



プログラミング演習2

クラスと継承と色々

中村、小林、辻野、石井

実験協力者募集中

明治大学総合数理学部
先端メディアサイエンス学科
中村研究室



- 現在、中村研では様々な実験協力者を随時募集しています
 - これから卒論・修論シーズンで他の研究室からも募集が多くかかると思います！
 - お金になることもありますし、研究室の雰囲気を知る切っ掛けにもなるので、興味のある人はFMS学科全体の「#募集-実験協力者」チャンネルへぜひ入りましょう！

講義予定（案）



- 09/22 第01回 ガイダンス + 前期の復習 + α
- 09/29 第02回 クラス
- 10/06 第03回 クラスと継承
- 10/13 第04回 クラスと継承と色々
- 10/20 第05回 復習
- **10/27 第06回 中間テスト（50点）**
- 11/10 第07回 振り返りとファイル入出力
- 11/17 第08回 ネットワーク
- 11/24 第09回 Web API
- 12/01 第10回 フィジカルコンピューティング
- 12/08 第11回 フィジカルコンピューティング
- 12/15 第12回 フィジカルコンピューティング
- 12/22 第13回 色々
- 01/19 第14回 発表会



- メディア教室を使って講義資料＋課題＋宿題から75%はそのまま出す中間試験（確認テスト）を実施します。
 - 範囲は春学期＋秋学期の第1回～4回のもの
 - ただし解き方については制限を入れないため、解くことができればOKです！
- ぜひ、課題と宿題だけでもちゃんとできるようになっておいてください！



