บทที่ 5
คลาสและวัตถุ(Class and Object)

การเขียนโปรแกรมเชิงวัตถุจะมองทุกๆ อย่างของปัญหาเป็นวัตถุหรือออบเจ็กต์ (Object) โดย object ประกอบด้วยส่วนสำคัญ 2 ส่วนคือ คุณลักษณะ (attribute หรือ data) และคุณสมบัติ (property หรือ method) โดย attribute จะบอกข้อมูลประจำตัวของ object ส่วน method จะบอกว่า object สามารถทำสิ่งใดได้บ้าง ส่วน class คือ พิมพ์เขียวที่กำหนดไว้ล่วงหน้าก่อนจะมี object (instance) โดยในพิมพ์เขียวจะประกอบไปด้วย attitude และ method ที่เอาไว้กำหนดค่าคุณลักษณะและคุณสมบัติของ object

5.1 Class and Object definition

1) รูปแบบการนิยาม class

```
[modifier] class className
{
    attributes;
    methods( );
}
```

class	คำสงวนและใช้เริ่มต้นโปรแกรมของทุกครั้งและไม่ต้องมี semicolon (;) หลัง className
className	ชื่อ class ที่สร้างขึ้น นิยมใช้ตัวอักษรตัวใหญ่นำหน้า เช่น Employee, Student เป็นต้น
{ }	ขอบเขตการนิยาม class โดย “{” = จุดเริ่มของ class และ “}” = จุดสิ้นสุดของ class
attributes	คุณลักษณะ (attribute) จะบอกข้อมูลประจำตัวของ object
methods( )	method ที่บอกว่า object สามารถทำสิ่งใดได้บ้าง
[modifier]	บ่งบอกลักษณะการเข้าถึงของ class สัญลักษณ์ [ ] เป็น option จะมีหรือไม่มีก็ได้ขึ้นอยู่กับการใช้งาน class

ตัวอย่าง 5.1 Employee1

```
1  class Employee1
2  {
3     int      age;
4     String   name;
5     double   salary;
6     double   getSalary()
7     {
8        return salary;
9     }
10 }
11
12
13
14
```

Note

class Employee1 มี attribute อยู่ 3 ค่า คือ age, name และ salary ส่วน method มีอยู่ 1 วิธี คือ getSalary( ) ซึ่งในการกำหนดค่า attribute หรือ ค่า method ให้แก่ นิยาม class อาจจะมีมากกว่าหรือน้อยกว่าที่กล่าวมาข้างต้นก็ได้ ขึ้นอยู่กับโปรแกรมเมอร์ต้องการให้ object ของ class ที่กำหนดขึ้นสามารถทำอะไรได้บ้างหรืออาจจะกล่าวได้ว่าจำนวน attribute และ method ขึ้นอยู่กับปัญหาที่ต้องการแก้ไขก็ได้ เช่น เราอาจจะเพิ่ม attribute double id; method int getAge( ); ก็ได้

ตัวอย่าง 5.2 Employee2		Note
1	class Employee2	จากตัวอย่าง 5.1 เพิ่ม attribute double id; และ method int getAge( );
2	{	
3	int age;	
4	String name;	
5	double salary;	
6	double id;	
7	double getSalary(){	
8	return salary;	
9	}	
10	int getAge(){	
11	return age;	
12	}	
13	}	

2) รูปแบบการสร้าง Object จาก class

แบบที่ 1 dataType objName; objName = new dataType();		**แบบที่ 2** dataType objName = new objName();
dataType	ชนิดของตัวแปร object ที่เป็นไปตาม นิยาม class	
objName	ชื่อตัวแปร object	
new dataType( );	คำสั่ง create object จาก นิยาม class	

ตัวอย่าง	class Circle{
	double r = 1.0;
	double findArea(){
	return r*r*3.14159;
	}
	}	

	แบบที่ 1 Circle a; a = new Circle( );	แบบที่ 2 Circle a = new Circle( );

ตัวอย่าง 5.3 TestBox		Note
1	class Box	class Box ประกอบด้วย attribute: width, height, depth;
2	{	
3	double width;	
4	double height;	
5	double depth;	
6	}	
1	class TestBox	โปรแกรมเริ่มทำงานที่บรรทัดที่ 3 บรรทัดที่ 3-11 method main( ) บรรทัดที่ 4 สร้าง object ด้วยคำสั่ง Box a = new Box( ); บรรทัดที่ 5 คำสั่ง double vol; บรรทัดที่ 6-8 กำหนดค่าให้กับ attribute ต่างๆ ของ object a
2	{	
3	public static void main(String[] args){	
4	Box a = new Box();	
5	double vol;	
6	a.width = 10;	
7	a.height = 20;	
8	a.depth = 15;	
9	vol = a.width*a.height*a.depth;	
10	System.out.println("vol: " + vol);	
11	}	
12	}	
1	---- exec: TestBox	
2		
3	vol is: 3000.0	
4		
5	---- operation: complete	

ตัวอย่าง 5.4 TetsCounter

```
1  class  Counter
2  {
3     int value;
4     Counter(){
5        value = 0;
6     }
7     void increment(){
8        value++;
9     }
10    int getValue(){
11       return value;
12    }
13 }
```

Note

class Counter ประกอบด้วย
attribute: value;
method: increment( );
getValue( );
constructor: Counter( );

```
1  class TetsCounter
2  {
3   public static void main(String[] args){
4     Counter aCounter = new Counter();
5     System.out.println("Initial counter"
6          + aCounter.value);
7     for(int i = 1; i<=10 ; i++){
8          aCounter.increment();
9     }
10    int aCounterValue=aCounter.getValue();
11    System.out.println("Final counter "
12         + aCounterValue);
13  }
14 }
15
16
17
18
```

โปรแกรมเริ่มทำงานที่บรรทัดที่ 3
บรรทัดที่ 3-13 method main( )
บรรทัดที่ 4 สร้าง object ด้วยคำสั่ง
Counter aCounter = new Counter( );
บรรทัดที่ 7-9 คำสั่ง
for(int i = 1; i<=10 ; i++){
บรรทัดที่ 8 เรียกใช้ method increment( ) ของ object aCounter ด้วยคำสั่ง aCounter.increment( );
}
บรรทัดที่ 10 คำสั่ง
aCounterValue=Counter.getValue( );

