บทที่ 3

Class และ Object

ผศ.ดร.ธีระยุทธ ทองเครือ

ภาควิชาวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์

มหาวิทยาลัยขอนแก่น

ทำไมต้องสร้างคลาส

- ชนิดและโครงสร้างของตัวแปรพื้นฐาน int, float, double, string หรือ array เป็นต้น ยังไม่เพียงพอต่อการพัฒนาโปรแกรม

- การออกแบบและนิยามชนิดตัวแปรใหม่ตามความต้องการของนักพัฒนา โดยใช้ Class ช่วยให้มีชนิดของตัวแปรที่หลากหลายและตรงตามความต้องการมากขึ้น เช่น Student, Room, Point, Circle, Player

คลาส

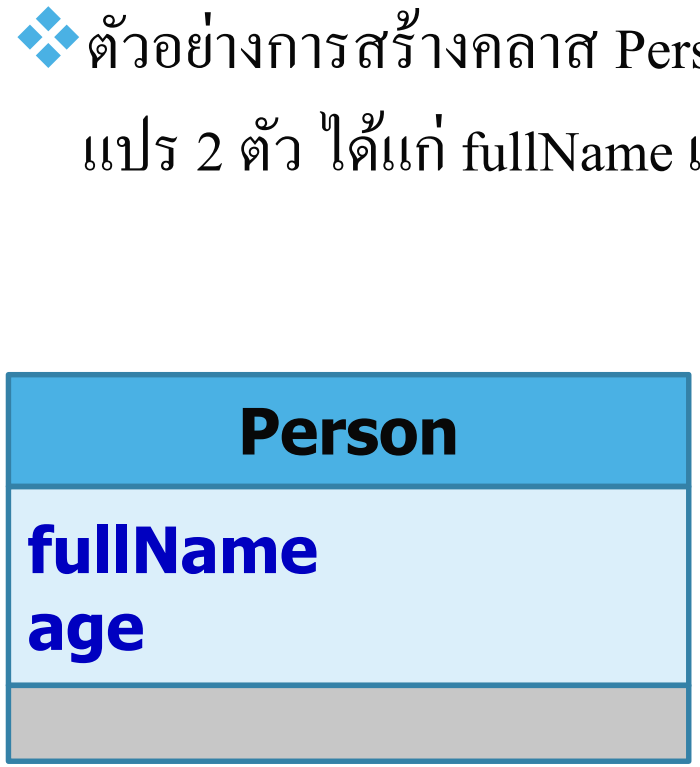
- คลาส (class) คือ ต้นแบบ หรือ template สำหรับนำไปใช้ในการสร้างตัวแปรชนิดใหม่ ที่เรียกว่า Object


```
class ชื่อคลาส {
        // Attribute
        ชนิดตัวแปร   ชื่อตัวแปรที่1;
        ชนิดตัวแปร   ชื่อตัวแปรที่2;
        ชนิดตัวแปร   ชื่อตัวแปรที่N;

        // Method
        returnType   เมธอดที่1();
        returnType   เมธอดที่2();
        returnType   เมธอดที่N();
}
```

คลาส Person

- ตัวอย่างการสร้างคลาส Person ที่มีเฉพาะส่วน attribute ประกอบด้วยตัวแปร 2 ตัว ได้แก่ fullName และ age




```
class Person {

    String fullName;
    int age;

}
```

Object

- Object คือ การประกาศใช้ตัวแปรที่มีโครงสร้างตามคลาสที่นิยามเอาไว้ มีรูปแบบดังนี้

ชื่อคลาส ชื่อobject = new ชื่อคลาส();

Person john = new Person();

ตัวแปร john มีชนิดเป็น Person

สั่งให้สร้างพื้นที่เก็บข้อมูลบนหน่วยความจำตามที่คลาส Person นิยามเอาไว้

การใช้ attribute และ method

- เมื่อต้องการอ่านค่าจากตัวแปร หรือเรียกใช้เมธอด จะเรียกผ่านชื่อ object โดยมีรูปแบบดังนี้

