข้อมูล> [<ขนาดแถว>] [<ขนาดหลัก>] ;
```

```
เช่น int a [ ] [ ] = new int [3] [4] ;  
int [ ] [ ] a = new int [3] [4] ;
```

เครื่องหมาย [ ] [ ] จะอยู่หน้าหรือหลังชื่ออาร์เรย์ก็ได้

**หมายเหตุ** ขนาดแถวและขนาดหลักต้องระบุเป็นตัวเลขจำนวนเต็มเท่านั้น (ห้ามใส่จำนวนจริงเด็ดขาด)

## 5) ข้อสังเกตในการประกาศและสร้างอาร์เรย์สองมิติ

อย่าสับสน ตำแหน่งสมาชิกของอาร์เรย์ (สมาชิกตำแหน่งที่) เริ่มต้นที่ 0

ลำดับสมาชิกของอาร์เรย์ (สมาชิกลำดับที่ / สมาชิกตัวที่) เริ่มต้นที่ 1

## 6) อาร์เรย์สองมิติที่ได้หลังจากการประกาศและการสร้างในขั้นตอนที่ 4

a[0][0]	a[0][1]	a[0][2]	a[0][3]
a[1][0]	a[1][1]	a[1][2]	a[1][3]
a[2][0]	a[2][1]	a[2][2]	a[2][3]

ถ้าประกาศและสร้างอาร์เรย์แบบรวมจะได้อาร์เรย์สองมิติที่มีจำนวนช่องของแถวแต่ละแถวเท่ากันทุกแถว โดยไม่มีการบิดเบี้ยว

## 3. การหาขนาดและความยาวของอาร์เรย์สองมิติ

ใช้คำสั่ง `.length` เหมือนกับการหาขนาดของอาร์เรย์หนึ่งมิติ แต่จะต้องระบุตำแหน่งของแถวหรือหลักที่ต้องการ

### 1) การหาขนาดของแถว (ขนาดของมิติที่ 1)

```
<ชื่ออาร์เรย์>.length
```

### 2) การหาขนาดของหลัก (ขนาดของมิติที่ 2)

```
<ชื่ออาร์เรย์>[<ตำแหน่งแถว>].length
```

### 3) ตัวอย่างการหาขนาดของอาร์เรย์สองมิติ

```
1 int[][] a = new int[3][4];
2 System.out.println(a.length);    → 3
3 System.out.println(a[0].length); → 4
4 System.out.println(a[3].length); → Error
```

ความยาวแถวของอาร์เรย์ a

ความยาวหลักของแถว ณ ตำแหน่งที่ 0 ของอาร์เรย์ a

	[0]	[1]	[2]	[3]
[0]	0	0	0	0
[1]	0	0	0	0
[2]	0	0	0	0

## 4. การประมวลผลกับอาร์เรย์สองมิติ

### 1) การอ้างอิงค่าจากอาร์เรย์

```
<ประเภทข้อมูล> <ชื่อตัวแปร> = <ชื่ออาร์เรย์>[<ตำแหน่งแถว>][<ตำแหน่งหลัก>];
```

เช่น `int n = num[0][0];` (ให้จำนวนเต็ม n เก็บค่าจากอาร์เรย์ num แถวตำแหน่งที่ 0 และหลักตำแหน่งที่ 0)

### 2) การกำหนดค่าลงไปให้อาร์เรย์

```
<ชื่ออาร์เรย์>[<ตำแหน่งแถว>][<ตำแหน่งหลัก>] = <ค่าข้อมูล>;
```

เช่น `num[0][0] = 13;` (ให้อาร์เรย์ num แถวตำแหน่งที่ 0 และหลักตำแหน่งที่ 0 มีค่าเท่ากับ 13)

### 3) การประมวลผลกับสมาชิกแต่ละตัวของอาร์เรย์

ต้องใช้คำสั่ง `for` ซ้อน `for` ในการประมวลผล ดังตัวอย่างต่อไปนี้

```
1 int a[][] = new int[5][8];
2 for (int i = 0; i < a.length; i++) {
3     for (int j = 0; j < a[i].length; j++) {
4         a[i][j] = 13;
5     }
6 }
```

คำสั่ง `for i` วนตั้งแต่แถวแรกจนถึงแถวสุดท้าย

คำสั่ง `for j` วนตั้งแต่หลักแรกจนถึงหลักสุดท้ายของแถวตำแหน่งที่ i ใดๆ

เริ่มพิจารณาที่สมาชิกในแถวแต่ละแถวของอาร์เรย์และตามด้วยสมาชิกแต่ละตัว (แต่ละหลัก) ของอาร์เรย์ ณ แถวนั้นๆ



**โจทย์ข้อที่ 16 [ระดับง่าย]** จงเขียนคำสั่งภาษาจาวาเพื่อประกาศและสร้างอาร์เรย์ต่อไปนี้ (18 คะแนน)

- 1) ประกาศและสร้างอาร์เรย์สองมิติชื่อ a ขนาด  $2 \times 3$  โดยให้สมาชิกทุกตัวมีค่าเป็น 0 (2 คะแนน)
- 2) ประกาศและสร้างอาร์เรย์สองมิติชื่อ s ขนาด  $3 \times 1$  โดยให้สมาชิกทุกตัวเป็นคำว่า "Java" (2 คะแนน)
- 3) ประกาศและสร้างอาร์เรย์สองมิติชื่อ t ขนาด  $1 \times 3$  โดยให้สมาชิกทุกตัวเป็นคำว่า "Java" (2 คะแนน)
- 4) ประกาศและสร้างอาร์เรย์ชื่อ intMatrix เป็นเมตริกซ์เก็บตัวเลขจำนวนเต็มขนาด  $6 \times 7$  (2 คะแนน)
- 5) ประกาศและสร้างอาร์เรย์ชื่อ tranMatrix เป็นเมตริกซ์ประเภทจำนวนเต็มที่เกิดจากการทรานสโพสของเมตริกซ์ขนาด  $9 \times 4$  (2 คะแนน)
- 6) ประกาศและสร้างอาร์เรย์ชื่อ mulMatrix เป็นเมตริกซ์ประเภทจำนวนจริงที่เกิดจากผลคูณของเมตริกซ์ขนาด  $8 \times 5$  กับเมตริกซ์ขนาด  $5 \times 3$  (2 คะแนน)
- 7) ประกาศและสร้างอาร์เรย์ชื่อ chess เพื่อสร้างตารางหมากรุกที่ใช้เก็บชื่อของตัวหมากบนตาราง (2 คะแนน)
- 8) ประกาศและสร้างอาร์เรย์ชื่อ data เพื่อเก็บคะแนนรวมของรายวิชา CSC4202 ของนิสิต 300 คน โดยในแต่ละคนประกอบไปด้วยคะแนนการบ้าน คะแนนกลางภาค และคะแนนปลายภาค (2 คะแนน)
- 9) ประกาศและสร้างอาร์เรย์ชื่อ hotel เพื่อเก็บสถานะของห้องพักของโรงแรมที่มี 15 ชั้นโดยแต่ละชั้นมี 12 ห้อง ซึ่งกำหนดให้สถานะของห้องมีเพียงแค่ 2 สถานะคือว่าง และไม่ว่าง (2 คะแนน)

**โจทย์ข้อที่ 17 [ระดับง่าย]** จงเขียนโปรแกรมภาษาจาวาที่สมบูรณ์เพื่อประกาศอาร์เรย์ชนิดจำนวนเต็มขนาด  $10 \times 20$  แล้วกำหนดค่าสมาชิกแต่ละช่องของอาร์เรย์ให้เท่ากับผลคูณของหมายเลขแถวและหลัก (10 คะแนน)

เช่น อาร์เรย์แถวที่ 3 หลักที่ 4  
เก็บค่า 12 ซึ่งเกิดจาก  $3 \times 4$

**โจทย์ข้อที่ 18 [ระดับง่าย]** จงเขียนโปรแกรมภาษาจาวาที่สมบูรณ์เพื่อหาผลบวกของสมาชิกทุกช่องในอาร์เรย์ n พร้อมทั้งแสดงผลลัพธ์ที่ได้ขึ้นบนจอภาพ (10 คะแนน)

