บทที่ 4
เมธอด (Method)

การเขียนโปรแกรมให้เข้าใจง่าย คือ การแบ่งโปรแกรมทั้งหมดออกเป็นส่วนย่อย ๆ (function) ในแต่ละส่วนย่อยทำงานเฉพาะเรื่องใดเรื่องหนึ่ง method คือ กลุ่มของ code ที่ถูกกำหนดขึ้นเพื่อทำงานอย่างใดอย่างหนึ่ง โดย method เปรียบเหมือนกับ function ของการเขียนโปรแกรมเชิงกระบวนการ

4.1 รูปแบบ Method

[modifier] returnType methodName([parameter]) **{** **body-statements;** **return varValue;** **}**	
[modifier]	ส่วนขยาย method เป็นการกำหนดวิธีการเข้าถึง สัญลักษณ์ [ ] เป็น Option มีหรือไม่ก็ได้ จะกล่าวอีกครั้งในหัวข้อ modifiers
returnType	เป็นชนิดข้อมูลที่ส่งค่ากลับเมื่อ method ทำงานเสร็จ ถ้าไม่มีการส่งค่ากลับกำหนดให้เป็น void
methodName	ชื่อ method
parameter	ตัวแปรที่ใช้ในการรับค่าข้อมูลที่นำเอาไว้ใช้ประมวลผลใน method โดย [ ] เป็น Option มีหรือไม่ก็ได้
body-statement	คำสั่งที่ใช้ทำงานใน method
return	คำสงวนที่จำเป็นต้องมีใช้ในการส่งค่าผลลัพธ์กลับ แต่ในกรณีที่ returnType เป็น void ไม่จำเป็นต้องมีคำสงวน return
varValue	ตัวแปรที่เก็บผลลัพธ์ของ method ที่จะส่งค่ากลับ ชนิดของ varValue จำเป็นต้องเป็นชนิดเดียวกับ returnType

ตัวอย่าง 4.1 TestMethod		Note

1
2
3
4
5
6
7
8
9
10
11
12
	`class TestMethod`
	`{`
	`static double PI = 22/7;`
	`static double volume(double radius){`
	`return 4.0/3.0*PI*radius*radius*radius;`
	`}`
	`public static void main(String[]args){`
	`double vol, radius = 7 ;`
	`vol = volume(radius);`
	`System.out.println("volume is:"+vol);`
	`}`
	`}`
		โปรแกรมเริ่มทำงานที่บรรทัดที่ 8
		บรรทัดที่ 8-11 method main( )
		บรรทัดที่ 9 คำสั่ง double vol, radius = 7;
		บรรทัดที่ 10 เรียก method volume( )
		โดยใช้คำสั่ง vol = volume(radius);
		บรรทัดที่ 3 คำสั่ง static double PI = 22/7;
		บรรทัดที่ 4-6 method volume(radius)
		parameter: radius
		return type: double

1
2
3
4
5
	`---- exec: TestMethod`
	
	`volume is: 1372.0`
	
	`---- operation: complete`	

4.2 อาร์กิวเมนต์ (Argument) และพารามิเตอร์ (Parameter)

อาร์กิวเมนต์ คือ ตัวแปรที่ส่งไปให้ method ในขณะเรียกใช้ method และถ้ามีการส่งอาร์กิวเมนต์มากกว่าหนึ่งค่าให้คั้นด้วยเครื่องหมาย “,” ส่วนพารามิเตอร์ คือ ตัวแปรใน method ที่ทำหน้าที่รับค่าอาร์กิวเมนต์ที่ส่งมาตอนเรียกใช้ method นั้นๆ และถ้ามีการส่งอาร์กิวเมนต์มากกว่าหนึ่งค่าพารามิเตอร์ก็ต้องมีจำนวนเท่ากับจำนวนอาร์กิวเมนต์และให้คั้นด้วยเครื่องหมาย “,” รูปแบบของ อาร์กิวเมนต์และพารามิเตอร์แสดงดังตารางที่ 4.1 ซึ่งจำนวนอาร์กิวเมนต์กับจำนวนพารามิเตอร์จำเป็นต้องมีจำนวนเท่ากัน

