

# Using Class

Processing Tutorial 2

# INDEX

- 3 Classとは
- 4 Processingでの記述方式
- 5 演習
- 6 Processing標準クラス
- 7 ArrayList
- 8 PVector
- 9 外部ライブラリ
- 10 課題

# Classとは

前回は、ファンクションやサブルーチンとは何かということと、どんなプログラミングでも使われる基本的な概念（変数、配列、ループ、条件付け等）を覚えてもらいました。ここでさらに一つステップアップするために「クラス」という概念を覚えていただきたいと思います。

ところで、オブジェクト指向という言葉を聞いたことがあるでしょうか。オブジェクト指向とは、「クラス」が使えるということで、オブジェクト指向プログラミングとはこの「クラス」を使ったプログラミングであるという意味です。

そこで、クラスとは一体なんであるかという話ですが、まずは前回覚えていたファンクションとサブルーチンの役割を思い出してください。前回は、ファンクションとサブルーチンとは、ある目的をもった道具であると説明しました。例えば、紙を切るためのハサミ、ものを貼り合わせるためのセロテープなどです。それ自体が材料になることのない道具がサブルーチン、例えばハサミのようなもの、そしてそれ自体が材料になりうる道具、例えばセロテープなどがファンクションであると説明しました。

ここで気をつけてほしいのは、道具として定義されているサブルーチンやファンクションの中に記述されているのは、ハサミであれば「切る」というアクション、セロテープであれば「貼る」というアクションで、それ自体は非常にシンプルなものです。実際なにか物を作るといった場合、こういった道具を複数使い分けて材料を加工していく必要があります。

例えば、紙を切ったり貼ったりすることで何かの模型を作るとします。そのために使う道具として、はさみとセロテープを使ったとします。ここでは二つの、それぞれは「切る」と「貼る」というシンプルな使い方しかできません。ただ、それらの組み合わせによっては、結果として単純な紙の箱から車や飛行機の模型、果ては城、恐竜なども作れたりします。つまり、同じ道具でも、その組み合わせや使い方によって様々なバリエーションを作ることができます。例えば、この例でのバリエーションとは、「紙を使って作る模型」のバリエーションということになります。このような、様々なバリエーションをもつ物を作るためのツールキット（道具箱）をクラスといいます。

もう一度言いますが、クラスとはサブルーチンやファンクションといった道具が複数、その組み合わせや使い方とともにしまわれている道具箱ということです。そして様々な材料（紙でも色のついた紙とか、和紙だとか）を与えたり、道具箱に記述されている道具の使い方の指示を少し変えるだけで、道具箱に入っている道具を使って様々なバリエーションのものが出来上がるということです。

最初はとても分かりにくいと思うのですが、このクラス（道具箱）という概念は、いろいろなバリエーションのものを一気に作りたいときとかに便利です。一回この道具箱を作ってしまえば後はそれを何回も使えるため、ものを作るたびに道具を作る（買いに行く）必要がないのです。つまりプログラムのコードが非常にすっきりしたものになるのです。模型のマニュアルに、「まずはハサミを買ってきて紙を半分に切ってください。そして次にハサミを買ってきてさらに半分に切ってください」とあったらくどいです。そんなにいくつも道具を用意する必要はない、ということです。

クラスとは、ものを作るための道具が全部入った道具箱。忘れないでください。

# Processingでの記述方式

まずは、クラスの基本形を覚えてください。以下が基本の形になります（左側の数字は行数で、それをコードの一部として書かないでください）

```
1 //まずクラスのバリエントを作つて名前をつける
2 sampleClass classVariant1;
3
4 void setup(){
5   //クラスのバリエントに入れる初期値の宣言
6   int value = 50;
7   //作ったクラスのバリエントの初期値を入力する
8   classVariant1 = new
sampleClass(value);
9   //クラスの中のサブルーチンを起動する
10  classVariant1.update();
11 }
12
13 //クラスの宣言
14 class sampleClass{
15   int value;
16   sampleClass(int _value){
17     value = _value;
18   }
19
20   void update(){
21     //何かしらのアクション
22     ellipse(width/2,height/2,value,value);
23   }
24 }
```

