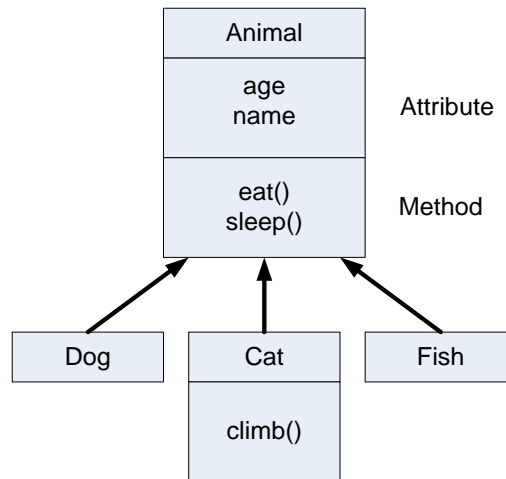


บทที่ 6

การสืบทอด (Inheritance)

การสืบทอดคุณสมบัติของ class ซึ่งเป็นวิธีการสร้าง class ใหม่จาก class ที่มีอยู่แล้ว ด้วยวิธีการนี้ เราสามารถใช้คุณสมบัติต่างๆ ของ class เดิมทั้ง attribute, method ที่ class เดิมมีอยู่แล้วได้ รวมทั้งเรายังสามารถเพิ่มเติมคุณสมบัติใหม่ๆ ที่ class เดิมไม่มีลงไปได้อีก ตัวอย่างแสดงดังรูปที่ 6.1



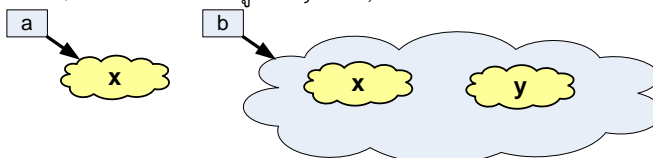
รูปที่ 6.1 class animal and her subclass

จากรูป 6.1 class Animal เป็นคลาสแม่ ที่ประกอบไปด้วย attribute age, name และ method eat(), sleep() และ class Dog, Cat, Fish เป็นคลาสลูกที่สืบทอดมาจาก class Animal โดย attribute age, name และ method eat(), sleep() ของ class แม่จะตามมามีด้วย นอกจากนั้น class Cat ยังได้เพิ่ม method climb() อีกด้วย

6.1 นิยามการสืบทอด

นิยาม SuperClass และ SubClass

| | |
|---|--|
| <pre>[modifier] class superClassName { } [modifier] class subClassName extends SuperClassName { }</pre> | |
| superClassName | ชื่อของ class แม่, parent class หรือ super class |
| subClassName | ชื่อของ class ลูก, child class หรือ sub class |
| extends | คำสงวนที่เอาไว้ใช้แสดงความสัมพันธ์ ลูก→แม่ (หัวลูกศรชี้ไปที่ superClass) |
| modifier | เป็นคำสงวนที่กำหนดคุณสมบัติการเข้าถึง เป็น option สัญลักษณ์ [] |

| ตัวอย่าง 6.1 TestMethod | | Note |
|-------------------------|-------------------|---|
| 1 | class A | <p>เมื่อสร้าง instances ของ class ด้วยคำสั่ง A a = new A(); B b = new B(); โดย object b จะมีทั้ง attribute x, y ซึ่ง x สืบทอดมาจาก class A ส่วน y เป็น attribute ที่มาจาก class B (ของ ตัวเอง) และสามารถวาดรูป object a, b ได้ ดังนี้</p>  |
| 2 | { | |
| 3 | int x; | |
| 4 | } | |
| 5 | class B extends A | |
| 6 | { | |
| 7 | int y; | |
| 8 | } | |

| ตัวอย่าง 6.2 TestSimpleInherit1 | | Note |
|---------------------------------|--|--|
| 1 | class Mom1 | class Mom1 ประกอบด้วย attribute: x1; y1; z1; method: show1(); |
| 2 | { | |
| 3 | int x1 =1, y1=2; | |
| 4 | int z1=12; | |
| 5 | void show1{ | |
| 6 | System.out.println("x1 =" +x1); | |
| 7 | System.out.println("y1 =" +y1); | |
| 9 | } | |
| 1 | class Child1 extends Mom1 | class Child1 extends Mom1 ประกอบด้วย attribute: n1; method: show2(); sum1(); โดย class Child1 ยังมีส่วนที่ขยายมาจาก แม่ Mom1 ด้วยเนื่องจากคำสั่ง extends |
| 2 | { | |
| 3 | int n1 =29; | |
| 4 | void show2{ | |
| 5 | System.out.println("n1 =" +n1); | |
| 6 | } | |
| 7 | void sum1{ | |
| 8 | System.out.println("z1 =" +z1); | |
| 10 | System.out.println("sum="+(x1+y1+n1)); | บรรทัดที่ 4 สร้าง object ด้วยคำสั่ง Child1 simple1 = new Child1(); บรรทัดที่ 5 เรียกใช้ simple1.show1(); ซึ่งเป็น method ที่สืบทอดมาจาก Mom1 บรรทัดที่ 6 เรียกใช้ simple1.show2(); ซึ่งเป็น method ของตัวเอง บรรทัดที่ 7 เรียกใช้ simple1.sum1(); ซึ่งเป็น method ของตัวเอง |
| 11 | } | |
| 12 | } | |
| 13 | } | |
| 14 | } | |
| 15 | } | |
| 16 | } | |
| 17 | } | |
| 1 | ---exec:TestSimpleInherit1 | ถ้า code line 4 เปลี่ยนเป็น private int z1=12; ผลลัพธ์จะเป็นเช่นไร? |
| 2 | | |
| 3 | x1 =1 | |
| 4 | y1 =2 | |
| 5 | n1 =29 | |
| 6 | z1 =12 | |
| 7 | sum =32 | |
| 8 | | |
| 9 | ---operation:complete | |

