Week5

Programming Fundamentals II

Course Outline

1. P2J (Basic)
2. P2J (Control structures)
3. P2J (Collection types)
4. Classes and methods
5. Inheritance
6. Polymorphism
7. Interfaces
8. Testing and debugging
9. Events
10. UI programming
11. Exceptions
12. Generics
13. Concurrency
14. Team project

Class and Objects

Class สิ่งที่ใช้อธิบายลักษณะและความสามารถของ Object เปรียบได้กับแม่แบบ ของ Object

Object สิ่งต่าง ๆ รอบตัว ซึ่งมีคุณลักษณะ (Attribute) และความสามารถในการทำงาน (Method)

ตัวอย่าง Object เช่น คน, รถยนต์, เครื่องคอมพิวเตอร์ เป็นต้น

Class and Objects

```
Class
{
        Attribute;

        Method
        {
                Statement;
        }

}
```

Argument and Parameter

Argument ตัวแปรที่ส่งไปให้เมธอดพร้อมกับการเรียกใช้เมธอด ในกรณีมีจำนวนมากกว่าหนึ่งค่าให้คั่นด้วยเครื่องหมาย “,”

Parameter ตัวแปรที่ทำหน้าที่รับค่าอาร์กิวเมนต์ที่ส่งมาใช้งานในเมธอด ในกรณีมีจำนวนมากกว่าหนึ่งค่าให้คั่นด้วยเครื่องหมาย “,”

Argument ต้องมีจำนวนเท่ากับตัวแปรที่เป็น **Parameter**

Datatype ของ **Argument** กับ **Parameter** ในแต่ละตำแหน่งจะต้องสอดคล้องกัน

Method

1. Instance Method ใช้งานบ่อยที่สุด การเรียกใช้ต้องสร้าง Object ขึ้นมาใช้งาน

2. Constructor method เป็น Method ที่มีการกำหนดชื่อ Method ให้เป็นชื่อเดียวกับชื่อ Class เพื่อกำหนดค่าเริ่มต้น

3. Static Method เป็น Method ที่สามารถเรียกใช้โดยไม่ต้องสร้าง Object ขึ้นมาใช้งาน

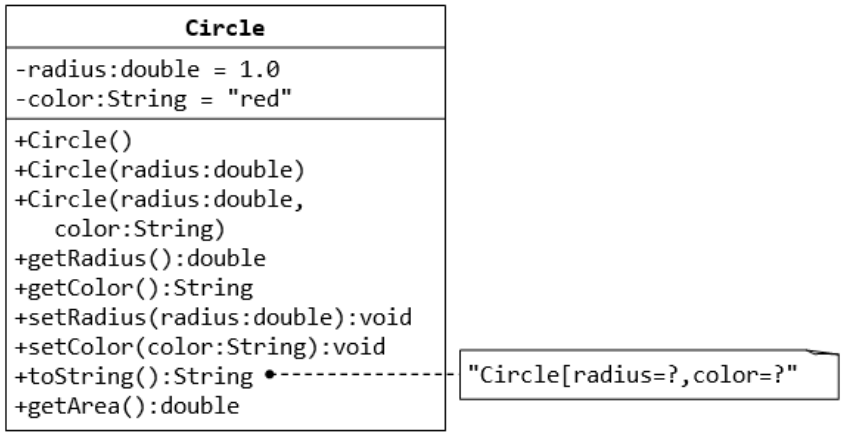
4. Overloading Method เป็น Method หลายๆ Method ที่มีชื่อเหมือนกัน แตกต่างที่ค่า Argument ที่แตกต่างกัน

5. Overriding Method เป็น Method ที่มีลักษณะที่ Class ลูกสามารถเขียนทับ Method ของ Class แม่ได้

Class diagram

Class name
// Attribute **- private** **+ public**
// Method **- private** **+ public**

Class diagram



Inheritance

Programming Fundamentals II

Week 5 **Inheritance**

1. Intro Inheritance
2. Superclasser & Subclasses
3. protected Members
4. Relationship between Superclasses and Subclasses
5. Constructors in Subclasses
6. Class Object