```
int n[][] = { { 1, 2, 3, 4, 5 }, { ... }, ...};
```

**โจทย์ข้อที่ 19 [ระดับปานกลาง]** จงเขียนโปรแกรมภาษาจาวาที่สมบูรณ์เพื่อหาค่าสมาชิกที่มากที่สุดในอาร์เรย์  $n$  พร้อมทั้งแสดงผลลัพธ์ที่ได้ขึ้นบนจอภาพ (10 คะแนน)

```
int n[][] = { { 1, 2, 3, 4, 5 }, { ... }, ...};
```

**โจทย์ข้อที่ 20 [ระดับยาก]** จงเขียนโปรแกรมภาษาจาวาที่สมบูรณ์เพื่อยึดอาร์เรย์สองมิติให้เป็นอาร์เรย์หนึ่งมิติ โดยกำหนดอาร์เรย์  $a = \{\{1,2\}, \{3,4\}, \{5,6\}\}$  เมื่อยึดแล้วจะได้อาร์เรย์  $b = \{1,2,3,4,5,6\}$  (10 คะแนน)

**โจทย์ข้อที่ 21 [ระดับยาก]** จงเขียนโปรแกรมภาษาจาวาที่สมบูรณ์เพื่อสร้างอาร์เรย์สามมิติชนิดจำนวนจริงเก็บปริมาณน้ำฝนของประเทศแห่งหนึ่งที่ประกอบไปด้วย 5 ภูมิภาคโดยแต่ละภูมิภาคมี 26 จังหวัดและในแต่ละจังหวัดมี 19 อำเภอ พร้อมรับข้อมูลจากแป้นพิมพ์เข้ามาเก็บในอาร์เรย์ดังกล่าว จากนั้นให้คำนวณหาปริมาณน้ำฝนสูงสุดของแต่ละจังหวัดพร้อมทั้งแสดงผลลัพธ์ขึ้นบนจอภาพ (ห้ามใช้เมทอดจากคลาส *Math*) (20 คะแนน)

**โจทย์ข้อที่ 22 [ระดับง่าย-ระดับยาก]** จงเขียนโปรแกรมภาษาจาวาให้สมบูรณ์เพื่อประมวลผลกับเมตริกซ์ 2 เมตริกซ์ที่กำหนดให้ต่อไปนี้ พร้อมทั้งแสดงผลลัพธ์ที่ได้ออกทางจอภาพให้สวยงาม (40 คะแนน)

$$\text{เมตริกซ์ } a = \begin{bmatrix} 1 & 3 & 5 & 9 \\ 2 & 6 & 4 & 3 \end{bmatrix} \quad \text{เมตริกซ์ } b = \begin{bmatrix} 3 & 6 \\ 4 & 8 \\ 1 & 0 \end{bmatrix} \quad \text{เมตริกซ์ } c = \begin{bmatrix} 3 & 1 & 1 & 2 \\ 10 & 9 & 7 & 4 \end{bmatrix}$$

กำหนดให้สร้างคลาสชื่อ *ApplicationOfMatrice* โดยมีรายละเอียดดังนี้

- 1) ประกาศ สร้าง และกำหนดค่าให้กับเมตริกซ์  $a$ ,  $b$  และ  $c$  ในรูปของ *Initialized List*
- 2) คำนวณหาค่าเมตริกซ์  $4a + c$  แล้วเก็บผลลัพธ์ที่ได้ไว้ในเมตริกซ์  $d$
- 3) คำนวณหาค่าทรานสโพสของเมตริกซ์  $b$  และ  $d$  เก็บผลลัพธ์ไว้ในเมตริกซ์  $bt$  และ  $dt$  ตามลำดับ
- 4) คำนวณหาผลคูณระหว่างเมตริกซ์  $dt$  และ  $bt$  แล้วเก็บผลลัพธ์ที่ได้ไว้ในเมตริกซ์  $m$
- 5) ดึงค่าสมาชิกแถวแรกทั้งแถวของเมตริกซ์  $m$  เข้าไปเก็บไว้ในอาร์เรย์  $x$
- 6) แสดงค่าสมาชิกทุกตัวในเมตริกซ์  $m$  และอาร์เรย์  $x$  ขึ้นบนจอภาพให้สวยงาม

## 3

## เมทอด (Methods)

## 1. นิยามของเมทอด

1) เมทอด (Methods) คือ โปรแกรมย่อยที่ทำหน้าที่เฉพาะเรื่อง ซึ่งมีลักษณะเช่นเดียวกับ Subroutine และ Function โดยเมทอดช่วยลดความซ้ำซ้อนของโปรแกรมที่ทำงานซ้ำๆ ได้ ซึ่งคลาจะรวบรวมเมทอดเอาไว้เป็นกลุ่มก้อน

## 2) การทำงานในภาพรวมของเมทอด

```

1 public class Operator {
2     public static void main(String[] args) {
3         int n = add(4, 10);
4         System.out.println(n);
5         int x = sub(6, 1);
6     }
7     public static int add(int x, int y) {
8         return x + y;
9     }
10    public static int sub(int x, int y) {
11        return add(x, -y);
12    }
13 }