- 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！



- 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



- 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！

ArrayList



```
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

何に使えるの？



- 継承とかポリモーフィズムとかなんか色々あるけど何に使うの？
 - シューティングで色んな動きをする敵を作る
 - RPGゲームで色んな戦い方をする敵を作る
 - 色々な光り方・広がり方をする花火を作る
 - 色々な結晶の雪を各々の特性で降らせる
 - 草原に色々な草木花を生えさせる
 - ひとの混雑に関するシミュレーションをする

**親クラスを定義して継承してクラスを作り
ArrayListで意識せずまとめて動かす！**

宿題3-2: hw_JankenGuriko



- グリコというゲーム（じゃんけんにグーで勝つと+3、チョキで勝つと+5、パーで勝つと+6の点が入る）がある
 - チョキで+6が一般的だがここでは+5とする
- 毎回じゃんけんを繰り返し、10000点を先に獲得したほうが勝ちになる
- 配布するhw_JankenGuriko内のJankenAgentクラスを継承して、自身のエージェントを作成せよ
- battleメソッドに自身のエージェントと、敵のエージェントを引数として与え、10000点になるまで戦わせて勝ちましょう！
 - 少なくともJankenAgent自身、AgentNakamura334には勝ちましょう
 - できれば、知り合いのAgentももらって、勝ちましょう！！

課題3-3:basic_CharacClass



- 配布するCharacterBaseクラスを継承し、CharacterBaseクラス内のcenterX, centerYを利用してその位置に自身のキャラクタを描画するようにせよ
 - 継承したクラスではdisplayActiveとdisplayInactiveをオーバーライドせよ
 - moveメソッドをオーバーライドして特殊な動き方をするように変更してもかまわないが、画面の外に出ないないようにせよ
- なお配布プログラムではCharacterBaseクラスのキャラクタ(□)と、ExNakamura334クラスのキャラクタを表示しているが、自分で作成したクラス(継承したクラス)のキャラクタをExNakamura334クラスのキャラクタに置き換えて一緒に表示するようにせよ
- クラスの名前は名前と番号が含まれるようにし、そのクラスのみ別ファイルとして作成せよ(来週配布します)
 - (例) X-3-34 中村の場合、クラス名を「ExNakamura334」とする

課題4-1: basic_ManyChara



- 色々な人が作成した ExXXXXX クラスを利用して多くのキャラクターが勝手に動き回るプログラムを作成せよ
- マウスのボタンを押している間はdisplayActive、押していないときはdisplayInactiveを利用して、すべてのキャラクターが異なる描画内容になるようにせよ
- そのために、各自が作成したキャラクタークラスが入ったファイルを研究室のSlackにアップせよ
 - 同じ研究室のものを5つ以上利用して（同じ研究室で足りなければ他の研究室のを使ってもよい）、キャラクターが画面内を動き回るプログラムを作成せよ
 - 他の研究室のものを使用して、もっと多くのキャラクターを入れてもよい

（ヒント） ArrayListに放り込むだけ！！

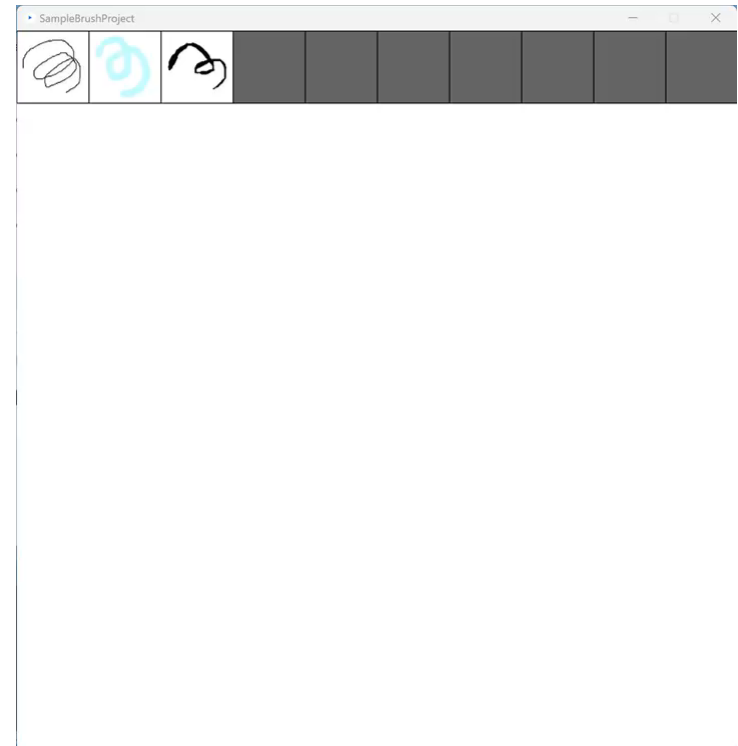
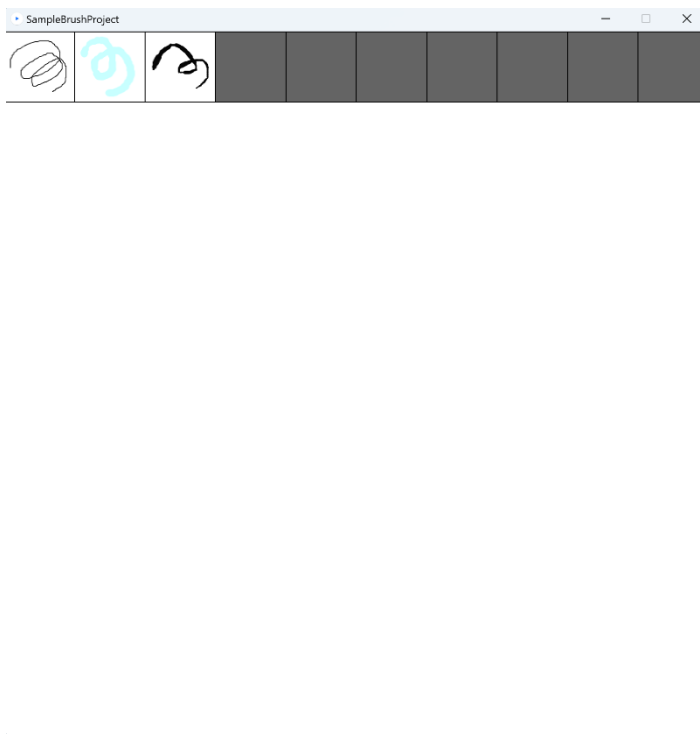
継承でペイントツールを作る

明治大学総合数理学部
先端メディアサイエンス学科
中村研究室



- hw_PaintTool

- hw_PaintToolには、BaseBrushと、BaseBrushを継承したMyBrush1、MyBrush2がある。
- BaseBrushクラスを継承して、他のペンを作ってみてどうでした？



覚える必要はないけれど

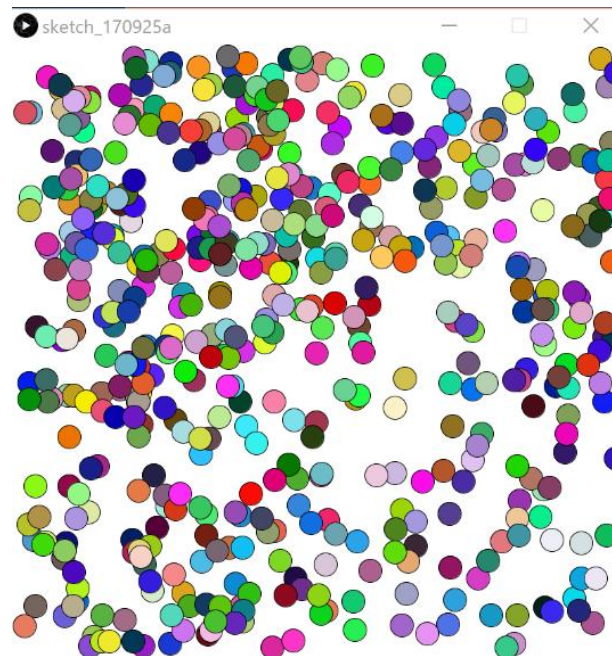


- アップキャスト
 - 親クラスで定義されたものに、継承されたクラスのインスタンスが代入されると、自動的に親の型に変換されること
- ちなみにダウンキャストもあるよ
 - ダウンキャストは安全じゃないよ！

クリックで増やしたい！



- ウィンドウ800x800を用意し、CircleClassを利用してマウスクリックするたびに**クリックした場所に丸が生まれ**，上下左右にランダムな速度で動き、上下左右の端で跳ね返るプログラムを作成せよ。なお、クリックでいくらでも作成できるようにせよ。
- たくさん増やして管理するには ArrayListを使う！



コンストラクタで定義



- インスタンスを作るときに初期位置をマウスの位置に！

コンストラクタ内で
mouseX, mouseYを使う？