ชื่อobject.ชื่อattribute;

ชื่อobject.ชื่อmethod( );

Object ของคลาส Person

object

john: Person
fullName = John Smith **age** = 50

คลาส

Person
fullName **age**

```
// คลาสหลัก
public class TestApp {
    public static void main(String args[]) {
        Person john = new Person();
        john.fullName = "John Smith";
        john.age = 50;
        System.out.println(
          john.fullName +" "+ john.age +" ปี");
    }
} // จบคลาสหลัก

// คลาสที่นิยามขึ้นมาใหม่
class Person {
    String fullName;
    int age;
}
```

ตัวอย่างหน้าจอ

```
John Smith 50 ปี
```

Object มีได้หลายตัว

object

john: Person
fullName = John Smith age = 50

object

robert: Person
fullName = Robert Morgan age = 15

คลาส

Person
fullName age

```
// คลาสหลัก
public class TestApp {
    public static void main(String args[]) {
        Person john = new Person();
        john.fullName = "John Smith";
        john.age = 50;

        Person robert = new Person();
        robert.fullName = "Robert Morgan";
        robert.age = 15;

        System.out.println(john.fullName +" "+ john.age +" ปี");
        System.out.println(robert.fullName +" "+ robert.age +" ปี");
    }
} // จบคลาสหลัก

// คลาสที่นิยามขึ้นมาใหม่
class Person {
    String fullName;
    int age;
}
```

ตัวอย่างหน้าจอ

```
John Smith 50 ปี
Robert Morgan 15 ปี
```

Method

- เมธอด คือ ฟังก์ชันที่ทำงานร่วมกับตัวแปรของคลาสที่นิยามขึ้น
- รูปแบบ

ชนิดข้อมูลที่ส่งกลับ ซึ่งต้องสอดคล้องกับตัวแปรที่ return
หากไม่ return จะใส่คำว่า **void**

รายการตัวแปรที่รับมาประมวลผลภายในเมธอด

```
returnType   ชื่อเมธอด (type param1, type param2, ... ) {

        ชุดคำสั่ง;

        return ตัวแปร;
}
```

Method ของคลาส Person

```
// คลาสหลัก
public class TestApp {
    public static void main(String args[]) {
        Person john = new Person();
        john.fullName = "John Smith";
        john.setAge(150);
        john.setAge(20);

        int year = john.calculateBirthYear();
        System.out.println(john.fullName
              + " เกิดปี " + year);
    }
}
```

```
class Person {
    String fullName;
    int age;

    void setAge(int newAge) {
        if (newAge >= 1 && newAge <= 120 ) {
            age = newAge;
        } else {
            System.err.println(
                "ผิดพลาด!! อายุอยู่ระหว่าง 1 ถึง 120 เท่านั้น");
        }
    }

    int calculateBirthYear( ) {
        int yearOld = 2562 - age;
        return yearOld;
    }
}
```

object

john: Person
fullName = John Smith **age** = 20
setAge(int newAge): void calculateBirthYear( ): int

ตัวอย่างหน้าจอ

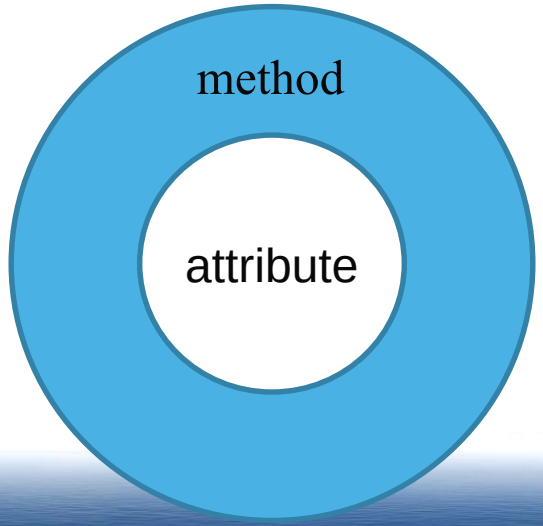
```
ผิดพลาด!! อายุอยู่ระหว่าง 1 ถึง 120 เท่านั้น
John Smith เกิดปี 2542
```

