



プログラミング演習2

クラスと継承

中村、小林、辻野、橋本

本日の流れ



- 13:30-14:00 宿題の解説
- 14:00-15:40 座学 + 演習 (10分休憩)
- 15:40-15:45 課題提示
- 15:45-16:45 課題に取り組む
- 16:45が課題の提出期限
- 16:45-17:00 課題の解説



- オブジェクト指向のさわりを学んだ
 - インスタンス化
 - `CircleClass circleObj = new CircleClass();`
 - インスタンス
 - `circleObj`
 - インスタンス変数
 - `circleObj.speed`
 - インスタンスメソッド
 - `circleObj.move()`
 - コンストラクタ
 - `CircleClass(){ 初期化処理 }`

動きは図形に任せたい



- 丸、正方形、三角形を定義
 - それぞれの座標は意識したくない
 - `circleObj.x`, `circleObj.y`, `squareObj.x`, `squareObj.y`,
`triangleObj.x`, `triangleObj.y`
 - 内部で適切に処理してもらう
 - それぞれの速度も意識したくない
 - `circleObj.vx`, `circleObj.vy`, `squareObj.vx`, `squareObj.vy`,
`triangleObj.vx`, `triangleObj.vy`
 - 描画はシンプルにしたい
 - `circleObj.draw()`, `squareObj.draw()`, `triangleObj.draw()`
 - 移動もシンプルにしたい
 - `circleObj.move()`, `squareObj.move()`, `triangleObj.move()`

すべてを **XXXX** . **機能** という形に！

オブジェクト指向（クラス）

宿題2-1: hw_boundA113



- CircleClassを改良し、正方形が動き回る SquareClass、xが動き回る CrossClass、△が動き回る TriangleClassを作成せよ
- またこれを利用して10個の○と、5個の正方形と、5個のxと、3個の△が画面内を動き回るプログラムを作成せよ
 - ただし、その速度はx、y方向それぞれ-5以上5未満の実数値とせよ
 - また、○と□は端で跳ね返り、xと△は跳ね返らずに反対側から出てくるようにせよ

Circle/Square/Cross



- 3つのクラスを作って、3つのオブジェクトをそれぞれ描画
 - CircleClass
 - SquareClass
 - CrossClass

CircleClass クラス



```
class CircleClass
{
    float x;
    float y;
    float vx;
    float vy;

    CircleClass() {
        x = random(width);
        y = random(height);
        vx = random(-5, 5);
        vy = random(-5, 5);
    }

    void display() {
        fill(255, 0, 0);
        circle(x, y, 30);
    }
}
```

```
void move() {
    x += vx;
    y += vy;
    if(x > width) {
        x = width * 2 - x;
        vx = -vx;
    }
    if(x < 0) {
        x = -x;
        vx = -vx;
    }
    if(y > height) {
        y = height * 2 - y;
        vy = -vy;
    }
    if(y < 0) {
        y = -y;
        vy = -vy;
    }
}
```

SquareClass



```
class SquareClass
{
    float x;
    float y;
    float vx;
    float vy;

    SquareClass {
        x = random(width);
        y = random(height);
        vx = random(-5, 5);
        vy = random(-5, 5);
    }

    void display() {
        fill(0, 0, 255);
        rect(x-15, y-15, 30, 30);
    }
}
```

```
void move() {
    x += vx;
    y += vy;
    if(x > width) {
        x = width * 2 - x;
        vx = -vx;
    }
    if(x < 0) {
        x = -x;
        vx = -vx;
    }
    if(y > height) {
        y = height * 2 - y;
        vy = -vy;
    }
    if(y < 0) {
        y = -y;
        vy = -vy;
    }
}
```

CrossClass



```
class CrossClass
{
    float x;
    float y;
    float vx;
    float vy;

    CrossClass() {
        x = random(width);
        y = random(height);
        vx = random(-5, 5);
        vy = random(-5, 5);
    }

    void display() {
        stroke(0, 0, 0);
        line(x-15, y-15, x+15, y+15);
        line(x-15, y+15, x+15, y-15);
    }
}
```

```
void move() {
    x += vx;
    y += vy;
    x = (x + width) % width;
    y = (y + height) % height;
}
}
```

Circle/Square/Cross



```
class CircleClass
{
  float x;
  float y;
  float vx;
  float vy;
```

```
CircleClass() {
  x = random(width);
  y = random(height);
  vx = random(-5, 5);
  vy = random(-5, 5);
}
```

```
void display() {
  fill(255, 0, 0);
  circle(x, y, 30);
}
```

```
class SquareClass
{
  float x;
  float y;
  float vx;
  float vy;
```

```
SquareClass() {
  x = random(width);
  y = random(height);
  vx = random(-5, 5);
  vy = random(-5, 5);
}
```

```
void display() {
  fill(0, 0, 255);
  rect(x-15, y-15, 30,
  30);
}
```

```
class CrossClass
{
  float x;
  float y;
  float vx;
  float vy;
```

```
CrossClass() {
  x = random(width);
  y = random(height);
  vx = random(-5, 5);
  vy = random(-5, 5);
}
```

```
void display() {
  stroke(0, 0, 0);
  line(x-15, y-15, x+15,
  y-15);
  line(x-15, y+15, x+15,
  y+15);
}
```