ตารางที่ 4.1 ตารางแสดงรูปแบบของอาร์กิวเมนต์และพารามิเตอร์

ลำดับ	รูปแบบ	Argument	Parameter

1	no argument, no parameter	`void main(){`
		`  f();`
		`}`
			`void f(){`
			`  int a;`
			`  a =a+1;`
			`}`

2	One argument, One parameter
	argument คือ a , parameter คือ x
		`void main(){`
		`  int a;`
		`  f(a);`
		`}`
			`void f(int x){`
			`  int y;`
			`  y =x+y;`
			`}`

3	Two arguments, Two parameter
	argument คือ a, b , parameter คือ x, y
		`void main(){`
		`  int a,b;`
		`  x =f(a,b);`
		`}`
			`void f(int x, int y){`
			`  int a`
			`  a =x+y;`
			`}`

4.3 รูปแบบการเรียกใช้งาน method

ตารางที่ 4.2 ตารางแสดงรูปแบบการใช้งาน method

ลำดับ	รูปแบบ	Caller	Receiver	คำอธิบาย

1	no parameter,
	no return
	value
		`void main(){`
		`  f();`
		`}`
			`void f(){`
			`  int a;`
			`  a =a+1;`
			`}`
				method main เรียกใช้งาน method f( ) โดย method f( ) ไม่มีการรับค่าพารามิเตอร์ และไม่ส่งผลลัพธ์กลับ

2	parameter,
	no return
	value
		`void main(){`
		`  f(a);`
		`}`
			`void f(int a){`
			`  int b;`
			`  b =b+a;`
			`}`
				method main เรียกใช้งาน method f(a) โดย method f( ) มีการรับค่าพารามิเตอร์ a แต่ไม่ส่งผลลัพธ์กลับ

3	no parameter,
	return value
		`void main(){`
		`  int x =f();`
		`}`
			`int f(){`
			`  int a`
			`  a =a+1;`
			`  return a;`
			`}`
				method main เรียกใช้งาน method f( ) โดย method f( ) ไม่มีการรับค่าพารามิเตอร์ แต่มีการส่งผลลัพธ์กลับ โดยผ่านตัวแปร a จากนั้นนำผลลัพธ์ที่ได้ไปเก็บที่ตัวแปร x ซึ่งจะเห็นได้ว่าชนิดของตัวแปร x และ a เป็นชนิดเดียวกันคือตัวแปร int

4	parameter,
	return value
		`void main(){`
		`  int x =`
		`f(a);`
		`}`
			`int f(int a){`
			`  a =a+1;`
			`  return a;`
			`}`
				method main เรียกใช้ method f(a) โดย method f() มีการรับค่าพารามิเตอร์ a และมีการส่งผลลัพธ์กลับผ่านตัวแปร a จากนั้นนำผลลัพธ์ที่ได้ไปเก็บที่ตัวแปร x ซึ่งจะเห็นได้ว่าชนิดของตัวแปร x และ a เป็นชนิดเดียวกันคือตัวแปร int

ตัวอย่าง 4.2 TestNoparaNoreturn

```
1  class TestNoparaNoreturn
2  {
3   public static void printName() {
4    System.out.println("My name is Bo");
5    System.out.println("I am 4 years old");
6   }
7   public static void main(String[]args) {
8    printName();
9   }
10 }
```

Note

โปรแกรมเริ่มทำงานที่บรรทัดที่ 7
บรรทัดที่ 7-9 method main( )
บรรทัดที่ 8 เรียก method printName( )
โดยใช้คำสั่ง printName( );
บรรทัดที่ 3-6 method printName( )
parameter: ไม่มี
return type: ไม่มี

```
1  ----exec: TestNoparaNoreturn
2
3  My name is Bo.
4  I am 4 years old.
5
6  ----operation: complete
```

ตัวอย่าง 4.3 TestParaNoreturn

```
1  class TestParaNoreturn
2  {
3   static void printName(String name,int age){
4    System.out.println("My name is"+name);
5    System.out.println("I am"+age+"years");
6   }
7   public static void main(String[]args) {
8    printName("Bo", 4);
9   }
10 }
```