กิจกรรม

- สร้างคลาส Dog ที่ประกอบไปด้วย ชื่อเล่นและเพศ โดยมีเมธอด setSex( ) ใช้สำหรับกำหนดค่าเพศสุนัข มีการทำงานดังนี้
  - ให้เมธอดมีตัวแปรรับค่าเพศสุนัข สำหรับกำหนดค่าให้กับ attribute sex
  - ตรวจสอบว่ากำหนดชื่อเล่นสุนัขแล้วหรือยัง ถ้ายังให้แสดง Error และยังไม่ set ค่าลงตัวแปร sex
  - เพศสุนัข ต้องมีค่าเป็น ‘M’ หรือ ‘F’ เท่านั้น ถ้าไม่ใช่ ให้แสดง Error และยังไม่ set ค่าลงตัวแปร sex

การห่อหุ้ม (Encapsulation)

- ความสามารถของภาษาโปรแกรมเชิงวัตถุในการซ่อนข้อมูล (Information hiding) ที่อยู่ในคลาส เพื่อปกป้องการเข้าถึงจากคลาสอื่น ๆ
- การห่อหุ้ม เกิดขึ้นเมื่อมีการใช้คำสั่งควบคุมระดับการเข้าถึง (Access Modifier)



ทำไมต้องห่อหุ้ม

- ลดความผิดพลาดจากการทำงาน ที่อาจเกิดจากการกำหนดค่าด้วยผู้ใช้เอง
- เพื่อให้ข้อมูลถูกจัดการโดยเมธอดภายในคลาสเท่านั้น
- สามารถตรวจสอบความถูกต้องของข้อมูล (validate data) ได้
- นักพัฒนาเรียกใช้โดยไม่จำเป็นต้องรู้โครงสร้างภายในคลาส เพราะสิ่งที่ต้องการคือ ผลลัพธ์เท่านั้น
- สามารถควบคุมทิศทางของข้อมูลได้ เช่น อ่านได้อย่างเดียว (read-only), กำหนดค่าได้อย่างเดียว (write-only)

คำสั่งควบคุมระดับการเข้าถึง

- **private**
  - จะถูกเรียกใช้ได้เฉพาะภายในคลาสเท่านั้น
  - มักใช้กับ attribute
  - มักใช้กับเมธอดที่ใช้เฉพาะภายในคลาส ไม่ต้องการให้คลาสอื่นเรียกได้
- **public**
  - ถูกเรียกใช้ได้ทั้งจากภายนอกหรือภายในคลาสได้
  - มักใช้กับเมธอดของคลาส เช่น เมธอดกำหนดค่า และขอค่า attribute

รูปแบบการใช้

- วางคำสั่งควบคุมระดับการเข้าถึงไว้ด้านหน้าสุด ซึ่งใช้ได้ทั้ง attribute และ method

```
class ชื่อคลาส {
        // Attribute
        public/private  ชนิดตัวแปร  ชื่อตัวแปรที่1;
        public/private  ชนิดตัวแปร  ชื่อตัวแปรที่2;
        public/private  ชนิดตัวแปร  ชื่อตัวแปรที่N;

        // Method
        public/private  returnType  เมธอดที่1();
        public/private  returnType  เมธอดที่2();
        public/private  returnType  เมธอดที่N();
}
```

รูปแบบคลาสที่มีการห่อหุ้มข้อมูล

การใช้ private จะทำให้คลาสภายนอกเข้าถึงไม่ได้ จะต้องทำผ่าน method เท่านั้น

```
class ชื่อคลาส {
        // Attribute
        private  ชนิดตัวแปร  ชื่อตัวแปรที่1;
        private  ชนิดตัวแปร  ชื่อตัวแปรที่2;
        private  ชนิดตัวแปร  ชื่อตัวแปรที่N;

        // Getter&Setter Method
        public  returnType  เมธอดอ่านค่า();
        public  void  เมธอดกำหนดค่า();
}
```