6.2 Modifier

modifier คือ คำสงวนของภาษา java ที่ใช้กำหนดคุณสมบัติการเข้าถึงข้อมูลของ class และสมาชิกของคลาส เพื่อรักษาความปลอดภัยและป้องกันการเปลี่ยนแปลงข้อมูลภายใน class รายละเอียดของ modifiers แสดงดังตารางที่ 6.1

ตารางที่ 6.1 ตารางแสดง modifiers และคำอธิบาย

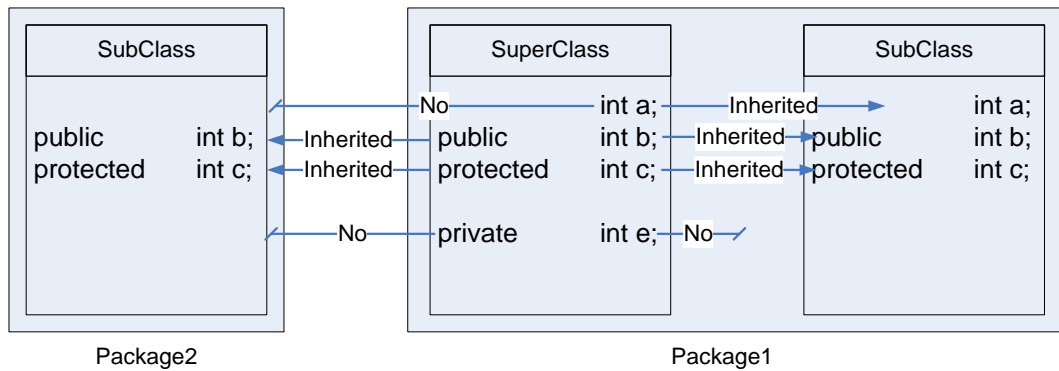
| Modifier | คำอธิบาย |
|-----------|--|
| final | เป็นการกำหนดค่าเริ่มต้นให้แก่ attribute และไม่สามารถเปลี่ยนแปลงค่าได้ตลอดการทำงาน |
| static | เป็นการกำหนดให้ attribute และ method ถูกเก็บอยู่ที่ class ที่เดียว ทุกๆ object เรียกใช้ผ่านชื่อ class โดยไม่ต้องสร้าง object |
| private | สามารถใช้งานได้ภายใน class เดียวกันเท่านั้น |
| protected | สามารถใช้งานได้ภายใน class เดียวกันและ class ที่สืบทอดเท่านั้น |
| public | ทุก class สามารถใช้งานได้ |
| package | ทุกๆ class ที่อยู่ใน package เดียวกัน |

| ตัวอย่าง 6.3 TestModi1 | | Note |
|------------------------|---|--|
| 1 | class TestModi1 | ตัวอย่างแสดงลักษณะการใช้งาน modifier ที่อยู่ใน class เดียวกัน |
| 2 | { | |
| 3 | static final double PI = 3.14; | |
| 4 | final double y = 32.23; | |
| 5 | private double radius = 8 ; | |
| 6 | public int x = 20; | |
| 7 | public static void main(String[] args) | |
| 8 | { | |
| 9 | { | |
| 10 | PI = 120.35 ; | |
| 11 | TestModi1 x1 = new TestModi1(); | |
| 12 | x1.y = 23.14; | |
| 13 | x1.radius = 5; | |
| 14 | x1.x = 32; | |
| 15 | } | |
| | } | |
| 1 | ---exec:TestModi1 | |
| 2 | | |
| 3 | TestModi1.java:10:error:cannot assign a | |
| 4 | value to final variable PI | |
| 5 | PI = 120.35 ; | |
| 6 | ^ | |
| 7 | TestModi1.java:12:error:cannot assign a | |
| 8 | value to final variable y | |
| 9 | x1.y = 23.14; | |
| 10 | ^ | |
| 11 | 2 errors | |
| 12 | | |
| 13 | ---operation:complete | |

