1. การใช้ this

2. Constructor

3. Package

4. การตั้งชื่อ

ผศ.ดร.ธีระยุทธ ทองเครือ

การใช้ this

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

  this.ชื่อattribute

- ช่วยลดความกำกวม เมื่อใช้ชื่อตัวแปรเหมือนกัน เช่น ชื่อ parameter กับชื่อ 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 ที่ใช้ this

```
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; }
}
```

Robert Morgan

15

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 เอง เท่านั้น**

Packages

- Package ใช้ในการจัดหมวดหมู่ของคลาส
- เปรียบเสมือนโฟลเดอร์ในการจัดเก็บโค้ดโปรแกรม
- เมื่อมีการเก็บคลาสต่างๆไว้ใน Package แล้วจะต้องระบุชื่อ package ที่ส่วนหัวของโค้ดด้วย เช่น

```
package oop.lab1; // โค้ดนี้จะเก็บในโฟลเดอร์ oop\lab1
```

- เมื่อต้องการใช้คลาสต่าง package ต้องระบุชื่อ package ในส่วนหัวของโค้ด เพื่อให้โค้ดรู้จักคลาสจากภายนอก เช่น

```
import java.util.Scanner;// ประกาศใช้คลาส Scanner จากแพ็คเกจ java.util ซึ่งเป็นคลาสมาตรฐาน
import oop.lab2.Student;  // ประกาศใช้คลาส Student จากแพ็คเกจ oop.lab2
import oop.lab2.*;  // ประกาศใช้ทุกคลาสจากแพ็คเกจ oop.lab2
```

ตัวอย่าง

ชื่อ package ของคลาส ***InputTest***

java.util คือชื่อ package ที่ต้องการนำมาใช้

*ชื่อคลาสที่ต้องการนำมาใช้ กรณีไม่ระบุชื่อคลาส ใช้ * ได้*

```
package lab01;

import java.util.Scanner;

public class InputTest {
    public static void main(String args[]){
        Scanner scan = new Scanner(System.in);

        System.out.print("Please enter number: ");
        int num = scan.nextInt();
        System.out.print("You enter " + num);
    }
}
```

การตั้งชื่อ (Naming conventions)

- หลักการตั้งชื่อที่นิยมปฏิบัติกันโดยทั่วไปจะใช้ Camel case
- ชื่อคลาส
  - เป็นคำนาม
  - คำแรกขึ้นต้นด้วยตัวพิมพ์ใหญ่ เช่น

    Circle **EuropeCar** **GraduateStudent**
- ชื่อ Attribute หรือ ตัวแปร
  - เป็นคำนาม
  - คำแรกขึ้นต้นด้วยตัวพิมพ์เล็ก เช่น

    firstName, lastName, yearOfBirth, studentName

Camel case

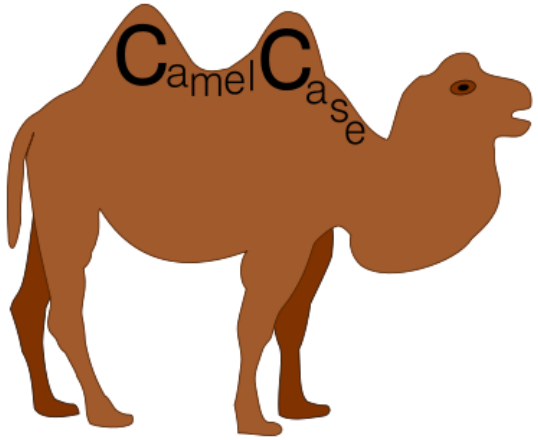
- รูปแบบการตั้งชื่อชนิดหนึ่ง สำหรับการเขียนโค้ดของภาษาโปรแกรม ที่มีตั้งแต่ 2 คำขึ้นไป (Compound Words)

- ใช้ตัวพิมพ์ใหญ่คั่น เมื่อขึ้นต้นคำต่อไป

  Word1Word2Word3

  หรือ

  word1Word2Word3



Naming conventions

- ชื่อ Method
  - เป็นคำกริยา
  - คำแรกขึ้นต้นด้วยตัวพิมพ์เล็ก เช่น

    getName **setName** **showDetails**

- ชื่อตัวแปรเก็บค่าคงที่ (Constant)
  - ใช้ตัวพิมพ์ใหญ่ทั้งหมด หากมีหลายคำให้แยกโดยใช้เครื่องหมาย _ (underscore) เช่น

    PI **MIN_SCORE** **MAX_SCORE**

Naming conventions

- ชื่อ Package
  - ใช้ตัวพิมพ์เล็กทั้งหมด เช่น

    subject.oop.assignment
    subject.oop.example
    subject.datastruc.assignment
    subject.datastruc.chapter1

  - อาจนำชื่อโดเมนขององค์กรมาใช้โดยใช้คำจากหลังไปหน้า เช่น

    com.pantip.controller
    com.pantip.model
    com.pantip.util