Circle/Square/Cross



class CircleClass	class SquareClass	class CrossClass
<pre>{ float x; float y; float vx; float vy; }</pre>	<pre>{ float x; float y; float vx; float vy; }</pre>	<pre>{ float x; float y; float vx; float vy; }</pre>
<pre>CircleClass() { x = random(width); y = random(height); vx = random(-5, 5); vy = random(-5, 5); }</pre>	<pre>SquareClass() { x = random(width); y = random(height); vx = random(-5, 5); vy = random(-5, 5); }</pre>	<pre>CrossClass() { x = random(width); y = random(height); vx = random(-5, 5); vy = random(-5, 5); }</pre>
<pre>void display() { fill(255, 0, 0); circle(x, y, 30); }</pre>	<pre>void display() { fill(0, 0, 255); rect(x-15, y-15, 30, }</pre>	<pre>void display() { stroke(0, 0, 0); line(x-15, y-15, x+15, line(x-15, y+15, x+15, }</pre>

Circle/Square/Cross



```
void move() {  
    x += vx;  
    y += vy;  
    if(x > width) {  
        x = width * 2 - x;  
        vx = -vx;  
    }  
    if(x < 0) {  
        x = -x;  
        vx = -vx;  
    }  
    if(y > height) {  
        y = height * 2 - y;  
        vy = -vy;  
    }  
    if(y < 0) {  
        y = -y;  
        vy = -vy;  
    }  
}
```

```
void move() {  
    x += vx;  
    y += vy;  
    if(x > width) {  
        x = width * 2 - x;  
        vx = -vx;  
    }  
    if(x < 0) {  
        x = -x;  
        vx = -vx;  
    }  
    if(y > height) {  
        y = height * 2 - y;  
        vy = -vy;  
    }  
    if(y < 0) {  
        y = -y;  
        vy = -vy;  
    }  
}
```

```
void move() {  
    x += vx;  
    y += vy;  
    x = (x + width) % width;  
    y = (y + height) % height;  
}
```

Circle/Square/Cross



```
void move() {  
    x += vx;  
    y += vy;  
  
    if(x > width) {  
        x = width * 2 - x;  
        vx = -vx;  
    }  
    if(x < 0) {  
        x = -x;  
        vx = -vx;  
    }  
    if(y > height) {  
        y = height * 2 - y;  
        vy = -vy;  
    }  
    if(y < 0) {  
        y = -y;  
        vy = -vy;  
    }  
}
```

```
void move() {  
    x += vx;  
    y += vy;  
  
    if(x > width) {  
        x = width * 2 - x;  
        vx = -vx;  
    }  
    if(x < 0) {  
        x = -x;  
        vx = -vx;  
    }  
    if(y > height) {  
        y = height * 2 - y;  
        vy = -vy;  
    }  
    if(y < 0) {  
        y = -y;  
        vy = -vy;  
    }  
}
```

```
void move() {  
    x += vx;  
    y += vy;  
  
    x = (x + width) % width;  
    y = (y + height) % height;  
}
```

Circle/Square/Cross



- CircleClassと, SquareClassと, CrossClassは似ている

クラス名	CircleClass	SquareClass	CrossClass	
インスタンス変数	x, y, vx, vy	x, y, vx, vy	x, y, vx, vy	一致
コンストラクタ	CircleClass() 座標速度初期化	SquareClass() 座標速度初期化	CrossClass() 座標速度初期化	名は違うが 内部は一致
void move()	はねかえる	はねかえる	はねかえらない	一部一致
void display()	円を描く	正方形を描く	xを描く	違う

無駄じゃね？
もっと手軽にできないの？

どう無駄をなくす？



- そもそもCircleClassというのがダメなのでは？
 - ObjectClassというものを作って、objectTypeなどの変数を用意して、displayの時に表示するものを切り替えては？

```
class ObjectClass {  
    float x;  
    float y;  
    float vx;  
    float vy;  
    int objectType;  
  
    void display(){  
        if(objectType == 0){  
            circle(x, y, 30);  
        } else if(objectType == 1){  
            rect(x-15, y-15, 30, 30);  
        }  
    }  
}
```

これも一つの方法だが
跳ね返りでも条件が必要で
複雑なものだと厳しくなる

そこで継承！



- 大辞林 第三版

1. 先の人の身分・権利・義務・財産などを受け継ぐこと。「王位を－する」
2. インヘリタンス→（オブジェクト指向プログラミングにおいて、クラス間でデータの共有を行う機構。新しく定義するクラスを既存のクラスの下位クラスとして記述し、上位クラスより属性やメソッドを引き継ぐ仕組みをいう。上位クラスに対する差分のみを記述するだけで新しいクラスを定義することが可能となる。）

**スーパークラス（親クラス）を作って
そこから機能や変数を引き継ぐ！**



- ObjectBaseというクラスを作る
 - その機能を, CircleClassやSquareClass, CrossClassに提供する
 - そのためには共通項を探る

クラス名	ObjectBase	CircleClass	SquareClass	CrossClass
インスタンス変数	x, y, vx, vy	x, y, vx, vy	x, y, vx, vy	x, y, vx, vy
コンストラクタ	ObjectBase() 座標速度初期化	CircleClass() 座標速度初期化	SquareClass() 座標速度初期化	CrossClass() 座標速度初期化
void move()	はねかえる	はねかえる	はねかえる	はねかえない
void display()	点を描く	円を描く	正方形を描く	xを描く

ObjectBaseクラス



```
class ObjectBase
{
    float x;
    float y;
    float vx;
    float vy;

    ObjectBase() {
        x = random(width);
        y = random(height);
        vx = random(-5, 5);
        vy = random(-5, 5);
    }

    void display(){
        point(x, y);
    }
}
```

display() は点を描くに
move() は跳ね返るに

```
void move() {
    x += vx;
    y += vy;
    if(x > width) {
        x = width * 2 - x;
        vx = -vx;
    }
    if(x < 0) {
        x = -x;
        vx = -vx;
    }
    if(y > height) {
        y = height * 2 - y;
        vy = -vy;
    }
    if(y < 0) {
        y = -y;
        vy = -vy;
    }
}
```