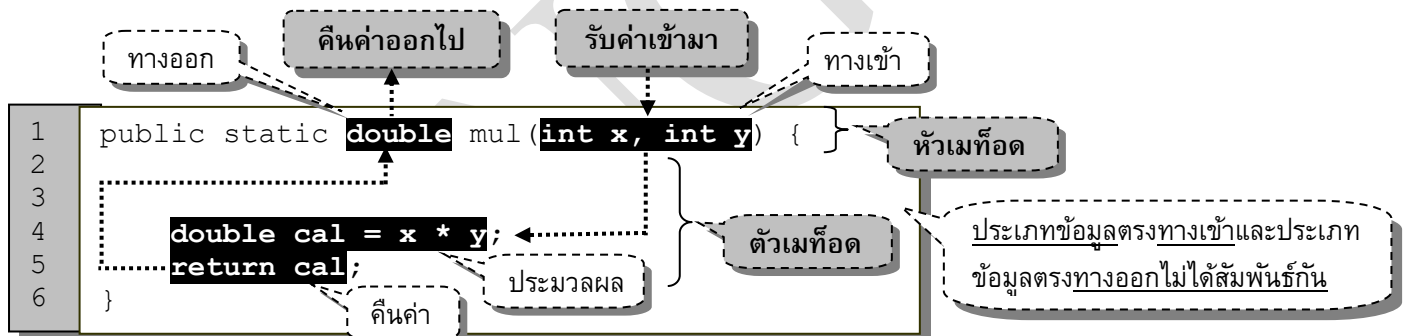
```

การประมวลผลจะเริ่มต้นขึ้นที่  
เมทอด main ก่อนเสมอ

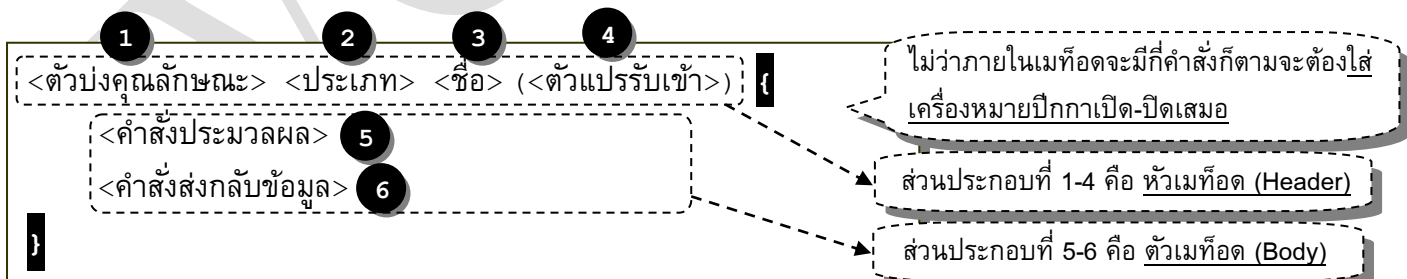
- 1 เรียก add(4, 10)
- 2 ส่งค่า 4 กับ 10
- 3 รับ add(int x, int y)
- 4 ทำงาน 4 + 10
- 5 คืนค่า return 14
- 6 รับ int n = 14
- 7 ใช้งาน System.out... (n)

## 2. โครงสร้างโดยรวมของเมทอด

สามารถแบ่งออกเป็น 2 ส่วนหลักได้แก่ หัวเมทอดและตัวเมทอด ดังตัวอย่าง



จากตัวอย่างสามารถแจกแจงส่วนประกอบย่อยของเมทอดออกเป็น 6 ส่วนดังต่อไปนี้



1) ตัวบ่งคุณลักษณะ หรือมอดิไฟเออร์ (Modifier) คือ ส่วนที่กำหนดคุณลักษณะและคุณสมบัติบางประการของเมทอดซึ่งได้แก่ public, private, protected, static ฯลฯ โดยจะอธิบายอีกครั้งในเรื่องคลาสและอ็อบเจกต์ แต่ในบทนี้ให้เขียนตัวบ่งคุณลักษณะของทุกเมทอดเป็นแบบ public static เท่านั้น ดังตัวอย่าง

```
public static double power(double n)
```

ส่วนนี้มีก็ได้ไม่มีก็ได้ ตัดออกได้

- 2) ประเภทเมทอด (Method Type) หรือประเภทข้อมูลที่ส่งกลับ (Return Type) คือ ประเภทข้อมูลที่เป็นผลลัพธ์สุดท้ายของเมทอดที่จะคืนค่ากลับ ซึ่งได้แก่ประเภทข้อมูล byte, short, int, long, float, double ฯลฯ หากเมทอดไม่มีการคืนค่ากลับให้ระบุเป็นประเภท void ดังตัวอย่าง

```
protected String show(float y, int s)
```

ส่วนนี้ต้องมีเสมอ ตัดออกไม่ได้

ในกรณีที่ประเภทข้อมูลที่ส่งกลับมีโครงสร้างข้อมูลเป็นแบบอาร์เรย์ หรือในกรณีที่คืนค่าข้อมูลมากกว่า 1 ค่าจะต้องคืนอยู่ในรูปของอาร์เรย์ โดยให้ระบุเครื่องหมายวงเล็บเหลี่ยม (Brackets) ต่อท้ายประเภทเมทอดนั้นดังตัวอย่าง

```
private static int[] getArray(int n)
```

ถ้าเป็นอาร์เรย์สองมิติให้ระบุเป็น `int[] []`

- 3) ชื่อเมทอด (Method Name) คือ ชื่อที่อ้างถึงหรือชื่อเรียกเมทอด ซึ่งควรขึ้นต้นด้วยอักษรตัวพิมพ์เล็ก (เป็นไปตามกฎการตั้งชื่อตัวแปร) และชื่อควรเป็นคำกริยาหรือคำที่แสดงถึงการกระทำ เช่น `getID(...)`, `addNumber(...)` เป็นต้น และหลังชื่อเมทอดต้องตามด้วยเครื่องหมายวงเล็บเปิด-ปิดเสมอ ดังตัวอย่าง

```
public static double add(int x, int y)
```

ส่วนนี้ต้องมีเสมอ ตัดออกไม่ได้

- 4) ตัวแปรรับเข้า (Parameters) คือ ค่าหรือตัวแปรที่รับเข้ามายังเมทอดเพื่อใช้ประมวลผลภายในเมทอด ซึ่งตัวแปรรับเข้าของเมทอดหนึ่งๆ จะมีหรือไม่ก็ได้ ถ้ามีจะมีกี่ตัวก็ได้ ถ้ามีหลายตัวแต่ละตัวต้องคั่นด้วยเครื่องหมายจุลภาค (Comma) และทุกตัวต้องมีประเภทข้อมูลกำกับเสมอ เช่น `set(int a, int b)` ห้ามเขียนเป็น `set(int a, b)` โดยเด็ดขาด ดังตัวอย่าง