เมธอดอ่านค่า (getter method)

- เมื่อ attribute ถูกห่อหุ้มด้วย private แล้ว หากต้องการให้ผู้ใช้อ่านค่าได้ จะต้องสร้างเมธอดสำหรับใช้ในการอ่าน/ดึงค่าจาก attribute ของคลาส
- รูปแบบ

```
public returnType get...() {
    return ... ;
}
```

- ตัวอย่าง

```
public String getFullName() {
    return fullName;
}
```

เมธอดกำหนดค่า (setter method)

- เมื่อ attribute ถูกห่อหุ้มด้วย private แล้ว หากต้องการให้ผู้ใช้กำหนดค่าให้กับ attribute จะต้องสร้างเมธอดสำหรับใช้ในการกำหนดค่า
- รูปแบบ

```
public void set... (type ตัวแปรรับค่า) {
    ชื่อattribute = ตัวแปรรับค่า ;
}
```

- ตัวอย่าง

```
public void setFullName(String newFullName) {
    fullName = newFullName;
}
```

คลาส Person ที่ถูกห่อหุ้มข้อมูล

```
class Person {
    // Attribute
    private String fullName;
    private int age;

    // Getter&Setter Method
    public String getFullName( ) {
        return fullName;
    }
    public void setFullName (String newFullName) {
        fullName = newFullName;
    }

    public int getAge( ) {
        return age;
    }
    public void setAge(int newAge) {
        age = newAge;
    }
}
```

fullName และ *age* ระบุให้ไม่สามารถเข้าถึงได้จากภายนอกได้

เมธอดอ่านค่า *fullName*

เมธอดกำหนดค่า *fullName*

เมธอดอ่านค่า *age*

เมธอดกำหนดค่า *age*

สัญลักษณ์คลาส

Person
- fullName - age
+ getFullName(): String + setFullName(String) + getAge(): int + setAge(int)

- แทน private

+ แทน public

ตัวอย่างการใช้คลาส Person

```
// คลาสหลัก
public class TestApp {
    public static void main(String args[]) {

        Person john = new Person();
        john.setFullName("John Smith");
        john.setAge(50);

        Person robert = new Person();
        robert.setFullName("Robert Morgan");
        robert.setAge(15);

         System.out.println(john.getFullName()
                + " " + john.getAge() + " ปี");
         System.out.println(robert.getFullName()
                + " " + robert.getAge() + " ปี");
    }
}
```

```
class Person {
    // Attribute
    private String fullName;
    private int age;

    // Getter&Setter Method
    public String getFullName( ) {
        return fullName;
    }
    public void setFullName(String newFullName) {
        fullName = newFullName;
    }

    public int getAge( ) {
        return age;
    }
    public void setAge(int newAge) {
        age = newAge;
    }
}
```

object

john: Person
fullName = John Smith **age** = 50

object

robert: Person
fullName = Robert Morgan **age** = 15

ตัวอย่างหน้าจอ

```
John Smith 50 ปี
Robert Morgan 15 ปี
```

กิจกรรม

- พิจารณาโค้ดในคลาสหลัก ที่มีการอ้างถึง attribute ของคลาส Person โดยตรง การเขียนแบบนี้สามารถทำงานได้หรือไม่

```
// คลาสหลัก

public class TestApp {
    public static void main(String args[]) {

        Person john = new Person();
        john.fullName = "John Smith";
        john.age = 50;

        System.out.println(john.fullName + " " +
                john.age + " ปี");
    }
}
```

```
class Person {
    // Attribute
    private String fullName;
    private int age;

    // Getter&Setter Method
    public String getFullName( ) {
        return fullName;
    }
    public void setFullName (String newFullName) {
        fullName = newFullName;
    }

    public int getAge( ) {
        return age;
    }
    public void setAge(int newAge) {
        age = newAge;
    }
}
```