ObjectBaseと ○ □ X



```
class ObjectBase
{
    float x;
    float y;
    float vx;
    float vy;

    ObjectBase() {
        x = random(width);
        y = random(height);
        vx = random(-5, 5);
        vy = random(-5, 5);
    }

    void display(){
        point(x, y);
    }
}
```

```
class CircleClass
{
    float x;
    float y;
    float vx;
    float vy;

    CircleClass() {
        x = random(width);
        y = random(height);
        vx = random(-5, 5);
        vy = random(-5, 5);
    }

    void display() {
        fill(255, 0, 0);
        circle(x, y, 30);
    }
}
```

```
class SquareClass
{
    float x;
    float y;
    float vx;
    float vy;

    SquareClass() {
        x = random(width);
        y = random(height);
        vx = random(-5, 5);
        vy = random(-5, 5);
    }

    void display() {
        fill(0, 0, 255);
        rect(x-15, y-15, x+15, y+15);
    }
}
```

```
class CrossClass
{
    float x;
    float y;
    float vx;
    float vy;

    CrossClass() {
        x = random(width);
        y = random(height);
        vx = random(-5, 5);
        vy = random(-5, 5);
    }

    void display() {
        stroke(0, 0, 0);
        line(x-15, y-15, x+15, y-15);
        line(x-15, y+15, x+15, y+15);
    }
}
```

ObjectBaseと ○ □ X



```
void move() {  
    x += vx;  
    y += vy;  
    if(x > width) {  
        x = width * 2  
        vx = -vx;  
    }  
    if(x < 0) {  
        x = -x;  
        vx = -vx;  
    }  
    if(y > height) {  
        y = height * 2  
        vy = -vy;  
    }  
    if(y < 0) {  
        y = -y;  
        vy = -vy;  
    }  
}
```

```
void move() {  
    x += vx;  
    y += vy;  
    if(x > width) {  
        x = width * 2  
        vx = -vx;  
    }  
    if(x < 0) {  
        x = -x;  
        vx = -vx;  
    }  
    if(y > height) {  
        y = height * 2  
        vy = -vy;  
    }  
    if(y < 0) {  
        y = -y;  
        vy = -vy;  
    }  
}
```

```
void move() {  
    x += vx;  
    y += vy;  
    if(x > width) {  
        x = width * 2  
        vx = -vx;  
    }  
    if(x < 0) {  
        x = -x;  
        vx = -vx;  
    }  
    if(y > height) {  
        y = height * 2  
        vy = -vy;  
    }  
    if(y < 0) {  
        y = -y;  
        vy = -vy;  
    }  
}
```

```
void move() {  
    x += vx;  
    y += vy;  
    x = (x + width)  
    y = (y + height)
```

一緒の部分をまとめる



- CircleClassはObjectBaseの変数 (x , y , vx , vy) や機能 (移動や初期化) をもち, 独自の丸の描画の機能をもつ
- SquareClassはObjectBaseの変数 (x , y , vx , vy) や機能 (移動や初期化) をもち, 独自の□の描画の機能をもつ
- CrossClassはObjectBaseの変数 (x , y , vx , vy) や機能 (初期化) をもち, 独自の移動とxの描画の機能をもつ

- インスタンス変数や, インスタンスメソッドを引き継ぐことを**継承**と呼ぶ!

一緒の部分をまとめる



- **CircleClass**は**ObjectBase**の変数 (x, y, vx, vy) や機能 (移動や初期化) をもち, **独自の丸の描画の機能をもつ**
- **SquareClass**は**ObjectBase**の変数 (x, y, vx, vy) や機能 (移動や初期化) をもち, **独自の□の描画の機能をもつ**
- **CrossClass**は**ObjectBase**の変数 (x, y, vx, vy) や機能 (初期化) をもち, **独自の移動とxの描画の機能をもつ**
- インスタンス変数や, インスタンスメソッドを引き継ぐことを**継承**と呼ぶ!

ObjectBase と etc



<pre>class ObjectBase { float x; float y; float vx; float vy; }</pre>	<pre>class CircleClass { float x; float y; float vx; float vy; }</pre>	<pre>class SquareClass { float x; float y; float vx; float vy; }</pre>	<pre>class CrossClass { float x; float y; float vx; float vy; }</pre>
<pre>ObjectBase() { x = random(width); y = random(height); vx = random(-5, 5); vy = random(-5, 5); }</pre>	<pre>CircleClass() { x = random(width); y = random(height); vx = random(-5, 5); vy = random(-5, 5); }</pre>	<pre>SquareClass() { x = random(width); y = random(height); vx = random(-5, 5); vy = random(-5, 5); }</pre>	<pre>CrossClass() { x = random(width); y = random(height); vx = random(-5, 5); vy = random(-5, 5); }</pre>
<pre>void display(){ point(x, y); }</pre>	<pre>void display() { fill(255, 0, 0); circle(x, y, 30); }</pre>	<pre>void display() { fill(0, 0, 255); rect(x-15, y-15, 30, 30); }</pre>	<pre>void display() { stroke(0, 0, 0); line(x-15, y-15, x+15, y-15); line(x-15, y+15, x+15, y+15); }</pre>