| ตัวอย่าง 6.4 TestModi2 | | Note |
|--|---|------|
| <pre>1 class Sphere 2 { 3 static final double PI =3.14; 4 final double y =32.23; 5 private double radius =8 ; 6 public int x =20; 7 }</pre> | ตัวอย่างแสดงลักษณะการใช้งาน modifier ที่อยู่ต่าง class กัน | |
| <pre>1 class TestModi2 2 { 3 public static void main(String[]args) 4 { 5 Sphere.PI =120.35 ; 6 Sphere x1 =new Sphere(); 7 x1.y =23.14; 8 x1.radius =5; 9 x1.x =32; 10 } 11 }</pre> | | |
| <pre>1 ---exec:TestModi2 2 3 TestModi2.java:13:error:cannot assign a 4 value to final variable PI 5 Sphere.PI =120.35 ; 6 ^ 7 8 TestModi2.java:15:error:cannot assign a 9 value to final variable y 10 x1.y =23.14; 11 ^ 12 TestModi2.java:16:error:radius has private 13 access in Sphere 14 x1.radius =5; 15 ^ 16 3 errors 17 18 ---operation:complete</pre> | <pre>ถ้า code เปลี่ยนเป็น class TestModi3 extends Sphere { public static void main(String[] args) { Sphere.PI = 120.35 ; Sphere x1 = new Sphere(); x1.y = 23.14; x1.radius = 5; x1.x = 32; } }</pre> <p>ผลลัพธ์จะเป็นเช่นไร?</p> | |

6.3 Inherit Attribute

การสืบทอด attribute ของ SubClass จาก SuperClass สามารถแสดงดังรูปที่ 6.2



รูปที่ 6.2 แสดงการจำแนกการสืบทอด attribute ของ SubClass จาก SuperClass

จากรูป 6.2 สามารถกล่าวได้ว่า

- ถ้า superclass และ subclass อยู่ใน packet เดียวกัน ทุกๆ accessType สามารถ Inherit ได้ หมดยกเว้น private
- ถ้า superclass และ subclass อยู่คนละ packet ตัวแปรชนิด private และในกรณีที่ไม่ระบุ accessType นั้นไม่สามารถ Inherit ได้ ส่วนตัวแปรชนิด public และ protected สามารถ Inherit ได้

| ตัวอย่าง 6.5 TestInheritAtt | | Note |
|---|---|--|
| 1 2 3 4 5 6 | <pre> class ClassX { protected int m; public String toString() { return new String("(" + m + ")"); } } </pre> | class ClassX ประกอบด้วย attribute: m; method: toString(); |
| 1 2 3 4 5 6 7 | <pre> class ClassY extends ClassX { private int n; public String toString() { return new String("(" + m + ", " + n + ")"); } } </pre> | class ClassY extends Class X ประกอบด้วย attribute: n; method: toString(); |
| 1 2 3 4 5 6 7 8 9 | <pre> class TestInheritAtt { public static void main(String[] args){ ClassX x = new ClassX(); System.out.println("x = " + x); ClassY y = new ClassY(); System.out.println("y = " + y); } } </pre> | บรรทัดที่ 4 สร้าง object ด้วยคำสั่ง ClassX x = new ClassX(); บรรทัดที่ 6 สร้าง object ด้วยคำสั่ง ClassY y = new ClassY(); |
| 1 2 3 4 5 6 | <pre> ---- exec: TestInheritAtt x = (0) y = (0,0) ---- operation: complete </pre> | |

6.4 Inherit Method

ทุกๆ method ใน superclass ยกเว้น Constructor method สามารถ inherit ได้เหมือนกับการ inherited attribute โดย method ที่มีการประกาศเป็น private ไม่สามารถ inherit ได้และถ้าหน้า method ไม่มีการกำหนด accessType แล้ว method นั้นจะ inherit ได้ก็ต่อเมื่อ subclass อยู่ใน package เดียวกับ superclass

ส่วน method constructor นั้นแตกต่างไปจาก method อื่นๆ โดย constructor ของ superclass ไม่สามารถถูก inherit ได้

| ตัวอย่าง 6.6 TestInheritAtt | | Note |
|--|--|--|
| <pre> 1 class Coin{ 2 int value; 3 public int getValue(){ 4 return value; 5 } 6 }</pre> | | class Coin ประกอบด้วย attribute: value; method: getValue(); |
| <pre> 1 class Quarter extends Coin{ 2 Quarter() 3 { 4 value =26; 5 } 6 }</pre> | | class Quarter extends Coin ประกอบด้วย constructor: Quarter(); |
| <pre> 1 class TestAccessMethod1 2 { 3 public static void main(String[] args){ 4 Quarter q =new Quarter(); 5 System.out.println("Value is "+ 6 q.getValue()); 7 } 8 }</pre> | | บรรทัดที่ 4 สร้าง object ด้วยคำสั่ง Quarter q = new Quarter(); บรรทัดที่ 6 เรียกใช้ method getValue() โดยใช้คำสั่ง q.getValue() |
| <pre> 1 ----exec:TestAccessMethod1 2 3 Value is 26 4 5 ----operation:complete</pre> | | ถ้า code line 3 เปลี่ยนเป็น private int getValue(); ผลลัพธ์จะเป็นเช่นไร? |