上の形が基本形で、これをまずは空で書けるようにしてください。そして何か試しに自分でクラスを作ってみてください。これからプログラムを書くというとき、実際のところクラスを使わずにファンクションやサブルーチンをつくるだけで済むことはいくらでもあるのですが、そういう場面でも積極的にクラスを使うようにしてください。これも慣れです。

本来クラスはあってもなくてもプログラムは動きます。あくまで、クラスはサブルーチンやファンクションを入れたツールボックスであると、考えてください。ただ、そのツールボックスが用意されていた方が、欲しい道具の組み合わせをすぐ取り出せるという、効率的か否かという話です。効率化は重要です。

以下はクラスをもうすこしふくらせたものの例です。

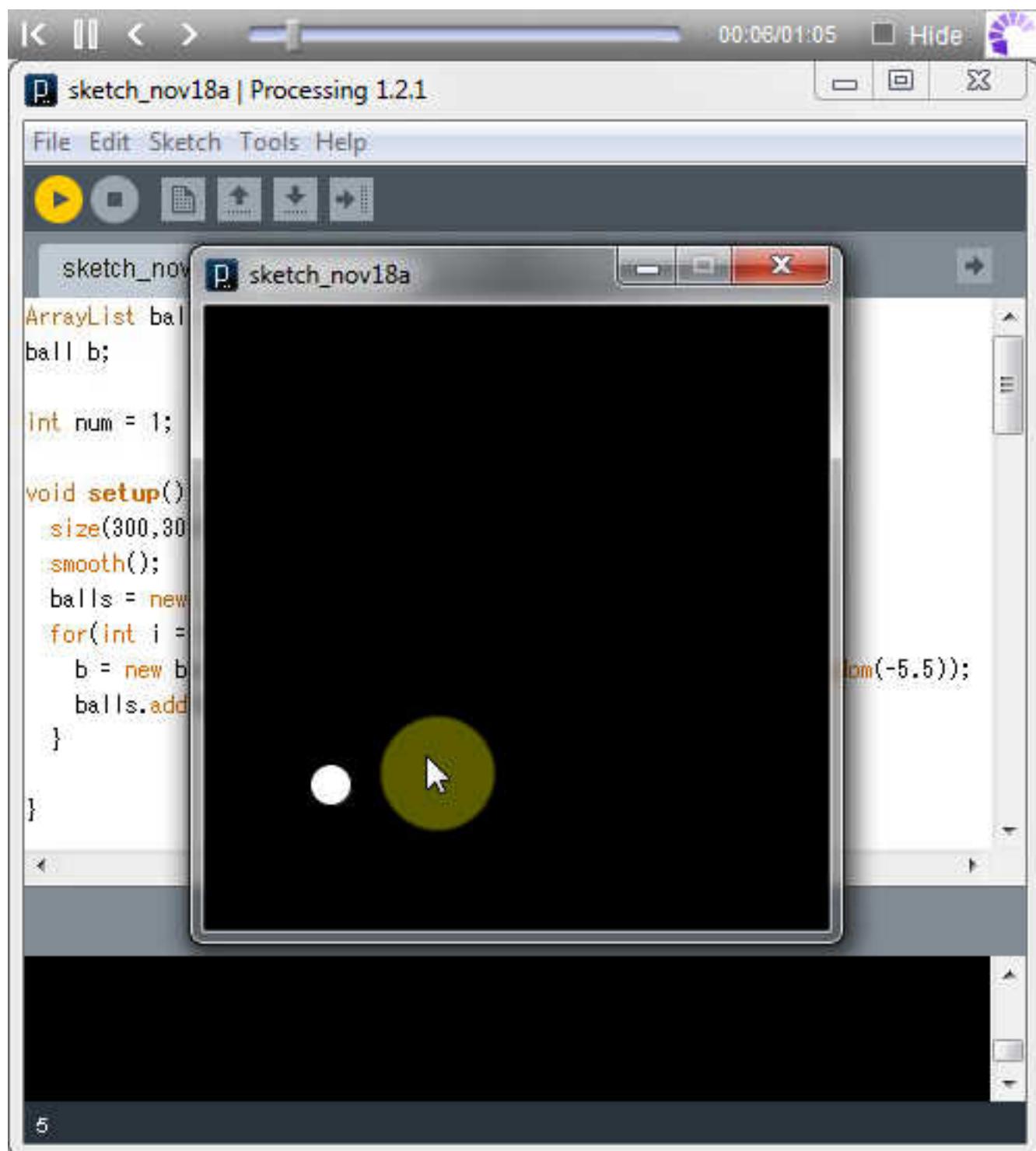
```
1 modelMaker model1;
2
3 void setup(){
4   model1 = new modelMaker(200);
5   model1.scissors();
6 }
7
8 class modelMaker{
9   float paperSize;
10
11 pipeMaker(float _paperSize){
12   paperSize = _paperSize;
13 }
14
15 void scissors(){
16   if(tapeResult == true){
17     print("Cutting");
18   }else{
19     print("No Cutting");
20   }
21 }
22
23 boolean tape(paperSize){
24   boolean tapeResult;
25   if(paperSize>100){
26     tapeResult = true;
27   }else{
28     tapeResult = false;
29   }
30   return tapeResult;
31 }
32 }
```