```
public double mul(int x, double y, int z)
```

ส่วนนี้มีก็ได้ไม่มีก็ได้ ตัดออกได้

ในกรณีที่พารามิเตอร์มีโครงสร้างข้อมูลเป็นแบบอาร์เรย์ ให้ระบุเครื่องหมายวงเล็บเหลี่ยมไว้ข้างหน้าหรือต่อท้ายชื่อพารามิเตอร์นั้นด้วย ดังตัวอย่าง

```
public static int find(int[] x, int y[][])
```

มีพารามิเตอร์ที่เป็นอาร์เรย์สองมิติ

ให้คิดว่าพารามิเตอร์ทุกตัวมีค่าพร้อมใช้งาน

- 5) การประมวลผล ซึ่งได้แก่การประกาศตัวแปรเพื่อใช้ภายในเมทอด และการใช้คำสั่งต่างๆ เช่น if-else, while, for, Scanner ในการประมวลผลภายในเมทอด ซึ่งหลักการเหล่านี้จะเหมือนกับเนื้อหาก่อนหน้านี้ที่ได้เรียนมาแล้วทุกประการ
- 6) การส่งกลับข้อมูล เป็นการส่งค่าหรือคืนค่าคำตอบที่ได้จากการประมวลผลในเมทอดกลับไปยังเมทอดที่เรียกใช้ ซึ่งค่าที่ส่งกลับ (Return Value) จะต้องตรงกับประเภทเมทอด (Method Type) หรือ ประเภทข้อมูลที่ส่งกลับ (Return Type) ที่ประกาศไว้ที่หัวเมทอดเสมอหรือถ้าไม่ตรงกันจะต้องสามารถสอดคล้องกันได้ ซึ่งการส่งกลับจะใช้คำสั่ง `return` และสามารถทำได้เพียงครั้งเดียวเท่านั้น ถ้ามีคำสั่ง `return` หลายกรณีจะทำเพียงกรณีแรกที่เป็นจริง แต่ถ้าประเภทเมทอดเป็น void ไม่ต้องมีคำสั่ง `return` ดังตัวอย่าง

```
1 public static double div(int a, int b){
2     double x = (double) a / (double) b;
3     return x;
4 }
```

เมทอดเป็นประเภท double

ค่าที่จะส่งกลับ (ค่าของตัวแปร x) จะต้องมีประเภทข้อมูลเป็น double

- 7) การส่งกลับข้อมูลแบบหลายค่า ถ้าเมทอดประมวลผลแล้วได้ผลลัพธ์ที่เป็นคำตอบสุดท้ายมากกว่า 1 ค่า จะต้องส่งค่ากลับโดยใช้อาเรย์ (หนึ่งมิติ) โดยการเก็บค่าคำตอบทุกค่าไว้ในอาเรย์แล้วคืนค่าของอาเรย์นั้นเพียงค่าเดียวกลับไปยังเมทอดที่เรียกใช้ ซึ่งจะทำให้ได้ผลคำตอบทุกค่าคืนกลับมาด้วย ดังตัวอย่าง

```

1 public static int[] findFirstLast(int a[]) {
2     int x[] = new int[2];
3     x[0] = a[0];
4     x[1] = a[a.length - 1];
5     return x;
6 }

```

ประเภทข้อมูลที่ส่งกลับที่หัวเมทอดต้องเป็นอาเรย์

การส่งกลับข้อมูลแบบอาเรย์จะเขียนเฉพาะชื่อของอาเรย์โดยไม่ต้องใส่เครื่องหมายวงเล็บเหลี่ยม

### 3. วิธีการเรียกใช้เมทอดอย่างง่าย (การเรียกใช้งานเมทอดภายในคลาสเดียวกัน)

#### 1) วิธีการเรียกใช้เมทอดที่คืนค่า

หัวเมทอด <ตัวบ่งคุณลักษณะ> <ประเภทข้อมูล> <ชื่อเมทอด> (<พารามิเตอร์>) { ... }

การเรียกใช้ <ประเภทข้อมูล> <ตัวแปร> = <ชื่อเมทอด> (<พารามิเตอร์>);

ตัวอย่าง public static double mul(int x, double y) { ... } (หัวเมทอด)  
double n = mul(5, 10.5); (การเรียกใช้เมทอด)

#### 2) วิธีการเรียกใช้เมทอดที่ไม่คืนค่าหรือเป็น void

หัวเมทอด <ตัวบ่งคุณลักษณะ> <void> <ชื่อเมทอด> (<พารามิเตอร์>) { ... }

การเรียกใช้ <ชื่อเมทอด> (<พารามิเตอร์>);

ตัวอย่าง public static void show(String x, int y, double z) { ... } (หัวเมทอด)  
show("PBank Java", 13, 123.4); (การเรียกใช้เมทอด)

### 5. การใช้งานอาเรย์กับเมทอด

- 1) การคืนค่าข้อมูลเป็นอาเรย์ ในกรณีที่ข้อมูลในการคืนค่ากลับเป็นอาเรย์ หรือกรณีที่คำตอบสุดท้ายมีมากกว่า 1 ค่า ก็จะต้องคืนค่ากลับเป็นอาเรย์เช่นกัน โดยให้เขียนเพียงชื่อของอาเรย์อย่างเดียวไม่ต้องใส่เครื่องหมายวงเล็บเหลี่ยมกำกับดังตัวอย่าง (หัวเมทอดต้องมี Return Type เป็นอาเรย์ด้วย)

```

1 public static int[] findFirstLast(int a[]) {
2     int x[] = new int[2];
3     x[0] = a[0];
4     x[1] = a[a.length - 1];
5     return x;
6 }

```

ประเภทข้อมูลที่ส่งกลับที่หัวเมทอดต้องเป็นอาเรย์

การส่งกลับข้อมูลแบบอาเรย์จะเขียนเฉพาะชื่อของอาเรย์โดยไม่ต้องใส่เครื่องหมายวงเล็บเหลี่ยม

- 2) การเรียกใช้เมทอดที่รับพารามิเตอร์เป็นอาเรย์ เมื่อต้องการส่งค่าอาเรย์เข้าไปยังเมทอดผ่านทางพารามิเตอร์นั้นทำได้โดยการใส่ชื่อของอาเรย์ที่ต้องการจะส่งค่าโดยไม่ต้องใส่วงเล็บเหลี่ยม ณ ขณะที่เรียกใช้เมทอด เช่น