ObjectBase と etc



```
void move() {  
    x += vx;  
    y += vy;  
    if(x > width) {  
        x = width * 2;  
        vx = -vx;  
    }  
    if(x < 0) {  
        x = -x;  
        vx = -vx;  
    }  
    if(y > height) {  
        y = height * 2;  
        vy = -vy;  
    }  
    if(y < 0) {  
        y = -y;  
        vy = -vy;  
    }  
}
```

```
void move() {  
    x += vx;  
    y += vy;  
    if(x > width) {  
        x = width * 2;  
        vx = -vx;  
    }  
    if(x < 0) {  
        x = -x;  
        vx = -vx;  
    }  
    if(y > height) {  
        y = height * 2;  
        vy = -vy;  
    }  
    if(y < 0) {  
        y = -y;  
        vy = -vy;  
    }  
}
```

```
void move() {  
    x += vx;  
    y += vy;  
    if(x > width) {  
        x = width * 2;  
        vx = -vx;  
    }  
    if(x < 0) {  
        x = -x;  
        vx = -vx;  
    }  
    if(y > height) {  
        y = height * 2;  
        vy = -vy;  
    }  
    if(y < 0) {  
        y = -y;  
        vy = -vy;  
    }  
}
```

```
void move() {  
    x += vx;  
    y += vy;  
    x = (x + width) * 2;  
    y = (y + height) * 2;  
}
```

継承



- 継承すると、親の力をすべて引き継ぐ！
 - 親の資産、能力、機能をもらいうける
- 継承の方法は extends とやるだけ！

```
class クラス名 extends 親クラス名  
{  
}
```

CircleClassをどう作る？



- ObjectBaseクラスを継承してCircleClassを作る！

ObjectBase

x
y
vx
vy

ObjectBase()

move()

display()

CircleClassをどう作る？



```
class CircleClass extends ObjectBase  
{  
}
```

ObjectBase

x
y
vx
vy

ObjectBase()

move()

display()

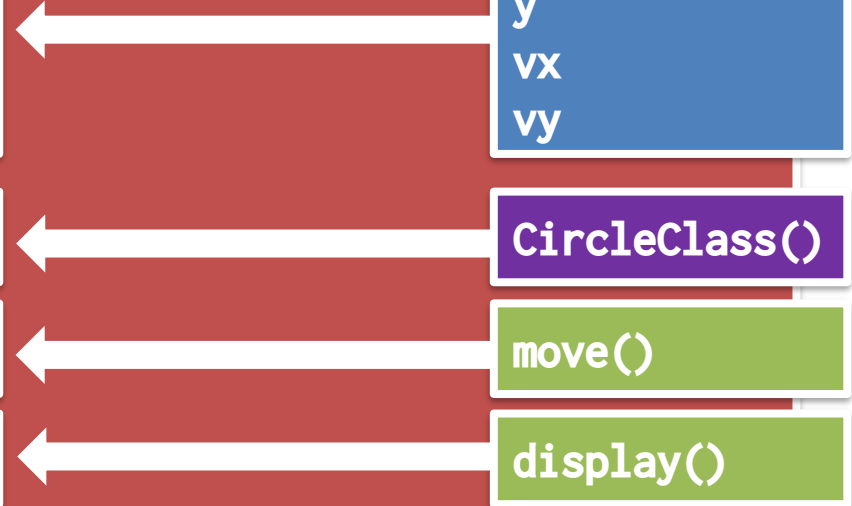
CircleClass

x
y
vx
vy

CircleClass()

move()

display()



使ってみよう！



```
CircleClass circle1;
CircleClass circle2;

void setup() {
    size( 400, 300 );

    circle1 = new CircleClass();
    circle2 = new CircleClass();
}

void draw() {
    background(255);

    circle1.move();
    circle2.move();
    circle1.display();
    circle2.display();
}
```

- 点が描画されるだけ
 - なんで？
- ObjectBaseのdisplay()は店の描画だから！
 - 丸を描画するにはどうしたら良い？
 - ObjectBaseのdisplayを書き換える？
 - → だめ！！

CircleClassをどう作る？



- ObjectBase の変数やメソッドを引き継ぎつつ、必要なところを上書きする！

ObjectBase

x
y
vx
vy

ObjectBase()

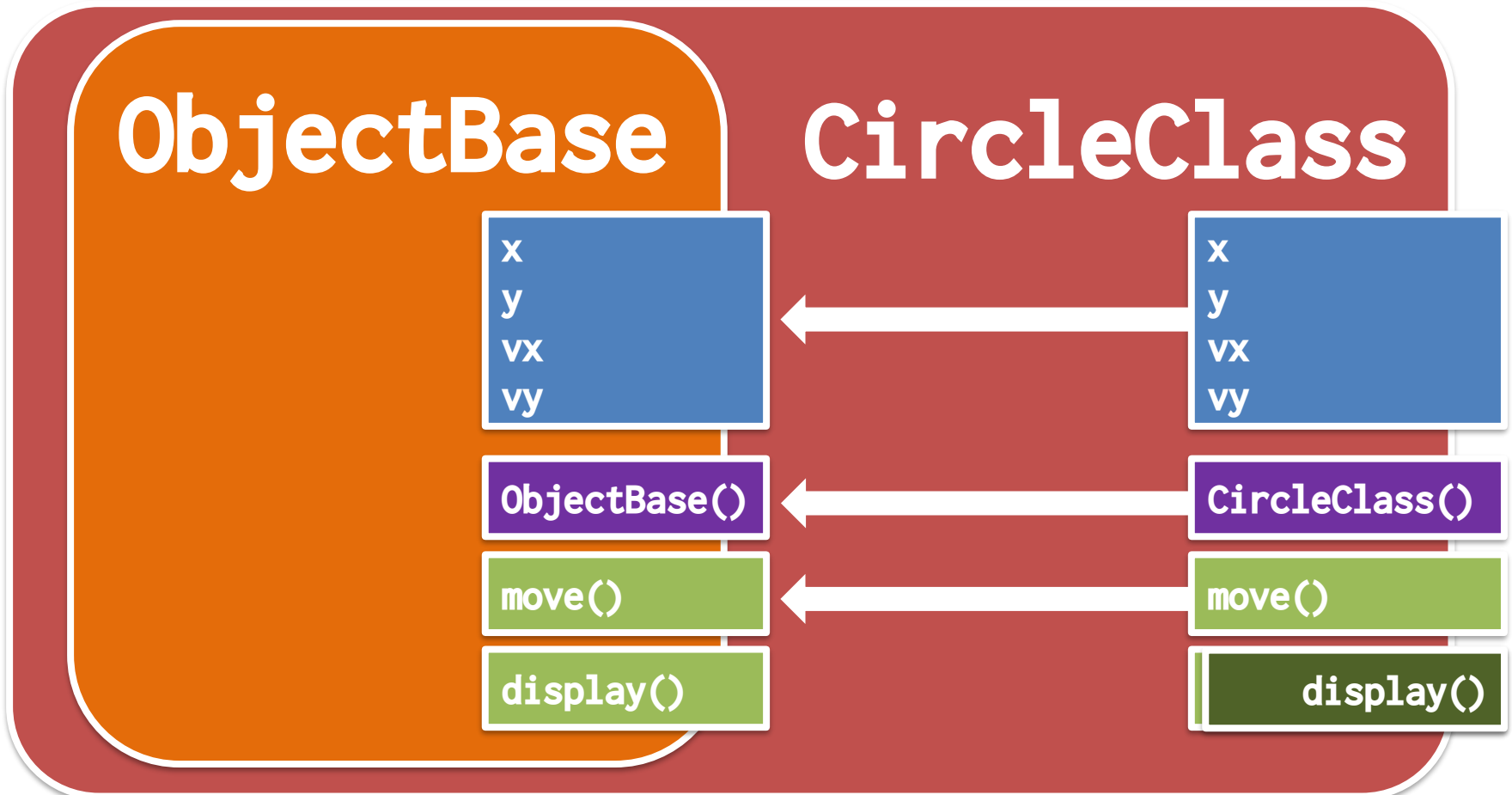
move()