```
class CircleClass {  
    float x;  
    float y;  
    float vx;  
    float vy;  
  
    CircleClass() {  
        x = mouseX;  
        y = mouseY;  
        vx = random(1,5);  
        vy = random(1,5);  
    }  
}
```

- これだと使い回せないのが困る！！！！
- 普通に作ったときにおかしくなる

関数を作る？



- 関数を作れば一応実現は可能

```
XXX.setPos(mouseX, mouseY);
```

- 初期値を設定するだけなのでわざわざ関数を作るのは...
- コンストラクタで設定する！

```
class CircleClass {  
    float x;  
    float y;  
    float vx;  
    float vy;  
  
    CircleClass() {  
        x = random(width);  
        y = random(height);  
        vx = random(1,5);  
        vy = random(1,5);  
    }  
  
    void setPos(float _x, float _y) {  
        x = _x;  
        y = _y;  
    }  
}
```

コンストラクタで定義！



- インスタンスを作るときに

```
new CircleClass(mouseX, mouseY)
```

みたいな感じで初期位置を明示的に決めたい！

コンストラクタとして座標を指定できるものを用意する！

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

  CircleClass() {
    x = random(width);
    y = random(height);
    vx = random(1,5);
    vy = random(1,5);
  }

  CircleClass(float _x, float _y) {
    x = _x;
    y = _y;
    vx = random(1,5);
    vy = random(1,5);
  }
}
```

new CircleClass(mouseX, mouseY)



