การสืบทอดคุณสมบัติ
Inheritance

Benjamas Panyangam
Matinee Kiewkanya
Computer Science, CMU

Deriving a New Class from the Base Class

- การสืบทอดคุณสมบัติ (Inheritance) เป็นการสร้างคลาสใหม่ขึ้นมาจากคลาสพื้นฐาน
- คลาสใหม่นี้จะมีลักษณะพื้นฐานเหมือนกับคลาสเดิม แต่จะมีลักษณะพิเศษที่เพิ่มเติมขึ้นมา โดยจะมีความเฉพาะเจาะจงมากยิ่งขึ้น
- การสืบทอดในที่นี้จะเป็นการถ่ายทอดคุณลักษณะต่าง ๆ ของคลาสต้นสกุล (Ancestor Class) ให้แก่คลาสผู้สืบสกุลนั่นเอง (Descendant Class)

Deriving a New Class from the Base Class

- สำหรับการสืบทอดคลาสนั้น ชั้นฐานหรือคลาสพื้นฐาน (Base Class) หรืออาจเรียกว่าคลาสแม่ (Parent Class) ถือเป็นคลาสต้นสกุล หากเปรียบกับพันธุ์ไม้ก็ถือว่าเป็นพันธุ์ต้นแบบ (บางตำรา เรียกว่า Super Class)
- ส่วนชั้นใหม่หรือคลาสสืบทอด (Derived Class) หรืออาจเรียกว่าคลาสลูก (Child Class) เปรียบได้กับพันธุ์ไม้ใหม่ที่สืบสายพันธุ์มานั่นเอง (บางตำราเรียกว่า Sub Class)

การสืบทอดคุณสมบัติ (Inheritance)

คลาสรถยนต์ : มีล้อ มีเครื่องยนต์

Generalization

Specialization

คลาสรถบรรทุก :คุณสมบัติของรถยนต์ + บรรทุกของได้

Generalization

Specialization

คลาสรถเก๋ง :คุณสมบัติของรถยนต์ + มี 4 ประตู

Generalization

Specialization

คลาสรถสปอร์ต :คุณสมบัติของรถเก๋ง + วิ่งเร็ว

ตัวอย่างการสืบทอดคุณสมบัติในชีวิตประจำวัน

- คลาสสิ่งมีชีวิต กับ คลาสมนุษย์
- คลาสยานพาหนะ กับ คลาสรถยนต์
- คลาสแพทย์ กับ คลาสกุมารแพทย์
- คลาสสิ่งพิมพ์ กับ คลาสนิตยสาร
- คลาสวัสดุสำนักงาน กับ คลาสโต๊ะ

สิ่งที่คลาสแม่จะสืบทอดให้แก่คลาสลูก

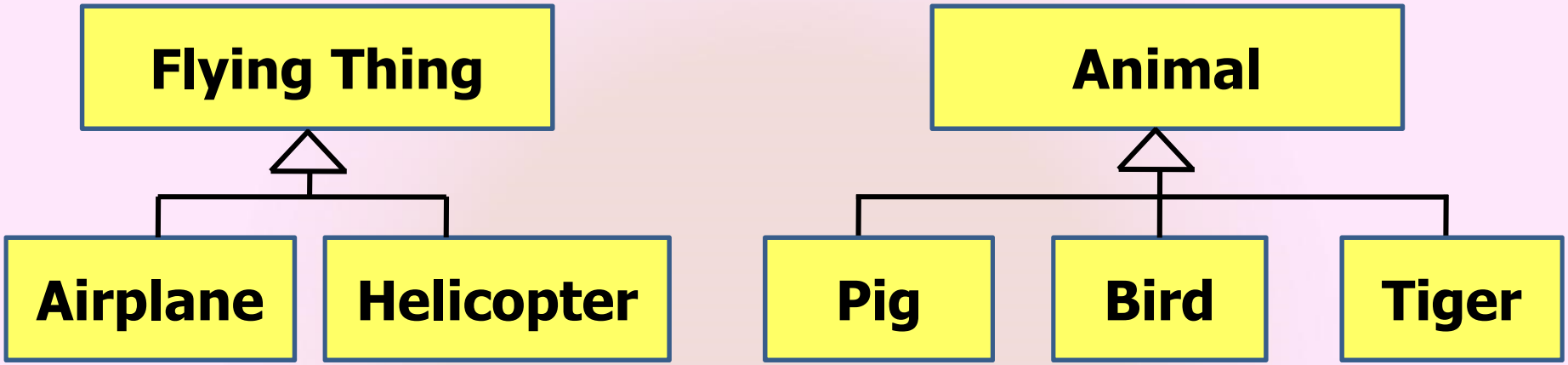
- **Member Data ที่มีชนิดเป็น public และ protected**
- **Member Method ที่มีชนิดเป็น public และ protected**
- **ความสัมพันธ์ (Relationship) ที่คลาสแม่มีต่อคลาสอื่น**

สิ่งที่คลาสแม่จะไม่สืบทอดให้แก่คลาสลูก

- **Member Data ที่มีชนิดเป็น private**
- **Member Method ที่มีชนิดเป็น private**
- **Constructor และ Destructor**

- ความสัมพันธ์ของคลาสที่มีการสืบทอดคุณสมบัติระหว่างคลาส เรียกว่า **Generalization Relationship**
- ในแผนภาพคลาสจะแทนความสัมพันธ์นี้โดยใช้ลูกศรสามเหลี่ยมโปร่งชี้ไปยังคลาสพื้นฐาน
- ความสัมพันธ์แบบ **Generalization** จะเป็นความสัมพันธ์ที่อยู่ในรูปแบบของ **"is-a relationship"** กล่าวคือคลาสสองคลาสจะมีความสัมพันธ์แบบ **Generalization** ได้ ประโยค "คลาสที่สืบทอด **is a** คลาสพื้นฐาน" จะต้องเป็นความจริง

Generalization (Is-a relationship)

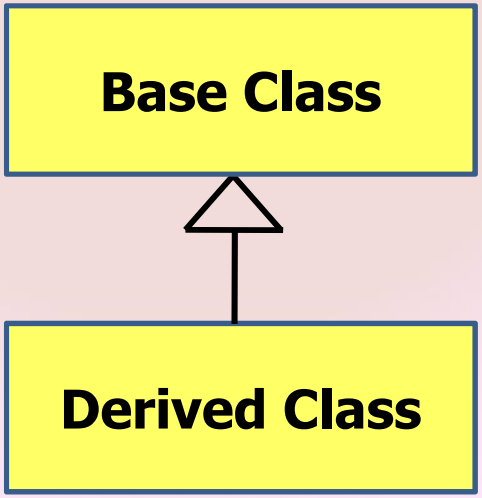


- An airplane is a flying thing.
- A helicopter is a flying thing.
- A pig is an animal.
- A bird is an animal.
- A tiger is an animal.

Types of Inheritance

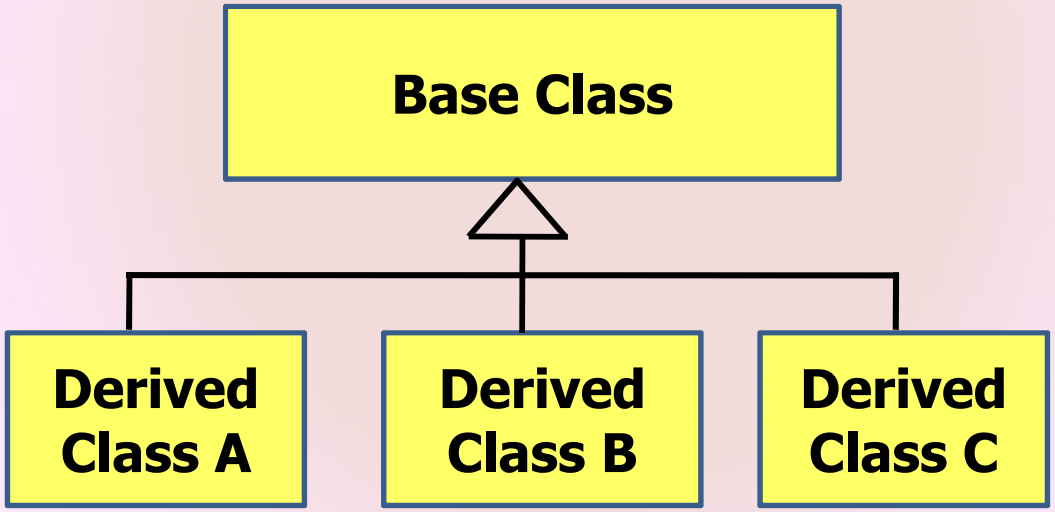
1) การสืบทอดคุณสมบัติแบบทางเดียว (Single Inheritance)