Error จากการอ้างถึงส่วน private

TestApp.java

```
1  public class TestApp {
2      public static void main(String args[]) {
3          Person john = new Person();
4          john.fullName = "John Smith";
5          john.age = 50;
6          System.out.println(john.fullName + " " + john.age + " ปี");
7      }
8  }
9
10 class Person {
11     // Attribute
12     private String fullName;
13     private int age;
```

Problems | Javadoc | Declaration | Console

4 errors, 19 warnings, 0 others

Description	Location
Errors (4 items)	
The field Person.age is not visible	line 5
The field Person.age is not visible	line 6
The field Person.fullName is not visible	line 4
The field Person.fullName is not visible	line 6

การใช้ this

- แต่ละ Object ที่สร้างขึ้นด้วยคำสั่ง **new** จะสร้างพื้นที่เก็บ attribute บนหน่วยความจำแยกส่วนกัน แต่ส่วน method จะยังคงใช้ร่วมกัน
- เมธอดที่อ้างถึง attribute ของ object ปัจจุบันที่กำลังทำงานอยู่ อาจใช้ **this** ร่วมได้
- รูปแบบการใช้

  this.ชื่อattribute

- มักใช้ในการแยกระหว่างตัวแปรรับค่า (parameter) ที่มีชื่อตรงกับชื่อ attribute

```
class Person {
    private int age;

    public void setAge(int newAge) {
        age = newAge;
    }
}
```

ไม่ใช้ this

```
class Person {
    private int age;

    public void setAge(int age) {
        this.age = age;
    }
}
```

ใช้ this

Getter/Setter Method

```
public class MobilePhone {
    // attributes
    private String model;
    private int year;

    public String getModel() {
        return model;
    }
    public void setModel(String model) {
        this.model = model;
    }

    public int getYear() {
        return year;
    }
    public void setYear(int year) {
        this.year = year;
    }
}
```

Getter/Setter Method สำหรับตัวแปร model

Getter/Setter Method สำหรับตัวแปร year

this หมายถึง object นี้ ในกรณีที่ชื่อตัวแปรพารามิเตอร์ใช้ชื่อเดียวกับตัวแปรในคลาสจะใช้ this เพื่อบอกว่านี่คือตัวแปรของส่วนคลาส

เมธอด Constructor

- การทำงานของบางเมธอดในคลาส จำเป็นต้องกำหนดค่าเริ่มต้นให้กับบาง attribute ก่อนเรียกใช้เมธอด
- มีลักษณะเดียวกับเมธอด Setter คือ ใช้กำหนดค่าให้กับ attribute
- ช่วยรันชุดคำสั่งบางอย่าง ก่อนเรียกใช้เมธอดอื่นๆ เช่น การสร้างการเชื่อมต่อฐานข้อมูล จะต้องเกิดก่อน การเรียกเมธอดดึงข้อมูลจากฐานข้อมูล

รูปแบบ Constructor

- มีชื่อเมธอดเดียวกับชื่อคลาส
- ไม่มี return type และไม่ต้องกำหนด void ให้เมธอด
- คำสั่งควบคุมระดับการเข้าถึง เป็นแบบ public เท่านั้น
- รูปแบบ

```
class ชื่อคลาส {
        // Attribute
        private  ชนิดตัวแปร  ชื่อตัวแปร;

        // เมธอด Constructor
        public เมธอดชื่อเดียวกับคลาส (ตัวแปรรับค่า) {
            ...
        }
}
```

คลาส Person ที่มี Constructor

```
class Person {
    // Attribute
    private String fullName;
    private int age;

     // Constructor
    public Person (String newFullName, int newAge) {
        fullName = newFullName;
        age = newAge;
    }

    // Getter&Setter Method
    public String getFullName( ) {  return fullName;  }
    public void  setFullName (String newFullName) {
        fullName = newFullName;
    }
}
```