Note

โปรแกรมเริ่มทำงานที่บรรทัดที่ 8
บรรทัดที่ 8-11 method main( )
บรรทัดที่ 10 เรียก method printName( )
โดยใช้คำสั่ง printName("Bo", 4);
บรรทัดที่ 3-6 method printName(name,age)
parameter: name,age
return type: ไม่มี

```
1  ----exec: TestParaNoreturn
2
3  My name is Bo.
4  I am 4 years.
5
6  ----operation: complete
```

ตัวอย่าง 4.4 TestNoparaReturn

```
1  class TestNoparaReturn
2  {
3   static int max(){
4    int num1 =10;
5    int num2 =20;
6    if (num1 > num2)
7     return num1;
8    else
9     return num2;
10  }
11  public static void main(String[]args){
12   int result =max();
13   System.out.print("result is "+result);
14  }
15 }
```

Note

โปรแกรมเริ่มทำงานที่บรรทัดที่ 12

บรรทัดที่ 12-16 method main( )
บรรทัดที่ 14 เรียก method max( )
โดยใช้คำสั่ง int result = max( );
บรรทัดที่ 3-11 method max( )
parameter: ไม่มี
return type: int

```
----exec: TestNoparaReturn

result is 20

----operation: complete
```

ตัวอย่าง 4.5 TestParaReturn1

```
class TestParaReturn1
{
 static int max(int num1,int num2)
 {
  if (num1 > num2)
    return num1;
  else
    return num2;
 }
 public static void main(String[]args)
 {
  int result =max(8,25);
  System.out.print("result is "+result);
 }
}
```

Note

โปรแกรมเริ่มทำงานที่บรรทัดที่ 10

บรรทัดที่ 10-14 method main( )
บรรทัดที่ 12 เรียก method max( )
โดยใช้คำสั่ง int result = max(8,25);
บรรทัดที่ 3-9 method max(num1, num2)
parameter: num1,num2
return type: int

```
----exec: TestParaReturn1

result is 25

----operation: complete
```

ตัวอย่าง 4.6 TestParaReturn2

```
class TestParaReturn2
{
 public static int min(int m, int n){
 if (m < n)return m;
 else return n;
 }
 public static int max(int m, int n){
 if (m > n)return m;
 else return n;
 }
 public static int gcd(int m, int n){
 int min =min(m, n);
 int max =max(m, n);
 if (max %min ==0)return min;
 else return gcd(min, max %min);
 }
 public static void main(String[]args){
 int number1 =25;
 int number2 =20;
 System.out.println(gcd(number1,number2));
 }
}
```

Note

โปรแกรมเริ่มทำงานที่บรรทัดที่ 17
บรรทัดที่ 17-21 method main( )
บรรทัดที่ 18 คำสั่ง int number1 = 25;
บรรทัดที่ 19 คำสั่ง int number2 = 20;
บรรทัดที่ 20 เรียก method gcd( )
โดยใช้คำสั่ง gcd(number1,number2)
บรรทัดที่ 3-6 method min(m, n)
parameter: m, n
return type: int
บรรทัดที่ 7-10 method max(m, n)
parameter: m, n
return type: int
บรรทัดที่ 11-16 method gcd(m, n)
parameter: m, n
return type: int

```
----exec: TestParaReturn2

5
```

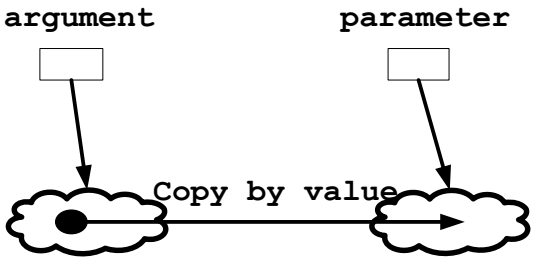
4 5	 ---- operation: complete	

4.4 Parameter passing

การส่งค่าพารามิเตอร์จะเกิดขึ้นเมื่อ method นั้นถูกเรียก โดยผ่านทาง arguments กลไกกลการส่งค่าพารามิเตอร์ (parameter passing mechanism) ที่นิยมใช้อยู่มี 2 แบบคือ pass by value กับ pass by reference รายละเอียดของแต่ละวิธีแสดงดังนี้