| | |
|-----------------------------|------|
| ตัวอย่าง 6.7 TestInheritAtt | Note |
|-----------------------------|------|

| | | |
|---|---|--|
| 1 2 3 4 5 6 7 8 9 10 11 12 13 | <pre> class Mother { protected int i; private int j; void setIJ(int x,int y){ i=x; j=y; } private void showIJ(){ System.out.println("i in Mother :"+i); System.out.println("j in Mother :"+j); } } </pre> | class Mother ประกอบด้วย attribute: protected i; private j; method: setIJ(x,y); private showIJ(); |
| 1 2 3 4 5 6 7 8 | <pre> class Child extends Mother{ int iSquare; int total; void calculate(){ iSquare =i*i; total =i+j; } } </pre> | class Child extends Mother ประกอบด้วย attribute: iSquare; total; method: calculate(); |
| 1 2 3 4 5 6 7 8 9 10 11 12 | <pre> class TestAccessMethod2 { public static void main(String[]args){ Child subObj =new Child(); subObj.setIJ(1,2); subObj.showIJ(); subObj.calculate(); System.out.println("Sum of I and J is:" +subObj.total +"and square of I is:" +subObj.iSquare); } } </pre> | บรรทัดที่ 4 สร้าง object ด้วยคำสั่ง Child subObj = new Child(); บรรทัดที่ 5 เรียกใช้ subObj.setIJ(1,2); บรรทัดที่ 6 เรียกใช้ subObj.showIJ(); บรรทัดที่ 7 เรียกใช้ subObj.calculate(); |
| 1 2 3 4 5 6 7 8 9 10 11 12 13 14 | <pre> ---exec:TestAccessMethod2 2 3 subObj.showIJ(); 4 ^ 5 symbol: method showIJ() 6 location:variable subObj of type Child 7 TestAccessMethod1.java:29:error:j has private 8 access in Mother 9 total =i+j; 10 ^ 11 2 errors 12 13 14 ---operation:complete </pre> | |

6.5 Keyword super

ในการสืบทอดของ subclass จาก superclass นั้นในบางครั้งอาจเกิดเหตุการณ์ที่ ตัวแปรหรือ method ใน subclass และ superclass มีชื่อซ้ำกัน และเมื่อเกิดเหตุการณ์ดังกล่าวขึ้น Java จัดการโดยการ ทำ hiding data member ของ superclass โดยใช้ตัวแปร และ method ของ subclass แทน และถ้า ต้องการเข้าถึงตัวแปรหรือ method ของ superclass ก็สามารทำได้โดยใช้ keyword super ซึ่งมีลักษณะ การใช้งานดังนี้

1) keyword super สำหรับอ้างถึง ตัวแปรและ Method ของ superclass

| | |
|------------------|--|
| super.dataMember | |
| super | คำสงวน super ที่เอาไว้ใช้เรียก ตัวแปรและ Method ของ superclass |
| dataMember | ตัวแปร attribute และ method ใน class |

| ตัวอย่าง 6.8 TestSuper1 | | Note |
|---|--|--|
| 1 2 3 4 5 6 7 | <pre> class A { int a; void print0{ System.out.println("A "+a); } } </pre> | class A ประกอบด้วย attribute: a; method: print(); |
| 1 2 3 4 5 6 7 8 9 10 11 12 | <pre> class B extends A { int a; B(int x,int y){ super.a=x; this.a =y; } void print0{ super.print0; System.out.println("B "+a); } } </pre> | class B extends A ประกอบด้วย attribute: a; method: print(); constructor: B(x,y); |
| 1 2 3 4 5 6 7 | <pre> class TestSuper1 { public static void main(String[] args){ B b =new B(1,2); b.print0; } } </pre> | บรรทัดที่ 4 สร้าง object ด้วยคำสั่ง B b = new B(1,2); บรรทัดที่ 5 เรียกใช้ b.print(); โดยเป็น print ของ object b |
| 1 2 3 4 5 6 | <pre> ---exec:TestSuper1 A 1 B 2 ---operation:complete </pre> | |

2) keyword super สำหรับอ้างถึง Constructor Method ของ superclass

| |
|-----------------|
| super(argument) |
|-----------------|

| | |
|----------|--|
| Super | คำสงวน super ที่เอาไว้ใช้เรียก ตัวแปรและ Method ของ superclass |
| argument | อาร์กิวเมนต์ |