```

1 public static int underX(int a[], int x) {
2     //some java code
3 }
4 public static void main(String[] args) {
5     int num[] = { 1, 2, 3, 4 };
6     int count = underX(num, 3);
7     System.out.println(count);
8 }

```

ก่อนการเรียกใช้เมทอด `underX` จะต้องกำหนดอาเรย์ใหม่ขึ้นมาเพื่อใช้ในการส่งค่า

- 3) การเรียกใช้เมทอดที่คืนค่าเป็นอาเรย์ เมื่อต้องการรับค่าอาเรย์ที่คืนค่าออกมาจากเมทอดนั้น ทำได้โดยการประกาศตัวแปรอาเรย์ที่มีประเภทข้อมูลเดียวกับอาเรย์คำตอบโดยไม่ต้องใช้คำสั่ง `new` เช่น

```

1 public static int[] getArray(int x) {
2     //some java code
3 }
4 public static void main(String[] args) {
5     int arr[] = getArray(50);
6     for (int i = 0; i < arr.length; i++) {
7         System.out.println(arr[i]);
8     }
9 }

```

สามารถนำอาเรย์ `arr` มาใช้งานได้เลยโดยไม่ต้อง `new` (มีข้อมูลอยู่ในอาเรย์ `arr` ครบสมบูรณ์ตั้งแต่คืนค่า)

## 6. โอเวอร์โหลดดิงเมทอด (Overloading Methods)

คือเมทอดตั้งแต่ 2 เมทอดใดๆ ขึ้นไปที่มีชื่อเหมือนกันแต่มีรายการของพารามิเตอร์ต่างกัน ดังรายละเอียดต่อไปนี้

- 1) จำนวนพารามิเตอร์ต่างกัน

```

public static int test(int x)
public static int test(int x, int y)

```

เมทอด `test` ตัวที่หนึ่งมีพารามิเตอร์ 1 ตัว  
เมทอด `test` ตัวที่สองมีพารามิเตอร์ 2 ตัว

- 2) ประเภทข้อมูลของพารามิเตอร์ต่างกัน

```

public static int test(int x)
public static int test(double x)

```

เมทอด `test` ตัวที่หนึ่งมีพารามิเตอร์ประเภท `int`  
เมทอด `test` ตัวที่สองมีพารามิเตอร์ประเภท `double`

- 3) ข้อสังเกต เมทอดแต่ละเมทอดจะมีประเภทข้อมูลที่ส่งกลับ (Return Type) ที่ต่างกันหรือเหมือนกันก็ได้

โจทย์ข้อที่ 23 [ระดับง่าย] จงพิจารณาเมทอดต่อไปนี้เขียนผิด (✗) หรือถูก (✓) ตามหลักไวยากรณ์ของภาษาจาวาพร้อมทั้งบอกเหตุผลกำกับ (10 คะแนน)

- 1) ☐ `public static printError(String msg) {  
 System.err.println(msg);  
}`
- 2) ☐ `protected static int flip(int n) {  
 if (n == 1) n = 0;  
 else n = 1;  
}`
- 3) ☐ `public static float max(long x, y) {  
 if (x > y) return x;  
 else return y;  
}`

- 4) ☐

```
private static void showChar() {
    for(char i = '0'; i <= '9'; i++) {
        System.out.println(i);
    }
    return;
}
```
- 5) ☐

```
static String calGrade(int x) {
    if (x >= 80) return "A";
    if (x < 50) return "F";
    if (x < 80 && x >= 50) System.out.println("C");
}
```
- 6) ☐

```
double getLen(double dx, double dy) {
    double dx = Math.abs(dx);
    return Math.sqrt(dx*dx + dy*dy);
}
```
- 7) ☐

```
public private boolean checkLen(int x[], int y[]) {
    return x.length == y.length;
}
```
- 8) ☐

```
public float getLocationPoint() {
    return 0.0;
}
```
- 9) ☐

```
public static int fac(int x) {
    if(x <= 1) return 1;
    else return fac(x - 1) * x;
}
```
- 10) ☐