display()

CircleClassをどう作る？



- display() を上書きしてしまう！
 - displayメソッドをオーバーライドする



ObjectBaseクラスを使う



```
class CircleClass extends ObjectBase
{
    void display(){
        fill(255, 0, 0);
        circle(x, y, 30);
    }
}
```

displayをオーバーライドして
親のdisplayメソッドが
呼ばれないようにする

CircleClassが劇的に短く！

使ってみよう！



```
CircleClass circle1;
CircleClass circle2;

void setup() {
    size( 400, 300 );

    circle1 = new CircleClass();
    circle2 = new CircleClass();
}

void draw() {
    background(255);

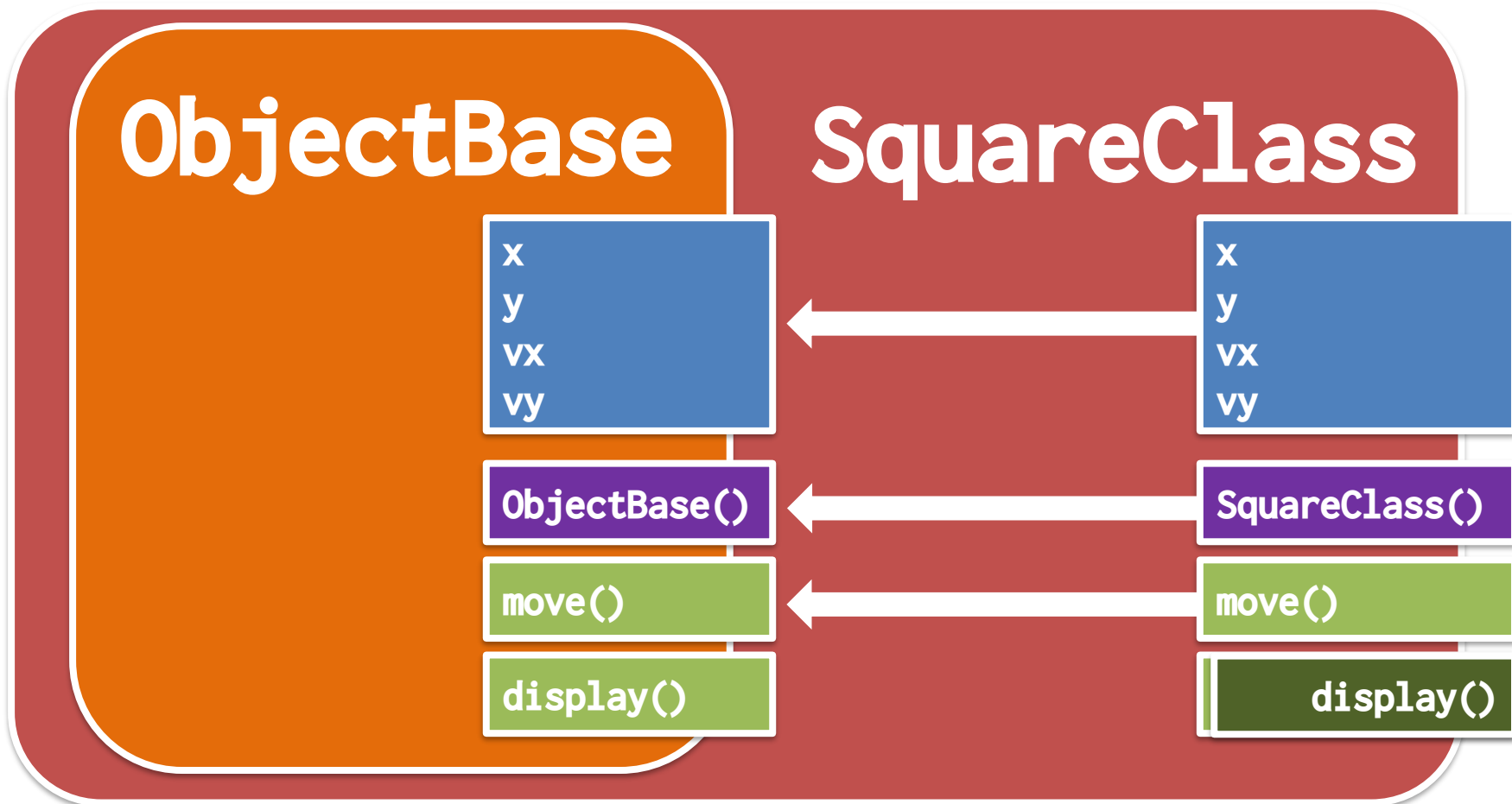
    circle1.move();
    circle2.move();
    circle1.display();
    circle2.display();
}
```