Intro Inheritance

1. สร้าง/พัฒนา Class จาก Class เดิมที่มีอยู่แล้ว
2. Class ใหม่จะนำ (Attribute + Method) ของ Class เดิมมาใช้
3. เป็นการถ่ายทอดคุณสมบัติจาก Class สู่อีก Class
   - Class ที่ถ่ายทอดคุณสมบัติ = คลาสแม่ (SuperClass)
   - Class ที่ได้รับการถ่ายทอดว่า = คลาสลูก (Subclass)
4. Subclass สามารถพัฒนาต่อเติม (Attribute + Method) ของตัวเอง
5. Subclass สามารถปรับปรุงแก้ไข (Attribute + Method) เดิมที่ได้รับมาจาก SuperClass ได้

Ex1 Inheritance

Superclass	Subclasses
Student	GraduateStudent, UndergraduateStudent
Shape	Circle, Triangle, Rectangle, Sphere, Cube
Loan	CarLoan, HomeImprovementLoan, MortgageLoan
Employee	Faculty, Staff
BankAccount	CheckingAccount, SavingsAccount

Fig. 9.1 | Inheritance examples.

Ex1 Inheritance

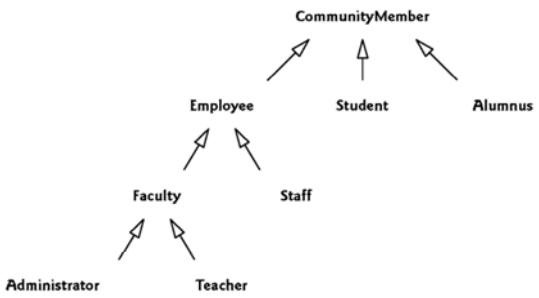


Fig. 9.2 | Inheritance hierarchy UML class diagram for university `CommunityMembers`.

Ex1 Inheritance

การสืบทอด University community class

–Each arrow in the hierarchy represents an is-a relationship.

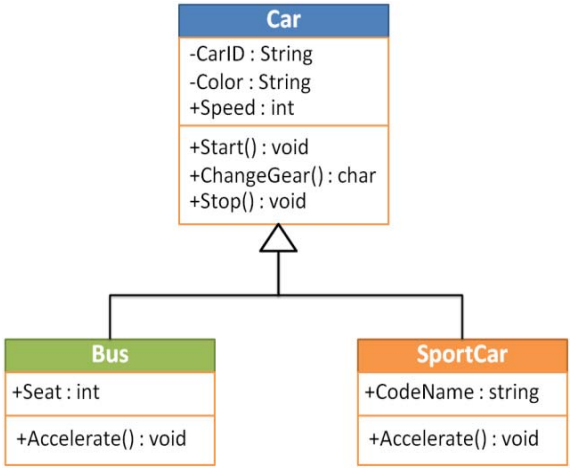
– Follow the arrows upward in the class hierarchy

1. an Employee is a CommunityMember”
2. “a Teacher is a Faculty member.”

– **CommunityMember** is the direct superclass of **Employee**, **Student** and **Alumnus** and is an indirect superclass of all the other classes in the diagram.

– Starting from the bottom, you can follow the arrows and apply the is-a relationship up to the topmost superclass.

Ex2 Inheritance



Ex2 Inheritance

1. Class Car เป็น Class รถทั่วไปที่มีสามารถ
   - Start() ได้
   - ChangeGear() ได้
   - Stop() ได้
2. Class Bus เป็น Class รถบัสที่มีสามารถ
   - Start() , ChangeGear() , Stop() ได้
   - มีผู้โดยสาร - Seat , Accelerate() ได้