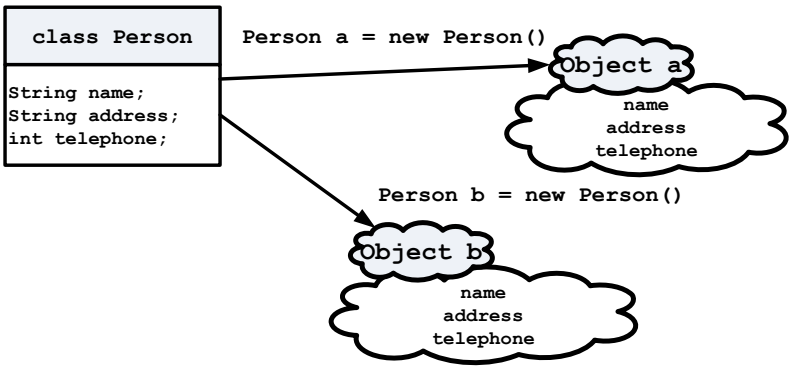
```
1  ---- exec: TetsCounter
2
3  Initial counter 0
4  Final counter is 10
5
6  ---- operation: complete
```

5.2 Attribute or Variable

จากที่กล่าวแล้ว class จะประกอบด้วย attribute และ method ซึ่งเนื้อหาในส่วนนี้จะกล่าวถึง attribute หรืออีกความหมายหนึ่งก็คือ ตัวแปร (variable) โดยจะกล่าวถึง instance variable และ class variable รวมถึงวิธีการกำหนดค่าเริ่มต้นของตัวแปร ภาษา Java กำหนดชนิดของตัวแปรดังต่อไปนี้

1) Instance Variables (Non-Static Fields)

เรารู้ว่าทุกๆ object ที่สร้างมาจาก class จะเป็น ตัวแทน (instance) ที่แสดงถึง class นั้นๆ เพราะฉะนั้น object หรือ instance ที่เกิดจะมีตัวแปรตามมาด้วย โดยแต่ละ instance ก็จะมี variable เป็นของตนเอง ตัวอย่างแสดงดังรูปที่ 5.1



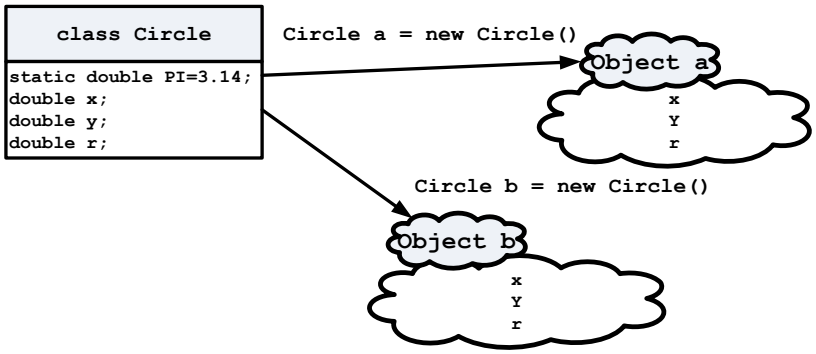
รูปที่ 5.1 Instance Variables a, b ของ class Person

จากรูปที่ 5.1 class Person ประกอบไปด้วย attribute String name, String address, int telephone จากนั้นทำการสร้าง object a, b ด้วยคำสั่ง Person a = new Person( ); และ Person b = new Person( ); จะได้ Object a, b ที่ประกอบด้วยตัวแปร name, address และ telephone โดยแต่ละ instance ก็จะมีตัวแปรเหล่านี้เป็นของตัวเอง เราเลยเรียกตัวแปรลักษณะนี้ ว่า instance variables เวลาใช้ต้องเรียกใช้ผ่าน object ที่สร้างจาก class เช่น object a เรียกใช้ attribute name สามารถเขียนได้ดังนี้ a.name = “Seaoil”;

ตัวอย่าง 5.5 TestBox2		Note
1	class Box2	class Box2 ประกอบด้วย attribute: non-static width, height, depth;
2	{	
3	double width;	
4	double height;	
5	double depth;	
6	}	
1	class TestBox2	โปรแกรมเริ่มทำงานที่บรรทัดที่ 3 บรรทัดที่ 3-16 method main( ) บรรทัดที่ 4-5 สร้าง object ด้วยคำสั่ง Box2 a = new Box2( ); Box2 b = new Box2( ); บรรทัดที่ 6 คำสั่ง double vol1,vol2; บรรทัดที่ 7-12 กำหนดค่าให้กับ attribute ต่างๆ ของ object a,b
2	{	
3	public static void main(String[] args){	
4	Box2 a = new Box2();	
5	Box2 b = new Box2();	
6	double vol1,vol2;	
7	a.width = 10;	
8	a.height = 20;	
9	a.depth = 15;	
10	b.width = 1;	
11	b.height = 2;	
12	b.depth = 3;	
13	vol1 = a.width*a.height*a.depth;	
14	System.out.println("vol1 : " + vol1);	
15	vol2 = b.width*b.height*b.depth;	
16	System.out.println("vol2 : " + vol2);	
17	}	
18	}	
1	---- exec: TestBox2	
2		
3	vol1 : 3000.0	
4	vol2 : 6.0	
5	---- operation: complete	

2) Class Variable (Static Variable)

ตัวแปร class variable นั้นเป็นตัวแปรที่มีค่าอยู่ที่เดียว คืออยู่ที่ class (หรือมีการอ้างอิงที่ตำแหน่งเดียวเท่านั้นในแต่ละ class) ไม่ว่าจะมี instance กี่ตัวก็ตาม หรือกล่าวได้ว่าเป็นตัวแปรที่ถูกใช้ร่วมกันของทุกๆ instance ยกตัวอย่างดังรูปที่ 5.2



รูปที่ 5.2 class Variables ของ class Circle

จากรูป 5.2 class Circle จะเห็นได้ว่า ค่า PI=3.14 นั้นไม่จำเป็นต้องไปอยู่ในทุกๆ instance ที่เกิดขึ้น เพราะไม่ว่าจะอยู่ที่ instance ใดก็ตามค่า PI = 3.14 เสมอ สำหรับภาษา Java ตัวแปร class variable นั้นจะถูกนำหน้าด้วยคำสงวน static ตอนประกาศตัวแปรเพื่อเป็นการบอกตัว compiler ให้รู้ว่าตัวแปรที่นำหน้าด้วย static เป็นตัวแปร class variable เช่น static double PI = 3.14; เวลาใช้สามารถเรียกใช้ผ่าน class ได้เลย ไม่ต้องสร้าง object ก่อนเรียกใช้ เช่น Circle.PI;