- 表示された！
 - display()が上書きされている！

SquareClassをどう作る？



- display() を上書きしてしまう！
 - displayメソッドをオーバーライドする



ObjectBaseクラスを使う



```
class SquareClass extends ObjectBase
{
  void display(){
    fill(0, 0, 255);
    rect(x-15, y-15, 30, 30);
  }
}
```

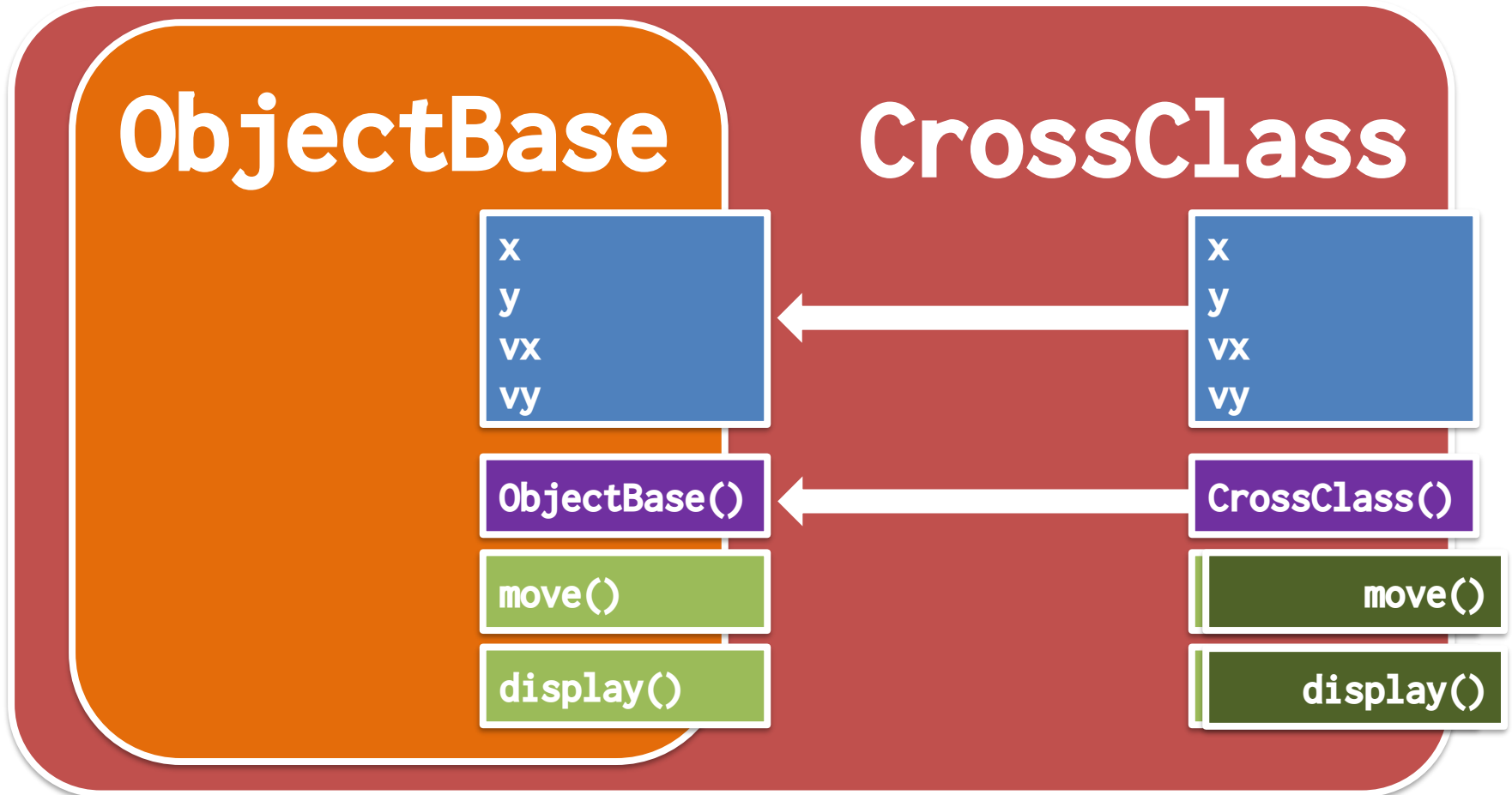
displayをオーバーライドして
親のdisplayメソッドが
呼ばれないようにする

SquareClassが劇的に短く！

CrossClassをどう作る？



- move() と display() を上書きしてしまう
– 跳ね返らないなので move をオーバーライド！



CrossClassはどう作る？



```
class CrossClass extends ObjectBase
{
    void display(){
        stroke(0, 0, 0);
        line(x-15, y-15, x+15, y+15);
        line(x-15, y+15, x+15, y-15);
    }

    void move() {
        x += vx;
        y += vy;
        x = (x + width) % width;
        y = (y + height) % height;
    }
}
```