ดังนั้น

Class Car เป็น Superclass

Class Bus เป็น Subclass ที่สืบทอดคุณสมบัติจาก Class Car

Ex3 Inheritance

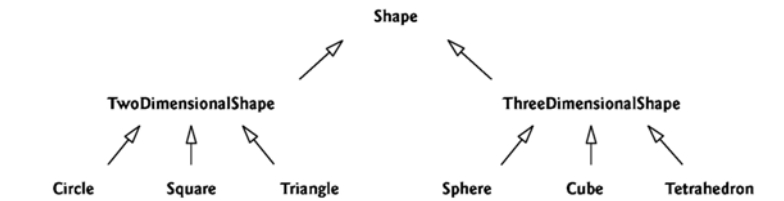


Fig. 9.3 | Inheritance hierarchy UML class diagram for Shapes.

Ex3 Inheritance

การสืบทอด Shape inheritance hierarchy.

1. Follow the arrows from the bottom of the diagram to the topmost superclass in this class hierarchy to identify several is a relationships.

- A Triangle is a TwoDimensionalShape and is a Shape
- A Sphere is a ThreeDimensionalShape and is a Shape.

Superclass

จาก Ex3 มาทำความรู้จักกับ Superclass Shape

รูปแบบ

```
[modifier] class SuperClassName {
        [AttributeName]
        [MethodName]
}
```

โดยที่ modifier Keyword ที่กำหนดคุณสมบัติการเข้าถึง Class

SuperClassName ชื่อของ Superclass

AttributeName การประกาศ Attribute

MethodName การประกาศ Method

Superclass

```
2  class Shape
3  {
4      protected float height;
5      protected float width;
6      protected float radius;
7
8      protected float getHeight()
9      {
10         return this.height;
11     }
12 }
```

Subclass

จาก Ex3 มาทำความรู้จักกับ Subclass TwoDimensionalShap

รูปแบบ

```
[modifier] class SubClassName extends SuperClassName {
        [AttributeName]
        [MethodName]
}
```

โดยที่

modifier	Keyword ที่กำหนดคุณสมบัติการเข้าถึง Class
SubClassName	ชื่อของ Subclass
SuperClassName	ชื่อของ Superclass
AttributeName	การประกาศ Attribute
MethodName	การประกาศ Method

Subclass

```
15 class TwoDimensionalShap extends Shape
16 {
17     protected float area;
18
19     protected float getArea()
20     {
21         return this.area;
22     }
23 }
```


Superclass

```
2  class Shape
3  {
4      protected float height;
5      protected float width;
6      protected float radius;
7
8      protected float getHeight()
9      {
10         return this.height;
11     }
12 }
13
14
15 class TwoDimensionalShap extends Shape
16 {
17     protected float area;
18
19     protected float getArea()
20     {
21         return this.area;
22     }
23 }
24
25 class Triangle extends TwoDimensionalSha
26 {
27     float base;
28 }
```

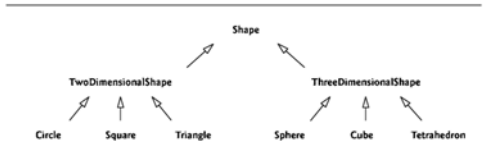


Fig. 9.3 | Inheritance hierarchy UML class diagram for Shapes.

Ex. โปรแกรมกำหนดความสูง Super/Sub Class

```
1  import java.util.Scanner;
2
3  public class ShapeTest
4  {
5      public static void main(String[] args)
6      {
7          float height;
8          Scanner scan = new Scanner(System.in);
9
10         Shape shape1 = new Shape();
11         System.out.printf("Shape height = %.2f \n\n",shape1.height);
12
13         System.out.print("Enter 2D height = ");
14         height = scan.nextFloat();
15         TwoDimensionalShap shape2 = new TwoDimensionalShap();
16         shape2.height = height;
17         System.out.printf("2D Shape : %.2f \n\n", shape2.height);
18
19         System.out.printf("Enter Triangle height : ");
20         height = scan.nextFloat();
21         Triangle shape3 = new Triangle();
22         shape3.height = height;
23         System.out.printf("Triangle : %.2f",shape3.height);
24     }
25 }
```

```
Shape height = 20.00

Enter 2D height = 12.2
2D Shape : 12.20

Enter Triangle height : 26.4
Triangle : 26.40
```