ตัวอย่าง 5.6 TestCircle		Note
1	class Circle	class Circle ประกอบด้วย
2	{	attribute: static PI;
3	static double PI = 3.14;	non-static x, y r;
4	double x;	
5	double y;	
6	double r;	
7	}	
1	class TestCircle	โปรแกรมเริ่มทำงานที่บรรทัดที่ 3
2	{	
3	public static void main(String args[]){	
4	Circle a = new Circle();	บรรทัดที่ 3-13 method main( )
5	Circle b = new Circle();	บรรทัดที่ 4-5 สร้าง object ด้วยคำสั่ง
6	double area1,area2;	Circle a = new Circle( );
7	a.r = 10;	Circle b = new Circle( );
8	b.r = 20;	บรรทัดที่ 6 คำสั่ง double area1,area2;
9	area1 = a.r*a.r*Circle.PI;	บรรทัดที่ 7- 8 กำหนดค่าให้กับ
10	System.out.println("area1:" + area1);	attribute ต่างๆ ของ object a,b
11	area2 = b.r*b.r*Circle.PI;	บรรทัดที่ 9 คำสั่งเรียกใช้ Circle.PI;
12	System.out.println("area2:" + area2);	
13	}	
14	}	
1	---- exec: TestCircle	
2		
3	area1: 314.0	
4	area2: 1256.0	
5		
6	---- operation: complete	

ตัวอย่าง 5.7 TestStaticVarBox3

```
1  class Box3
2  {
3     double width;
4     static int numBoxes = 0;
5     public Box3(){
6        width = 5.0;
7        numBoxes++;
8     }
10 }
```

Note

class Box3 ประกอบด้วย
attribute:
static numBoxes;
non-static width;
constructor: Box3( );

```
1  class TestStaticVarBox3
2  {
3    public static void main(String[] args){
4      System.out.println("Num obj= "+ Box3.numBoxes);
5      Box3 box1 = new Box3();
6      System.out.println("Num obj= "+ box1.numBoxes);
7      Box3 box2 = new Box3();
8      System.out.println("Num obj= "+ box2.numBoxes);
9      System.out.println("Num obj= "+ Box3.numBoxes);
10   }
11 }
```

โปรแกรมเริ่มทำงานที่บรรทัดที่ 3
บรรทัดที่ 3-11 method main( )
บรรทัดที่ 5, 7 สร้าง object
Box3 box1 = new Box3( );
Box3 box2 = new Box3( );

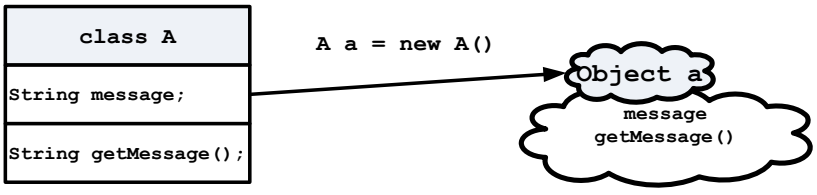
```
1  ---- exec: TestStaticVarBox3
2
3  Num obj = 0
4  Num obj = 1
5  Num obj = 2
6  Num obj = 2
7
8  ---- operation: complete
```

5.3 Instance method และ Class method

ในลักษณะเช่นเดียวกับ attribute การทำงานของ method ก็สามารถแบ่งออกได้เป็น instance method และ class method โดย class method (static method) คือ method ที่ทำงานในระดับ class เราสามารถประมวลผล class method ได้ถึงแม้ว่ายังไม่ได้สร้าง object ขึ้นมา และ class method ต้องมี keyword static อยู่หน้า methodName เสมอ แต่ไม่สามารถเรียกใช้หรือประมวลผล instance method ได้ ถ้าไม่มีการสร้าง object ขึ้นก่อน เนื่องจากไม่มี instance ให้ประมวลผล

1) Instance method

เป็น method ทั่วไปที่สร้างภายในคลาส หรือสามารถมองเหมือนเป็น function ในภาษา C ก็ได้ แต่เวลาเรียกใช้ต้องเรียกใช้ผ่าน object ที่สร้างจาก class เช่น object a สร้างจาก class A และถ้าต้องการเรียกใช้ method getMessage( ) สามารถเขียนได้ดังนี้ a.getMessage( ) รายละเอียดของ class A แสดงดังรูป 5.3



รูปที่ 5.3 การเรียกใช้ instance method ผ่าน object

ตัวอย่าง 5.8 TestInsMethod1 (instance-instance in class)

```
1  class TestInsMethod1
2  {
3    int min(int m, int n) {
4      if (m < n) return m;
5        else return n;
6    }
7    int max(int m, int n) {
8      if (m > n) return m;
9      else return n;
10   }
11   int gcd(int m, int n) {
12     int min = min(m, n);
13     int max = max(m, n);
14     if (max % min == 0) return min;
15     else return gcd(min, max % min);
16   }
17   public static void main(String[] args){
18     int number1 = 25;
19     int number2 = 20;
20     TestInsMethod1 a = new TestInsMethod1();
21     System.out.println("gcd "+number1+" and "
22       + number2+" is: "+ a.gcd(number1, number2));
23   }
24 }
```

Note

การเรียกใช้ instance method ที่อยู่ใน class เดียวกัน
class TestInsMethod1
ประกอบไปด้วย
method: min(m, n);
max(m, n);
gcd(m, n);

โปรแกรมเริ่มทำงานที่บรรทัดที่ 17
บรรทัดที่ 17-23 method main( )
บรรทัดที่ 18-19 คำสั่ง
int number1 = 25;
int number2 = 20;
บรรทัดที่ 20 สร้าง object
TestInsMethod1 a =
new TestInsMethod1( );

```
1  ---- exec: TestInsMethod1
2
3  gcd 25 and 20 is: 5
4
5  ---- operation: complete
```

ตัวอย่าง 5.9 TestInsMethod2 (instance-instance out class)

```
1  class InsMethod
2  {
3    int min(int m, int n) {
4      if (m < n) return m;
5      else return n;
6    }
7    int max(int m, int n) {
8      if (m > n) return m;
9      else return n;
10   }
11   int gcd(int m, int n) {
12     int min = min(m, n);
13     int max = max(m, n);
14     if (max % min == 0) return min;
15     else return gcd(min, max % min);
16   }
17 }
```

Note

การเรียกใช้ instance method ที่ไม่ได้อยู่ใน class เดียวกัน
class InsMethod1
ประกอบไปด้วย
method: min(m, n);
max(m, n);
gcd(m, n);

```
1  class TestInsMethod2
2  {
3    public static void main(String[] args){
4      int number1 = 25, number2 = 20;
5      InsMethod a = new InsMethod();
6      System.out.println("gcd "+number1+" and "
7        +number2+" is: "+ a.gcd(number1, number2));
8    }
9  }
```