**display と move を
オーバーライドして
親の move と display が
呼ばれないようにする**



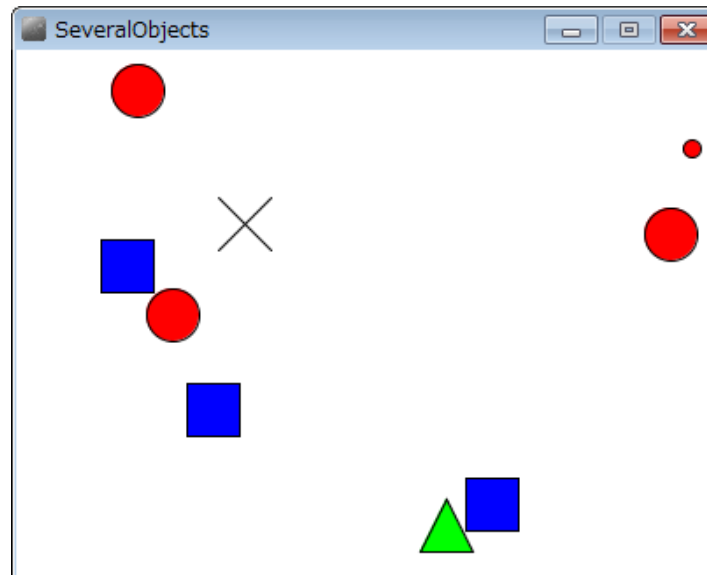
ObjectBaseクラスを使う

- 使うときは，継承していることは気にしないで，対象とするクラスを使うだけでよい！
 - move()
 - display()
 - すばらしい！！

```
CircleClass circleObj;  
SquareClass squareObj;  
CrossClass crossObj;  
  
void setup() {  
    size( 400, 300 );  
    circleObj = new CircleClass();  
    squareObj = new SquareClass();  
    crossObj = new CrossClass();  
}  
  
void draw() {  
    background(255);  
    circleObj.move();  
    squareObj.move();  
    crossObj.move();  
    circleObj.display();  
    squareObj.display();  
    crossObj.display();  
}
```



- ObjectBase クラスを継承して緑色の三角形を描画するTriangleClassを作るには？
 - 三角形は壁をすり抜けて反対側から出てくる



三角形のクラス



- 動く緑色の三角形のクラス

```
class TriangleClass extends ObjectBase
{
  void display(){
    stroke(0, 255, 0);
    triangle(x, y-15, x+10, y+15, x-10, y+15);
  }

  void move() {
    x += vx;
    y += vy;
    x = (x + width) % width;
    y = (y + height) % height;
  }
}
```

宿題2-1: hw_boundA113



- CircleClassを改良し、正方形が動き回る SquareClass、xが動き回るCrossClass、△が動き回る TriangleClassを作成せよ
- またこれを利用して10個の○と、5個の正方形と、5個のxと、3個の△が画面内を動き回るプログラムを作成せよ
 - ただし、その速度はx、y方向それぞれ-5以上5未満の実数値とせよ
 - また、○と□は端で跳ね返り、xと△は跳ね返らずに反対側から出てくるようにせよ

配列 + クラス



```
CircleClass[] circles = new CircleClass[10];
SquareClass[] squares = new SquareClass[5];
CrossClass[] crosses = new CrossClass[5];
TriangleClass[] triangles = new TriangleClass[3];

void setup() {
  size(400, 300);
  for (int i=0; i<10; i++)
    circles[i] = new CircleClass();
  for (int i=0; i<5; i++)
    squares[i] = new SquareClass();
  for (int i=0; i<5; i++)
    crosses[i] = new CrossClass();
  for (int i=0; i<3; i++)
    triangles[i] = new TriangleClass();
}
```

```
void draw() {
  background(255);
  for(int i=0; i<10; i++){
    circles[i].move();
    circles[i].display();
  }
  for(int i=0; i<5; i++){
    squares[i].move();
    squares[i].display();
  }
  for(int i=0; i<5; i++){
    crosses[i].move();
    crosses[i].display();
  }
  for(int i=0; i<3; i++){
    triangles[i].move();
    triangles[i].display();
  }
}
```



- CircleClass、SquareClass、CrossClass、TriangleClassの親クラス（継承元）はObjectBaseで、同じメソッドを持っている
- ポリモーフィズム（多態性，多様性）
 - 複数の型に属することを許すこと
 - 親クラスから継承されたクラスのインスタンスは，それぞれ親クラスの型に代入できる．呼び出されるのは，継承されたクラスのメソッド
 - つまり、ObjectBaseとして定義しておき、CircleClassもSquareClassもCrossClassもTriangleClassも放り込んで，ただのメソッドで呼び出しが可能！

配列 + ObjectBase



- CircleClassと
SquareClassと
CrossClassと
TriangleClassは
ObjectBaseを継承
- ObjectBase型は
CircleClassも
SquareClassも
CrossClassも
TriangleClassも含ん
でいるので、入れるこ
とができるよ！
- ポリモーフィズム！

```
ObjectBase[] list = new ObjectBase[10+5+5+3];

void setup() {
  size(400, 300);
  for (int i=0; i<10; i++)
    list[i] = new CircleClass();
  for (int i=0; i<5; i++)
    list[i+10] = new SquareClass();
  for (int i=0; i<5; i++)
    list[i+10+5] = new CrossClass();
  for (int i=0; i<3; i++)
    list[i+10+5+5] = new TriangleClass();
}

void draw() {
  background(255);
  for(int i=0; i<list.length; i++){
    list[i].move();
    list[i].display();
  }
}
```