```
int[] getThreeMember(int[] x) {
    int n[] = { x[0], x[1], x[2] };
    return n[];
}
```

**โจทย์ข้อที่ 24 [ระดับง่าย]** จงเขียนเมทอดอย่างง่ายตามรูปแบบการทำงานที่ระบุไว้ต่อไปนี้ (25 คะแนน)

- เมทอดชื่อ `addRealNumber` มีข้อมูลรับเข้าเป็นตัวเลขจำนวนจริง 3 จำนวน ซึ่งใช้ในการหาผลบวกของตัวเลข 3 จำนวนนั้นแล้วมีการคืนค่ากลับ (5 คะแนน)
- เมทอดชื่อ `printX` มีข้อมูลรับเข้าเป็นตัวเลขจำนวนเต็ม 1 จำนวน เพื่อใช้ในการแสดงผลลัพธ์ออกทางจอภาพภายในเมทอดนั้นโดยไม่คืนค่ากลับ (5 คะแนน)
- เมทอดชื่อ `divideByInt` ใช้ในการหาผลหารระหว่างตัวเลข 2 จำนวน ซึ่งเป็นจำนวนจริงและจำนวนเต็มอย่างละ 1 จำนวน แล้วมีการคืนค่ากลับ (5 คะแนน)
- เมทอดชื่อ `fullName` ใช้ในการรวมชื่อและนามสกุลที่รับเข้ามาให้เป็นชื่อเต็ม (ระหว่างชื่อและนามสกุลต้องมีการเว้นวรรคด้วย) แล้วคืนค่ากลับ (5 คะแนน)
- ฟังก์ชันชื่อ `absolute` ใช้ในการคำนวณหาค่าสัมบูรณ์ (Absolute) ของตัวเลขจำนวนเต็มที่ระบุผ่านทางพารามิเตอร์ เช่น `abs(-21) = 21`, `abs(9) = 9` เป็นต้น (5 คะแนน)

ห้ามใช้เมทอดจากคลาส `Math`

โจทย์ข้อที่ 25 [ระดับง่าย] จงเขียนเมทอด `fac(...)` ที่สมบูรณ์เพื่อใช้ในการคำนวณค่าแฟกทอเรียล (Factorial) ของตัวเลขจำนวนเต็มที่ระบุ เช่น `fac(3) = 6`, `fac(5) = 120` เป็นต้น (10 คะแนน)

โจทย์ข้อที่ 26 [ระดับง่าย] จงเขียนเมทอด `underX(...)` ที่สมบูรณ์เพื่อใช้ระบุว่าสมาชิกกี่จำนวนในอาร์เรย์หนึ่งมิติที่มีค่า a ที่มีค่าน้อยกว่าจำนวนเต็ม x โดยให้รับอาร์เรย์ a และตัวแปร x ผ่านทางพารามิเตอร์ (10 คะแนน)

โจทย์ข้อที่ 27 [ระดับง่าย] จงเขียนเมทอด `count(...)` ที่สมบูรณ์เพื่อรับอาร์เรย์สองมิติชนิดจำนวนเต็ม และจำนวนเต็มอีกหนึ่งค่าทางพารามิเตอร์ แล้วนับจำนวนสมาชิกในอาร์เรย์ที่มีค่าเท่ากับจำนวนเต็มนั้น (10 คะแนน)

โจทย์ข้อที่ 28 [ระดับง่าย] จงเขียนเมทอด `sumOfArray(...)` ที่สมบูรณ์เพื่อรับอาร์เรย์สองมิติชนิดจำนวนจริงเข้ามาหนึ่งตัว แล้วหาผลบวกของสมาชิกทุกช่องในอาร์เรย์นั้นพร้อมทั้งคืนค่ากลับ (10 คะแนน)

โจทย์ข้อที่ 29 [ระดับปานกลาง] จงเขียนเมทอด `fillInArray(...)` ที่สมบูรณ์เพื่อรับอาร์เรย์หนึ่งมิติชนิดจำนวนเต็มหนึ่งชุดทางพารามิเตอร์ แล้วทำการใส่ค่าสมาชิกทุกตัวของอาร์เรย์ด้วยตัวเลขจำนวนเต็มที่ได้จากการร่นรับค่าจากทางแป้นพิมพ์ทีละค่าจนครบสมาชิกทุกตัวของอาร์เรย์ (10 คะแนน)

โจทย์ข้อที่ 30 [ระดับปานกลาง] จงเขียนเมทอดชื่อ `findMaxMember(...)` ที่สมบูรณ์เพื่อรับอาร์เรย์หนึ่งมิติชนิดจำนวนจริง 1 ชุดเข้ามาทางพารามิเตอร์ แล้วนำมาหาค่าสมาชิกที่มากที่สุดภายในอาร์เรย์นั้น (10 คะแนน)

ห้ามใช้เมทอดจากคลาส `Math`



โจทย์ข้อที่ 31 [ระดับปานกลาง] เมื่อกัดชื่อ `reverseString(...)` ที่สมบูรณ์เพื่อรับสตริง 1 ตัวเข้ามาทางพารามิเตอร์ แล้วทำการกลับสตริงนั้นเสียใหม่ โดยเรียงอักขระทุกตัวจากหลังมาหน้า (ขวาไปซ้าย) (10 คะแนน)

โจทย์ข้อที่ 32 [ระดับยาก] จงเขียนเมทอด `isArrayEquals(...)` ที่สมบูรณ์เพื่อรับอาร์เรย์หนึ่งมิติชนิดจำนวนเต็มสองตัวเข้ามาทางพารามิเตอร์ เพื่อใช้ตรวจสอบว่าอาร์เรย์ทั้งสองเท่ากันหรือไม่ โดยอาร์เรย์หนึ่งมิติจะเท่ากันก็ต่อเมื่อความยาวของอาร์เรย์ต้องเท่ากัน และค่าของสมาชิกทุกตัวตำแหน่งต่อตำแหน่งต้องเท่ากัน (10 คะแนน)

โจทย์ข้อที่ 33 [ระดับยาก] จงเขียนเมทอด `appendArray(...)` เพื่อรับพารามิเตอร์สองค่าที่เป็นอาร์เรย์ 1 มิติชนิดจำนวนเต็มทั้งคู่แล้วคืนค่าเป็นอาร์เรย์ชนิดจำนวนเต็มที่เกิดจากการเอาข้อมูลในอาร์เรย์ของพารามิเตอร์ตัวที่สองไปต่อท้ายตัวที่หนึ่ง เช่น `a[] = {1, 2, 3}` และ `b[] = {5, 6}` จะได้ผลลัพธ์ดังนี้ `appendArray(a, b)` คือ `{1, 2, 3, 5, 6}` เป็นต้น (10 คะแนน)

โจทย์ข้อที่ 34 [ระดับยาก] จงเขียนเมทอด `stretchArray(...)` ที่สมบูรณ์เพื่อยืดอาร์เรย์สองมิติชนิดจำนวนเต็มที่ได้รับเข้ามาทางพารามิเตอร์ ให้เป็นอาร์เรย์หนึ่งมิติพร้อมทั้งคืนค่ากลับ โดยวิธีการยืดจะเริ่มต้นที่แถวที่หนึ่งของอาร์เรย์สองมิติเดิมต่อด้วยแถวที่สอง และต่อด้วยแถวถัดไปเรื่อย ๆ จนถึงแถวสุดท้าย (10 คะแนน)

กำหนดให้อาร์เรย์ไม่มีการบิดเบี้ยว

โจทย์ข้อที่ 35 [ระดับปานกลาง-ระดับยาก] จงเขียนโปรแกรมภาษาจาวาที่สมบูรณ์เพื่อออกแบบเมทอด (หนึ่งหรือหลายเมทอด เช่น `isPalindrome(...)`, `removeSpace(...)`, `reverseString(...)` เป็นต้น) ซึ่งใช้ในการตรวจสอบสตริงชุดใดชุดหนึ่งที่ได้รับเข้ามาทางพารามิเตอร์ว่าเป็นพาลินโดรม (Palindrome) หรือไม่ โดยพาลินโดรมคือสตริงที่อ่านจากซ้ายไปขวาเหมือนอ่านจากขวาไปซ้าย เช่น "level", "089 599 5980", "A Toyota", "I prefer pi", "Never odd or even" เป็นต้น ทั้งนี้จะไม่สนใจช่องว่างและตัวอักษรพิมพ์ใหญ่หรือพิมพ์เล็กในการพิจารณาพาลินโดรมแต่อย่างใด โดยโปรแกรมในข้อนี้จะต้องมีเมทอด `main(...)` เพื่อใช้ประมวลผล (20 คะแนน)

โจทย์ข้อที่ 36 [ระดับง่าย-ระดับยาก] จงเขียนเมทอดในคลาส `ArrayUtility` ให้สมบูรณ์เพื่อใช้ประมวลผลกับอาร์เรย์ โดยมีรายละเอียดของเมทอดดังต่อไปนี้ และกำหนดให้เมทอดแต่ละเมทอดสามารถเรียกใช้งานซึ่งกันและกันได้อย่างอิสระ (75 คะแนน)

- 1) สร้างเมทอด `count(a[], k, x)` ใช้หาค่าความถี่ของสมาชิกในอาร์เรย์ `a` ที่มีค่าเท่ากับจำนวนเต็ม `k` โดยเริ่มตั้งแต่ตำแหน่งที่ `x` ของอาร์เรย์เป็นต้นไป แล้วคืนค่าจำนวนสมาชิกที่นับได้ (10 คะแนน)
- 2) สร้างเมทอด `count(a[], k)` ที่ **Overload** กับเมทอด `count(a[], k, x)` ด้านบนโดยจะทำงานเหมือนกันแต่จะเริ่มนับค่าสมาชิกที่ตำแหน่ง 0 (ตำแหน่งแรก) ของอาร์เรย์เป็นต้นไป (5 คะแนน)
- 3) สร้างเมทอด `mode(...)` ใช้หาค่าฐานนิยม โดยจะคืนค่าสมาชิกในอาร์เรย์ที่มีค่าความถี่สูงสุดหรือปรากฏจำนวนครั้งมากที่สุด ถ้ามีหลายค่าให้คืนค่าใดก็ได้เพียงค่าเดียว (เรียกใช้เมทอด `count(...)`) (10 คะแนน)
- 4) เมทอด `majority(...)` ใช้หาค่าหมู่มาก โดยจะคืนค่าสมาชิกในอาร์เรย์ที่มีค่าความถี่มากกว่าครึ่งหนึ่งของจำนวนสมาชิกทั้งหมด ถ้าหาไม่ได้ให้คืนค่า -1 (เรียกใช้เมทอด `count(...)` และ `mode(...)`) (10 คะแนน)
- 5) สร้างเมทอด `maxIndex(a[], x, y)` ใช้หาค่าตำแหน่งของสมาชิกในอาร์เรย์ที่มีค่าสูงสุด ที่อยู่ในช่วงตั้งแต่ตำแหน่งที่ `x` ถึง `y` พร้อมทั้งคืนค่ากลับ (10 คะแนน)
- 6) สร้างเมทอด `maxIndex(a[])` ที่ **Overload** กับเมทอด `maxIndex(a[], x, y)` ด้านบนโดยจะทำงานเหมือนกันแต่จะประมวลผลตั้งแต่ตำแหน่งที่ 0 ถึงตำแหน่งสุดท้ายของอาร์เรย์ (5 คะแนน)
- 7) สร้างเมทอด `swap(a[], x, y)` ใช้ในการสลับค่าสมาชิกในอาร์เรย์ `a` ตำแหน่งที่ `x` และ `y` โดยไม่ต้องคืนค่ากลับ (5 คะแนน)
- 8) สร้างเมทอด `sort(...)` ใช้ในการเรียงลำดับข้อมูลในอาร์เรย์จากน้อยไปหามากโดยใช้อัลกอริทึมการเรียงลำดับแบบเลือก (Selection Sort) และไม่ต้องคืนค่ากลับ (ให้เรียกใช้เมทอด `maxIndex(...)` และ `swap(...)` ในการประมวลผล) (10 คะแนน)