เป็นการสืบทอด โดยคลาสลูก 1 คลาส มีการสืบทอดมาจากคลาสแม่เพียง 1 คลาส และคลาสลูกไม่มีการสืบทอดต่อ



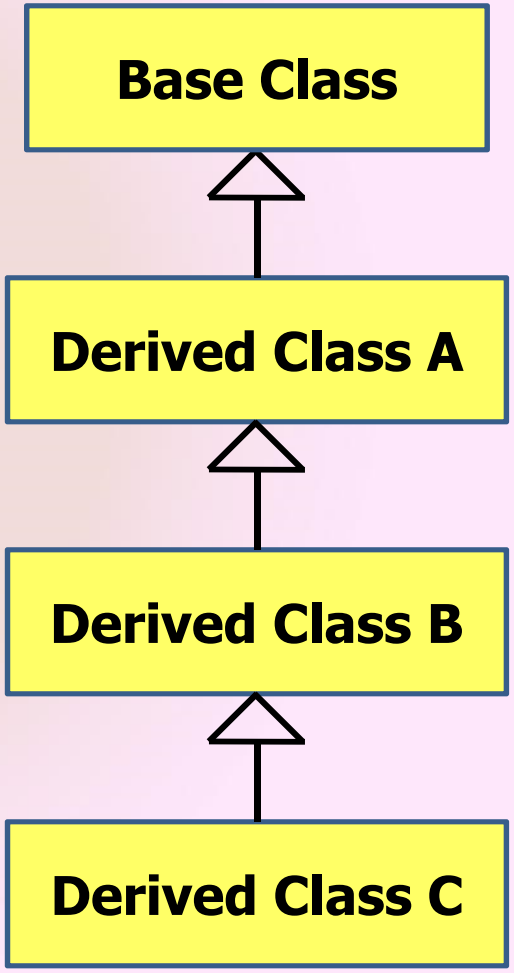
2) การสืบทอดคุณสมบัติแบบลำดับขั้น (Hierarchical Inheritance)

เป็นการสืบทอด โดยคลาสลูกหลายคลาสมีการสืบทอดมาจากคลาสแม่เพียง 1 คลาส



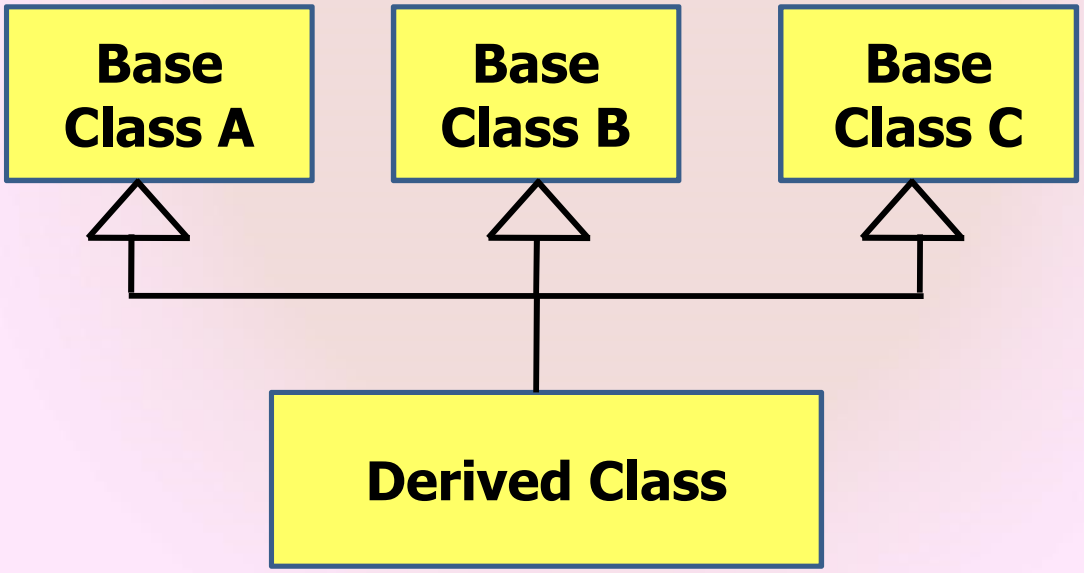
3) การสืบทอดคุณสมบัติแบบหลายระดับ (Multilevel Inheritance)

เป็นการสืบทอด โดยคลาสลูกมีการสืบทอดมาจากคลาสแม่ต่อเนื่องกันไปหลายระดับ



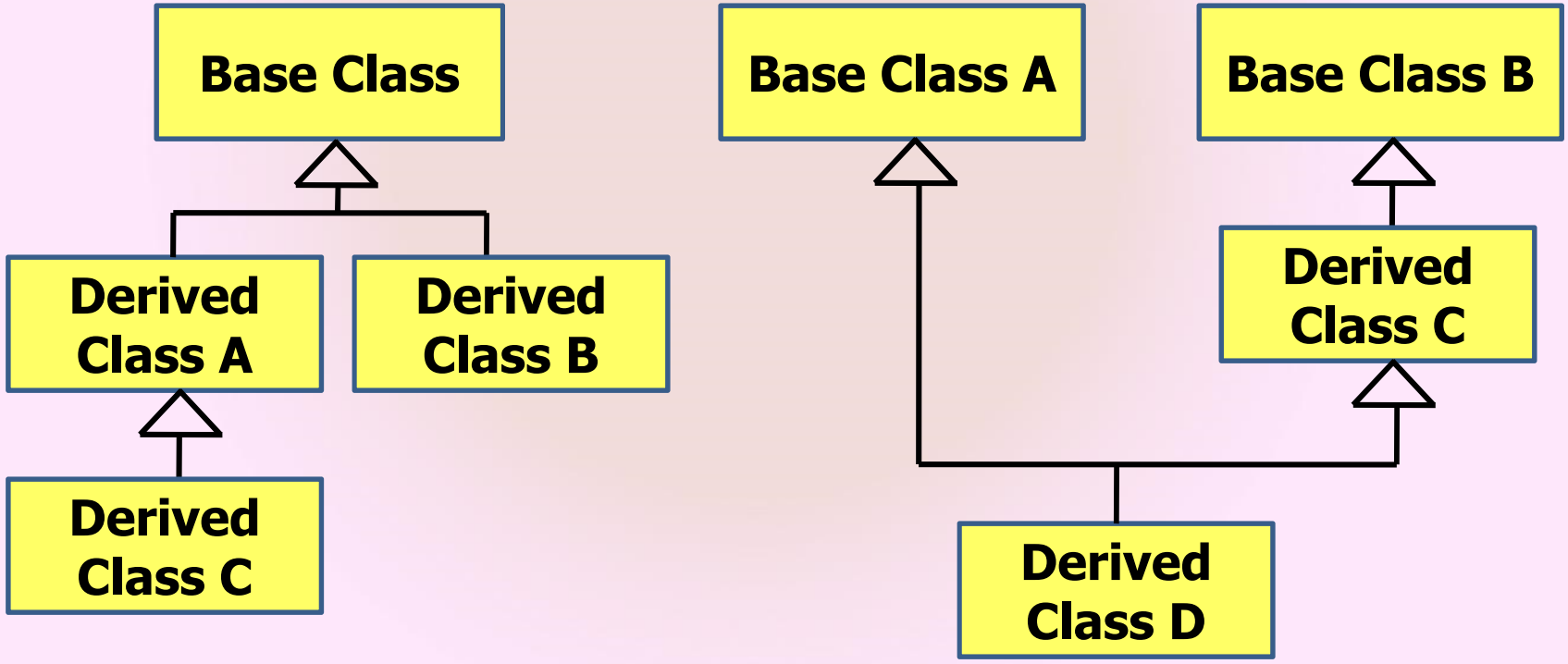
4) การสืบทอดคุณสมบัติแบบหลายทาง (Multiple Inheritance)

เป็นการสืบทอดโดยคลาสลูก 1 คลาส มีการสืบทอดมาจากคลาสแม่หลายคลาส



5) การสืบทอดคุณสมบัติแบบลูกผสม (Hybrid Inheritance)