protected

1. Subclass สามารถใช้ Method + Attribute ที่มีการเข้าใช้แบบ **protected** ได้
2. Class ที่อยู่ในแพ็คเกจเดียวกันก็สามารถใช้ Method + Attribute ที่มีการเข้าใช้แบบ **protected** ได้
3. การเข้าใช้แบบนี้มีความเข้มงวดน้อยกว่าแบบ **private** ซึ่งห้าม Class อื่นใดเข้าใช้แต่ก็อิสระน้อยกว่าแบบ **public** ที่ใครๆก็สามารถเข้าใช้ได้

Method

Java มีรูปแบบการสร้าง Method หลายประเภท ซึ่งแต่ละประเภทมีหน้าที่ที่แตกต่างกัน แบ่งได้ดังนี้

1. Instance Method
2. Constructor method
3. Static Method
4. Overloading Method

➡ 5. Overriding Method

Overriding Method

- Overriding เป็นการเปลี่ยนแปลงการทำงานของ Method ใน Subclassที่สืบทอดมาจาก Superclass
- Method ใน Subclass จะมีชื่อ, ชนิดข้อมูลที่คืนค่า, จำนวนและชนิดข้อมูลของ Argument ที่เหมือนกับ Superclass
- สามารถพัฒนา Method ให้มีการทำงานในเรื่องเดียวกัน แต่แตกต่างกันในรายละเอียดของการทำงาน

การคำนวณที่มีสูตรคำนวณที่แตกต่างกัน
Ex การคำนวณค่าแรงของ พนง.ที่มีวิธีคำนวณที่แตกต่างกันไปในแต่ละตำแหน่ง

Ex. Overriding Method

```
2  class Employee
3  {
4      protected float rate = 300.0f, work=10.0f; int hour;
5
6      float calOT()
7      {
8          return hour*rate/work;
9      }
10 }
11
12 class DailyEmployee extends Employee
13 {   }
14
15 class MonthlyEmployee extends Employee
16 {
17     float pay, bonus;
18
19     float calOT()
20     {
21         pay = hour*rate/work;
22
23         if (hour>99)
24             pay += bonus;
25
26         return pay;
27   }
28 }
```

Ex. Overriding Method

```
3  public class EmployeeTest
4  {
5      public static void main(String[] args)
6      {
7          Scanner scan1 = new Scanner(System.in);
8          Scanner scan2= new Scanner(System.in);
9
10         System.out.print("Enter total OT hour : ");
11         Employee emp1 = new Employee();
12         emp1.hour = scan1.nextInt();
13         System.out.println("\"Employee\"\nTotal OT Pay = "
14                 + "(" + emp1.rate + " * " + emp1.hour + ") = " + emp1.calOT() + " BAHT \n");
15
16         DailyEmployee emp2 = new DailyEmployee();
17         emp2.hour = emp1.hour;
18         System.out.print("\"Daily Employee\"\nRate per Day = ");
19         emp2.rate = scan2.nextFloat();
20         System.out.println("Total OT Pay = (" + emp2.rate + " * " +
21                 emp2.hour + ") = " + emp2.calOT()+ " BAHT \n");
22
23         MonthlyEmployee emp3 = new MonthlyEmployee();
24         emp3.hour = emp1.hour;
25         System.out.print("\"Monthly Employee\"\nEnter Salary =  ");
26         emp3.rate = scan2.nextFloat()/30;
27         System.out.print("Enter Bonus = ");
28         emp3.bonus = scan2.nextFloat();
29         System.out.println("Total OT Pay = (" + emp3.rate + " * " +
30         emp3.hour + ") + " + emp3.bonus + " = " + emp3.calOT() + " BAHT");
31     }
32 }
```