```
public static void sort(int a[]) {
    for (int i = a.length - 1; i >= 1; i--) {
        int m = 0;
        for (int j = 0; j <= i; j++) {
            if (a[j] > a[m]) m = j;
        }
        int temp = a[i];
        a[i] = a[m];
        a[m] = temp;
    }
}
```

ตัวอย่างอัลกอริทึมการ  
เรียงลำดับแบบเลือก  
(Selection Sort)

หมายเหตุ อัลกอริทึมการเรียงลำดับข้อมูลอยู่ในเอกสารบทที่ 10

- 9) สร้างเมทอด `range(...)` ใช้ในการหาค่าพิสัยของอาร์เรย์ (ค่ามากที่สุดลบด้วยค่าน้อยสุด) พร้อมทั้งคืนค่ากลับ (ให้เรียกใช้เมทอด `sort(...)` ในการประมวลผล) (5 คะแนน)
- 10) สร้างเมทอด `main(...)` เพื่อเรียกใช้งานเมทอดที่กำหนดให้พร้อมทั้งแสดงผลลัพธ์ขึ้นบนจอภาพให้สวยงาม โดยสามารถกำหนดค่าพารามิเตอร์ของแต่ละเมทอดได้อย่างอิสระ (5 คะแนน)

```
public static void main(String[] args) {
    int a[] = { 5, 1, 6, 1, 4, 1, 2, 1, 4, 3, 1, 5, 7, 2, 1, 1 };
    int b[] = { 4, 5, 7, 9, 7, 2, 1, 2, 4, 1, 1, 1 };
    int num = 5;
    System.out.println(sort(b));
}
```

ข้อ	คำสั่งเพื่อเรียกใช้งานเมทอดพร้อมแสดงผลลัพธ์ภายในบรรทัดเดียวกัน	เมทอดที่เรียก
1.		<code>count(...)</code>
2.		<code>mode(...)</code>
3.		<code>majority(...)</code>
4.		<code>maxIndex(...)</code>
5.		<code>range(...)</code>

```
} //End of main
```

## 4

## การเรียงลำดับ (Sorting)

1. การเรียงลำดับแบบฟอง (Bubble Sort) (ดูการสาธิตจากตัวอย่างการเรียงลำดับในโจทย์ข้อที่ 1)

```
1 public static void bubbleSort(int data[]) {
2     for (int i = data.length - 1; i >= 1; i--) {
3         for (int j = 0; j < i; j++) {
4             if (data[j] > data[j + 1]) {
5                 int temp = data[j];
6                 data[j] = data[j + 1];
7                 data[j + 1] = temp;
8             }
9         }
10    }
11 }
```

จำ

เรียงจากน้อยไปหามาก

จับคู่และขยับค่าที่สนใจในแต่ละรอบ ซึ่งอาจจะเป็นค่ามากที่สุดหรือค่าน้อยที่สุดในรอบนั้น เพื่อจัดเรียง

- 1) คำสั่ง `for i` ชั้นนอก เป็นการกำหนดช่องหรือเลือกช่องตำแหน่งที่ `i` ที่ต้องการจะเรียงลำดับ ซึ่งโดยทั่วไปนิยมเลือกช่องสุดท้ายก่อน (`i = data.length - 1`) แล้วค่อยขยับเข้ามายังช่องข้างหน้าจนถึงช่องแรก (`i = 1`)
- 2) คำสั่ง `for j` ชั้นใน เป็นการเปรียบเทียบค่าเป็นคู่ๆ (ตำแหน่งที่ `j` กับ `j + 1`) โดยเริ่มตั้งแต่ช่องแรกจนถึงช่องตำแหน่งที่ `i` ที่กำหนดไว้ เพื่อจะขยับค่าหรือค้นค่าสมาชิกที่สูงสุดไปไว้ในช่องตำแหน่งที่ `i` นั้น (เหมือนฟองสบู่ที่ลอยขึ้นไปเรื่อยๆ)

## 2. การเรียงลำดับแบบเลือก (Selection Sort) (ดูการสาธิตจากตัวอย่างการเรียงลำดับในโจทย์ข้อที่ 2)