ถ้า superClass มีการสร้าง constructor ไว้แล้ว subclass สามารถเรียกใช้ constructor ของ superClass ได้เลย ด้วยผ่าน keyword super และไม่ต้องสร้าง constructor ขึ้นมาตัวเอง โดยมีเงื่อนไขว่า keyword “super” จะต้องเป็นคำสั่งที่แรกที่เรียกใช้งานภายใน constructor ของ subclass เท่านั้น

| ตัวอย่าง 6.9 TestSuperConst1 | | Note |
|---|---|---|
| 1 2 3 4 5 6 7 8 | <pre> class Mom { protected String str; Mom(String str) { this.str = str; } } </pre> | class Mom ประกอบด้วย attribute: str; constructor: Mom(str); |
| 1 2 3 4 5 6 7 8 9 10 11 | <pre> class Child extends Mom { Child(String str) { super(str); } void showST() { System.out.println("massege :"+str); } } </pre> | class Child extends Mom ประกอบด้วย method: showST(); constructor: Child(str); |
| 1 2 3 4 5 6 7 | <pre> class TestSuperConst1 { public static void main(String[] args){ Child x = new Child("welcome"); x.showST(); } } </pre> | บรรทัดที่ 4 สร้าง object ด้วยคำสั่ง Child x = new Child("welcome"); บรรทัดที่ 5 เรียกใช้ x.showST(); |
| 1 2 3 4 5 | <pre> ---exec:TestSuperConst1 massege :welcome ---operation:complete </pre> | |

6.6 Overriding Method

เป็นวิธีการที่ชื่อ method เหมือนกัน รวมทั้ง signature ก็เหมือนกัน แต่ลักษณะการทำงานแตกต่างกันเรียกว่า “Overriding Method” ซึ่งการทำงานลักษณะนี้จะเป็นการทำงานของ method ของ subclass ที่สืบทอดมาจาก superclass โดย subclass สามารถแก้ไขปรับปรุงการทำงานของ method ได้ กรณีที่ compiler พบ Overriding method ใน subclass และ superclass compiler จะเลือกทำงานที่ method ของ subclass โดยอัตโนมัติ

| ตัวอย่าง 6.10 TestOverrideMethod1 | | Note |
|-----------------------------------|--|---|
| 1 2 3 4 5 6 | <pre> class Mom { void show(){ System.out.println("show of Mom"); } } </pre> | class Mom ประกอบด้วย method: show(); |
| 1 2 3 4 5 6 | <pre> class Child extends Mom { void show(){ System.out.println("show of Child"); } } </pre> | class Child extends Mom ประกอบด้วย method: show(); เป็น override method |
| 1 2 3 4 5 6 7 | <pre> class TestOverrideMethod1 { public static void main(String[] args){ Child obj =new Child(); obj.show(); } } </pre> | บรรทัดที่ 4 สร้าง object ด้วยคำสั่ง Child obj = new Child(); บรรทัดที่ 5 เรียกใช้ obj.show(); โดยจะ เรียก method show() ที่ object ทำ การ override |
| 1 2 3 4 5 | <pre> ---exec:TestOverrideMethod1 show of Child ---operation:complete </pre> | |

| ตัวอย่าง 6.11 TestOverrideMethod2 | Note |
|-----------------------------------|------|
|-----------------------------------|------|

| | | |
|---|--|---|
| 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 | <pre> class ClassX { protected int m; protected int n; void f0{ System.out.println("In ClassX.f0."); m =22; } void g0{ System.out.println("In ClassX.g0."); n =44; } public String toString0{ return new String("{m="+m+",n="+n+"}"); } } </pre> | class classX ประกอบด้วย attribute: protected m; protected n; method: f(); g(); toString(); |
| 1 2 3 4 5 6 7 8 9 10 11 | <pre> class ClassY extends ClassX { private double n; void g0{ System.out.println("In ClassY.g0."); n =3.1415926535897932; } public String toString0{ return new String("{m="+m+",n="+n+"}"); } } </pre> | class ClassY extends ClassX ประกอบด้วย attribute: private n; method: g(); เป็น override method toString(); เป็น override method |
| 1 2 3 4 5 6 7 8 9 10 11 12 13 | <pre> class TestOverrideMethod2 { public static void main(String[] args){ ClassX x =new ClassX(); x.f0; x.g0; System.out.println("x =" +x); ClassY y =new ClassY(); y.f0; y.g0; System.out.println("y =" +y); } } </pre> | บรรทัดที่ 4 สร้าง object ด้วยคำสั่ง ClassX x = new ClassX(); บรรทัดที่ 5-6 เรียกใช้ x.f(), x.g(); โดยเรียก method f(), g() ของ object x บรรทัดที่ 8 สร้าง object ด้วยคำสั่ง ClassY y = new ClassY(); บรรทัดที่ 9 เรียกใช้ y.f(); มาจาก class แม่ บรรทัดที่ 10 เรียกใช้ y.g(); มาจาก override method g() ของ classY |
| 1 2 3 4 5 6 7 8 9 10 | <pre> ---exec:TestOverrideMethod2 In ClassX.f0. In ClassX.g0. x = {m=22,n=44} In ClassX.f0. In ClassY.g0. y = {m=22,n=3.141592653589793} ---operation:complete </pre> | |