เป็นการสืบทอด โดยผสมชนิดของการสืบทอด 4 ชนิดข้างต้นเข้าด้วยกัน ซึ่งอาจจะผสมชนิดหนึ่งเข้ากับอีกชนิดหนึ่งหรือเข้ากับอีกหลายชนิดก็ได้



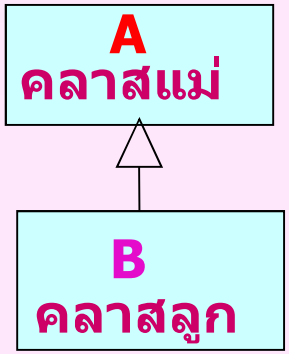
คำสั่งการสืบทอดคุณสมบัติ

รูปแบบการสืบทอดคุณสมบัติแบบทางเดียว

```
class ชื่อคลาสลูก  extends  ชื่อคลาสแม่ {
  ..........
}
```

- ใช้คีย์เวิร์ด extends เพื่อแสดงการสืบทอด
- สมมุติว่ามีคลาส A
  และต้องการสร้างคลาส B ที่สืบทอดมาจากคลาส A
  คำสั่งคือ

```
class B extends A { }
```



คำสั่งการสืบทอดคุณสมบัติ

ตัวอย่างเช่น

```
class GradStudent extends Student { }
class Car extends Vehicle { }
class Manager extends Employee { }
```

ตัวดัดแปร (Modifier)

- เป็นคำสำคัญที่ใช้ระบุการเข้าถึงสมาชิกของคลาส

	ใช้ได้ทั้งหมด	package เดียวกัน	ต่าง package กัน	ต่าง package กัน แต่เป็นคลาสแม่ คลาสลูกกัน	คลาสเดียวกัน
public	✓	✓	✓	✓	✓
protected	✗	✓	✗	✓	✓
package	✗	✓	✗	✗	✓
private	✗	✗	✗	✗	✓

private, package, protected, public
(เรียงลำดับจากความเข้มงวดมากถึงน้อย (อิสระที่สุด))

ตัวอย่างการสืบทอด

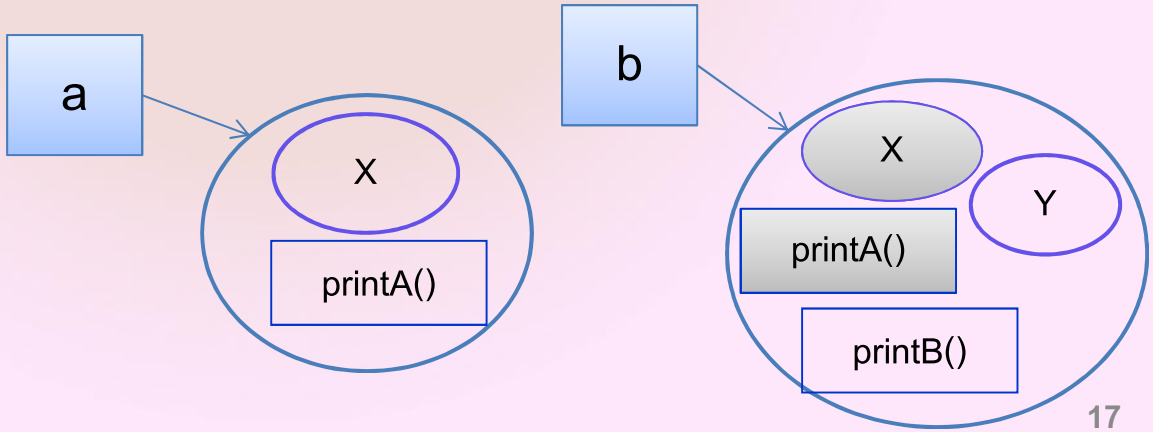
```
class A {
    protected int x;
    public void printA() {
        System.out.println('A'+" "+x);
    }
}
class B extends A {
    protected int y;
    public void printB() {
        System.out.println('B'+" "+y);
    }
}
```

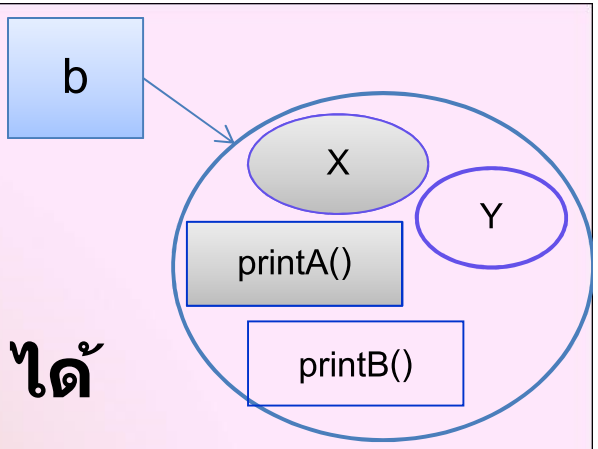
คลาส B ได้รับการสืบทอดสมาชิก (data และ method) จากคลาส A (Super class) ดังนี้

- public member
- protected member

(ยกเว้น private member)

```
A  a = new A();
B  b = new B();
```



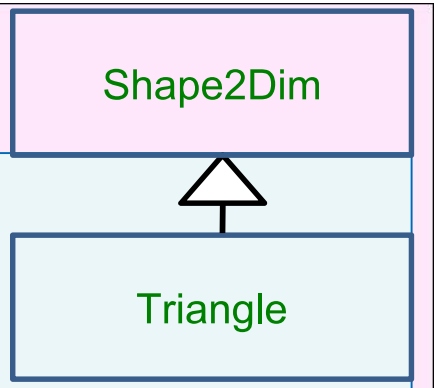


- ออบเจ็กต์ที่สร้างจากคลาส B สามารถเรียกใช้เมทอด printA() ของคลาส A ได้

```
public class JappExInherit{
    public static void main(String args[]) {
        A a = new A();
        a.x = 5;
        a.printA();

        B b = new B();
        b.x = 10;       b.printA();
        b.y = 20;       b.printB();
    }
}
```

ผลการทำงาน
A 5
A 10
B 20

ตัวอย่างโปรแกรมที่ 1




```
class Shape2Dim {
   private double width;
   private double height;
   public void print() {
         System.out.println("Width=" + width +", height="+height);
   }
}
class Triangle extends Shape2Dim {
   private String styleTri="Triangle";
   public double calArea() {
          return(0.5*width*height);
   }
   public  void printStyle() {
          System.out.println("Style=" + styleTri);
   }
}
```

คลาส Triangle ไม่สามารถเข้าถึง private data ของคลาส Shape2Dimได้

ตัวอย่างโปรแกรมที่ 2

(แก้ไขโปรแกรมที่ 1 ให้ทำงานได้ด้วยการเพิ่ม get, set methods)

```
class Shape2Dim {
   private double width;
   private double height;
   public void setWidth(double w) {
         width=w;
   }
   public  void setHeight(double h) {
         height=h;
   }
   protected double getWidth() { return width;}
   protected double getheight() { return height;}
   public void print() {
         System.out.println("Width=" + width +", height="+
         height);
   }
}
```

```
class Triangle extends Shape2Dim {
   private String styleTri="Triangle";

   public double calArea() {
          return(0.5* getWidth()* getHeight() );
   }
   public void printStyle() {
          System.out.println("Style=" + styleTri);
    }
}
```

คลาส Triangle สามารถเข้าถึง private data ของคลาส Shape2Dim ผ่านเมท็อดได้

```
public class JappExInherit{
   //method main
    public static void main(String[] args) {
          Shape2Dim s = new Shape2Dim();
          s.setWidth(10.0);
          s.setHeight(10.0);
          s.print();
          System.out.println(s.getWidth());
          System.out.println(s.getHeight());
```