1) Pass by value

Pass by value เป็นวิธี argument (อาจเป็นค่าคงที่ ตัวแปร) จะส่งค่า (value) ที่เก็บอยู่ให้กับ parameter โดยการก็อบปี้ value ให้แก่พารามิเตอร์ สามารถแสดงได้ดังรูปที่ 4.1



รูปที่ 4.1 Pass by value

ตัวอย่าง 4.7 TestPassByValue		Note

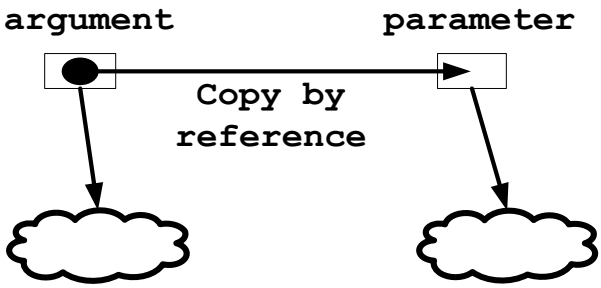
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
	class TestPassByValue
	{
	static void swap(int n1,int n2){
	int temp;
	temp =n1;
	n1 =n2;
	n2 =temp;
	System.out.println("2:n1:"+n1 +"
	n2:"+n2);
	}
	public static void main(String[]args){
	int num1 =5;
	int num2 =10;
	System.out.println("1:num1:"+num1
	+"num2:"+num2);
	swap(num1,num2);
	System.out.println("3:num1:"+num1
	+"num2:"+num2);
	}
	}
		โปรแกรมเริ่มทำงานที่บรรทัดที่ 11
		
		บรรทัดที่ 11-19 method main( )
		บรรทัดที่ 12 คำสั่ง int num1 = 5;
		บรรทัดที่ 13 คำสั่ง int num2 = 10;
		บรรทัดที่ 16 เรียก method swap()
		โดยใช้คำสั่ง swap(num1, num2);
		บรรทัดที่ 3-10 method swap(n1, n2)
		parameter: n1,n2
		return type: ไม่มี

1 2 3 4 5 6	---- exec: TestPassByValue 1:num1:5 num2:10 2:n1:10 n2:5 3:num1:5 num2:10	

7	---- operation: complete	

2) Pass by reference

Pass by reference เป็นวิธีที่ argument (ต้องเป็นเฉพาะตัวแปรเท่านั้นห้ามเป็นค่าคงที่) จะส่งค่า address ของตัวแปรให้กับพารามิเตอร์ โดยการก็อบปี้ address ไปให้แก่พารามิเตอร์ สามารถแสดงได้ ดังรูปที่ 4.2



รูปที่ 4.2 Pass by reference

ตัวอย่าง 4.8 TestPassByRef		Note
1	`class TestPassByRef`	โปรแกรมเริ่มทำงานที่บรรทัดที่ 11
2	`{`	
3	`  static void swap(int[] n){`	บรรทัดที่ 11-18 method main( )
4	`    int temp;`	บรรทัดที่ 12 คำสั่ง int num1 = 5;
5	`    temp =n[0];`	บรรทัดที่ 13 คำสั่ง int num2 = 10;
6	`    n[0]=n[1];`	บรรทัดที่ 16 เรียก method swap( )
7	`    n[1]=temp;`	โดยใช้คำสั่ง swap(num1, num2);
8	`    System.out.println("2:n1:"+n[0]+`	บรรทัดที่ 3-10 method swap(n1, n2)
9	`            "n2:"+n[1]);`	parameter: n1,n2
10	`  }`	return type: ไม่มี
11	`  public static void main(String[] args){`	
12	`    int[] num ={5,10};`	
13	`    System.out.println("1:num1:"+`	
14	`            num[0]+"num2:"+num[1]);`	
15	`    swap(num);`	
16	`    System.out.println("3:num1:"+`	
17	`            num[0]+"num2:"+num[1]);`	
18	`  }`	
19	`}`	