6.7 Polymorphism Method

คำว่า polymorphism หมายถึง การที่คำ 1 คำ สามารถมีรูปหรือรูปร่างที่แตกต่างกันได้หลายแบบ แต่ในทาง programming นั้นหมายถึง การที่ตัวแปร 1 ตัว (single variable) สามารถใช้อ้างอิง object ที่แตกต่างกันได้หลายตัว

Polymorphism method เป็น method ชนิดหนึ่งที่สามารถทำงานได้หลายแบบ ขึ้นอยู่กับค่า Argument ที่ method ได้รับ ซึ่งส่วนใหญ่ Argument จะเป็น Object ที่ถูกสร้างขึ้น และ object ตัวนี้และที่เป็นตัวบอกว่า จะต้องไปทำงานที่ class ไດ

| ตัวอย่าง 6.12 TestOverrideMethod2 | | Note |
|---|--|--|
| 1 2 3 4 5 6 | <pre> class Mom { void show(){ System.out.println("LOVE"); } } </pre> | class Mom ประกอบด้วย method: show(); |
| 1 2 3 4 5 6 | <pre> class Child extends Mom { void show(){ System.out.println("BBK"); } } </pre> | class Child extends Mom ประกอบด้วย method: show(); เป็น override method |
| 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 | <pre> class TestPolymorphism1 { public static void main(String[] args) { Mom mom = new Mom(); Mom child = new Child(); System.out.println("From Mom"); run(mom); System.out.println("From Child"); run(child); } static void run(Mom obj) { obj.show(); } } </pre> | บรรทัดที่ 4 สร้าง object ด้วยคำสั่ง Mom mom = new Mom(); บรรทัดที่ 5 สร้าง object ด้วยคำสั่ง Mom child = new Child(); บรรทัดที่ 8 เรียกใช้ run(mom); โดย object mom มาจาก class Mom method run() ก็จะเรียก method show() ของ class Mom บรรทัดที่ 10 เรียกใช้ run(child); โดย object child มาจาก class Child method run() ก็จะเรียก method show ของ class Child |
| 1 2 3 4 5 6 7 8 | <pre> ---exec:TestPolymorphism1 From Mom LOVE From Child BBK ---operation:complete </pre> | |

6.8 แบบฝึกหัดท้ายบท

1. จงหาข้อผิดพลาดและแก้ไขข้อผิดพลาดของโปรแกรมต่อไปนี้

| ข้อ | Source code | ผลลัพธ์ |
|-----|---|---------|
| 1 | <pre> 1 class Circle 2 { 3 private double radius; 4 public Circle(double radius){ 5 radius=radius; 6 } 7 public double getRadius(){ 8 return radius; 9 } 10 public double findArea(){ 11 return radius*radius*Math.PI; 12 } 13 }</pre> | |
| | <pre> 1 class Cylinder extends Circle 2 { 3 private double length; 4 Cylinder(double radius, double length){ 5 Circle(radius); 6 length=length; 7 } 8 }</pre> | |
| 2 | <pre> 1 class A 2 { 3 int a; 4 A(){ 5 System.out.println("AAAAAAA"); 6 } 7 A(char c){ 8 System.out.println("CCCCCCCC"); 9 } 10 void print(){ 11 System.out.println(a); 12 } 13 } 14 15 class B extends A 16 { 17 int a; 18 B(){ 19 super('a'); 20 } 21 B(int x,int y){ 22 super.a=x; 23 this.a =y; 24 } 25 void print(){ 26 super.print(); 27 System.out.println(a); 28 } 29 }</pre> | |
| | <pre> 1 class Test6811 2 { 3 public static void main(String args[]){ 4 B b =new B(3); 5 b.print(); 6 }</pre> | |

| | | | |
|--|---|---|--|
| | 7 | } | |
| | | } | |

2. โปรแกรมต่อไปนี้ข้อใดเป็น Overriding ข้อใดเป็น Overloading พร้อมคำอธิบาย

| ข้อ | Source code | ผลลัพธ์ |
|-----|--|---------|
| 1 | <pre> 1 class Test6821 2 { 3 public static void main(String args[] { 4 A a =new A(); 5 a.p(10); 6 a.p(10.0); 7 } 8 } 9 10 class B 11 { 12 void p(double i){ 13 System.out.println(i*2); 14 } 15 } 16 class A extends B 17 { 18 void p(double i){ 19 System.out.println(i); 20 } 21 }</pre> | |
| 2 | <pre> 1 class Test6822 2 { 3 public static void main(String args[] { 4 A a =new A(); 5 a.p(10); 6 a.p(10.0); 7 } 8 } 9 10 class B 11 { 12 void p(double i){ 13 System.out.println(i*2); 14 } 15 } 16 class A extends B 17 { 18 void p(int i){ 19 System.out.println(i); 20 } 21 }</pre> | |