ผลการทำงาน

```
Width=10.0, height =10.0
10.0
10.0
```

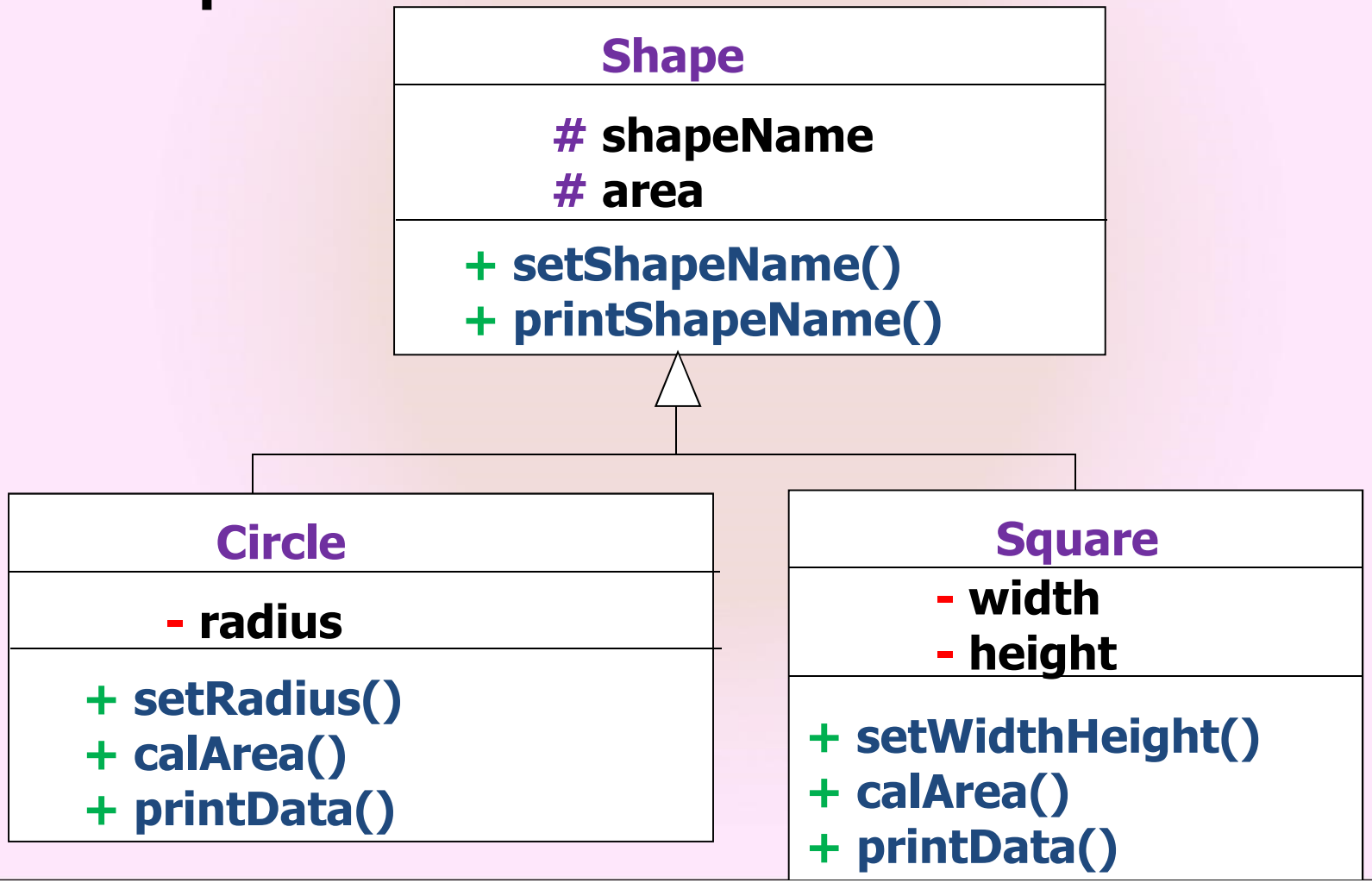
```
//method main (ต่อ)
     Triangle t = new Triangle();
      t.setWidth(10.0);
      t.setHeight(10.0);
      t.print();

      System.out.println(t.getWidth());
      System.out.println(t.getHeight());
      System.out.println(t.calArea());
      t.printStyle();
}
}
```

ผลการทำงาน

```
Width=10.0, height =10.0
10.0
10.0
50.0
Style=Triangle
```

ตัวอย่างโปรแกรมที่ 3

แสดงตัวอย่าง Hierarchical Inheritance
โปรแกรมเพื่อสืบทอดคลาส Shape ให้แก่คลาส Circle และคลาส Square



โปรแกรม

```
import java.util.Scanner;

class Shape {
    protected String shapeName;
    protected float area;
    public void setShapeName(){
        Scanner inVar = new Scanner(System.in);
        System.out.print("Enter shape name : ");
        shapeName = inVar.nextLine();
    }
    public void printShapeName(){
        System.out.println("You are " + shapeName);
    }
}
```

```
class Circle  extends Shape {  //Inherit
    private int radius;
    public  void setRadius() {
          Scanner inVar = new Scanner(System.in);
          System.out.print("Enter Radius : ");
          radius = inVar.nextInt();
    }
    public  void calArea(){
          area = 3.14f * radius * radius;
    }
    public  void printData() {
          System.out.print("Radius = " + radius);
          System.out.println(" Area= " + area);
    }
}
```

```
class Square extends Shape { //Inherit
    private int width;
    private int height;
    public void setWidthHeight() {
        Scanner inVar = new Scanner(System.in);
        System.out.print("Enter Width and Height ");
        width  = inVar.nextInt();
        height = inVar.nextInt();
    }
    public void calArea(){
        area = width * height;
    }
    public void printData(){
        System.out.print("Width = " + width);
        System.out.print(" Height= " + height);
        System.out.println(" Area= " + area);
    }
}
```

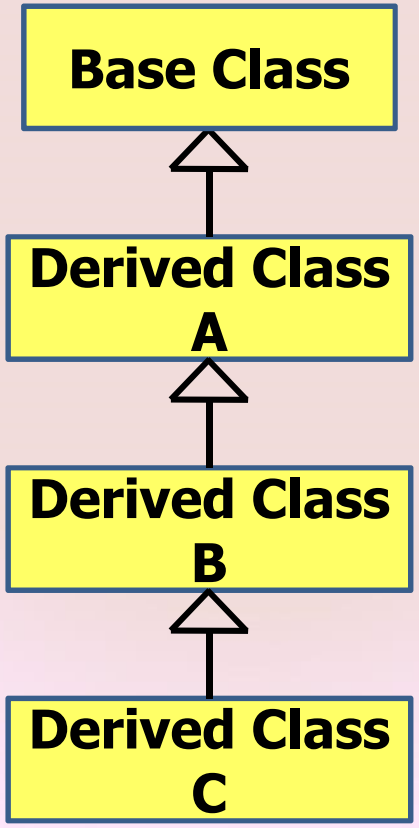
```
public class JappExInherit{
   public static void main(String[] args) {  //main
         Circle myCircle = new Circle();
         myCircle.setShapeName();
         myCircle.setRadius();
         myCircle.calArea();
         myCircle.printShapeName();
         myCircle.printData();

         System.out.print("\n\n");

         Square mySquare = new Square();
         mySquare.setShapeName();
         mySquare.setWidthHeight();
         mySquare.calArea();
         mySquare.printShapeName();
         mySquare.printData();
   }
}
```

ตัวอย่างผลการทำงาน

```
Enter shape name : Circle
Enter Radius : 4
You are Circle
Radius = 4 Area = 50.24

Enter shape name : Square
Enter Width and Height : 4 2
You are Square
Width = 4 Height = 2 Area = 8.0
```

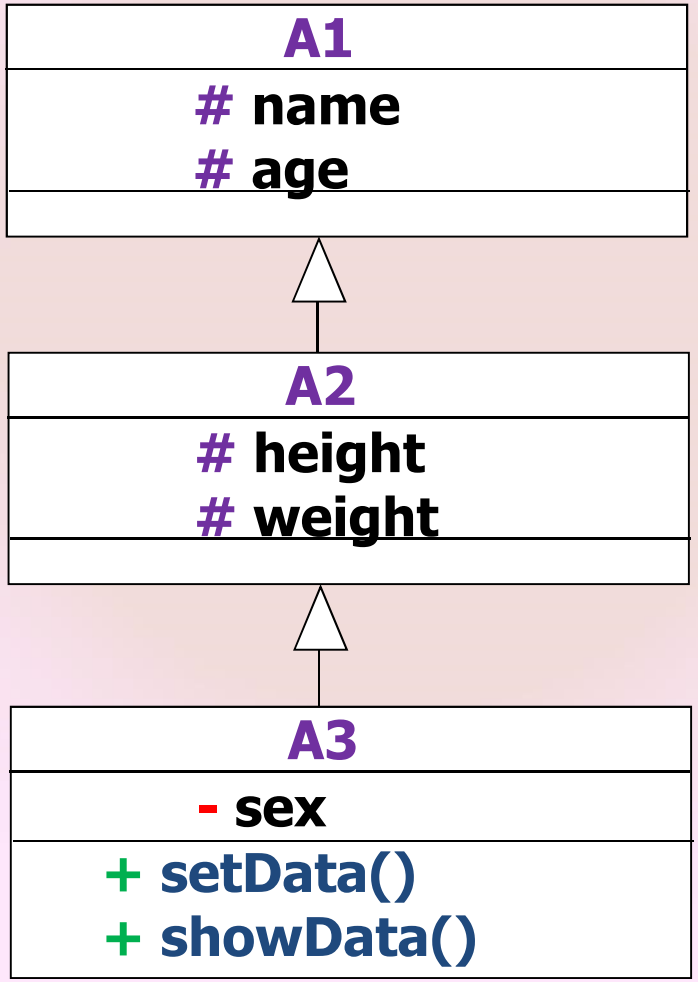
การสืบทอดคุณสมบัติแบบหลายระดับ (Multilevel Inheritance)