Ex. Overriding Method

```
Enter total OT hour : 100
"Employee"
Total OT Pay = (300.0 * 100) = 3000.0 BAHT

"Daily Employee"
Rate per Day = 200
Total OT Pay = (200.0 * 100) = 2000.0 BAHT

"Monthly Employee"
Enter Salary =  20000
Enter Bonus = 200
Total OT Pay = (666.6667 * 100) + 200.0 = 6866.667 BAHT
```


Method

Java มีรูปแบบการสร้าง Method หลายประเภท ซึ่งแต่ละประเภทมีหน้าที่ที่แตกต่างกัน แบ่งได้ดังนี้

1. Instance Method
2. Constructor method
3. Static Method
4. Overloading Method
5. Overriding Method

Overloading Method

- Constructor เป็นสิ่งที่ Subclass ไม่สามารถสืบทอดจาก Superclass ได้
- แต่ Subclass สามารถใช้งาน Constructor ของ Superclass ได้

  ใช้ Keyword “ super ”
- สามารถทำ overload constructor ใน Superclass ได้ตามปกติ

Overloading Method

```
3  class Employee
4  {
5      protected float rate = 300.0f, work=10.0f; int hour;
6      protected float pay;
7
8      public Employee(int h)
9      {
10         pay = h*rate/work;
11     }
12     public Employee(int h, float r)
13     {
14         pay = h*r/work;
15     }
16     public Employee(int h, float r, float b)
17     {
18         this(h,r);
19         if (h>99)
20           pay += b;
21     }
22 }
23
24 class DailyEmployee extends Employee
25 {
26     public DailyEmployee (int h,float r)
27     {
28         super(h,r);
29     }
30 }
31
32 class MonthlyEmployee extends Employee
33 {
34     public MonthlyEmployee (int h, float r, float b)
35     {
36         super(h,r,b);
37     }
38 }
```


Overloading Method

```
Enter total OT hour : 100
"Employee"
Total OT Pay = (300.0 * 100) = 3000.0 BAHT

"Daily Employee"
Rate per Day = 200
Total OT Pay = (200.0 * 100) = 2000.0 BAHT

"Monthly Employee"
Enter Salary =  20000
Enter Bonus = 200
Total OT Pay = (666.6667 * 100) + 200.0 = 6866.667 BAHT
```

```
2  import java.util.Scanner;
3
4  public class EmployeeTest
5  {
6      public static void main(String[] args)
7      {
8          Scanner scan1 = new Scanner(System.in);
9          Scanner scan2= new Scanner(System.in);
10
11         System.out.print("Enter total OT hour : ");
12         int hour = scan1.nextInt();
13         Employee emp1 = new Employee(hour);
14         System.out.println("\"Employee\"\nTotal OT Pay = "
15                 + "(" + emp1.rate + " * " + hour + ") = " + emp1.pay + " BAHT \n");
16
17         System.out.print("\"Daily Employee\"\nRate per Day = ");
18         float rate = scan2.nextFloat();
19         DailyEmployee emp2 = new DailyEmployee(hour,rate);
20         System.out.println("Total OT Pay = (" + emp2.rate + " * " +
21                 hour + ") = " + emp2.pay + " BAHT \n");
22
23         System.out.print("\"Monthly Employee\"\nEnter Salary =  ");
24         rate = scan2.nextFloat();
25         System.out.print("Enter Bonus = ");
26         float bonus = scan2.nextFloat();
27         MonthlyEmployee emp3 = new MonthlyEmployee(hour,rate/30,bonus);
28             System.out.println("Total OT Pay = (" + rate/30 + " * " +
29                     hour + ") + " + bonus + " = " + emp3.pay + " BAHT");
30     }
31 }
```


FinalClass and FinalMethod

Final เป็น Keyword ที่ใช้นำหน้า Variable, Method, และ Class ได้

ทำให้มีคุณสมบัติดังนี้

1. Variable เป็นค่าคงที่ ไม่สามารถเปลี่ยนแปลงค่าได้
2. Method ไม่สามารถถูก override ได้
3. Class ไม่สามารถถ่ายทอดคุณสมบัติได้ คือไม่สามารถเป็น Superclass ได้