3. จงเขียนผลลัพธ์ที่ได้จากคลาสดังกล่าว พร้อมแสดงลำดับของการทำงานของคลาส อธิบายเหตุผลประกอบ

| ข้อ | Source code | ผลลัพธ์ |
|-----|--------------|---------|
| 1 | 1 class Test | |

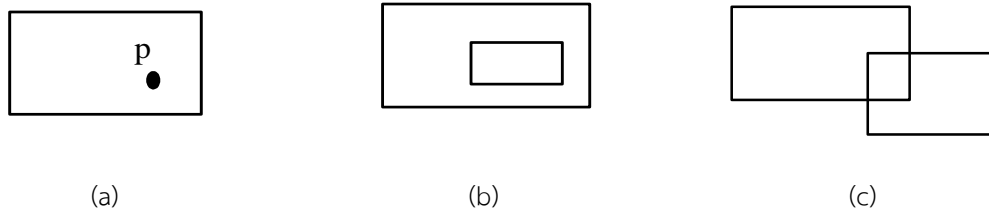
| | | | |
|---|--|--|--|
| | 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 | <pre> { public static void main(String[] args){ Animalia a =new Animalia (); Mammalia b =new Mammalia (); Taxonomy zoo =new Taxonomy (); System.out.println(zoo.getDetails(a)); System.out.println(zoo.getDetails(b)); } } class Taxonomy { String getDetails(Animalia a){ return a.m0+" "+a.p0; } String getDetails(Mammalia b){ return b.p0+" "+b.m0; } } class Animalia { String m0{ return "Tiger"; } String p0 { return "Lion"; } } class Mammalia extends Animalia { String m0{ return "Small Tiger"; } String p0{ return "Small Lion"; } } </pre> | |
| 2 | 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 | <pre> class SpecialActor { int power=1; void inc0{power++;} int get0{return power;} } class Spiderman extends SpecialActor { int power=2; void inc0{power++;} int get0{return power;} } class Test2 { public static void main(String[] args){ Spiderman b =new Spiderman(); SpecialActor a =b; a.inc0; b.inc0; System.out.println(a.power+", "+b.power); System.out.println(a.get0+", "+b.get0()); } } </pre> | |

| | | |
|---|--|--|
| 3 | <pre> 1 class Test3 2 { 3 public static void main(String[] args){ 4 Animal x=new Tiger(); 5 System.out.println("1.x.news is "+x.news); 6 System.out.println("2.((Tiger)x).news is "+((Tiger)x).news); 7 System.out.println("3.x.smile() is "+x.smile()); 8 System.out.println("4.((Tiger)x).smile() is "+((Tiger)x).smile()); 9 System.out.println("5.x.getNews() is "+x.getNews()); 10 System.out.println("6.x.getMessage() is "+x.getMessage()); 11 } 12 } 13 class Animal 14 { 15 String news="Animal's news"; 16 String message="Animal's message"; 17 static String smile(){ 18 return "smile from Animal"; 19 } 20 String getNews(){ 21 return news; 22 } 23 String getMessage(){ 24 return message; 25 } 26 } 27 28 29 class Tiger extends Animal 30 { 31 String news="Tiger's news"; 32 String message="Tiger's message"; 33 static String smile(){ 34 return "smile from Tiger"; 35 } 36 37 String getNews(){ 38 return news; 39 } 40 } </pre> | |
|---|--|--|

4. กำหนดให้คลาส MyRectangle2D ที่ประกอบไปด้วยรายละเอียดต่อไปนี้:

- ตัวแปรชนิด double ชื่อ x และ y ที่กำหนดจุดศูนย์กลางของสี่เหลี่ยม โดยมีเมธอด get และ set สำหรับกำหนดค่าและดึงค่า
- ตัวแปรชนิด double ชื่อ width and height โดยมีเมธอด get และ set สำหรับกำหนดค่าและดึงค่า
- no-arg constructor สำหรับการสร้างสี่เหลี่ยมแบบ default rectangle ที่มีการกำหนด (0, 0) สำหรับตัวแปร (x, y) และ 1 สำหรับ width and height.
- constructor สำหรับการสร้างสี่เหลี่ยมแบบกำหนดค่า x, y, width, and height.
- เมธอด getArea() ที่คืนพื้นที่ของสี่เหลี่ยม.
- เมธอด getPerimeter() ที่คืนเส้นรอบวงของสี่เหลี่ยม.