เป็นการสืบทอด โดยคลาสลูกมีการสืบทอดมาจากคลาสแม่ต่อเนื่องกันไปหลายระดับ



ตัวอย่างโปรแกรม

แสดงตัวอย่าง Multilevel Inheritance
โปรแกรมเพื่อสืบทอดคลาส A1 ให้แก่ A2 และสืบทอดคลาส A2 ให้แก่ A3



โปรแกรม

```
import java.util.Scanner;
class A1 {
    protected String name;
    protected int age;
}
class A2 extends A1 {
    protected  float height;
    protected  float weight;
};
class A3 extends A2 {
    private char sex;
    public void setData() {
        Scanner inVar = new Scanner(System.in);

        System.out.print("Name    : ");
        name = inVar.nextLine();
```

```
        System.out.print("Age     : ");
        age = inVar.nextInt();
        System.out.print("Sex     : ");
        sex = inVar.next().charAt(0);
        System.out.print("Height  : ");
        height = inVar.nextFloat();
        System.out.print("Weight  : ");
        weight = inVar.nextFloat();
    }
    public void showData()  {
        System.out.println("\nName   : " + name);
        System.out.println("Age    : " + age + " years");
        System.out.println("Sex    : " + sex);
        System.out.println("Height : " + height + " cm.");
        System.out.println("Weight : " + weight + " kg.");
    }
} // end class A3
```

```
public class JappExInherit{
    public static void main(String[] args) {  //main
        A3  objA = new A3();
        objA.setData( );
        objA.showData( );
}
```

ตัวอย่างผลการทำงาน

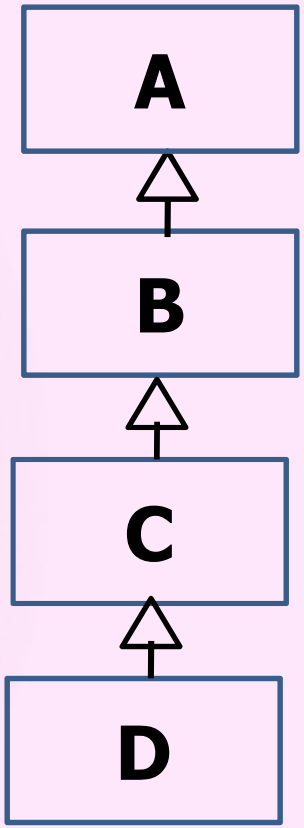
```
Name        : Matinee
Age         : 18
Sex         : F
Height      : 160
Weight      : 50

Name        : Matinee
Age         : 18 years
Sex         : F
Height      : 160.0 cm.
Weight      : 50.0 kg.
```

การห้ามไม่ให้มีการสืบทอดต่อ

- ถ้าไม่ต้องการให้ Class นั้นสืบทอดได้อีก ให้ใส่ keyword "final" ไว้หน้า Class

ตัวอย่าง

```
class A {
    protected int a = 1;
}
class B extends A {
    protected int b = 2;
}
final class C extends B {
    protected int c = 3;
}
class D extends C {
        ....
}
```

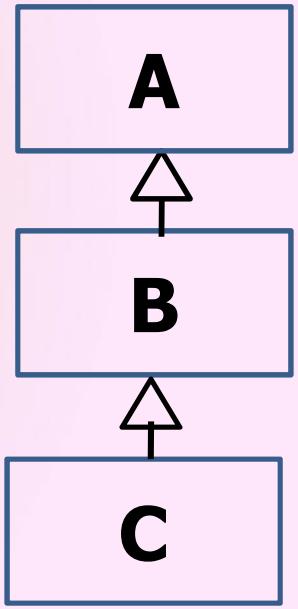


จะฟ้อง Error "cannot inherit from final C"

Constructor

- ในภาษา Java การสร้าง instance ของคลาสลูก จะทำให้ constructors ของคลาสบรรพบุรุษทั้งหมดถูกทำงาน

```
class A {
    protected int x =1;
    A(){ System.out.println("AAA");  }
}
class B extends A {
    protected int y=2;
    B(){  System.out.println("BBB");  }
}
class C extends B {
    private int z=2;
    C(){  System.out.println("CCC");  }
}
```



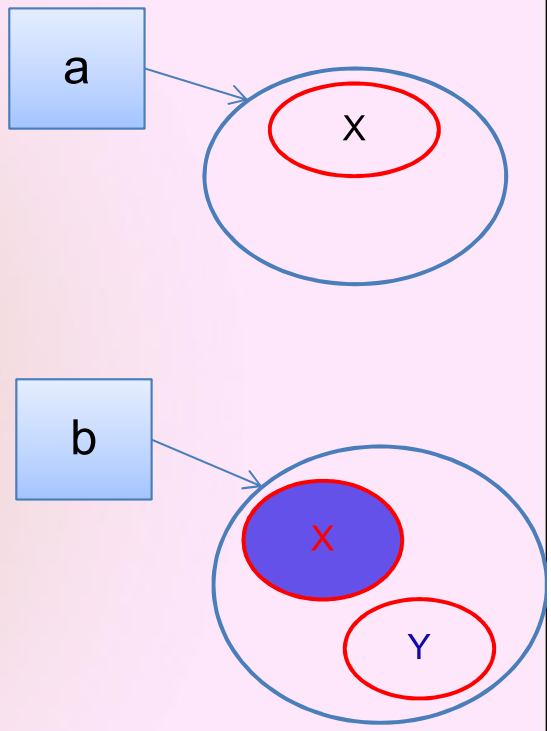
```
public class JappExInherit{
    public static void main(String[] args) {  //main
        A  a = new A();
        B  b = new B();
        C  c = new C();
}
```

ผลลัพธ์

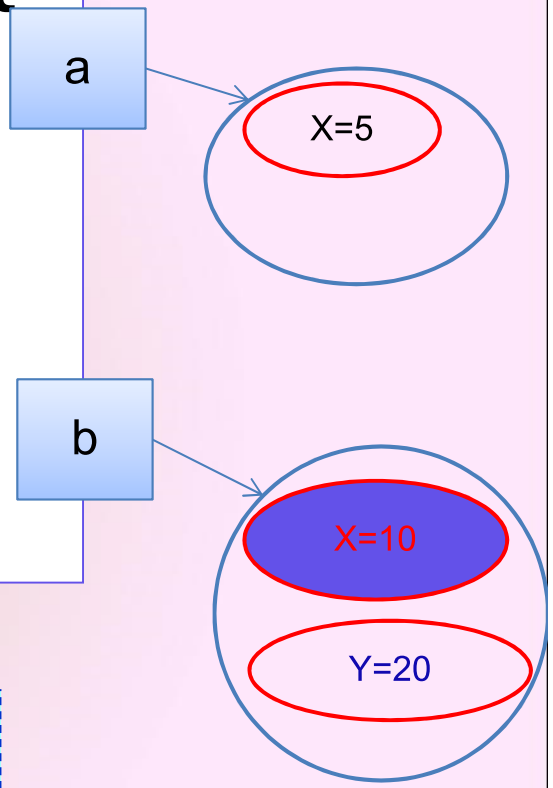
AAA
AAA
BBB
AAA
BBB
CCC

การเรียกใช้ constructor ของคลาสแม่ ด้วย keyword "super"

```
class A {
    protected int x ;
    A(int m) {  //constructor of A
          x = m;
     }
    public void printA() {
      System.out.println("in printA x=" + x);
    }
}
class B extends A {
     private int y;
     B( int m, int n) {//constructor of B
          super(m);       y = n;   }
     public void printB() {
        System.out.print("in printB x=" + x );
        System.out.println(" y=" + y );
     }
}
```



```
public class JappExInherit {
    public static void main(String args[]) {
      A a = new A(5);

      a.printA();
      B b = new B(10,20);
      b.printA();
      b.printB();
    }
}
```