FinalClass

```
3  final class Employee
4  {
5      float rate = 300.0f, work=10.0f; int hour;
6
7      float calOT()
8      {
9          return hour * rate / work;
10     }
11     float calOT(float bonus)
12     {
13         float pay = hour*rate/work;
14
15         if (hour>99)
16           pay += bonus;
17         return pay;
18     }
19 }
```

FinalClass

```
Enter total OT hour : 100
"Employee"
Total OT Pay = (300.0 * 100) = 3000.0 BAHT

"Daily Employee"
Rate per Day = 200
Total OT Pay = (200.0 * 100) = 2000.0 BAHT

"Monthly Employee"
Enter Salary =  20000
Enter Bonus = 200
Total OT Pay = (666.6667 * 100) + 200.0 = 6866.667 BAHT
```

```
4  public class EmployeeTest
5  {
6      public static void main(String[] args)
7      {
8          Scanner scan1 = new Scanner(System.in);
9          Scanner scan2= new Scanner(System.in);
10
11         Employee emp = new Employee();
12         System.out.print("Enter total OT hour : ");
13         emp.hour = scan1.nextInt();
14         System.out.println("\n\"Employee\"\nTotal OT Pay = "
15                 + "(" + emp.rate + " * " + emp.hour + ") = " + emp.calOT() + " BAHT\n");
16
17         Employee dailyEmp = new Employee();
18         dailyEmp.hour = emp.hour;
19         System.out.print("\"Daily Employee\"\nRate per Day = ");
20         dailyEmp.rate = scan2.nextFloat();
21         System.out.println("Total OT Pay = (" + dailyEmp.rate + " * " +
22                 dailyEmp.hour + ") = " + dailyEmp.calOT()+ " BAHT\n");
23
24         Employee monthlyEmp = new Employee();
25         monthlyEmp.hour = emp.hour;
26         System.out.print("\"Monthly Employee\"\nEnter Salary =  ");
27         monthlyEmp.rate = scan2.nextFloat();
28         monthlyEmp.rate = monthlyEmp.rate/30;
29         System.out.print("Enter Bonus = ");
30         int b = scan1.nextInt();
31         System.out.println("Total OT Pay = (" + monthlyEmp.rate + " * " +monthlyEmp.hour + ") "
32                 + " + "  + b + " = " + monthlyEmp.calOT(b) + " BAHT");
33
34     }
35 }
```


FinalMethod

```
3  class Employee
4  {
5      float rate = 300.0f, work=10.0f; int hour;
6
7      final float calOT()
8      {
9          return hour*rate/work;
10     }
11 }
12
13 class DailyEmployee extends Employee
14 {    }
15
16 class MonthlyEmployee extends Employee
17 {
18     float calmOT(float bonus)
19     {
20         float pay = hour*rate/work;
21         if (hour>99)
22             pay += bonus;
23         return pay;
24     }
25 }
```