1	---- exec: TestPassByRef	
2		
3	1:num1:5 num2:10	
4	2:n1:10 n2:5	
5	3:num1:10 num2:5	
6		
7	---- operation: complete	

4.5 Scope of Visibility

การประกาศตัวแปรในโปรแกรมอาจซ้ำกันได้ ดังนั้นการกำหนดขอบเขตการมองเห็นตัวแปรด้วย block (block จะเริ่มด้วย { และจบด้วย }) ซึ่งเป็นกลไกที่เป็นประโยชน์อย่างมาก เช่น ตัวแปร x ใน block หนึ่งจะเป็นคนละตัวกับตัวแปร x ของอีก block หนึ่ง โดยขอบเขตการมองของ java เป็นไปตามกฎดังนี้

1) ตัวแปรที่ประกาศอยู่ใน block หนึ่งจะเป็นตัวแปรเฉพาะที่ (local variable) ของ block นั้นๆ คำสั่งที่อยู่นอก block จะไม่เห็นตัวแปรเฉพาะที่ใน block ตัวอย่างเช่น

บรรทัด	ตัวอย่าง	คำอธิบาย

1
2
3
4
5
6
7
8
9
	`x is not visible here`
	`void f()`
	`{`
	`  //x is not visible here`
	`  int x;`
	`  //x is visible here`
	`}`
	`x is not visible here`
		บรรทัดที่ 1 ก่อน method f( ) ไม่สามารถใช้ตัวแปร x ได้ method f( )
		บรรทัดที่ 4 ก่อนประกาศตัวแปร int x ไม่สามารถใช้ได้
		บรรทัดที่ 5 int x;
		บรรทัดที่ 6 หลังประกาศตัวแปร int x สามารถใช้ได้
		บรรทัดที่ 9 หลัง method f( ) ไม่สามารถใช้ตัวแปร x ได้

2) method ที่อยู่ใน class เดียวกันจะมองเห็นกันหมด ไม่ว่า method นั้นจะอยู่ก่อนหรืออยู่หลัง ทำให้ภาษา java ไม่ต้องประกาศ function prototype ตัวอย่างเช่น

ตัวอย่าง	คำอธิบาย

`class A`
`{`
`  void f(){`
`  g(); //g()is visible here`
`  }`
`  void g(){`
`  f(); //f()is visible here`
`  }`
`}`
	method f( ) {
	สามารถเรียกใช้ method g( ) ได้
	}
	method g( ) {
	สามารถเรียกใช้ method f( ) ได้
	}

3) ตัวแปรที่เป็น attribute ของ class จะสามารถมองเห็นได้จาก method โดยสามารถวาง attribute ก่อนหรือหลัง method ก็ได้ ตัวอย่างเช่น

ตัวอย่าง	คำอธิบาย

`class A`
`{`
`  int x;`
`  void f(){`
`  y =12;`
`  x =13;`
`  }`
`  int y;`
`}`
	ประกาศตัวแปร x;
	method f( ) {
	สามารถเรียกใช้ x และ y ได้ ถึงแม้ว่าจะประกาศ ตัวแปร y ที่หลัง
	}
	ประกาศตัวแปร y;

4) หากใน method มีตัวแปรชื่อเหมือนกับตัวแปร attribute ของ class ชื่อตัวแปรที่อยู่ใน method จะถูกเรียกใช้ก่อนหรือบังตัวแปร attribute ของ class

ตัวอย่าง	คำอธิบาย

class A
{
static int x =1;
public static void main(String args[])
{
System.out.println(x);
int x =30;
System.out.println(x);
}
}
	ประกาศตัวแปร x; (attribute ของ class)
	method main( ) {
	พิมพ์ค่า x ได้ ซึ่ง x มีค่า 1 (attribute)
	ประกาศตัวแปร x ใน method main()
	พิมพ์ค่า x ได้ ซึ่ง x มีค่า 30 (x ของ main)
	}

