บทที่ 5

Overloading

และ

Override

Overloading

- Overloading คือ การเขียนเมธอดชื่อซ้ำกันในคลาส แต่มี parameter แตกต่างกัน ทำให้มีเมธอดหลากหลาย version

- ช่วง runtime จะถูกเลือกเมธอดแบบอัตโนมัติ โดยพิจารณาจากค่าที่ส่งให้กับเมธอด

- Overloading จะหมายถึง method ที่อยู่ในคลาสเดียวกัน

เมธอด 2 version

```
class Person {
    private String fullName;
    private int age;
    public Person(String fullName, int age) {
        this.fullName = fullName;
        this.age = age;
    }

    public int calculateBirthYear( ) {                    // version A
        int yearOld = 2562 - age;
        return yearOld;
    }

    public int calculateBirthYear(int currentYear, int age) {    // version B
        int yearOld = currentYear - age;
        return yearOld;
    }
}
```

*ชื่อเมธอด **calculateBirthYear** ซ้ำกัน แต่ parameter ต่างกัน*

ตัวอย่างการทำงาน

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

    int year = john.calculateBirthYear();
    System.out.println(year);
    year = john.calculateBirthYear(2563, 15);
    System.out.println(year);
  }
}
```

object

john: Person
fullName = John **age** = 20
Person(String fullName, int age) calculateBirthYear( ): int calculateBirthYear(int currentYear, int age): int

```
class Person {
    private String fullName;
    private int age;
    public Person(String fullName, int age) {
        this.fullName = fullName;
        this.age = age;
    }

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

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

ตัวอย่างหน้าจอ

```
2542
2545
```

static attribute

- คลาสบางคลาส อาจไม่ต้องการปกป้องข้อมูล เนื่องจากอาจมีความจำเป็นต้องใช้ข้อมูลร่วมกับคลาสอื่น เช่น
  - ค่าคงที่ ที่ไม่มีการเปลี่ยนแปลง เช่น ค่าใน Math.PI
  - สถานะ หรือ การตั้งค่าบางอย่าง (Configuration)
- รูปแบบการใช้

  static return_type ชื่อตัวแปร;
- รูปแบบการอ้างถึง

  ชื่อคลาส.ชื่อattribute

ตัวอย่าง

```
public class TestApp {
    public static void main(String args[]) {
        System.out.println( Visitor.count );  // อ่านค่าได้โดยตรง
        Visitor.count = 1;  // กำหนดค่าได้โดยตรง
    }
}

class Visitor {
    static int count = 0;
}
```

อ้างถึงตัวแปร count จากชื่อคลาส Visitor ได้ทันทีโดยไม่ต้องใช้คำสั่ง new Visitor( ) ก่อน

ตัวอย่างหน้าจอ

```
0
```

static method

- เมธอดที่ไม่ได้กำหนดให้เป็น static method (หรือ instance method) จะต้องสร้าง object ก่อนจึงจะเรียกใช้ได้

- เมธอดที่ต้องการให้ผู้ใช้เรียกใช้ได้โดยตรง โดยไม่ต้องสร้าง object ก่อน เรียกว่า static method (หรือ class method)
- รูปแบบการใช้

  static return_type ชื่อเมธอด(ตัวแปรรับค่า) { ... }

- รูปแบบการอ้างถึง

  ชื่อคลาส.ชื่อเมธอดstatic( );

ตัวอย่าง

```
public class TestApp {
    public static void main(String args[]) {
        System.out.println( Calculator.plus(5, 7) );
    }
}

class Calculator {
    static int plus(int a, int b) {
        return a+b;
    }
}
```

เรียกใช้เมธอด plus( ) ผ่านคลาส Calculator
โดยไม่ต้อง new Calculator( )

ตัวอย่างหน้าจอ

Overloading เมธอดแบบ static

```
public class TestMethodOverloading {
  public static int average(int n1, int n2) {                    // version A
    System.out.println("Run version A");
    return (n1+n2)/2;
  }

  public static double average(double n1, double n2) {           // version B
    System.out.println("Run version B");
    return (n1+n2)/2;
  }

  public static int average(int n1, int n2, int n3) {            // version C
    System.out.println("Run version C");
    return (n1+n2+n3)/3;
  }

  public static void main(String[] args) {
    System.out.println(average(1, 2));       // Use A
    System.out.println(average(1.0, 2.0));   // Use B
    System.out.println(average(1, 2, 3));    // Use C
    System.out.println(average(1.0, 2));     // Use B – ค่า 2 ถูกแปลงเป็น double อัตโนมัติ
    // average(1, 2, 3, 4);        // Compilation Error – ไม่ตรงกับ method ใดๆ
  }
}
```

Overloading Constructors

- การสร้าง object ใหม่ด้วยคำสั่ง new อาจทำได้หลายรูปแบบ

```
Person robert = new Person( );
Person john = new Person("John Smith");
Person robert = new Person("Robert Morgan", 15);
```

- การเขียน Constructor ที่มี parameter หลายแบบ เรียกว่าเป็นการ Overloading ด้วย
- Constructor ที่มี parameter ตรงตามรูปแบบที่เขียนไว้ จะถูกเลือกอัตโนมัติ เมื่อสร้าง Object ด้วยคำสั่ง **new**