- クリックした場所から丸がうまれる！

```
ArrayList<CircleClass> circles = new ArrayList<CircleClass>();

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

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

void mousePressed() {
  circles.add( new CircleClass(mouesX, mouseY) );
}
```

newで指定！

オーバーロード！



- CircleClass(){...}と
CircleClass(float _x, float _y){...} と
2つあってもよいの？

– 引数の数とか型が違えば大丈夫！！

CircleClass() は

```
CircleClass c = new CircleClass();
```

CircleClass(float _x, float _y) は

```
CircleClass c = new CircleClass(500.0, 100.0);
```

で，それぞれ呼び出される。

オーバーロード（多重定義）

同じ関数の名前だけれど，引数の数や型を変えて振る舞いを変えることができるもの

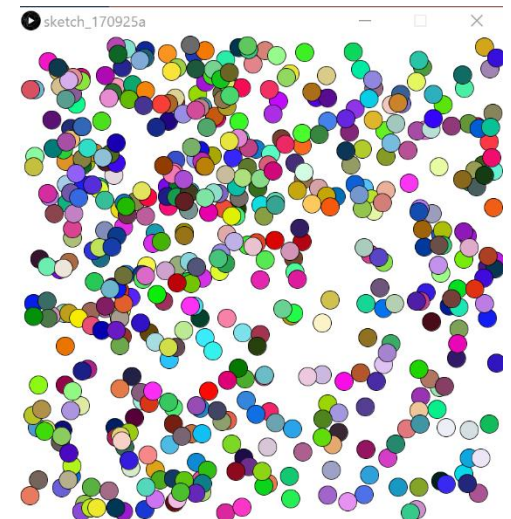
クリックによる丸の生成



- ウィンドウ800x800を用意し, CircleClassを利用してマウスをクリックするたびにクリックした場所に丸が生まれ, 上下左右にランダムな速度で動き, 上下左右の端で跳ね返るプログラムを作成せよ. なお, クリックでいくらでも作成できるようにせよ
- また, CircleClassを改良してそれぞれの丸の色は初期値としてランダムに設定するようにせよ

[ヒント] 色情報を持つインスタンス変数を追加する

- int red;
- int green;
- int blue;





- 引数がある場合は要コンストラクタの定義

ObjectBase

```
posX  
posY  
speedX  
speedY
```

```
ObjectBase()
```

```
ObjectBase(float x, float y)
```

```
move()
```

```
display()
```

CircleClass

```
CircleClass()
```

```
CircleClass(float x, float y)
```

```
move()
```

```
display()
```