ผลลัพธ์

```
in printA x=5
in printA x=10
in printB x=10 y=20
```

การเรียกใช้ constructor ของคลาสตนเอง ด้วย keyword "this"

```
class A {
    private int x;
    private int y;
    private int z;
    A(){x = 10;}        //constructor 1
    A(int y) {          //constructor 2
          this();
          this.y = y;
    }
    A(int y, int z) {   //constructor 3
          this(y);
          this.z = z;
     }
    public void printA() {
      System.out.println("x=" + x + " y=" + y + " z=" + z);
    }
}
```

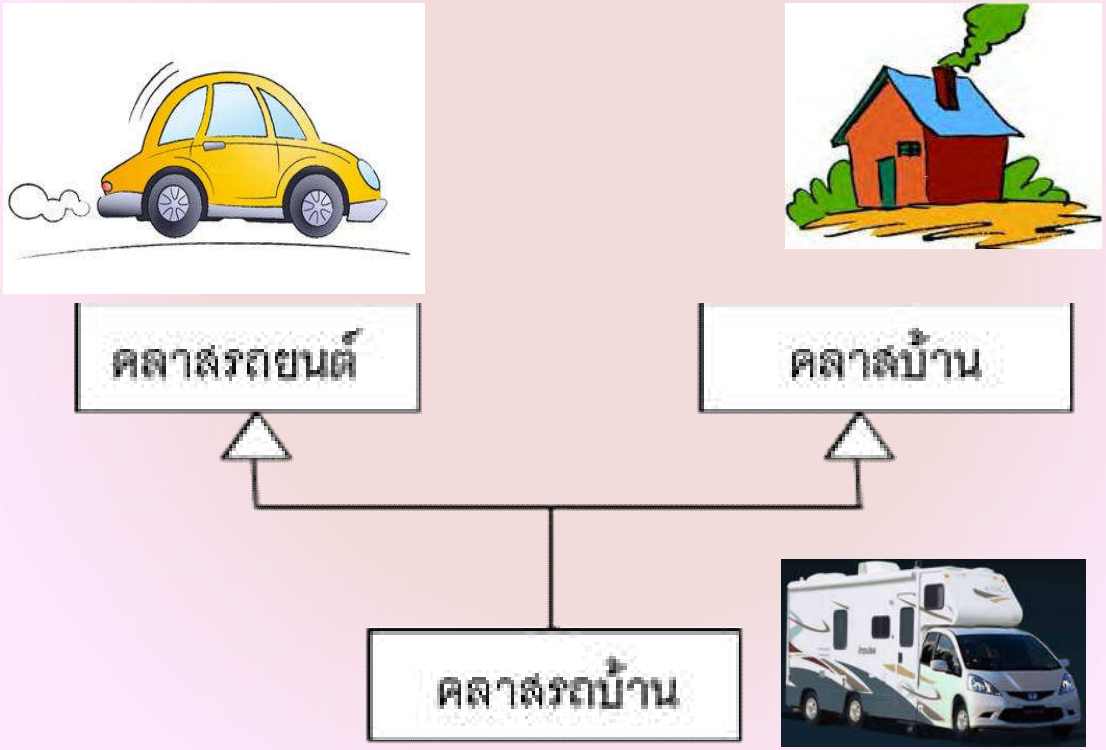
```
public class JappExInherit {
   public static void main(String args[]) {

        A a1 = new A();
        a1.printA();
        A a2 = new A(22);
        a2.printA();
        A a3 = new A(33,44);
        a3.printA();
    }
}
```

ผลลัพธ์

```
x=10 y=0 z=0
x=10 y=22 z=0
x=10 y=33 z=44
```

การสืบทอดคุณสมบัติแบบหลายทาง (Multiple Inheritance)

เป็นการสืบทอดโดยคลาสลูก 1 คลาส มีการสืบทอดมาจากคลาสแม่หลายคลาส



ตัวอย่างการสืบทอดคุณสมบัติแบบหลายทาง ในชีวิตประจำวัน

- คลาสแม่ : คลาสม้า คลาสลา
  คลาสลูก : คลาสล่อ
- คลาสแม่ : คลาสโต๊ะ คลาสเก้าอี้
  คลาสลูก : คลาสเก้าอี้เลคเชอร์
- คลาสแม่ : คลาสอาจารย์ คลาสแพทย์ คลาสทหาร
  คลาสลูก : คลาสอาจารย์แพทย์ทหาร

*** ภาษาจาวา ไม่สนับสนุนการเขียนโปรแกรมแบบ Multiple Inheritance**

รูปแบบการสืบทอดคุณสมบัติแบบหลายทางภาษา C++

```
class  derived_class_name  : inherit_modifier base_class_name1,
                             inherit_modifier base_class_name2,..

{   [modifier] :
                  [member_data]
    [modifier] :
                   [member_method(s)]
};
```

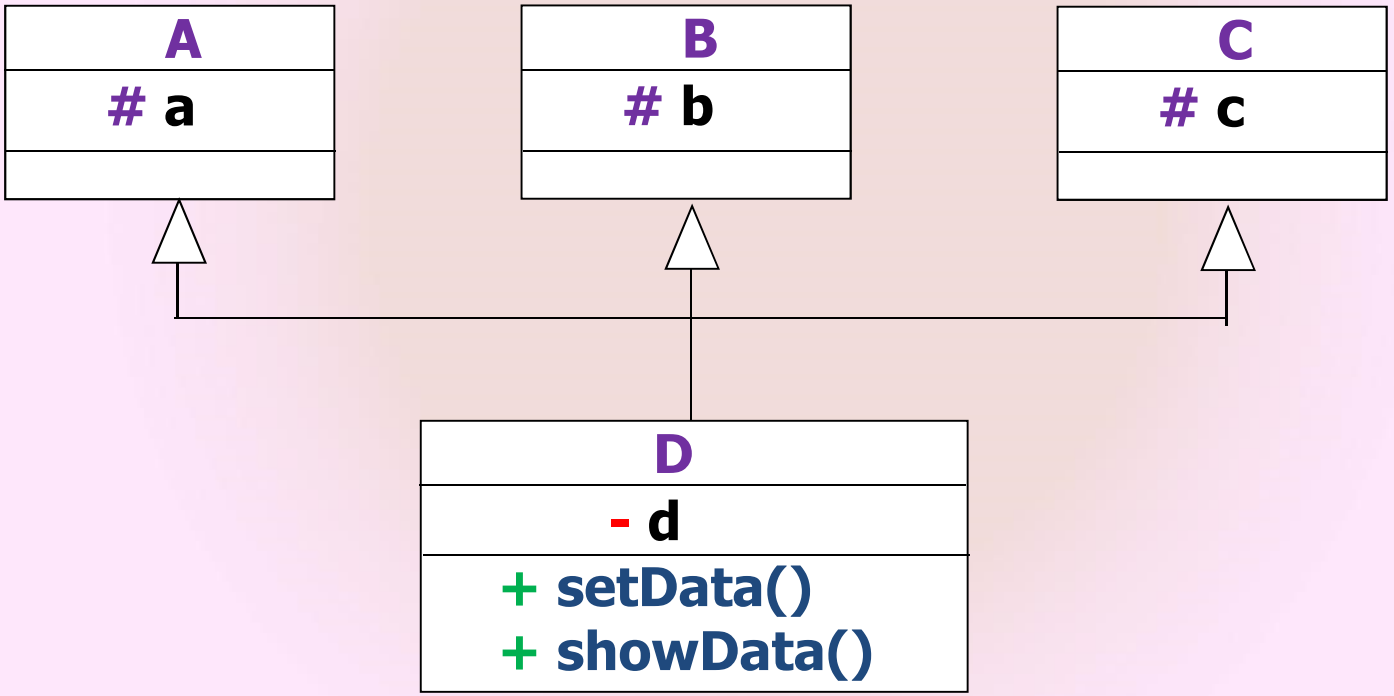
โดยที่

- **derived_class_name** คือ ชื่อคลาสลูก
- **base_calss_name** คือ ชื่อคลาสแม่
- **inherit_modifier** คือ คำสำคัญที่ใช้ระบุประเภทของการสืบทอดคุณสมบัติ ได้แก่ **private public** และ **protected** ซึ่งจะมีผลทำให้สมาชิกที่ถูกถ่ายทอดมีระดับของการเข้าถึงที่แตกต่างกัน
- **modifier** คือ คำสำคัญที่ใช้ระบุระดับการเข้าถึงสมาชิกของคลาส ได้แก่ **private public** และ **protected**
- **member_data** คือ การนิยามข้อมูลที่เป็นสมาชิกของคลาส
- **member_method** คือ การนิยามเมท็อดที่เป็นสมาชิกของคลาส