4.6 แบบฝึกหัดท้ายบท

1. จงแสดงผลลัพธ์ของโปรแกรมในแต่ละข้อ

ข้อ	Source code	ผลลัพธ์

1	1 class Test2 {
	2 public static void main(String[]args){
	3 int[]list ={9, 8, 7, 6, 5};
	4 list =dosomething(list);
	5 for (int i =0; i < list.length; i++)
	6 System.out.print(list[i]+"");
	7 }
	8 static int[]dosomething(int[]list){
	9 int[]newList =new int[list.length];
	10 for (int i =0; i < list.length; i++)
	11 newList[i]=list[list.length -1 -i];
	12 return newList;
	13 }
	14 }	

2	1 class Midterm{
	2 public static void main(String[]args){
	3 int k =1;
	4 int[]a={10, 11,12, 13, 14};
	5 f(k,a);
	6 System.out.println(k);
	7 doSomething(a);
	8 }
	9 static void f(int k, int[]b){
	10 if (k>=b.length)return;
	11 for(int i=k;i<b.length;i++){
	12 b[i]=b[b.length-i];
	13 }
	14 k=0;
	15 }
	16 static void doSomething(int[]a){
	17 for (int i =0;i<a.length; i++){
	18
	19	

```
20    System.out.println(a[i]);
21   }
22
    }
   }
```

LAB 3

1. ให้เขียนโปรแกรมรับค่าจำนวนเต็ม 2 จำนวน a และ b โดยโปรแกรมจะ return true ถ้ามีตัวใดตัวหนึ่งมีค่าเป็น 10 หรือผลรวมของ a และ b มีค่าเท่ากับ 10

```
makes10(9, 10) → true
makes10(9, 9) → false
makes10(1, 9) → true
```

2. เขียนโปรแกรมรับค่าจำนวนเต็ม 2 จำนวน โดยโปรแกรมจะ return ค่าผลบวกของจำนวนดังกล่าว ถ้าจำนวนทั้ง 2 ไม่เท่ากัน แต่ถ้าจำนวนทั้ง 2 เท่ากันโปรแกรมจะ return ค่า 2 เท่าของผลบวก แสดงดังตัวอย่าง

```
sumDouble(1, 2) → 3
sumDouble(3, 2) → 5
sumDouble(2, 2) → 8
```

3. เขียนโปรแกรมรับค่าจำนวนจริง 1 จำนวน โดยโปรแกรมจะ return ค่าผลต่างระหว่างจำนวนนั้นกับ 21 แต่ถ้าจำนวนนั้นมีค่ามากกว่า 21 โปรแกรม return ค่า 2 เท่าของผลต่าง แสดงดังตัวอย่างเช่น

```
diff21(19) → 2
diff21(10) → 11
diff21(41) →4 0
```

4. กำหนดอาเรย์มา 1 ชุด โดยโปรแกรมจะ return ค่า true เมื่อผลรวมของค่า 2 ในอาเรย์มีค่าเท่ากับ 8 แสดงดังตัวอย่าง

```
sum28({2, 3, 2, 2, 4, 2}) → true
sum28({2, 3, 2, 2, 4, 2, 2}) → false
sum28({1, 2, 3, 4}) → false
```

5. กำหนดอาร์ยมาให้ 1 ชุด โดยโปรแกรมจะ return ค่า true เมื่อจำนวนเลข 1 มีค่ามากกว่าจำนวนเลข 4 แสดงดังตัวอย่าง

```
more14({1, 4, 1}) → true
more14({1, 4, 1, 4})→false
more14({1, 1}) → true
```

6. เซต A และ B มีขนาดและข้อมูลจากรูปที่ 1 จงเขียนโปรแกรมแสดงว่าเซต A เท่ากับ B

A	18	3	48	2	18	78	9

B	9	18	3	18	48	2	78

7. จงเขียนโปรแกรมที่อ่านค่า a b แบบ int จากนั้นเขียนโปรแกรมคำนวณหาค่า a ยกกำลัง b โดย 0<a<200 , 0<b<200

ตัวอย่างเช่น

Input	Output
3 2	8
199 2	39601
0 -126	Out of range

8. จงเขียนโปรแกรมอ่านค่าตัวเลขที่อยู่ในช่วง 0-1000 ลงในตัวแปร integer จากนั้นหาผลรวมของตัวเลขดังกล่าว

ตัวอย่างเช่น

Input	Output
328	13
400	4
-126	Out of range