親コンストラクタの呼び出し



- 空っぽで同じ引数のコンストラクタを作る

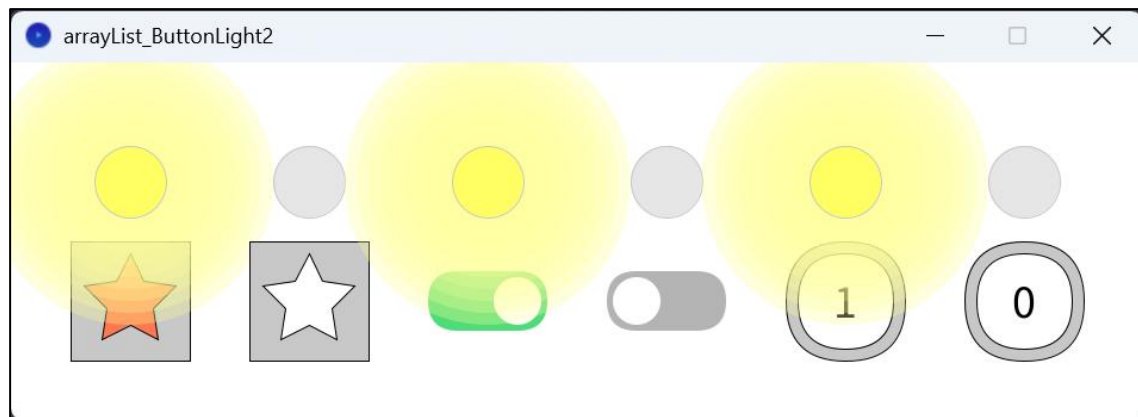
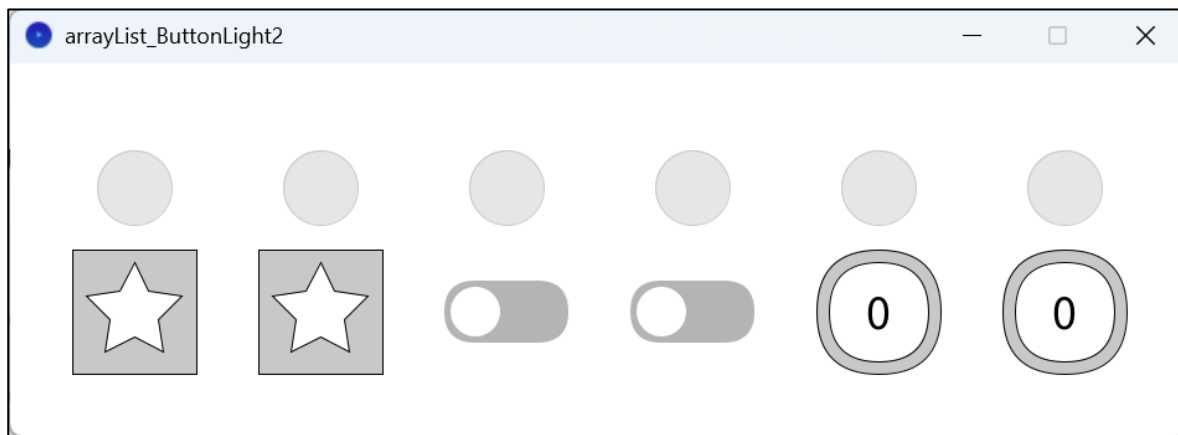
```
class ObjectBase {  
    float x;  
    float y;  
    float vx;  
    float vy;  
    ObjectBase(){  
        x = random(width);  
        y = random(height);  
        vx = random(-5, 5);  
        vy = random(-5, 5);  
    }  
    ObjectBase(float _x, float _y){  
        x = _x;  
        y = _y;  
        vx = random(-5, 5);  
        vy = random(-5, 5);  
    }  
}
```

```
class CircleClass extends ObjectBase {  
  
    CircleClass(){  
    }  
  
    CircleClass(float _x, float _y){  
        super(_x, _y);  
    }  
}
```

宿題3-1: hw_ButtonLight



- これまで作成したLightClassやButtonClassを継承して、光り方やボタンの形状が異なるものをそれぞれ2つ作成するとともに、その3つのボタンとライトをそれぞれ2つずつ計6個描画せよ



こんな方法で解いた？



```
ButtonClass[] buttons = new ButtonClass [6];
LightClass[] lights = new LightClass [6];

void setup() {
  size(600, 600);
  for(int i=0; i<6; i++)
    lights[i] = new LightClass();
  buttons[0] = new StarButton();
  buttons[1] = new StarButton();
  buttons[2] = new SwitchButton();
  buttons[3] = new SwitchButton();
  buttons[4] = new CountButton();
  buttons[5] = new CountButton();
  // setPositionは省略
}

void mousePressed() {
  for(int i=0; i<6; i++){
    buttons[i].click(mouseX, mouseY);
    if(button[i].isOn()) lights[i].turnOn();
    else lights[i].turnOff();
  }
}
```