ตัวอย่างโปรแกรมภาษา C++

แสดงตัวอย่าง **Multiple Inheritance**

โปรแกรมเพื่อสืบทอดคลาส A คลาส B และคลาส C ให้แก่คลาส D



โปรแกรม

```
#include <iostream>
using namespace std;

class A {
        protected :  int a;
};

class B {
        protected :  int b;
};

class C {
        protected :  int c;
};
```

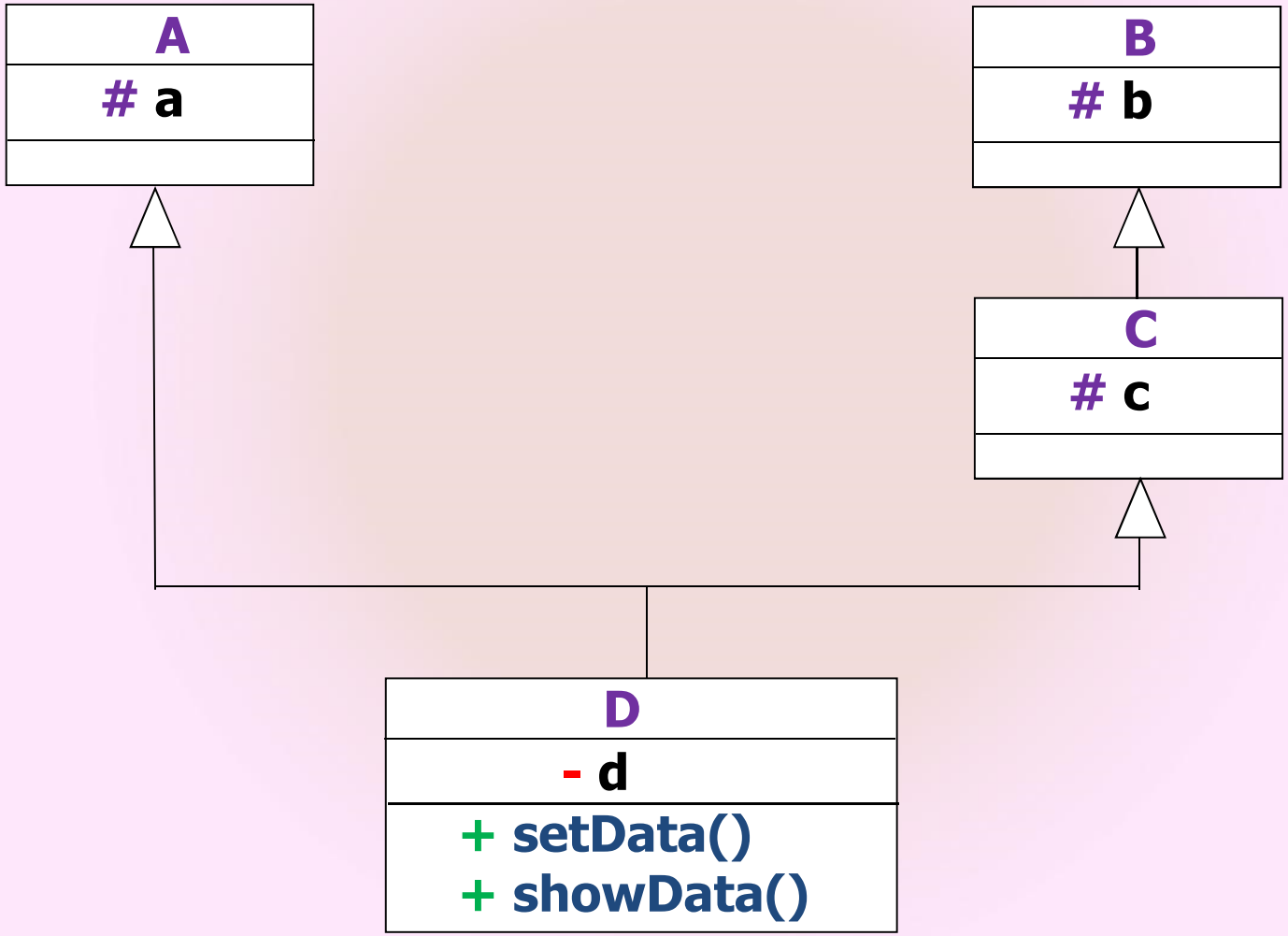
```
class D : public  A, public B, public C  { //public A,B,C
        private :
                int d;
        public :
                void setData( )  {
                        cout <<"Enter a b c d : " ;
                        cin>>a>>b>>c>>d;
                }

                void showData( )  {
                        cout <<"a="<<a <<" b=" << b ;
                        cout << " c="<<c <<" d="<<d;
                }
 };
int main( ) {
        D  objD;
        objD.setData( );
        objD.showData( );
}
```

ตัวอย่างผลการทำงาน

```
Enter a b c d : 10 20 30 40
a=10 b=20 c=30 d=40
```

ตัวอย่างโปรแกรมภาษา C++

แสดงตัวอย่าง Hybrid Inheritance
โปรแกรมเพื่อสืบทอดคลาสแบบลูกผสม



โปรแกรม

```
#include <iostream>
using namespace std;

class A {
        protected :  int a;
};

class B {
        protected :  int b;
};

class C:public B {
        protected :  int c;
};
```

```
class D : public  A, public C  { //public A,C
        private :
                int d;
        public :
                void setData( )  {
                        cout <<"Enter a b c d : " ;
                        cin>>a>>b>>c>>d;
                }

                void showData( )  {
                        cout <<"a="<<a <<" b=" << b ;
                        cout << " c="<<c <<" d="<<d;
                }
};
```

```
int main( ) {
        D  objD;
        objD.setData( );
        objD.showData( );
}
```

ตัวอย่างผลการทำงาน

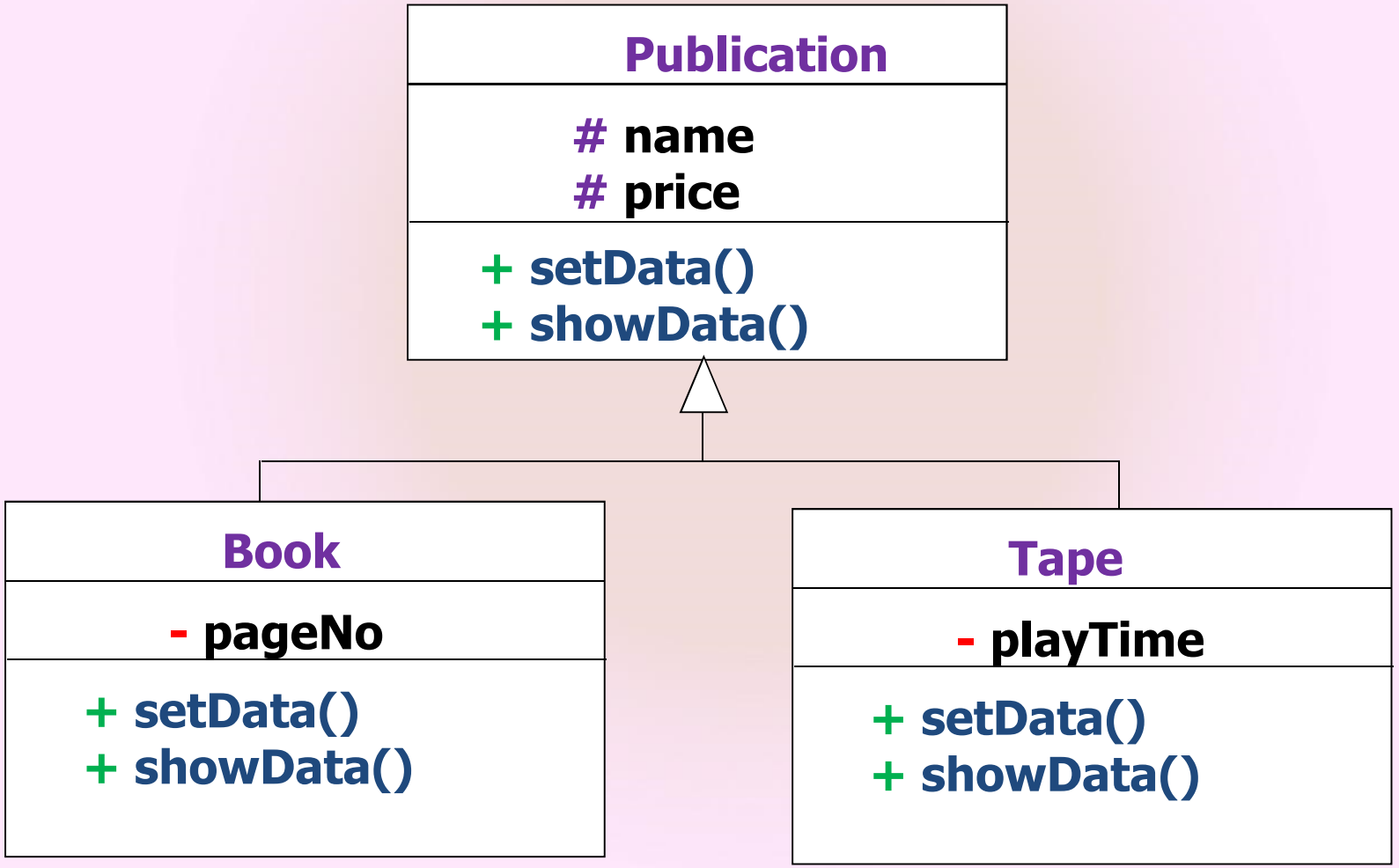
```
Enter a b c d : 10 20 30 40
a=10 b=20 c=30 d=40
```