FinalMethod

```
Enter total OT hour : 100
"Employee"
Total OT Pay = (300.0 * 100) = 3000.0 BAHT

"Daily Employee"
Rate per Day = 200
Total OT Pay = (200.0 * 100) = 2000.0 BAHT

"Monthly Employee"
Enter Salary =  20000
Enter Bonus = 200
Total OT Pay = (666.6667 * 100) + 200.0 = 6866.667 BAHT
```

```
2  import java.util.Scanner;
3
4  public class EmployeeTest
5  {
6      public static void main(String[] args)
7      {
8          Scanner scan1 = new Scanner(System.in);
9          Scanner scan2= new Scanner(System.in);
10
11         Employee emp1 = new Employee();
12         System.out.print("Enter total OT hour : ");
13         emp1.hour = scan1.nextInt();
14         System.out.println("\"Employee\"\nTotal OT Pay = (" +
15         emp1.rate +  " * " + emp1.hour + ") = " + emp1.calOT() + " BAHT\n");
16
17         DailyEmployee emp2 = new DailyEmployee();
18         emp2.hour = emp1.hour;
19         System.out.print("\"Daily Employee\"\nRate per Day = ");
20         emp2.rate = scan2.nextFloat();
21         System.out.println("Total OT Pay = (" + emp2.rate + " * " +
22                 emp2.hour + ") = " + emp2.calOT()+ " BAHT\n");
23
24         MonthlyEmployee emp3 = new MonthlyEmployee();
25         emp3.hour = emp1.hour;
26         System.out.print("\"Monthly Employee\"\nEnter Salary =  ");
27         emp3.rate = scan2.nextFloat();
28         emp3.rate = emp3.rate/30;
29         System.out.print("Enter Bonus = ");
30         int b = scan1.nextInt();
31         System.out.println("Total OT Pay = (" + emp3.rate + " * " +
32                 emp3.hour + ") + " + b + " = " + emp3.calmOT(b) + " BAHT");
33
34     }
35 }
```

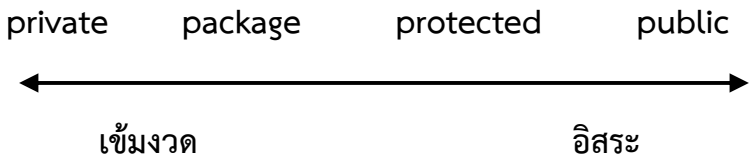

Encapsulation

1. เป็นการซ่อนรายละเอียดเพื่อป้องกันไม่ให้ Object ภายนอกเข้าถึงข้อมูลได้อย่างอิสระ
2. Object ไม่สามารถเรียกใช้หรือเปลี่ยนแปลงค่าข้อมูลได้
3. สามารถจำกัดสิทธิการใช้งาน Attribute และ Method ได้ด้วยระดับการเข้าใช้งานของ access modifier

หากต้องการซ่อนรายละเอียด Key “ private ”

หากต้องการใช้งาน Attribute หรือ Method ใดๆ ให้กำหนดเป็นแบบ “ public ”

การเข้าถึงข้อมูล



Class Object

- All classes in Java inherit directly or indirectly from class `Object`, so its 11 methods are inherited by all other classes.
- Every array has an overridden `clone` method that copies the array.
  - If the array stores references to objects, the objects are not copied—a *shallow copy* is performed.

Class Object

Method	Description
`equals`	This method compares two objects for equality and returns `true` if they're equal and `false` otherwise. The method takes any `Object` as an argument. When objects of a particular class must be compared for equality, the class should override method `equals` to compare the *contents* of the two objects. For the requirements of implementing this method (which include also overriding method `hashCode`), refer to the method's documentation at `docs.oracle.com/javase/7/docs/api/java/lang/Object.html#equals(java.lang.Object)`. The default `equals` implementation uses operator `==` to determine whether two references *refer to the same object* in memory. Section 14.3.3 demonstrates class `String`'s `equals` method and differentiates between comparing `String` objects with `==` and with `equals`.
`hashCode`	Hashcodes are `int` values used for high-speed storage and retrieval of information stored in a data structure that's known as a hashtable (see Section 16.11). This method is also called as part of `Object`'s default `toString` method implementation.

Fig. 9.12 | `Object` methods. (Part 1 of 3.)

Class Object

Method	Description
`toString`	This method (introduced in Section 9.4.1) returns a `String` representation of an object. The default implementation of this method returns the package name and class name of the object's class typically followed by a hexadecimal representation of the value returned by the object's `hashCode` method.
`wait`, `notify`, `notifyAll`	Methods `notify`, `notifyAll` and the three overloaded versions of `wait` are related to multithreading, which is discussed in Chapter 23.
`getClass`	Every object in Java knows its own type at execution time. Method `getClass` (used in Sections 10.5 and 12.5) returns an object of class `Class` (package `java.lang`) that contains information about the object's type, such as its class name (returned by `Class` method `getName`).
`finalize`	This `protected` method is called by the garbage collector to perform termination housekeeping on an object just before the garbage collector reclaims the object's memory. Recall from Section 8.10 that it's unclear whether, or when, `finalize` will be called. For this reason, most programmers should avoid method `finalize`.