ちょっと面倒で
わかりにくい

LightClassにまとめたい



- 色々ボタン定義した場合どうする？

```
class LightClass {
  int centerX, centerY; // 位置
  int r;
  ButtonClass button;   // ボタン

  LightClass() {
    // Lightの中でButtonを定義しよう
    button = new ButtonClass();
    r = 30;
  }

  void setPosition(int cx, int cy) {
    centerX = cx;
    centerY = cy;
    button.setPosition(cx, cy+100);
  }

  void click(int mx, int my) {
    button.click(mx, my);
  }
}
```

```
LightClass light;

void setup() {
  size(600, 600);
  light = new LightClass();
  light.setPosition(width/2, height/2);
}

void draw() {
  background(255);
  light.display();
}

void mousePressed() {
  light.click(mouseX, mouseY);
}
```

LightClassにまとめたい



```
class LightClass {
    int centerX, centerY; // 位置
    int r;
    ButtonClass button;    // ボタン

    LightClass() {
        button = null;
        r = 30;
    }

    LightClass(ButtonClass btn) {
        button = btn;
        r = 30;
    }

    void setPosition(int cx, int cy) {
        centerX = cx;
        centerY = cy;
        button.setPosition(cx, cy+100);
    }
}
```

```
LightClass[] lights = new LightClass[6];

void setup() {
    size(600, 600);
    for(int i=0; i<6; i++){
        lights[i].setPosition(xxx, xxx);
        if(i==0 || i==1)
            lights[i] = new LightClass(new StarButton());
        if(i==2 || i==3)
            lights[i] = new LightClass(new SquareButton());
        if(i==4 || i==5)
            lights[i] = new LightClass(new SwitchButton());
    }
}

void draw() {
    background(255);
    for(int i=0; i<6; i++) lights[i].display();
}

void mousePressed() {
    for(int i=0; i<6; i++)
        lights[i].click(mouseX, mouseY);
}
```

LightClassにまとめたい



- ArrayListでもやれるよ

```
class LightClass {
    int centerX, centerY; // 位置
    int r;
    ButtonClass button;    // ボタン

    LightClass() {
        button = null;
        r = 30;
    }

    void setPosition(int cx, int cy) {
        centerX = cx;
        centerY = cy;
        button.setPosition(cx, cy+100);
    }

    void setButton(ButtonClass btn) {
        button = btn;
    }
}
```

```
ArrayList<LightClass> lights
    = new ArrayList<LightClass>();

void setup() {
    size(600, 600);
    for(int i=0; i<6; i++)
        lights.add(new LightClass(new StarButton()));
    // 場所の設定
    // ボタンの設定
}

void draw() {
    background(255);
    for (LightClass l : lights) l.display();
}

void mousePressed() {
    for (LightClass l : lights)
        l.click(mouseX, mouseY);
}
```

クリックで敵を倒すゲーム

明治大学総合数理学部
先端メディアサイエンス学科
中村研究室



- 独特の動きをする5個の○と、同じく独特の動きをする5個の□、5個の×をクリックにより倒すゲームを作成せよ

どう実現する？



- EnemyRootクラスを作り下記のクラスを継承して作成する
 - EnemyCircle
 - EnemySquare
 - EnemyCross
- どんな変数が必要だろうか？
 - 場所、速度、生きているか
- どんなメソッドが必要だろうか？
 - 初期化、移動、描画、当たり判定

EnemyRootクラス



```
class EnemyRoot
{
    float x;
    float y;
    float vx;
    float vy;
    boolean live;

    EnemyRoot() {
        x = random(width);
        y = random(height);
        vx = random(-10, 10);
        vy = random(-10, 10);
        live = true;
    }
}
```