ไม่มี ***void*** หรือการส่งกลับใดๆ

สัญลักษณ์คลาส

Person
- fullName - age
+ Person(String, int) + getFullName(): String + setFullName(String) + getAge(): int + setAge(int)

- แทน private

+ แทน public

การทำงานของ Constructor

เมธอด Constructor จะถูกเรียกอัตโนมัติ เมื่อใช้คำสั่ง new หรือเมื่อมีการสร้าง object เท่านั้น และไม่สามารถเรียกซ้ำได้อีก

```
// คลาสหลัก
public class TestApp {
  public static void main(String args[]) {
    Person john = new Person("John Smith", 50);
    Person robert = new Person("Robert Morgan", 15);

    System.out.println(john.getFullName()
            + " " + john.getAge() + " ปี");
    System.out.println(robert.getFullName()
            + " " + robert.getAge() + " ปี");
  }
}
```

```
class Person {
   // Attribute
   private String fullName;
   private int age;

   // Constructor
   public Person (String newFullName, int newAge) {
       fullName = newFullName;
       age = newAge;
   }

   // Getter&Setter Method
   public String getFullName( ) { return fullName; }
   public void setFullName(String newFullName) {
       fullName = newFullName;
   }
   public int getAge( ) {  return age; }
   public void setAge(int newAge) { age = newAge; }
}
```

object

john: Person
fullName = John Smith **age** = 50

object

robert: Person
fullName = Robert Morgan **age** = 15

ตัวอย่างหน้าจอ

```
John Smith 50 ปี
Robert Morgan 15 ปี
```

เปรียบเทียบ ใช้/ไม่ใช้ Constructor

- ช่วยให้การสร้าง object และการกำหนดค่าให้กับตัวแปรใน object ทำได้ภายในบรรทัดเดียว

```
public class TestApp {
    public static void main(String args[]) {

        Person john = new Person();
        john.setFullName("John Smith");
        john.setAge(50);

        Person robert = new Person();
        robert.setFullName("Robert Morgan");
        robert.setAge(15);
    }
}
```

ใช้เมธอด setter กำหนดค่า

การสร้าง object และกำหนดค่าอาจต้องใช้โค้ดหลายบรรทัด และนักพัฒนา อาจลืมกำหนดค่าให้ fullName กับ age

```
public class TestApp {
    public static void main(String args[]) {

        Person john = new Person("John Smith", 50);

        Person robert = new Person("Robert Morgan", 15);



    }
}
```

ใช้เมธอด constructor กำหนดค่า

กำหนดค่าเริ่มต้นให้ fullName กับ age แต่สามารถกำหนดค่าใหม่ได้ จากเมธอด setter

Default Constructor

- Default Constructor หมายถึง เมธอด Constructor ที่ไม่มีตัวแปรรับค่า (parameter) ใดๆ

อาจใช้กำหนดค่าเริ่มต้นให้กับ attribute ก็ได้

```
class Person {
    // Attribute
    protected String fullName;
    protected int age;

    // Default Constructor
    public Person() {
        fullName = "";
        age = 0;
    }

    // Setter Method
    public void setFullName(String newFullName) {
        fullName = newFullName;
    }
    public void setAge(int newAge) {
        age = newAge;
    }
}
```

คลาสที่ไม่มี Constructor

```
class Person {
    // Attribute
    protected String fullName;
    protected int age;

    // Setter Method
    public void setFullName(String newFullName) {
        fullName = newFullName;
    }
    public void setAge(int newAge) {
        age = newAge;
    }
    // Default Constructor
    public Person( ) {

    }
}
```

- หากนักพัฒนา**ไม่เขียน** Constructor ใดๆ ไว้ Default Constructor จะถูกสร้างโดยอัตโนมัติ
- **แต่** หากเขียน Constructor แบบใดก็ตามไว้แล้ว Default Constructor จะไม่ถูกสร้างอัตโนมัติ นักพัฒนา **ต้องเขียน Default Constructor เองเท่านั้น**