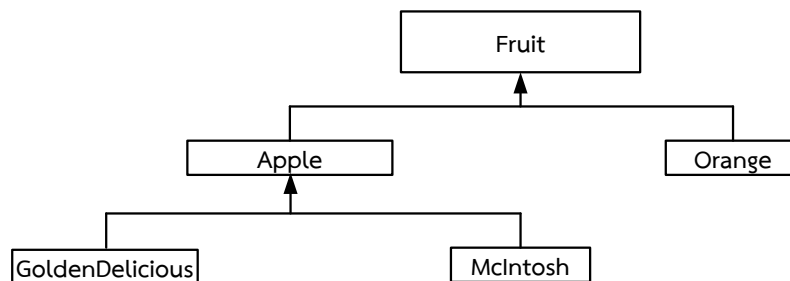
- เมธอด `contains(double x, double y)` ที่คืนค่า `true` ถ้ามีจุด `point (x, y)` อยู่ภายในสี่เหลี่ยม ดังรูปที่ 1(a).
- เมธอด `contains(MyRectangle2D r)` ที่คืนค่า `true` ถ้ามีสี่เหลี่ยม `rectangle` อยู่ภายในสี่เหลี่ยม ดังรูปที่ 1(b).
- เมธอด `overlaps(MyRectangle2D r)` ที่คืนค่า `true` ถ้ามีสี่เหลี่ยม `rectangle` ที่มีบางส่วนซ้อนทับกัน ดังรูปที่ 1(c).



รูปที่ 1

- (a) จุดอยู่ในสี่เหลี่ยม. (b) สี่เหลี่ยมอยู่ภายในสี่เหลี่ยมตัวอื่น. (c) สี่เหลี่ยมที่มีบางส่วนซ้อนทับกัน
- วาดคลาสไดอะแกรมของคลาส `MyRectangle2D`.
 - สร้างเมธอด `getArea()`, `getPerimeter()`, `contains(double x, double y)`, `contains(MyRectangle2D r)`, และ `overlaps(MyRectangle2D r)`.

5. กำหนดให้ `Fruit`, `Apple`, `Orange`, `GoldenDelicious`, and `McIntosh` มีลำดับสืบทอดดังรูป



กำหนด code ดังนี้

```

Fruit fruit = new GoldenDelicious();
Orange orange = new Orange();
  
```

จงตอบคำถามต่อไปนี้

| ข้อ | ผลลัพธ์ |
|---|---------|
| 1. Is fruit instanceof Orange? | |
| 2. Is fruit instanceof Apple? | |
| 3. Is fruit instanceof GoldenDelicious? | |
| 4. Is fruit instanceof Macintosh? | |
| 5. Is orange instanceof Orange? | |
| 6. Is orange instanceof Fruit? | |
| 7. Is orange instanceof Apple? | |

6. จงหาข้อผิดพลาดและแก้ไขข้อผิดพลาดของโปรแกรมต่อไปนี้

| ข้อ | Source code | ผลลัพธ์ |
|-----|--|---------|
| 1 | <pre> 1 class Test6851 2 { 3 public static void main(String[] args) 4 { 5 Object fruit =new Fruit(); 6 Object apple =new (Apple)fruit; 7 8 } 9 } 10 } 11 class Apple extends Fruit 12 { 13 14 } 15 16 class Fruit 17 { 18 19 }</pre> | |

โจทย์เพิ่มทักษะการเขียนโปรแกรม

จงเขียนโปรแกรมเพื่อหาผลลัพธ์ของปัญหาข้อ 7 - 13

7. Given an int array length 2, return true if it does not contain a 2 or 3.

no23({4, 5}) → true

no23({4, 2}) → false

no23({3, 5}) → false

8. Given an int array, return a new array with double the length where its last element is the same as the original array, and all the other elements are 0. The original array will be length 1 or more. Note: by default, a new int array contains all 0's.

makeLast({4, 5, 6}) → {0, 0, 0, 0, 0, 6}

makeLast({1, 2}) → {0, 0, 0, 2}

makeLast({3}) → {0, 3}

9. Given an int array, return true if the array contains 2 twice, or 3 twice. The array will be length 0, 1, or 2.

double23({2, 2}) → true

double23({3, 3}) → true

double23({2, 3}) → false

10. Given an int array length 3, if there is a 2 in the array immediately followed by a 3, set the 3 element to 0. Return the changed array.

`fix23({1, 2, 3}) → {1, 2, 0}`

`fix23({2, 3, 5}) → {2, 0, 5}`

`fix23({1, 2, 1}) → {1, 2, 1}`

11. Start with 2 int arrays, a and b, of any length. Return how many of the arrays have 1 as their first element.

`start1({1, 2, 3}, {1, 3}) → 2`

`start1({7, 2, 3}, {1}) → 1`

`start1({1, 2}, {}) → 1`

12. Start with 2 int arrays, a and b, each length 2. Consider the sum of the values in each array. Return the array which has the largest sum. In event of a tie, return a.

`biggerTwo({1, 2}, {3, 4}) → {3, 4}`

`biggerTwo({3, 4}, {1, 2}) → {3, 4}`

`biggerTwo({1, 1}, {1, 2}) → {1, 2}`

13. Given an array of ints of even length, return a new array length 2 containing the middle two elements from the original array. The original array will be length 2 or more.

`makeMiddle({1, 2, 3, 4}) → {2, 3}`

`makeMiddle({7, 1, 2, 3, 4, 9}) → {2, 3}`

`makeMiddle({1, 2}) → {1, 2}`