ArrayList型



- ArrayList型は、いくつでも追加可能で色々なものを格納できるクラス

(例) `ArrayList list = new ArrayList();` で定義

- ArrayListの要素数を取得

```
list.size();
```

- ArrayListに対するaddで要素を追加

```
list.add( value ); // valueを追加
```

- ArrayListに対するremoveで要素を削除

```
list.remove( index ); // index番目を削除
```

- ArrayListに対するgetで要素を取得

```
list.get( index ); // index番目のオブジェクトを取得
```

ArrayListの定義



- 一応こちらでOK (getのとき要キャスト)

```
ArrayList list = new ArrayList();
```

- でも, こんな感じで型 (クラス) もセットにしたほうが後々わかりやすいので推奨
 - <> をGenericsと言います
 - intやfloatはなく, Integer, Floatと書きます

```
ArrayList<型> list = new ArrayList<型>();
```

```
ArrayList<Integer> list = new ArrayList<Integer>();  
ArrayList<Float> list = new ArrayList<Float>();  
ArrayList<String> member = new ArrayList<String>();
```

ArrayList<型>



- サイコロを1000回ふって、最後の10回分を表示するには？

```
ArrayList<Integer> history = new ArrayList<Integer>();
```

```
for(int i=0; i<1000; i++)  
{  
    int dice = (int)random(1, 7);  
    history.add( dice );  
}
```

```
for(int i=history.size()-10; i<history.size(); i++){  
    println(history.get(i));  
}
```

Generics で Integer を指定

ArrayList<型>



- サイコロを1000回ふって、最後の10回分を表示するには？

```
ArrayList history = new ArrayList();

for(int i=0; i<1000; i++)
{
    int dice = (int)random(1, 7);
    history.add( dice );
}

for(int i=history.size()-10; i<history.size(); i++){
    int num = (int)history.get(i);
    println( num );
}
```

Genericsを指定していないので
(int)でキャスト

追加/削除が自由！



- どうやって実現しているんだろう？という
ことを気にせずに，配列として使うことが
できる！
 - add(末尾に追加したいもの)
 - remove(削除したいアイテム番号)
 - get(取得したいアイテムの番号)
 - size()
 - を使えば，大きさを気にせず配列みたいなこと
を実現することができる！



ArrayList + ObjectBase

<ObjectBase> で型を定義

```
ArrayList<ObjectBase> list = new ArrayList<ObjectBase>();  
void setup() {  
    size(400, 300);  
    for (int i=0; i<10; i++){  
        CircleClass circleObj = new CircleClass();  
        list.add( circleObj );  
    }  
    for (int i=0; i<5; i++){  
        SquareClass squareObj = new SquareClass();  
        list.add( squareObj );  
    }  
    for (int i=0; i<5; i++){  
        CrossClass crossObj = new CrossClass();  
        list.add( crossObj );  
    }  
    for (int i=0; i<3; i++){  
        TriangleClass triangleObj = new TriangleClass();  
        list.add( triangleObj );  
    }  
}
```

CircleClassも
SquareClassも
CrossClassも
TriangleClassも
入れることができ
るよ！



ArrayList + ObjectBase

<ObjectBase> で型を定義

```
ArrayList<ObjectBase> list = new ArrayList<ObjectBase>();

void setup() {
  size(400, 300);
  for (int i=0; i<10; i++){
    list.add( new CircleClass() );
  }
  for (int i=0; i<5; i++){
    list.add( new SquareClass() );
  }
  for (int i=0; i<5; i++){
    list.add( new CrossClass() );
  }
  for (int i=0; i<3; i++){
    list.add( new TriangleClass() );
  }
}
```

new ClassName()
を直接 add できるよ！

ArrayList + ObjectBase



```
ArrayList<ObjectBase> list = new ArrayList<ObjectBase>();
```

```
void setup() {  
    size(400, 300);  
    for (int i=0; i<10; i++)  
        list.add( new CircleClass() );  
    for (int i=0; i<5; i++)  
        list.add( new SquareClass() );  
    for (int i=0; i<5; i++)  
        list.add( new CrossClass() );  
    for (int i=0; i<3; i++)  
        list.add( new TriangleClass() );  
}
```

```
void draw() {  
    background(255);  
    for( int i=0; i<list.size(); i++ ){  
        list.get(i).move();  
        list.get(i).display();  
    }  
}
```

<ObjectBase> で型を定義

list.size() で数を取得

list.get(i) でi番目を取得

ArrayList + ObjectBase



```
ArrayList list = new ArrayList();

void setup() {
  size(400, 300);
  for (int i=0; i<10; i++)
    list.add( new CircleClass() );
  for (int i=0; i<5; i++)
    list.add( new SquareClass() );
  for (int i=0; i<5; i++)
    list.add( new CrossClass() );
  for (int i=0; i<3; i++)
    list.add( new TriangleClass() );
}

void draw() {
  background(255);
  for(int i=0; i<list.size(); i++){
    ObjectBase obj = (ObjectBase)list.get(i);
    obj.move();
    obj.display();
  }
}
```

型を定義しなくても使えます
その場合は使うときにキャストを！



(ObjectBase) でキャスト

ArrayList + 拡張for文



```
ArrayList<ObjectBase> list = new ArrayList<ObjectBase>();

void setup() {
  size(400, 300);
  for (int i=0; i<10; i++)
    list.add( new CircleClass() );
  for (int i=0; i<5; i++)
    list.add( new SquareClass() );
  for (int i=0; i<5; i++)
    list.add( new CrossClass() );
  for (int i=0; i<3; i++)
    list.add( new TriangleClass() );
}

void draw() {
  background(255);
  for( ObjectBase obj: list ){
    obj.move();
    obj.display();
  }
}
```

めっちゃシンプル！



- 継承とArrayListを紹介
 - 継承は変数や関数を引き継ぐ
 - 継承されたものはまとめて扱うことが可能
 - ArrayListは柔軟な配列みたいなもので便利なのでマスターしよう！