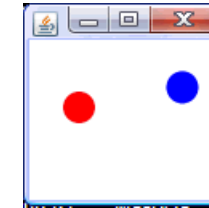


การสร้างคลาสใหม่ ด้วยการรับทอด

สร้างคลาสใหม่จากเก่า

- ทำสำเนา source code ของคลาสเดิม แล้วแก้ไข
- เขียนคลาสใหม่เก็บอ็อบเจกต์ของคลาสเดิม
- เขียนคลาสใหม่ให้ขยายความสามารถจากคลาสเดิม



วิธีที่ 1 : ลูกบอลสีดำ → ตั้งสีได้

```
import jlab.graphics.DWindow;

public class BallC {
    public double x, y;
    public double dx, dy;
    public double r;
    public int color;

    public BallC(double x1, double y1, double dx1, double dy1,
                 double r, int c) {
        x = x1; y = y1; dx = dx1; dy = dy1; r = r1; color = c;
    }
    public void draw(DWindow w) {
        w.fillEllipse(color, x, y, 2 * r, 2 * r);
    }
    public void move(DWindow w) {
        x = x + dx;
        y = y + dy;
        if (x + r > w.getWidth() || x < r) dx = -dx;
        if (y + r > w.getHeight() || y < r) dy = -dy;
    }
}
```

ทำสำเนาห้สตันฉบับของ
Ball แล้วแก้ไขได้ BallC

วิธีที่ 2 : ลูกบอลสีดำ → ตั้งสีได้

```
import jlab.graphics.DWindow;

public class BallC {
    public Ball ball;
    public int color;

    public BallC(double x1, double y1, double dx1, double dy1,
                 double r1, int c1) {
        ball = new Ball(x1, y1, dx1, dy1, r1);
        color = c1;
    }
    public void draw(DWindow w) {
        w.fillEllipse(color, ball.x, ball.y, 2*ball.r, 2*ball.r);
    }
    public void move(DWindow w) {
        ball.move(w);
    }
}
```

สร้างอ็อบเจกต์ของเดิมเก็บไว้ใช้งาน
เขียนเพิ่มเท่าที่จำเป็น

ถ้า Ball เปลี่ยนพฤติกรรมการ move
BallC ก็เปลี่ยนตามโดยอัตโนมัติ

วิธีที่ 3 : ลูกบอลสีดำ → ตั้งสีได้

```
import jlab.graphics.DWindow;
```

```
public class BallC extends Ball {
```

```
public Ball ball;
public int color;
```

```
public BallC(double x1, double y1, double dx1, double dy1,
             double r1, int c1) {
    super(x1, y1, dx1, dy1, r1);
    color = c1;
```

super(...) แทนตัวสร้างของ
ที่คลาสนี้ไป extends มา

```
public void draw(DWindow w) {
    w.fillEllipse(color, super.x, super.y, 2*super.r, 2*super.r);
}
```

```
public void move(DWindow w) {
    super.move(w);
}
```

super. แทนของที่
คลาสนี้ไป extends มา

ถ้าเหมือนของคลาสที่ extends มา
ทุกประการ ไม่ต้องเขียนก็ได้

เขียนคลาสใหม่ให้ "ขยาย"
ลักษณะจากคลาสเก่า

วิธีที่ 3 : ลูกบอลสีดำ → ตั้งสีได้

```
import jlab.graphics.DWindow;
```

```
public class BallC extends Ball {
```

```
public int color;
```

```
public BallC(double x1, double y1, double dx1, double dy1,
             double r1, int c1) {
    super(x1, y1, dx1, dy1, r1);
    color = c1;
```

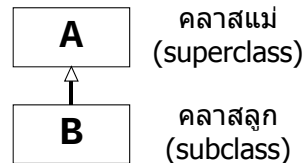
```
public void draw(DWindow w) {
    w.fillEllipse(color, super.x, super.y, 2*super.r, 2*super.r);
}
```

ถ้า Ball เปลี่ยนพฤติกรรม move
BallC ก็เปลี่ยนตามโดยอัตโนมัติ

เขียนคลาสใหม่ให้ "ขยาย"
ลักษณะจากคลาสเก่า

การรับทอด (Inheritance)

```
public class B extends A {
    ...
}
```



- B รับทอดลักษณะสมบัติจาก A
 - อ็อบเจกต์ของ B มีตัวแปรประจำอ็อบเจกต์ทุกตัวของ A
 - B ใช้ตัวแปรและเมทอด (ประจำคลาสและประจำอ็อบเจกต์) ทั้งหมดของ A ที่ไม่เป็น private ได้
 - B ไม่ได้ตัวสร้างของ A (แต่เรียกใช้ได้)
- B เพิ่มตัวแปรใหม่และเพิ่มเมทอดใหม่ได้
- B เขียนเมทอดใหม่แทนของที่ได้รับทอดได้

