บทที่ 3

คลาสและวัตถุ

Class & Object

สอนโดย อ.เรวดี พิพัฒน์สูงเนิน สาขาวิชาวิทยาการคอมพิวเตอร์และเทคโนโลยีสารสนเทศ มหาวิทยาลัยราชภัฏอุดรธานี

หลักการเขียนโปรแกรมเชิงวัตถุ

วัตถุประสงค์การเรียนรู้

- อธิบายความหมายของ Class และ Object ได้
- ออกแบบและสร้าง Class ได้ด้วยภาษา Java
- ใช้งาน Object จาก Class ได้อย่างถูกต้อง
- เขียนโปรแกรม Java โดยใช้หลักการ Class และ Object

ความหมายของ Class

- **Class** คือแม่แบบ (Blueprint) สำหรับสร้าง Object
- กำหนดว่า Object หนึ่ง ๆ จะมีอะไรบ้าง
- ภายใน Class ประกอบด้วย
  - Attributes (คุณสมบัติ / fields)
  - Methods (พฤติกรรม / Behaviors)
- ตัวอย่างเช่น Class Student
  - อาจมี Attribute studentId, name, gpa
  - อาจมี Method registerCourse(), calculateGpa()

ความหมายของ Object

- Object คือสิ่งที่สร้างจาก Class
- เป็นหน่วยที่ใช้จริงในโปรแกรม
- มีค่าข้อมูลเป็นของตัวเอง
- มีพฤติกรรมที่สามารถสั่งงานได้

เปรียบเทียบ Class และ Object

Class	Object
แม่แบบ	ตัวอย่างจริง
ยังไม่ใช้หน่วยความจำ	ใช้หน่วยความจำแล้ว
ประกอบด้วย method และ attribute	มีค่าของ attribute และสามารถเรียกใช้ method ได้

ตัวอย่าง Class และ Object

- Class : Car
- Attributes: brand, color
- Methods: start(), stop()
- Object : car1
- brand = “Toyota”
- color = “red”

ตัวอย่าง Class และ Object

- Class : Student
- Attributes: studentId, name, gpa
- Methods: regisCourse(), calculateGpa()
- Object : s1
- studentId = “67040233101”
- name = “Jiraporn”
- gpa = 3.25

ตัวอย่าง Class และ Object

- Class : Student
- Attributes: studentId, name, gpa
- Methods: regisCourse(), calculateGpa()
- Object : s1
- studentId = “67040233101”
- name = “Jiraporn”
- gpa = 3.25

การประกาศ Class ใน Java

```
class ClassName {
    // Attributes (Fields)
    type attributeName;

    // Methods
    returnType methodName() {
        // code
    }
}
```

- การประกาศคลาสใน Java ใช้คีย์เวิร์ด class ตามด้วยชื่อคลาส และบล็อกของโคด {} ที่บรรจุสมาชิกของคลาส
- **ชื่อคลาส** ควรเป็นคำนาม, ขึ้นต้นด้วยตัวพิมพ์ใหญ่, และสื่อถึงความหมายของวัตถุ
- หมายเหตุ: modifier อาจเป็น public, private, หรือ protected แต่โดยทั่วไปแล้ว
  - หากเป็นการประกาศ class มักจะใช้ public
  - หากเป็นการประกาศ attribute มักจะใช้ private
  - หากเป็นการประกาศ method มักจะใช้ public
- การใช้ private และ protected กับคลาส จะได้อธิบายในภายหลัง

การประกาศ Attribute

- รูปแบบการประกาศในภาษา Java

```
[access modifier] [data type] [attribute name];
```

- ตัวอย่างการประกา ศ

```java
//Dog.java
public class Dog {
    String name;
    String breed;
    int age;
}
```

```java
//Student.java
public class Student {
    private String studentId;
    private String name;
    private double gpa;
}
```

การประกาศ Method

- รูปแบบการประกาศในภาษา Java

```
[access modifier] [return type] [method name](parameter list) {
    // คำสั่งภายในเมธอด
}
```

- ตัวอย่าง

```
//Student.java
public class Student {
    private String studentId;
    private String name;
    private double gpa;

    void sayHello() {
        System.out.println("Hello...My name is "+ name);
    }
}
```