โปรแกรมเริ่มทำงานที่บรรทัดที่ 3
บรรทัดที่ 3-8 method main( )
บรรทัดที่ 4 คำสั่ง
int number1 = 25, number2 = 20;
บรรทัดที่ 5 สร้าง object
InsMethod a = new InsMethod();

```
1  ---- exec: TestInsMethod2
2
3  gcd 25 and 20 is: 5
4
5  ---- operation: complete
```

2) Static method

static method เหมือนกับ static variable ที่ถูกกำหนดมาเพื่อ class ไม่ได้เกิดมาเพื่อ object ของ class, method static จะถูกเรียกจาก method ที่อยู่ต่าง class ได้โดยอ้างอิงผ่านทางชื่อ class ซึ่งต่างจาก method non-static จะเรียกใช้ได้ก็ต่อเมื่อต้องมีการสร้าง instance ของ class ขึ้นมาก่อน กรณีที่อยู่ใน class เดียวกัน method ที่เป็น static จะเรียกใช้ได้แต่ method ที่เป็น static ได้เท่านั้น เช่น main( ) ที่เป็น static จะเรียก method ที่อยู่ในคลาสเดียวกันได้แต่เฉพาะที่เป็น static เท่านั้น

ตัวอย่าง 5.10 TestStaMethod1
(static-static in class)

```
1  class TestStaMethod1
2  {
3   static String getBlah(){
4     return "Zzz";
5   }
6   public static void main(String[] args){
7     String str = getBlah();
8     System.out.println(str);
9   }
10 }
```

Note

การเรียกใช้ static method ที่อยู่ใน class เดียวกัน
class TestStaMethod1 ประกอบไปด้วย
method: statistic geyBlah( );
บรรทัดที่ 7 เรียกใช้ method getBlah( ) โดยใช้คำสั่ง String str = getBlah( );

```
1  ---- exec: TestStaMethod1
2
3  Zzz
4
5  ---- operation: complete
```

ถ้าบรรทัดที่ 3 เอา keyword static ออก ผลลัพธ์ของโปรแกรมจะได้อะไร?

ตัวอย่าง 5.11 TestStaMethod2
(static-static out class)

```
1  class StaticMethod
2  {
3     public static String getBlah(){
4        return "ZzZZz";
5     }
6  }
```

Note

การเรียกใช้ static method ที่อยู่ต่าง class กัน
class StaticMethod ประกอบไปด้วย
method: static geyBlah( );

```
1  class TestStaMethod2
2  {
3     public static void main(String[] args){
4        String str = StaticMethod.getBlah();
5        System.out.println(str);
6     }
7  }
```

บรรทัดที่ 4 เรียกใช้ getBlah( ) โดยใช้คำสั่ง str = StaticMethod.getBlah( );
เนื่องจาก getBlah() เป็น static จึงสามารถเรียกผ่านชื่อ class ได้

```
1  ---- exec: TestInsMethod2
2
3  ZzZZz
4
5  ---- operation: complete
```

ตัวอย่าง 5.12 TestStaMethod3
(static-static out class)

```
1  class Circle
2  {
3    static final double PI = 3.14159;
4    static double area(double radius) {
5       return (PI*radius*radius);
6    }
```

Note

การเรียกใช้ static method ที่อยู่ต่าง class กัน
class Circlae ประกอบไปด้วย
attribute: static PI = 3.14159

7
8
9
10
	static double circum(double radius){
	return (PI*(radius+radius));
	}
	}
		method: static area(radius);
		static circum(radius);

1
2
3
4
5
6
7
8
9
10
11
	class TestStaMethod3
	{
	public static void main(String[] args){
	double radius = 20;
	System.out.println("radius "+radius);
	System.out.println("area " +
	Circle.area(radius));
	System.out.println("circum " +
	Circle.circum(radius));
	}
	}
		โปรแกรมเริ่มทำงานที่บรรทัดที่ 3
		บรรทัดที่ 4 คำสั่ง double radius = 20;
		บรรทัดที่ 7 เรียกใช้ Circle.area(radius);
		เนื่องจาก area( ) เป็น static จึง สามารถเรียกผ่านชื่อ class ได้
		บรรทัดที่ 9 เรียกใช้ Circle.circum(radius);
		เนื่องจาก circum( ) เป็น static จึง สามารถเรียกผ่านชื่อ class ได้

1 2 3 4 5 6 7	---- exec: TestInsMethod3 A circle radius 20.0 area 1256.636 circum 125.6636 ---- operation: complete	

จาก Instance variable and method และ Class variable and method สามารถสรุปการนำไปใช้งานได้ดังตารางที่ 5.1

ตารางที่ 5.1 รูปแบบการใช้งาน Instance variable and method และ Class variable and method

รูปแบบ	ใน class เดียวกัน	ข้าม class

Static VS Static	class A
	{
	static int x;
	static void f(){
	x = g();
	}
	static int g(){
	return 1;
	}
	}
		class A
		{
		static int x;
		static int g(){
		return 1;
		}
		}
		class B
		{
		static void f(){
		A.x = A.g();
		}
		}

Static VS non-Static ไม่สามารถใช้งานได้	class A
	{
	int x;
	static void f(){
	x = g();
	}
	int g(){
	return 1;
	}
	}
		class A
		{
		int x;
		int g(){
		return 1;
		}
		}
		
		class B
		{
		A a = new A();
		static void f(){
		a.x = a.g();
		}
		}

Non-Static VS Static	class A
	{
	static int x;
	void f(){
	x = g();
	}
	static int g(){
	return 1;
	}
	}
		class A
		{
		static int x;
		
		static int g(){
		return 1;
		}
		}
		class B
		{
		void f(){
		A.x = A.g();
		}
		}

Non-Static VS non-Static	class A
	{
	int x;
	void f(){
	x = g();
	}
	int g(){
	return 1;
	}
	}
		class A
		{
		int x;
		
		int g(){
		return 1;
		}
		}
		class B
		{
		A a = new A();
		void f(){
		a.x = a.g();
		}
		}

5.4 Constructor

Constructor เป็น method ที่มีชื่อเหมือนกับชื่อ class ซึ่งมีหน้าที่ในการกำหนดค่าเริ่มต้นให้แก่ object เมื่อมีการสร้าง object ใหม่