Fig. 9.12 | `Object` methods. (Part 2 of 3.)

Class Object

Method	Description
`clone`	This `protected` method, which takes no arguments and returns an `Object` reference, makes a copy of the object on which it's called. The default implementation performs a so-called **shallow copy**—instance-variable values in one object are copied into another object of the same type. For reference types, only the references are copied. A typical overridden `clone` method's implementation would perform a **deep copy** that creates a new object for each reference-type instance variable. *Implementing `clone` correctly is difficult. For this reason, its use is discouraged.* Some industry experts suggest that object serialization should be used instead. We discuss object serialization in Chapter 15. Recall from Chapter 7 that arrays are objects. As a result, like all other objects, arrays inherit the members of class `Object`. Every array has an overridden `clone` method that copies the array. However, if the array stores references to objects, the objects are not copied—a shallow copy is performed.

Fig. 9.12 | `Object` methods. (Part 3 of 3.)

Class Object : toString

toString () เป็น Method ที่ช่วยในการพิมพ์ Attribute ของวัตถุ Method นี้จะส่ง String กลับมาเพื่อที่เราสามารถนำมาพิมพ์ออกไปที่จอภาพได้ด้วยคำสั่ง print, println

Ex. System.out.println(rect.toString());

หรือ

System.out.println(rect);

toString เป็น Method ที่ถูกโอเวอร์ไรด์จาก Class Object

Class Object : toString()

```
public class Rectangle extends Shape
{
      …
      public String toString()
      {
            String str= "Rectangle";
            str+= " color=" + getColor();
            str+= " width=" + width;
            str+= " height=" + height;
            str+= " area=" + getArea();
            return str;
      }
}
```


Class Object : equals()

```
public boolean equals(Object otherObject)
{
  if (otherObject instanceof Rectangle)
  {
      Rectangle otherRect = (Rectangle) otherObject;
      boolean equalWidth = width == otherRect.width;
      boolean equalHeight= height == otherRect.height;

      return equalWidth&& equalHeight;
  }
  return false;
}
```

Class Object : clone()

สร้างวัตถุที่เหมือนกับวัตถุที่ได้รับข้อความ

```
public Object clone()
{
      Rectangle clone = new Rectangle(width, height);
      return clone;
}
```


ข้อดีของการสืบทอด

- การนำกลับมาใช้ใหม่

  ถ้าต้องการสร้าง Class ที่มีความสามารถคล้ายๆกับ Class ที่มีอยู่แล้ว สามารถใช้การสืบทอดแทนที่จะเขียนขึ้นมาใหม่หมด
- ความเป็นมาตรฐานเดียวกัน

  Class พื้นฐานเป็นการกำหนดโครงแบบในการระบุความสามารถของ Object ใน Subclass
- เข้าใจสาระสำคัญได้ง่าย

ข้อเสียของการสืบทอด

- โปรแกรมทำงานช้าลง

  มี (overhead) ในค้นหาและเรียกใช้ Method ของ Subclass แต่ overhead ดังกล่าวถือว่าน้อยมากเมื่อเทียบกับประโยชน์ที่ได้จากการสืบทอด
- โปรแกรมมีขนาดใหญ่ขึ้น

  แต่หน่วยความจำราคาไม่แพง
- ความซับซ้อนเพิ่มขึ้น

  ผู้ใช้ต้องหาตาม Class ต่างๆ ที่อยู่ในผังการสืบทอดจนกว่าจะพบ Class ที่อิมพลีเมนต์ Method นั้น ปัญหาที่กล่าวถึงนี้มีชื่อว่าปัญหาลูกดิ่ง (Yo-yoproblem)