ตัวอย่างการประกาศคลาส Dog

```java
//Dog.java
public class Dog {
    String name;
    String breed;
    int age;

    public void bark() {
        System.out.println("Bok Bok!!!");
    }

    public void eat() {
        System.out.println(name + " is eating.");
    }
}
```

การสร้าง Object ใน Java

- **Object (วัตถุ) คืออะไร?**
- Object (วัตถุ) คือ "ตัวตน" ที่ถูกสร้างขึ้นมาจาก Class (An instance of a class)
- เมื่อ Object ถูกสร้างขึ้น มันจะถูกเก็บไว้ในหน่วยความจำของคอมพิวเตอร์
- Object แต่ละตัวจะมีสถานะ (ค่าของ Attributes) ที่แตกต่างกันได้ แม้จะมาจากคลาสเดียวกัน
- เปรียบเทียบ Object คือบ้านแต่ละหลังที่ถูกสร้างขึ้นตามพิมพ์เขียวของบ้าน

การสร้าง Object ใน Java

- รูปแบบ

```
ClassName  objectName = new  Constructor();
```

- ตัวอย่าง

```
Student std1 = new Student();

Dog myDog = new Dog();
```

การเข้าถึงสมาชิกของ Object

- เราสามารถเข้าถึง Attributes และ Methods ของอ็อบเจ็กต์ได้โดยใช้ dot operator (.)
- **การเข้าถึง Attribute**

  objectName.attributeName
- **ตัวอย่าง** myDog.name = "Buddy";
- **การเรียกใช้งาน Method**

  objectName.methodName();
- **ตัวอย่าง** myDog.bark();

ตัวอย่างการสร้างและใช้งาน Object ใน Java

```java
//TestDog.java
public class TestDog {
    public static void main(String[] args) {
        Dog dog1 = new Dog();
        dog1.name = "Soba";
        dog1.breed = "Shih tzu";
        dog1.age = 5;

        Dog dog2 = new Dog();
        dog2.name = "Lucky";
        dog2.breed = "Pug";
        dog2.age = 10;

        System.out.println("Name : "+dog1.name+"\nBreed : "+ dog1.breed
                + "\nAge : "+dog1.age);

        System.out.println("Name : "+dog2.name+"\nBreed : "+ dog2.breed
                + "\nAge : "+dog2.age);
    }
}
```

Constructor

- **Constructor (คอนสตรัคเตอร์)** คือเมธอดพิเศษของคลาสที่ถูกเรียกโดยอัตโนมัติเมื่อมีการสร้าง Object
- ใช้สำหรับ "กำหนดค่าเริ่มต้น" ให้กับ Fields ของ Object ทันทีที่ Object ถูกสร้างขึ้น
- กฎของ Constructor
  - ต้องมีชื่อเดียวกับชื่อ Class
  - ห้ามมีชนิดข้อมูลส่งคืน (Return Type) แม้กระทั่ง void

การประกาศ Constructor ใน Java

- รูปแบบ

```
[modifier] Construct_name ([argument]) {
    //statements
}
```

- ตัวอย่าง

```
//Dog.java
public class Dog {
    public Dog() {
        //code
    }
}
```

Constructor ที่ไม่มีพารามิเตอร์ (Default constructor)

```java
//Person.java
public class Person {
    String name;

    public Person() {
        name = "Unnamed";
    }

    public void showName() {
        System.out.println("Name : " + name);
    }
}
```

```java
public class TestPerson {
    public static void main(String[] args) {
        Person p = new Person();
        p.showName();
    }
}
```

```
Name : Unnamed
```

ข้อสังเกต: *ถ้าไม่สร้าง Constructor เลย Java จะมี* ***default constructor*** *ให้เองโดยอัตโนมัติ*

Constructor แบบมีพารามิเตอร์ (Parameterized Constructor)

```java
//Person.java
public class Person {
    String name;
    int age;

    public Person(String n, int a) {
        name = n;
        age = a;
    }

    public void showName() {
        System.out.println("Name : " + name);
        System.out.println("Age  : " + age);
    }
}
```

```java
public class TestPerson {
    public static void main(String[] args) {
        Person p = new Person("Somsak", 35);
        p.showName();
    }
}
```

```
Name : Somsak
Age  : 35
```