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

void hit(int _x, int _y) {
    if((int)x == _x && (int)y == _y){
        live = false;
    }
}

void display() {
    if(live) {
        point(x, y);
    }
}

boolean isLive() {
    return live;
}
}
```

**ObjectBase と似てる
動くものは似がち**



- 継承して move を工夫し色々面白くする
 - EnemyCircle
 - めちゃくちゃ早い
 - EnemySquare
 - 不規則運動
 - EnemyCross
 - 途中で消える



- ポイント：カプセル化
 - 中の変数はいじらせずに、メソッドで命令できるようにします（エラーのもとなので、変数は基本的に直接いじらない！）
 - これが上手にはまると、きれいなコードになっていき、再利用可能性が高まります。
- まあクラスの勘所は、何度も設計して身につけていくものですので、恐れず設計してみてください。で、あーこうしたら良かったと気づいて下さい。



- ベクトル表現を扱うクラス
 - `.x` と `.y` の実数値で2次元の値を格納
 - `PVector v = new PVector(x, y);` で定義
 - なお、引数は実数値 `float`
 - `v.add(v2);` `v`に`v2`を足し合わせる
 - `v.sub(v2);` `v`から`v2`を引く
 - `v.mult(n);` `v`を`n`倍する
 - `v.rotate(theta);` `v`を`theta`回転する



PImage型 / 画像型

- 画像を格納および描画するクラス
 - .width や .height で画像の縦横のサイズ取得
 - .resize() で画像サイズを変更可能
 - .save() で画像を保存可能
 - .filter() で各種フィルタをかけることが可能

```
PImage img;  
size(400, 400);  
img = loadImage("画像ファイル名");  
img.filter(BLUR, 6);  
image(img, 0, 0);
```

フィルタ例 ()内はオプション
THRESHOLD (0-1.0)
GRAY
OPAQUE
INVERT
POSTERIZE (2-255)
BLUR (1以上, 半径)
ERODE
DILATE

音楽の再生



```
import processing.sound.*;
SoundFile bgm; //BGM格納用の変数
int x = 0;
int vx = 4;

void setup()
{
  size(400, 400);
  bgm = new SoundFile(this, "bgm.mp3"); //mp3ファイルを指定する
  bgm.loop(); //bgmを再生
}

void draw()
{
  background(0, 0, 0);
  x = x + vx;
  if(x >= width)
  {
    x = width;
    vx = -vx;
  }
  else if(x <= 0)
  {
    x = 0;
    vx = -vx;
  }
  circle(x, 200, 20);
}
```



- 準備

- 音楽再生ライブラリ（便利関数群）の読み込み

- `import processing.sound.*;`

- SoundFileの変数を初期化し，初期化

- **SoundFile はサウンド関係を扱うクラス**

- `SoundFile bgm;`

- `bgm = new SoundFile(this, "ファイル名");` でファイル読込

- `bgm.loop();` // 繰り返し再生

- `bgm.stop();` // 停止

- SoundFile に効果音を読み込む

- `SoundFile se = new SoundFile(this, "ファイル名");`

- `se.play();` // 効果音の再生

Stringクラスのメソッド



- `charAt(num)`; `num`文字目の文字を返す（0から始まる）
- `indexOf(文字列)`; 入力された文字列が何文字目か？
- `indexOf(文字列, n)`; `n`文字目以降で入力された文字列が何文字目か？
- `length()`; 入力された文字の文字数を返す
- `substring(x)`; `x`文字目から最後までを出力
- `substring(x, y)`; `x`文字目から`y-1`文字目までを出力
- `toLowerCase()`; 全てを小文字に変換する
- `toUpperCase()`; 全てを大文字に変換する
- `replace(文字列A, 文字列B)`;
 - 文字列Aを文字列Bに変更する
- `split(文字列)`; 文字列を分割

<http://processing.org/reference/String.html>

<http://docs.oracle.com/javase/6/docs/api/java/lang/String.html>



THE NATURE OF CODE

DANIEL SHIFFMAN

- The Nature of Code
 - Processingでクラスを使いつつ，物理世界の再現について学ぶことができるうえ，ニューラルネットワークとかも学べるよ！
 - 英語も難しくないから学習に良いよ！
 - <https://natureofcode.com/book/>