ถ้า class ไม่ได้กำหนด constructor method แล้ว java compiler จะทำการนำ default constructor มาประมวลผลให้ ตัวอย่างเช่น

ตัวอย่าง 5.11 Constructor		Note
1	class Student {	class Student ประกอบไปด้วย attribute: id; name; constructor: Student(i,n);
2	int id;	
3	String name;	
4	Student(int i, String n){	
5	id = i;	
6	name = n;	
7	}	
8	}	

method constructor มีคุณสมบัติที่ไม่เหมือนกับ method อื่นๆ ดังนี้ คือ

- constructor ไม่มีการส่งค่ากับ และไม่จำเป็นต้องกำหนด returnType void
- constructor มีชื่อเหมือนกับ className

ตัวอย่าง 5.13 TestStudent

```
1  class Student {
2     int id;
3     String name;
4     Student(){
5        id = 0;
6        name = null;
7     }
8     Student(int i, String n){
9        id = i;
10       name = n;
11    }
12    Student(Student s){
13       id = s.id;
14       name = s.name;
15    }
16 }
```

Note

class Student ประกอบไปด้วย
attribute: id;
name;
constructor: Student( );
Student(i, n);
Student(s);

```
1  class TestStudent
2  {
3     public static void main(String[] args) {
4        Student x1 = new Student();
5        Student x2 = new Student(12,"John");
6        Student x3 = new Student(x2);
7        System.out.println("x1: " + x1.name);
8        System.out.println("x2: " + x2.name);
9        System.out.println("x3: " + x3.name);
10    }
11 }
```

โปรแกรมเริ่มทำงานที่บรรทัดที่ 3
บรรทัดที่ 4-6 สร้าง object ด้วยคำสั่ง
Student x1 = new Student( );
Student x2 = new Student(12,"John");
Student x3 = new Student(x2);

```
1  ---- exec: TestStudent
2
3  x1: null
4  x2: John
5  x3: John
6
7  ---- operation: complete
```

ตัวอย่าง 5.14 TestStudent

```
1  class Rect
2  {
3     int width, height;
4     Rect(){
5        width = 2;
6        height = 1;
7     }
8     int getArea(){
9        return width*height;
10    }
11 }
```

Note

class Rect ประกอบไปด้วย
attribute: width;
height;
method: getArea( );
constructor: Rect( );

```
1  class TestRectConstruc
2  {
3    public static void main(String[] args) {
4     Rect r1 = new Rect();
5     System.out.println("area is:" + r1. getArea());
6   }
7  }
```

บรรทัดที่ 4 สร้าง object
Rect r1 = new Rect( );
บรรทัดที่ 5 เรียกใช้ method getArea( )
ของ object r1 ด้วยคำสั่ง
r1. getArea( );

```
1  ---- exec: TestRectConstruc
2
3  area is: 2
4
5  ---- operation: complete
```

5.5 This reference and This Constructor

1) This reference

ในระหว่างการเขียนโปรแกรมเพื่อกำหนด class เราจำเป็นต้องใช้ reference เพื่ออ้างถึงสมาชิกของ instance ตัวหนึ่ง แต่ปัญหา คือในขณะนั้นเรายังไม่ทราบว่าคลาสที่กำลังกำหนดนี้ถูกนำไปสร้าง instance ชื่ออะไรจนกว่าจะสร้างตัวแปรขึ้นมาก ภาษา java จึงได้จัดทำ reference ตัวหนึ่งชื่อ this ให้ใช้อ้างถึง instance ที่สร้างขึ้นจาก class นี้ แต่ this จะถูกใช้ได้จากภายใน class เท่านั้น กล่าวคือ this ที่อ้างถึงใน class ใดก็จะหมายถึง instance ที่สร้างขึ้นจาก class นั้น

ประโยชน์ของ this reference คือ ใช้ในการอ้างถึงชื่อสมาชิกใน class นั้นเพื่อให้แตกต่างจากชื่ออื่นที่เป็นเช่นนี้เนื่องจากสมาชิกของ class อาจมีชื่อเหมือนกับชื่อพารามิเตอร์ของ method ใน class ตัวอย่างเช่น

ตัวอย่าง 5.15 This reference		Note
1	class C	class C ประกอบด้วย attribute: r,i constructor: Complex(r, i) โดย parameter(r,i) ซึ่งไปซ้ำกับ attribute r, i ของ class อาจทำให้สับสนตอนเรียกใช้ r, i ดังนั้นจึงมีการใช้ this ช่วย ดังนี้ - this.r, this.i คือ r, i ที่เป็น attribute ของ class - r, i คือ พารามิเตอร์ที่ใช้ใน method C ( )
2	{	
3	double r, i;	
4	C(double r, double i)	
5	{	
6	this.r = r ;	
7	this.i = i ;	
8	}	
9	}	

ตัวอย่าง 5.16 TestThisRef1		Note
1	class Sphere	class Sphere ประกอบไปด้วย attribute: static PI; r; method: vol();
2	{	
3	static double PI = 3.14;	
4	double r = 7 ;	
5	double vol()	
6	{	
7	return 4.0/3.0*PI*this.r*this.r*this.r;	
8	}	
9	}	
1	class TestThisRef1	บรรทัดที่ 4 คำสั่ง double vol; บรรทัดที่ 5 สร้าง object ด้วยคำสั่ง Sphere x = new Sphere( ); บรรทัดที่ 6 เรียกใช้ method vol( ) ของ object x ด้วยคำสั่ง x.vol( );
2	{	
3	public static void main(String[] args){	
4	double vol;	
5	Sphere x = new Sphere();	
6	vol = x.vol();	
7	System.out.println(vol);	
8	}	
9	}	
1	---- exec: TestThisRef1	ถ้า code line 6 เปลี่ยนเป็น return 4.0/3.0*PI*r*r*r; ผลลัพธ์จะเป็นเช่นไร?
2		
3	1436.0266666666666	
4		
5	---- operation: complete	

ตัวอย่าง 5.17 TestThisRef2

```
1  class Sphere2
2  {
3     static double PI = 3.14;
4     double r = 8;
5     double vol(){
6        return 4.0/3.0*PI*this.r*this.r*this.r;
7     }
8     void changeRadius(double r){
9        r = r;
10       System.out.println("in shape "+r);
11    }
12 }
```

Note

class Sphere2 ประกอบไปด้วย
attribute: static PI;
r;
method: vol( );
changeRadius(r);

```
1  class TestThisRef2
2  {
3     public static void main(String[] args){
4        double vol1;
5        Sphere2 x = new Sphere2();
6        vol1 =  x.vol();
7        System.out.println("1: "+vol1);
8        x.changeRadius(4.0);
9        vol1 =  x.vol();
10       System.out.println("2: "+vol1);
11    }
12 }
```