การใช้ this keyword ใน Constructor

- เมื่อชื่อ parameter ซ้ำกับ attribute ของคลาส ต้องใช้ this เพื่ออ้างอิง attribute

```java
//Person.java
public class Person {
    String name;
    int age;

    public Person(String name, int age) {
        this.name = name;
        this.age = age;
    }

    public void showName() {
        System.out.println("Name : " + name);
        System.out.println("Age  : " + age);
    }
}
```

this.name ⟶ หมายถึง attribute ของ Object

name ⟶ หมายถึง parameter ที่ส่งเข้ามา

Constructor หลายแบบ (Constructor Overloading)

สามารถสร้าง Constructor ได้หลายตัวในคลาสเดียว โดยมีพารามิเตอร์ต่างกัน

```java
//Person.java
public class Person {
    String name;
    int age;

    public Person() {
        name = "Unknown";
        age = 0;
    }

    public Person(String name, int age) {
        this.name = name;
        this.age = age;
    }

    public void showName() {
        System.out.println("Name : " + name);
        System.out.println("Age  : " + age);
    }
}
```

```java
public class TestPerson {
    public static void main(String[] args) {
        Person p1 = new Person();
        Person p2 = new Person("Somsak", 35);

        p1.showName();
        p2.showName();
    }
}
```

```
Name : Unknown
Age  : 0
Name : Somsak
Age  : 35
```

เปรียบเทียบ Method vs Constructor

หัวข้อ	Method	Constructor
ชื่อ	ตั้งชื่อใดก็ได้	ต้องชื่อเดียวกับคลาส
คืนค่า	มีประเภทคืนค่า	ไม่มีประเภทคืนค่า
เรียกเมื่อใด	เรียกเมื่อผู้ใช้เรียกเอง	เรียกอัตโนมัติเมื่อ new object
ใช้สำหรับ	แสดงพฤติกรรม	กำหนดค่าเริ่มต้นให้ object

Access Modifier

- Access Modifiers เป็นคำสั่งในการควบคุมระดับการเข้าถึงของตัวแปรหรือเมธอดที่อยู่ภายในคลาส และยังเป็นคำสั่งที่ใช้ในการกำหนดการเข้าถึงของ Object ต่างๆ ใน Package เช่น คลาส และ Interfaces เป็นต้น
- ในภาษา Java นั้นมีคำสั่งในการควบคุมระดับการเข้าถึงอยู่ 4 ระดับด้วยกัน คือ public, protected, private และ no modifier (ไม่ต้องกำหนด)

Access Modifier

- **public :** คลาสหรือสมาชิกสามารถเข้าถึงได้จากที่ส่วนของโปรแกรม
- **protected :** คลาสหรือสมาชิกสามารถเข้าถึงได้ภายใน package เดียวกัน และ sub class ของมัน
- **no modifier (ไม่กำหนด) :** คลาสหรือสมาชิกสามารถเข้าถึงได้ภายใน package เดียวกัน และภายในคลาสเดียวกันเท่านั้น
- **private :** คลาสหรือสมาชิกสามารถเข้าถึงได้ภายในคลาสเดียวกันเท่านั้น

Access Modifier

Modifiers	Class	Package	Sub class	World
public	Yes	Yes	Yes	Yes
protected	Yes	Yes	Yes	No
no modifier	Yes	Yes	No	No
private	Yes	No	No	No

การใช้ Access Modifier

หัวข้อ	Method	Constructor
public	ทุกคลาสเข้าถึงได้	method ที่ใช้ทั่วไป
private	เฉพาะภายในคลาส	attribute ที่ไม่ต้องการให้แก้ไขโดยตรง
protected	คลาสลูกเข้าถึงได้	ใช้ในการสืบทอด

ตัวแปรอินสแตนซ์ (Instance Variables)

- คือตัวแปรที่ประกาศภายในคลาส แต่ภายนอกเมธอด
- แต่ละอ็อบเจ็กต์ที่สร้างจากคลาสจะมีชุดตัวแปรอินสแตนซ์เป็นของตัวเอง โดยมีค่าที่แตกต่างกันได้
- สถานะของอ็อบเจ็กต์ถูกกำหนดโดยค่าของตัวแปรอินสแตนซ์เหล่านั้น
- **ตัวอย่าง** ในคลาส Dog, ตัวแปร name, breed, และ age เป็นตัวแปรอินสแตนซ์

ตัวแปรคลาส (Class Variables)