```

1 public static void selectionSort(int data[]) {
2     for (int i = data.length - 1; i >= 1; i--) {
3         int maxIndex = 0;
4         for (int j = 0; j <= i; j++) {
5             if (data[j] > data[maxIndex]) {
6                 maxIndex = j;
7             }
8         }
9         int temp = data[i];
10        data[i] = data[maxIndex];
11        data[maxIndex] = temp;
12    }
13 }

```



เรียงจากน้อยไปหามาก

เลือกค่าที่สนใจในแต่ละรอบ ซึ่งอาจจะเป็นค่ามากที่สุดหรือค่าน้อยที่สุดในรอบนั้น เพื่อจัดเรียง

- 1) คำสั่ง `for i` ชั้นนอก เป็นการกำหนดช่องหรือเลือกช่องตำแหน่งที่ `i` ที่ต้องการจะเรียงลำดับ ซึ่งโดยทั่วไปนิยมเลือกช่องสุดท้ายก่อน (`i = data.length - 1`) แล้วค่อยขยับเข้ามายังช่องข้างหน้าจนถึงช่องแรกสุด (`i = 0`)
- 2) คำสั่ง `for j` ชั้นใน เป็นการหาค่าสมาชิกที่สูงสุด (หรือต่ำสุด) ตั้งแต่ช่องแรกจนถึงช่องตำแหน่งที่ `i` ที่เลือกไว้เพื่อจะได้สลับค่าสูงสุด (หรือต่ำสุด) ไปไว้ในช่องตำแหน่งที่ `i` นั้น

**โจทย์ข้อที่ 37 [ระดับง่าย]** จากอัลกอริทึมในการเรียงลำดับแบบฟอง (Bubble) และแบบเลือก (Selection) จงเขียนเมทอด `bubbleSort(...)` และ `selectionSort(...)` ที่สมบูรณ์ใหม่อีกครั้งหนึ่ง โดยรับอาร์เรย์หนึ่งมิติชนิดจำนวนจริงเข้ามาทางพารามิเตอร์ แล้วทำการเรียงลำดับข้อมูลในอาร์เรย์นั้นจากมากไปหาน้อย (20 คะแนน)

**โจทย์ข้อที่ 38 [ระดับปานกลาง]** จงเขียนเมทอด `sortByStringLength(...)` ที่สมบูรณ์เพื่อใช้ในการเรียงลำดับสตริงที่รับเข้ามาในรูปของอาร์เรย์ โดยให้เรียงลำดับสตริงที่มีความยาวน้อยที่สุดไปจนถึงสตริงที่มีความยาวมากที่สุดตามลำดับ เช่น สตริง `s[] = { "Computer", "Java", "Engineering", "Chula", "CU" }` หลังจากเรียงลำดับแล้วจะได้ `s[] = { "CU", "Java", "Chula", "Computer", "Engineering" }` (10 คะแนน)


ให้ใช้การเรียงลำดับแบบฟอง

**โจทย์ข้อที่ 39 [ระดับปานกลาง]** จงเขียนเมทอด `sortByDictionary(...)` ที่สมบูรณ์เพื่อใช้ในการเรียงลำดับสตริงที่รับเข้ามาในรูปของอาเรย์ โดยให้เรียงลำดับสตริงตามพจนานุกรม (Dictionary) โดยการเปรียบเทียบรหัสยูนิโค้ด (Unicode) เช่น สตริง `s[] = { "Computer", "Java", "Engineering", "Chula", "CU" }` หลังจากเรียงลำดับแล้วจะได้ `s[] = { "CU", "Chula", "Computer", "Engineering", "Java" }` (10 คะแนน)

### ให้ใช้การเรียงลำดับแบบเลือก

**โจทย์ข้อที่ 40 [ระดับยาก]** จงเขียนเมทอด `sort2DArray(...)` เพื่อเรียงลำดับข้อมูลในอาเรย์สองมิติชนิดจำนวนเต็มจากน้อยไปหามาก โดยให้เลือกใช้อัลกอริทึมที่เหมาะสมในการเรียงลำดับ ซึ่งอาเรย์สองมิติหลังจากเรียงลำดับแล้วจะมีข้อมูลดังตัวอย่างต่อไปนี้ (สมมติว่าอาเรย์ไม่มีการบิดเบี้ยว) (10 คะแนน)

10	4	2	6
8	1	12	9
5	11	3	7



1	2	3	4
5	6	7	8
9	10	11	12

คำแนะนำ ยึดอาเรย์สองมิติให้เป็นอาเรย์หนึ่งมิติแล้วทำการเรียงลำดับ พร้อมใส่กลับไปยังอาเรย์สองมิติเดิม

**โจทย์ข้อที่ 41 [ระดับง่าย-ระดับปานกลาง]** จงเขียนโปรแกรมภาษาจาวาที่สมบูรณ์เพื่อสร้างเมท็อดในการเรียงลำดับ (Sort) อาร์เรย์ชนิดจำนวนเต็มและนำไปประยุกต์ใช้ในการสร้างเมท็อดเพื่อหาค่าพิสัย (Range) กึ่งกลางพิสัย (Mid-range) และมีธฐาน (Median) ของอาร์เรย์ตามรายละเอียดต่อไปนี้ (30 คะแนน)

- 1) สร้างเมท็อด `swap(...)` เพื่อใช้สลับค่าสมาชิกในอาร์เรย์ตำแหน่ง `i` และ `j` ที่ระบุ เช่นคำสั่ง `swap(a, i, j)` คือการสลับค่าสมาชิกในอาร์เรย์ `a` ตำแหน่งที่ `i` และ `j` เป็นต้น โดยเมท็อดไม่ต้องคืนค่ากลับ (5 คะแนน)
- 2) สร้างเมท็อด `sort(...)` เพื่อเรียงลำดับข้อมูลในอาร์เรย์จากน้อยไปหามาก พร้อมทั้งเรียกใช้เมท็อด `swap(...)` ในการสลับข้อมูลระหว่างการเรียงลำดับในแต่ละรอบ (ให้ใช้การเรียงลำดับแบบเลือก) (5 คะแนน)
- 3) สร้างเมท็อด `range(...)` เพื่อหาค่าพิสัย (Range) ของอาร์เรย์ (5 คะแนน)
- 4) สร้างเมท็อด `midRange(...)` เพื่อหาค่ากึ่งกลางพิสัย (Mid-range) ของอาร์เรย์ (5 คะแนน)
- 5) สร้างเมท็อด `median(...)` เพื่อหาค่ามัธยฐาน (Median) ของอาร์เรย์ (5 คะแนน)
- 6) สร้างเมท็อด `main(...)` เพื่อแสดงค่าพิสัย (Range) กึ่งกลางพิสัย (Mid-range) และมัธยฐาน (Median) ขึ้นบนจอภาพให้สวยงาม โดยให้สมมติข้อมูลในอาร์เรย์ตามความเหมาะสม (5 คะแนน)