ตัวอย่าง

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

    public Person() {                                   // version A
        fullName = "";
        age = 0;
    }

    public Person(String fullName) {                    // version B
        this.fullName = fullName;
        age = 0;
    }

    public Person(String fullName, int age) {           // version C
        this.fullName = fullName;
        this.age = age;
    }
}
```

มี 3 Constructor ที่มี parameter ต่างกัน

ตัวอย่างการทำงาน

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

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

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

    }
}
```

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

    public Person() {                    // version A
        fullName = "";
        age = 0;
    }

    public Person(String fullName) { // version B
        this.fullName = fullName;
        age = 0;
    }

    public Person(String fullName, int age) { // version C
        this.fullName = fullName;
        this.age = age;
    }
}
```

object

john: Person
fullName = John Smith **age** = 0

object

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

Override

- คลาสลูกที่สืบทอดมาจากคลาสแม่ หากไม่ต้องการใช้สิ่งที่คลาสแม่มีมา และต้องการเขียนใหม่ขึ้นมาเอง สามารถทำได้โดยประกาศชื่อให้ตรงกับคลาสแม่
- การบดบังเมธอด หรือ attribute จากคลาสแม่ เรียกว่า **Override**
- การ Override จะเกิดในคลาสลูกเท่านั้น ขณะที่ Overloading จะเกิดภายในคลาส
- เมธอดที่จะถูก Override ต้องมีรายการ parameter ที่ตรงกับเมธอดแม่ด้วยจึงจะถูก Override ได้

ตัวอย่าง

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

    // Getter&Setter Method
    public String getFullName( ) {
        return fullName;
    }

    public void setFullName(String fullName) {
        this.fullName = fullName;
    }

    public int getAge( ) {
        return age;
    }

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

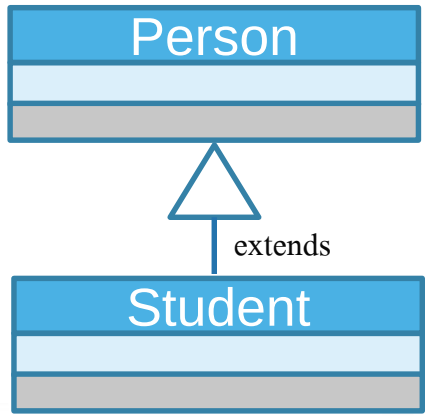
เกิดการ Override

```
class Student extends Person {
    private String studentId;

    public String getFullName() {  // Override
        return studentId  + " " + fullName;
    }

    public String getStudentId() {
        return studentId;
    }

    public void setStudentId(String id) {
        studentId = id;
    }
}
```



คลาสแม่

(superclass)

คลาสลูก

(subclass)

คลาสทดสอบ

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

        Student pop = new Student();
        pop.setFullName("Poppy");
        pop.setAge(21);
        pop.setStudentId("61100000-0");

        System.out.println(pop.getStudentId());
        System.out.println(pop.getFullName());
        System.out.println(pop.getAge() + " ปี");
    }
}
```

object

pop: Student
fullName = Poppy age = 21 studentId = 61100000-0

ตัวอย่างหน้าจอ

```
61100000-0
61100000-0 Poppy
21 ปี
```

คลาสแม่

```
class Person {
    // Attribute
    protected String fullName;
    protected 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;
    }
}
```

คลาสลูก

```
class Student extends Person {
    private String studentId;

    public String getFullName( ) {  // Override
        return studentId  + " " + fullName;
    }

    public String getStudentId() {
        return studentId;
    }
    public void setStudentId(String id) {
        studentId = id;
    }
}
```

กิจกรรม

- เมธอด setCost( ) ใดที่ถูก Overloading

ก

```
public class Invoice {

 public double setCost(double cost) {
  return cost;
 }
 public double setCost(double cost) {
  return cost;
 }
}
```

ข

```
public class Invoice {

 public double setCost(double cost) {
  return cost;
 }
 public double updateCost(double cost, int cID) {
  return cost;
 }
}
```

ค

```
public class Invoice {

 public double setCost(double cost) {
  return cost;
 }
 public double setCost(double cost, int cID) {
  return cost;
 }
}
```

ง

```
public class Invoice {

 public double setCost(double cost) {
  return cost;
 }
 public double cost(double cost) {
  return cost;
 }
}
```

กิจกรรม

- เมธอด getID( ) คือ การ _____ ในคลาสลูก VipOrder.

ก. Override

ข. Overload

ค. สืบทอด

ง. ถูกทุกข้อ

```
public class Order {
 public int getID(int orderID) {
  return orderID;
 }
}
public class VipOrder extends Order {
 public int getID(int orderID) {
  boolean vip = true;
  return orderID;
 }
}
```

กิจกรรม

- ข้อใดเป็นการ Override เมธอด setQoh

```
public long setQoh(long qty) {
 return qty;
}
```

ก

```
public setQoh(qty) {
    qty = qty + 5;
    return qty;
}
```

ข

```
public double setQoh() {
    qty = qty + 5;
    return qty;
}
```

ค

```
public void setQoh() {
    qty = qty + 5;
    return qty;
```

ง

```
public long setQoh(long qty) {
    qty = qty + 5;
    return qty;
}
```