- ตัวแปรคลาส (Class Variables) หรือ Static Variables คือตัวแปรที่ประกาศภายในคลาสด้วยคีย์เวิร์ด static
- มีเพียงสำเนาเดียวของตัวแปรคลาสที่ใช้ร่วมกันโดยทุกอ็อบเจ็กต์ที่สร้างจากคลาสนั้น
- หากค่าของตัวแปรคลาสถูกเปลี่ยนแปลง จะมีผลกระทบต่อทุกอ็อบเจ็กต์ของคลาสนั้น
- มักใช้สำหรับเก็บค่าคงที่หรือข้อมูลที่ทุกอินสแตนซ์ควรรู้

ตัวอย่างตัวแปรคลาส (Class Variables)

```java
//Dog.java
public class Dog {

    private String name;
    private String breed;
    private int age;
    public static int dogCount = 0; //class variable

    public Dog(String name, String breed, int age) {
        this.name = name;
        this.breed = breed;
        this.age  = age;
        Dog.dogCount++; //เพิ่มค่าให้ตัวแปร dogCount +1 เมื่อมีการสร้าง Object
    }
    public void bark() {
        System.out.println("Bok Bok!!!");
    }

    public void eat() {
        System.out.println(name + " is eating.");
    }
}
```

ตัวอย่างตัวแปรคลาส (Class Variables)

```java
//TestDog.java
public class TestDog {
    public static void main(String[] args) {
        Dog dog1 = new Dog("Soba", "Shih Tzu", 5);
        Dog dog2 = new Dog("Lucky", "Pug", 10);

        System.out.println("Total number of dogs : " + Dog.dogCount);
    }
}
```

```
Total number of dogs : 2
```

เมธอดอินสแตนซ์ (Instance Methods)

- **เมธอดอินสแตนซ์** (Instance Methods) คือเมธอดที่ไม่ได้ประกาศด้วยคีย์เวิร์ด static
- การเรียกใช้งานเมธอดอินสแตนซ์จะต้องทำผ่านอ็อบเจ็กต์
- ภายในเมธอดอินสแตนซ์ สามารถเข้าถึงและทำงานกับตัวแปรอินสแตนซ์ของอ็อบเจ็กต์นั้นๆ ได้
- **ตัวอย่าง** ในคลาส Dog, bark() และ eat() เป็นเมธอดอินสแตนซ์

เมธอดคลาส (Class Methods)

- **เมธอดคลาส** (Class Methods) **หรือ** Static Methods คือเมธอดที่ประกาศด้วยคีย์เวิร์ด static
- สามารถเรียกใช้งานได้โดยตรงผ่านชื่อคลาส โดยไม่ต้องสร้างอ็อบเจ็กต์ก่อน
- ภายในเมธอดคลาส ไม่สามารถเข้าถึงตัวแปรอินสแตนซ์ได้โดยตรง (เพราะไม่มีอ็อบเจ็กต์อ้างอิง) แต่สามารถเข้าถึงตัวแปรคลาสได้
- มักใช้สำหรับเมธอดที่เป็น utility function หรือทำงานที่เกี่ยวข้องกับคลาสโดยรวม
- **ตัวอย่าง** เมธอดสำหรับคำนวณบางอย่างที่ใช้ค่าคงที่ของคลาส

เมธอดคลาส (Class Methods)

```java
//Dog.java
public class Dog {

    private String name;
    private String breed;
    private int age;
    public static int dogCount = 0; //class variable

    public Dog(String name, String breed, int age) {
        this.name = name;
        this.breed = breed;
        this.age  = age;
        Dog.dogCount++; //เพิ่มค่าให้ตัวแปร dogCount +1 เมื่อมีการสร้าง Object
    }
```

เมธอดคลาส (Class Methods)

```
    public void bark() {
        System.out.println("Bok Bok!!!");
    }

    public void eat() {
        System.out.println(name + " is eating.");
    }

    public static void displayTotalDogs() {
        System.out.println("Total dogs created : " + dogCount);
    }
}
```

เมธอดคลาส (Class Methods)

```java
//TestDog.java
public class TestDog {
    public static void main(String[] args) {
        Dog dog1 = new Dog("Soba", "Shih Tzu", 5);
        Dog dog2 = new Dog("Lucky", "Pug", 10);

        Dog.displayTotalDogs();
    }
}
```

```
Total dogs created : 2
```

END.

Q & A