บรรทัดที่ 4 คำสั่ง double vol1;
บรรทัดที่ 5 สร้าง object
Sphere2 x = new Sphere2( );
บรรทัดที่ 6 เรียกใช้ method vol ของ object x ด้วยคำสั่ง x.vol( );
บรรทัดที่ 8 เรียกใช้ method changeRadius ของ object x โดยส่งค่า 4.0 ไป ด้วยคำสั่ง x.changeRadius(4.0);

```
1  ---- exec: TestThisRef2
2
3  1: 2143.5733333333333
4  in shape 4.0
5  2: 2143.5733333333333
6
7  ---- operation: complete
```

2) This Constructor

ภาษา java ยังมีการใช้ this ในอีกความหมายหนึ่ง คือ การใช้ this เรียก method constructor ของ class นั้น

ตัวอย่าง 5.17 This reference

```
1  class C
2  {
3     double r,i;
4     C(){
5        this(0.0,0.0);
6     }
7     C(double r, bouble i){
8        this.r = r;
9        this.i = i;
10    }
11 }
```

Note

Class C มี constructor 2 ตัว คือ C( ) และ C(r, i) โดย C( ) เรียกใช้ this(0.0, 0.0) คือการเรียก constructor ของ class ตัวเอง ที่มีพารามิเตอร์ที่ตรงกับ argument ที่ส่งออกมา ซึ่งก็คือ constructor C(r, i) ประโยชน์ของกลไกนี้คือ ทำให้ constructor ตัวหนึ่งสามารถใช้โปรแกรมของ constructor อีกตัวหนึ่งได้ ถือเป็นการใช้โปรแกรมร่วมกัน

ภาษา java ยอมให้มีการเรียก constructor ด้วย this( ) จากภายใน constructor เท่านั้น ไม่สามารถเรียกจาก method อื่นที่ไม่ใช่ constructor

ตัวอย่าง 5.18 This reference		Note
1	class TestConstrucThis	class TestConstrucThis ประกอบไปด้วย
2	{	constructor: TestConstrucThis( );
3	TestConstrucThis(){	method: main( );
4	System.out.println("Hi");	
5	}	
6	public static void main(String args[]){	
7	this();	
8	}	
9	}	
1	---- exec: TestConstrucThis	ถ้า code line 7 เปลี่ยนเป็น
2	TtestConstrucThis.java:7: error: call to this must be first statement in constructor	TtestConstrucThis( );
3	this();	ผลลัพธ์จะเป็นเช่นไร?
4	^	
5	1 error	
6		ถ้า code line 7 เปลี่ยนเป็น
7	---- operation: complete	new TtestConstrucThis( );
8		ผลลัพธ์จะเป็นเช่นไร?
9		

5.6 Overloaded Method

Overloaded method คือ mehod ที่มีชื่อเหมือนกันได้แต่ Parameter list แตกต่างกัน เปรียบเสมือนกับ object คน มี method กินข้าว(x) กับ method กินข้าว(x, y) ซึ่งการกินข้าวอาจมีพารามิเตอร์และขั้นตอนในการกินไม่เหมือนกันแต่ใช้ชื่อ method กินข้าว เหมือนกัน เป็นต้น

ตัวอย่าง 5.19 TestMethodOverloading1		Note
1	class TestMethodOverloading1	class TestMethodOverloading1 ประกอบไปด้วย
2	{	method: max(n1,n2);
3	static int max(int n1, int n2) {	max(n1,n2,n3);
4	if (n1 > n2)return n1;	main( );
5	else return n2;	
6	}	
7	static int max(int n1, int n2, int n3) {	
8	if (n1 > n2){	
9	if (n1 > n3) return n1;	
10	else return n3;	
11	}	
12	else {	
13	if (n2 > n3) return n2;	
14	else return n3;	
15	}	
16	}	
17	public static void main(String[] args) {	
18	int n = max(4, 6);	
19	int m = max(7, -3, 3);	
20	System.out.println("n= "+n+", m= "+m);	
21	}	
22	}	
23		

โปรแกรมเริ่มทำงานที่บรรทัดที่ 17
บรรทัดที่ 4 เรียกใช้ method max( )
โดยส่งค่า 4,6 ไป ด้วยคำสั่ง
int n = max(4, 6);
บรรทัดที่ 19 เรียกใช้ method max( ) โดย
ส่งค่า 7,-3,3 ไป ด้วยคำสั่ง
int m = max(7, -3, 3);

Compiler จะทำหน้าที่ในการ Load method ที่มีจำนวน parameter เท่ากันกับจำนวน argument ที่ส่งให้เองอัตโนมัติ

1	---- exec: TestMethodOverloading1	
2		
3	n = 6, m = 7	
4		
5	---- operation: complete	

ตัวอย่าง 5.20 TestMethodOverloading2

```
1  class Rect
2  {
3    int width, height;
4    Rect(){
5      width = 1;
6      height = 1;
7    }
8    Rect(int w){
9      width = w;
10     height = w;
11   }
12   Rect(int w, int h){
13     width = w;
14     height = h;
15   }
16   int getArea(){
17     return width*height;
18   }
19 }
```

Note

class Rect ประกอบไปด้วย

attribute: width;
height;

method: getArea( );

constructor: Rect( );
Rect(w);
Rect(w, h);

```
1  class TestMethodOverloading2
2  {
3   public static void main(String[] args){
4    Rect r1 = new Rect();
5    Rect r2 = new Rect(5);
6    Rect r3 = new Rect(2,3);
7    System.out.println("area r1: "+r1.getArea());
8    System.out.println("area r2: "+r2.getArea());
9    System.out.println("area r3 :  "+r3.getArea());
10  }
11 }
12
```

บรรทัดที่ 4-6 สร้าง object ด้วยคำสั่ง
Rect r1 = new Rect( );
Rect r2 = new Rect(5);
Rect r3 = new Rect(2,3);
Object แต่ละตัวจะเรียกใช้ constructor ต่างกันตามค่า argument ที่ส่งไป
บรรทัดที่ 7-9 เรียกใช้ method getArea( ) ของ object r1,r2,r3

```
1  ---- exec: TestMethodOverloading2
2
3  area of r1 is: 1
4  area of r2 is: 25
5  area of r2 is: 6
6
7  ---- operation: complete
```