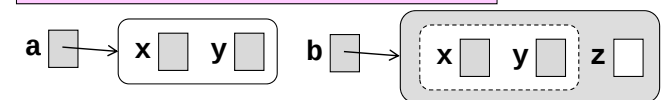
override

B extends A (แบบง่ายสุด)

```
public class A {
    public int x;
    private int y;
    public A() {}
    public A(int x1) {x = x1;}
    public void m1() {...}
    public int m2(double z) {...}
    private void m3() {...}
}
```

```
public class B extends A {
    ...
}
```

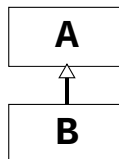
```
A a = new A();
B b = new B();
```



- อ็อบเจกต์ของ B มีตัวแปร x และ y (แต่ใช้ y ไม่ได้)
- เราเรียกเมทอด m1 และ m2 กับอ็อบเจกต์ของ B ได้
- คลาส B มีตัวสร้าง B(){} ที่ระบบเติมให้

B extends A

```
public class A {
    public int x;
    private int y;
    public A() {}
    public A(int x1) {x = x1;}
    public void m1() {...}
    public int m2(double z) {...}
    private void m3() {...}
}
```



```
public class B extends A {
    private int z;
    public B(int z1) {z = z1;}
    public void m1() {...}
    public void m4() {...}
}
```

m1 นี้ แทน
m1 ของแม่

เพิ่ม m4

- อ็อบเจกต์ของ B มีตัวแปร x, y, และ z (แต่ใช้ y ไม่ได้)
- เราเรียกเมทอด m1, m2, และ m4 กับอ็อบเจกต์ของ B ได้
- คลาส B มีตัวสร้าง B(int z1){...} ไม่มี B(){}

การเรียกตัวสร้างของแม่ : super(...)

- เราควรเขียนตัวสร้างเพื่อตั้งค่าเริ่มต้นของตัวแปรประจำอ็อบเจกต์ เมื่อมีการ new อ็อบเจกต์
- ลูกไม่มีตัวสร้างของแม่ แต่ลูกเรียกใช้ตัวสร้างของแม่ได้
- ใช้ super(...) เพื่อเรียกตัวสร้างของแม่
- ที่บรรทัดแรกในตัวสร้างของลูก ควรเรียกตัวสร้างของแม่ ถ้าไม่เรียก ระบบจะเติม super() ให้

```
public class A {
    private int x;

    public A(int x1) {
        x = x1;
    }
    ...
}
```

```
public class B extends A {
    private int z;

    public B(int z1) {
        super();
        z = z1;
    }
    ...
}
```

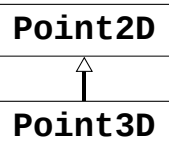
ผิด : แม่ไม่มี A()

super(0); ถูกต้อง

เขียน Point3D จาก Point2D

```
public class Point2D {
    public int x, y;

    public Point2D(int x1, int y1) {
        x = x1; y = y1;
    }
    ...
}
```



```
public class Point3D extends Point2D {
    public int z;

    public Point3D(int x1, int y1, int z1) {
        super(x1, y1);
        z = z1;
    }
    ...
}
```

public, private, protected

- **public** : ใคร ๆ ก็ใช้ได้
- **private** : ใช้ได้เฉพาะภายในคลาสตัวเอง
- **protected** : ให้คลาสลูก ๆ หลาน ๆ ใช้
- เขียนกำกับสมาชิกในคลาส

```
public class A {
    public int x;
    private int y;
    protected int z;

    protected A() {}
    public A(int x1) {x = x1;}

    public void m1() {...}
    private void m2() {...}
    protected void m3() {...}
}
```

เขียน CrookedDice จาก Dice

```
public class Dice {
    protected int value;
    public Dice() { roll(); }
    public int roll() {
        value = 1 + (int)(6 * Math.random());
        return value;
    }
    public int getValue() {
        return value;
    }
}

public class CrookedDice extends Dice {
    public CrookedDice() {}
    public int roll() {
        if (Math.random() < 0.9)
            value = 1;
        else
            value = 2 + (int)(5 * Math.random());
        return value;
    }
}
```

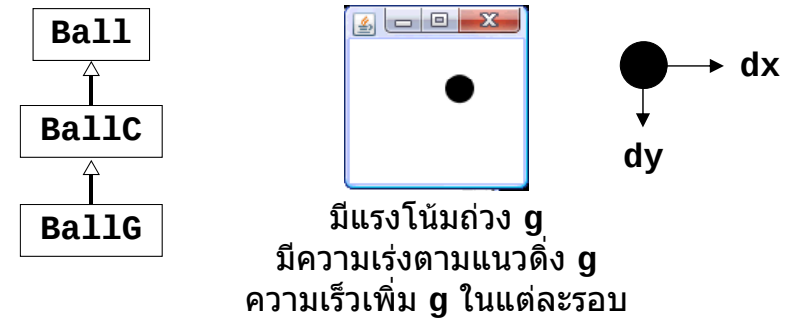
public ไม่ปลอดภัย
private ลูกใช้ไม่ได้
ต้องใช้ protected