การซ้อนทับของเมท็อด (Method Overriding)

Method Overriding คือ การที่เมท็อดในคลาสลูกมีชื่อเดียวกันกับเมท็อดในคลาสแม่ โดยเมท็อดของคลาสลูกจะไปซ้อนทับเมท็อดที่ควรจะสืบทอดมาจากคลาสแม่ ดังนั้นหากคลาสลูกอ้างถึงเมท็อดที่มีชื่อซ้ำกันนี้ จึงหมายถึงเมท็อดของคลาสลูกนั่นเอง

ตัวอย่างโปรแกรม

โปรแกรมเพื่อแสดงตัวอย่างการซ้อนทับของเมท็อด setData() และ showData()



โปรแกรม

```
class Publication {
    protected String name;
    protected float price;
    public void setData() {
        Scanner inVar = new Scanner(System.in);
        System.out.print("Name : ");
        name = inVar.nextLine();
        System.out.print("Price : ");
        price = inVar.nextFloat();
    }
    public void showData() {
         System.out.print("\n" + name);
         System.out.println(",Price: " + price);
    }
} //end class Publication
```

```
class Book extends Publication {
    private int pageNo;
    public void setData() {
        Scanner inVar = new Scanner(System.in);
        System.out.print("Book Name: ");
        name = inVar.nextLine();
        System.out.print("Book Price : ");
        price = inVar.nextFloat();
        System.out.print("Page number: ");
        pageNo = inVar.nextInt();
    }
    public void showData() {
        System.out.print("\n" + name);
        System.out.print(",Price: " + price);
        System.out.println(", " + pageNo + " Pages");
    }
} //end class Book
```

```
class Tape extends  Publication{
        private float playTime;
        public void setData() {
          Scanner inVar = new Scanner(System.in);
          System.out.print("Tape Name: ");
          name = inVar.nextLine();
          System.out.print("Tape Price : ");
          price = inVar.nextFloat();
          System.out.print("Play time (in minute): ");
          playTime = inVar.nextFloat();
        }
      public void showData() {
          System.out.print("\n" + name);
          System.out.print(",Price: " + price);
          System.out.println(",Play Time: " + playTime);
      }
} //end class Tape
```

```
public class JavaApp4 {
    public static void main(String[] args) {
        Book b = new Book();
        b.setData();
        b.showData();
        System.out.println();

        Tape t= new Tape();
        t.setData();
        t.showData();

    }
}
```

ถ้าคลาสลูกมีเมท็อดชื่อเหมือนกับเมท็อดในคลาสแม่

- พฤติกรรมของลูกจะลบล้าง (override) พฤติกรรมในคลาสแม่

ตัวอย่างผลการทำงาน

```
Book Name: Java
Book Price : 250
Page number: 100

Java,Price: 250.0, 100 Pages

Tape Name: Bird13
Tape Price : 199
Play time (in minute): 105

Bird13,Price: 199.0,Play Time: 105.0
```

การเรียกใช้สมาชิกของคลาสแม่ ด้วย keyword "super"

```
class A {
   protected int a;
   public void print() { System.out.println("in A, a="+a);}
}
class B extends A {
   private int a;
   B(int x, int y){super.a = x; this.a = y;}
   public void print() {
         super.print(); System.out.println("in B, a="+a);}
}
class InheritTest{
   public static void main(String args[]){
         B b = new B(1,2);
         b.print();
   }
}
```

ผลลัพธ์

in A, a=1

in B, a=2

Overriding

- เมท็อดในคลาสลูกที่มีชื่อเหมือนกับเมท็อดในคลาสแม่ จะเกิดการ override เมท็อดของคลาสแม่ เมื่อ
  - จำนวนและชนิดข้อมูลของ parameter เหมือนกันกับเมท็อดของคลาสแม่
  - ชนิดข้อมูลที่ส่งกลับ (return type) เหมือนกันกับของเมท็อดของคลาสแม่
- ถ้าจำนวนและชนิดข้อมูลของ parameter ไม่เหมือนกันกับของเมท็อดของคลาสแม่ จะไม่เกิดการ override

```
class A {
   protected int a;
   public void print()
         { System.out.println("hello from A");}
}

class B extends A {
   private int a;
   public void print(int x)  //ไม่เกิดการ override
         { System.out.println("hello from B");}
}
class InheritTest{
   public static void main(String args[]){
         B b = new B();
         b.print();
         b.print(1);
   }
```

ผลลัพธ์

hello from A

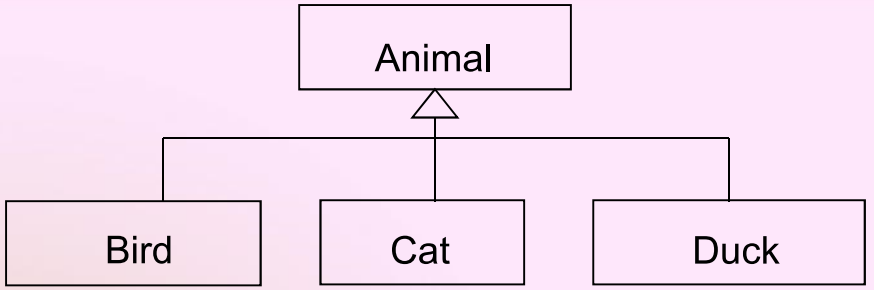
hello from B

Overriding

- **access modifier ของเมท็อดในคลาสลูก ที่จะทำการ override เมท็อดของคลาสแม่ จะต้องมีระดับการ access ไม่ต่ำกว่าเมท็อดของคลาสแม่**
- **เช่น ถ้าเมท็อดของคลาสแม่เป็น public เมท็อดของคลาสลูกจะไม่สามารถเปลี่ยนเป็น protected ได้**

ระดับการ access

private < package < protected < public




```
class Animal {
   protected void print() {
         System.out.println("in Animal Class");}
}
class Bird extends Animal {
   public void print() {
         System.out.println("in Bird Class");}
}
class Cat extends Animal  {
   protected void print() {
         System.out.println("in Cat Class");}
}
class Duck extends Animal  {//ทำไม่ได้
   private void print() {
         System.out.println("in Duck Class");}
}
```

```
class InheritTest{
   public static void main(String args[]){ //เมื่อตัด Duck ออก
         Animal a=new Animal();
         a.print();
         Bird b=new Bird();
         b.print();
         Cat c=new Cat();
         c.print();
}
```

ผลลัพธ์

in Animal Class

in Bird Class

in Cat Class

Shadow

- ถ้ากำหนดให้ **data** ในคลาสลูกมีชื่อเหมือนกับ **data** ในคลาสแม่ จะทำให้เกิดการบัง **(shadow)** กัน

```
class A {
      int x = 1;
}
class B extends A {
       float x = 2.0f;
}
class  exShadow{
   public static void main(String args[])
   {
       A a = new A();
      System.out.println(a.x);
       B b = new B();
       System.out.println(b.x);
   }
}
```

ผลลัพธ์

1

2.0