5.7 แบบฝึกหัดท้ายบท

1. จงหาข้อผิดพลาดและวิธีการแก้ไขของโปรแกรมต่อไปนี้

ข้อ		Source code	ผลลัพธ์
1	1	class Test5611	
	2	{	
	3	int value=2;	
	4	Test5611(int a){	
	5	this.value = a;	
	6	}	
	7	}	
	8	class ShowErrors5611	
	9	{	
	10	public static void main(String[] args){	
	11	ShowErrors5611 t = new ShowErrors5611(5);	
	12	}	
	13	}	
ข้อ		**Source code**	**ผลลัพธ์**
2	1	class Test5612	
	2	{	
	3	int value=2;	
	4	Test5612(int a, int b)	
	5	{	
	6	this.value = a;	
	7	}	
	8	}	
	9	class ShowErrors5612	
	10	{	
	11	public static void main(String[] args)	
	12	{	
	13	Test5612 t = new Test5612(5);	
	14	}	
	15	}	
3	1	class Test5613	
	2	{	
	3	int value=2;	
	4	Test5613(int a)	
	5	{	
	6	this.value = a;	
	7	}	
	8	}	
	9	class ShowErrors5613	
	10	{	
	11	public static void main(String[] args)	
	12	{	
	13	Test5613 c = new Test5613(5);	
	14	System.out.println("value"+ c.get());	
	15	}	
	16	}	
4	1	class Test5614	
	2	{	
	3	String s;	
	4	Test5614(String newS)	
	5	{	
	6	s= newS;	
	7	}	
	8	public void print(){	
	9	System.out.print(s);	

	10	} }	
	11	class ShowErrors	
	12	{	
	13	public static void main(String[] args)	
	14	{	
	15	Test5614 a = new Test5614(5.0);	
	16	a.print();	
	17	}	
	18	}	
	19		

2. จงหาข้อผิดพลาดและวิธีการแก้ไขของโปรแกรมต่อไปนี้

ข้อ		Source code	ผลลัพธ์
1	1	class T	
	2	{	
	3	static void g()	
	4	{	
	5	}	
	6	void f()	
	7	{	
	8	}	
	9	}	
	10	class Test5621	
	11	{	
	12	public static void main(String[] args)	
	13	{	
	14	T.g();	
	15	T.f();	
	16	}	
	17	}	
2	1	class A	
	2	{	
	3	double PI = 3.145;	
	4	}	
	5	class Test5622	
	6	{	
	7	public static void main(String[] args)	
	8	{	
	9	System.out.println(A.PI);	
	10	}	
	11	}	
3	1	class Test5623	
	2	{	
	3	public static void main(String[] args)	
	4	{	
	5	int larger = max(3,4);	
	6	System.out.println("Max is "+ larger);	
	7	}	
	8	int max(int num1,int num2){	
	9	if(num1>num2)	
	10	return num1;	
	11	else	
	12	return num2;	
	13	}	
	14	}	
4	1	class Test5624	
	2	{	
	3	int x;	
	4	void setX(int y){	

	5	x = y;	
	6	}	
	7	}	
5	1	class Test5625	
	2	{	
	3	int x;	
	4	static void setX(int y){	
	5	x = y;	
	6	}	
	7	}	
ข้อ		**Source code**	**ผลลัพธ์**
6	1	class T	
	2	{	
	3	static void g()	
	4	{ }	
	5	void f()	
	6	{ }	
	7	}	
	8	class Test5621	
	9	{	
	10	public static void main(String[] args){	
	11	T.g();	
	12	T.f();	
	13	}	
	14	}	
7	1	class Test5626	
	2	{	
	3	int x = f();	
	4	static int f() {	
	5	return 1;	
	6	}	
	7	}	
8	1	class Test5627	
	2	{	
	3	public static void main(String[] args){	
	4	int[] a = {77,44,99,66,33,55,88,22 };	
	5	print(a);	
	6	sort(a);	
	7	print(a);	
	8	}	
	9	void swap(int[] a, int i, int j){	
	10	if (i == j)	
	11	return;	
	12	int temp=a[j];	
	13	a[j] = a[i];	
	14	a[i] = temp;	
	15	}	
	16	static void print(int[] a){	
	17	for (int i=0; i<a.length; i++)	
	18	System.out.print(a[i]+" ");	
	19	System.out.println();	
	20	}	
	21	static void sort(int[] a){	
	22	for (int i=a.length-1; i>0; i--)	
	23	for (int j=1; j<=i; j++)	
	24	if (a[j-1]<a[j])	
	25	swap(a,j-1,j);	
	26	}	
	27	}	

9	1	class Test5628	
	2	{	
	3	static int x = f();	
	4	int f(){	
	5	return g(x);	
	6	}	
	7	int g(int a){	
	8	return x;	
	9	}	
	10	}	

3. จงแสดง output จากโปรแกรมต่อไปนี้พร้อมอธิบายการทำงานในแต่ละบรรทัด

ข้อ		Source code	ผลลัพธ์
1	1	class Test5631	
	2	{	
	3	static int i=0;	
	4	static int j=0;	
	5	public static void main(String[] agrs){	
	6	int i=2;	
	7	int k=2;	
	8	{	
	9	int j=3;	
	10	System.out.println("i+j is "+(i+j));	
	11	}	
	12	k=i+j;	
	13	System.out.println("k is " +k);	
	14	System.out.println("j is " +j);	
	15	}	
	16	}	
2	1	class Test5632	
	2	{	
	3	int i = 5;	
	4	static int k =2;	
	5	public static void main(String arq[])	
	6	{	
	7	Test5632 a = new Test5632();	
	8	int j = a.i;	
	9	a.f1();	
	10	}	
	11		
	12	void f1()	
	13	{	
	14	i = i + k + f2(i,k);	
	15	System.out.println(i);	
	16	}	
	17	int f2(int i, int j)	
	18	{	
	19	return (int)(Math.pow(i,j));	
	20	}	
	21	}	

โจทย์เพิ่มทักษะการเขียนโปรแกรม
จงเขียนโปรแกรมเพื่อหาผลลัพธ์ของปัญหาข้อ 4 – 12

4. Given an array of ints length 3, "rotate left" the elements, so {1, 2, 3} becomes {2, 3, 1}. Return the changed array.

rotateLeft3({1, 2, 3}) ⟶ {2, 3, 1}
rotateLeft3({5, 11, 9}) ⟶ {11, 9, 5}
rotateLeft3({7, 0, 0}) ⟶ {0, 0, 7}

5. Given an array of ints length 3, return a new array with the elements in reverse order, so {1, 2, 3} becomes {3, 2, 1}.