ระบบเดิม
super();

roll นี้ แทน
roll ของแม่



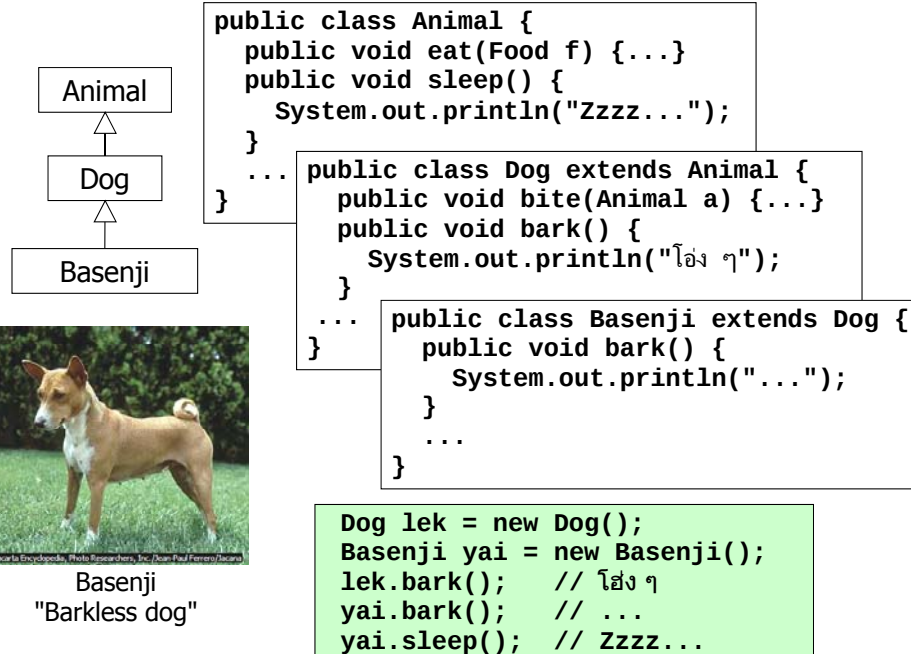
เขียน BallG จาก BallC



```
public class BallG extends BallC {
    private double gravity = 2;
    ...
    public void move(DWindow w) {
        dy += gravity;
        super.move(w);
        y = Math.min(y, w.getHeight() - r);
    }
}
```

ป้องกันลูกบอลจมน้ำ

Animal, Dog, Basenji



การสร้างคลาสใหม่แบบรับทอด

- เขียนคลาสใหม่ให้ extends จากคลาสเดิม
 - เขียน ตัวสร้าง ที่ต้องการให้บริการ
 - เพิ่ม ตัวแปรประจำอ็อบเจกต์ใหม่ที่เป็น
 - เพิ่ม เมทอดประจำอ็อบเจกต์ใหม่ ที่ superclass ไม่มี
 - แทน เมทอดประจำอ็อบเจกต์เดิมของ superclass ที่อยากเปลี่ยน

PiggyBank : กระปุกเก็บเหรียญ

```
public class PiggyBank {
    private int one, two, five, ten;

    public PiggyBank() {
        clear();
    }
    public void clear() {
        one = two = five = ten = 0;
    }
    public void add1(int coins) {one = one + coins;}
    public void add2(int coins) {two = two + coins;}
    public void add5(int coins) {five = five + coins;}
    public void add10(int coins) {ten = ten + coins;}
    public int getTotal() {
        return one + 2*two + 5*five + 10*ten;
    }
}
```



เขียน MoneyBox จาก PiggyBank

กล่องเก็บเงิน เก็บทั้งเหรียญและธนบัตร

```
public class MoneyBox extends PiggyBank {
    private int b20, b50, b100, b500, b1000;

    public MoneyBox() { super(); }
    public void clear() {
        super.clear();
        b20 = b50 = b100 = b500 = b1000 = 0;
    }
    public void add20(int b) { b20 = b20 + b; }
    public void add50(int b) { b50 = b50 + b; }
    public void add100(int b) { b100 = b100 + b; }
    public void add500(int b) { b500 = b500 + b; }
    public void add1000(int b) { b1000 = b1000 + b; }
    public int getTotal() {
        return super.getTotal() + 20*b20 + 50*b50 +
            100*b100 + 500*b500 + 1000*b1000;
    }
}
```



เขียน DWin จาก DWindow ให้มี fillCircle

```
import jlab.graphics.DWindow;

public class DWin extends DWindow {
    public DWin(double w, double h) {
        super(w, h);
    }
    public void fillCircle(double x, double y,
                           double r) {
        fillEllipse(x, y, 2 * r, 2 * r);
    }
    public void fillCircle(int color, double x,
                           double y, double r) {
        fillEllipse(color, x, y, 2 * r, 2 * r);
    }
}
```

ไม่ต้องมีรหัสต้นฉบับ
ของ DWindow

Inheritance Hierarchy