reverse3({1, 2, 3}) ⟶ {3, 2, 1}
reverse3({5, 11, 9}) ⟶ {9, 11, 5}
reverse3({7, 0, 0}) ⟶ {0, 0, 7}

6. Given an array of ints length 3, figure out which is larger between the first and last elements in the array, and set all the other elements to be that value. Return the changed array.

maxEnd3({1, 2, 3}) ⟶ {3, 3, 3}
maxEnd3({11, 5, 9}) ⟶ {11, 11, 11}
maxEnd3({2, 11, 3}) ⟶ {3, 3, 3}

7. Given an array of ints, return the sum of the first 2 elements in the array. If the array length is less than 2, just sum up the elements that exist, returning 0 if the array is length 0.

sum2({1, 2, 3}) ⟶ 3
sum2({1, 1}) ⟶ 2
sum2({1, 1, 1, 1}) ⟶ 2

8. Given 2 int arrays, a and b, each length 3, return a new array length 2 containing their middle elements.

middleWay({1, 2, 3}, {4, 5, 6}) ⟶ {2, 5}
middleWay({7, 7, 7}, {3, 8, 0}) ⟶ {7, 8}
middleWay({5, 2, 9}, {1, 4, 5}) ⟶ {2, 4}

9. Given an array of ints, return a new array length 2 containing the first and last elements from the original array. The original array will be length 1 or more.

makeEnds({1, 2, 3}) ⟶ {1, 3}
makeEnds({1, 2, 3, 4}) ⟶ {1, 4}
makeEnds({7, 4, 6, 2}) ⟶ {7, 2}

10. Given an int array length 2, return true if it contains a 2 or a 3.

has23({2, 5}) ⟶ true

has23({4, 3}) ⟶ true

has23({4, 5}) ⟶ false

11. จงเขียนโปรแกรมเพื่อแสดงรายละเอียดของคลาส Account ที่ประกอบด้วยสมาชิกต่อไปนี้

- ตัวแปรชนิดข้อมูล int ชื่อ id สำหรับเก็บหมายเลขบัญชี
- ตัวแปรชนิดข้อมูล double ชื่อ balance สำหรับเก็บยอดเงินคงเหลือ
- ตัวแปรชนิดข้อมูล double ชื่อ annualInterestRate สำหรับเก็บอัตราดอกเบี้ย
- ตัวแปรชนิดข้อมูล Date ชื่อ dateCreated สำหรับเก็บวันที่ที่บัญชีถูกสร้าง
- constructor ที่ไม่มี argument สำหรับการสร้างบัญชีแบบ default
- constructor ที่มี argument สำหรับการสร้างบัญชีแบบระบุเลขที่บัญชี และยอดเงินเริ่มต้น
- method(get) และ method(set) สำหรับตัวแปร id, balance, annualInterestRate
- method(get) สำหรับ ตัวแปร dateCreated
- เมธอด getMonthlyInterestRate() ที่คืนอัตราดอกเบี้ยรายเดือน
- เมธอด getMonthlyInterest() ที่คืนดอกเบี้ยรายเดือน
- เมธอด withdraw() ที่ถอนเงินตามจำนวนที่ระบุ
- เมธอด deposit() ที่ฝากเงินตามจำนวนที่ระบุ

วาดคลาสไดอะแกรมและเขียนส่วนของ Client สำหรับเรียกใช้คลาส Account โดยสร้างอ๊อบเจ็คของบัญชีเลขที่ (ID) 1122 ยอดเงินเปิดบัญชีคือ 20000 และอัตราดอกเบี้ยคือ 4.5 % หลังจากนั้นให้ใช้ withdraw method สำหรับถอนเงิน 2500 บาท และใช้ deposit method สำหรับฝากเงิน 3000 บาท และโปรแกรมสามารถแสดงยอดเงินคงเหลือและอัตราดอกเบี้ยรายเดือนได้

12. เกมส์ Angry Birds เป็นเกมส์ที่ได้รับความนิยมมากในปัจจุบัน ในเกมส์ Angry Birds จะประกอบด้วยนกหลากหลายชนิด และประกอบไปด้วยหลายฉากแต่ละฉากจะมีลักษณะแตกต่างกัน จงออกแบบคลาส Birds และ คลาส Background โดยแต่ละคลาสประกอบด้วยสมาชิกต่อไปนี้

คลาส Birds ประกอบด้วย

- ตัวแปรชนิดข้อมูล int ชื่อ id สำหรับเก็บหมายเลขนก
- ตัวแปรชนิดข้อมูล String ชื่อ name สำหรับเก็บชื่อของนก
- ตัวแปรชนิดข้อมูล String ชื่อ color สำหรับเก็บสีของนก
- ตัวแปรชนิดข้อมูล int ชื่อ birdSize สำหรับเก็บขนาดของนก
- ตัวแปรชนิดข้อมูล String ชื่อ type สำหรับเก็บประเภทของนก
- ตัวแปรชนิดข้อมูล int ชื่อ power สำหรับเก็บพลังของนก
- constructor ที่ไม่มี argument สำหรับการสร้างนกแบบ default
- constructor ที่มี argument สำหรับการสร้างนกแบบระบุ id name color birdSize type

- method(get), method(set) สำหรับ id name color birdSize type
- เมธอด bomb() ที่แสดงข้อความ “Bomb!”
- เมธอด showBirdDetails() ที่แสดงรายละเอียดของนกผ่านทางหน้าจอ

คลาส Background ประกอบด้วย

-ตัวแปร private ชนิดข้อมูล int ชื่อ id สำหรับเก็บหมายเลขฉาก
-ตัวแปร private ชนิดข้อมูล String ชื่อ name สำหรับเก็บชื่อฉาก
-ตัวแปร private ชนิดข้อมูล int ชื่อ numBuilding สำหรับเก็บจำนวนของตึก
-constructor ที่ไม่มี argument สำหรับการสร้างฉากแบบ default
-constructor ที่มี argument สำหรับการสร้างฉากแบบระบุ id name numBuilding
-accessor method และ mutator method สำหรับ ตัวแปร id name numBuilding
-เมธอด paintBg() ที่แสดงข้อความ “Now painting Background”
-เมธอด showBgDetails() ที่แสดงข้อความรายละเอียดของฉากผ่านทางหน้าจอ

จากนั้นวาดคลาสไดอะแกรมและเขียนส่วนของ Client สำหรับเรียกใช้คลาส Birds และ Background โดยสร้างออบเจ็คของนก และฉาก