最初は扱いが難しいクラスですが、書いているうちに、道具をいかにクラスへとパックすることが便利なことかということが分かってくると思います。

# 演習

ここで一つ演習問題を出してみたいと思います。以下の動画で描写されているものを、クラスを使って作ってみてください。書き方は人それぞれ絶対に変わらるはずですので、まったく同じコードということはありません。コードがどんなに長くなってしまっても今はかまいません。逆にすごく短いコードで同じプログラムを書けたとしたら、それはすばらしいです。レファレンスは何を参照していただいてもかまいません。ヒントとして、Processingに標準で装備されているクラスを組み合わせると簡単に作ることができます (PVector, ArrayList等)

まずは一人で一から書いてみることが重要です。これ一つできるだけで、これから伸びが速くなっていきます。がんばってください。



# Processing標準クラス

Processingには標準で便利なクラスが数多く用意されています。例えばArrayList、PVector、PImage、PShape、PGraphics、P3D、PFontなどがあります。これらは、Processingでのプログラミングを簡単に行うための非常に便利なツールボックス集です。いかに手だれたプログラマであっても、プログラムをまっさらな状態から書きはじめる人はほとんどいません。みんな、自分の目的にあったクラスを複数使い分けてプログラムを書いている場合がほとんどです。そして、ピンポイントで欲しい機能に関してだけ、自分でクラスを書くのです。

この章では、Processingでも本当によく使うクラスを二つ、使い方とともに紹介したいと思います。ひとつは配列を動的に扱うことが可能なArrayListというクラス、そしてもうひとつはベクターを扱うことのできるPVectorというクラスです。

# ArrayList

ArrayListは配列を扱うためのクラスです。前回は、Processing上での配列の作り方、扱い方は説明させていただきました。それとどう違うのかということですが、同じ配列ではありますが柔軟性がだいぶ違ってきます。最初に言ってしまうと、ArrayListの方が、書き方が少し難しいですが、その分Processing標準の配列と比べると断然柔軟性という意味でおすすめです。

まず、何が根本的に違うかというと、Processing標準の配列は静的配列であるのに対し、ArrayListというクラスは動的配列であるという点です。この「静的」「動的」の意味は、ぱっと聞いた限りでは分からなうと思います。

前回、Processingで配列を作る際には、最初、配列の大きさを先に決めてあげなければならぬと述べました。たとえば、三つのものをいれたい場合は、最初に「三つのものが入る箱である」という風に宣言しなければなりません。そしてそれに加えて、中に入れるものの種類も決めてあげる必要がありました。「三つの整数がはいる箱である」という風に宣言する必要がある、それがProcessing標準の配列です。これをいわゆる静的配列と呼びます。

この静的配列は、一行コードを書くだけで作れてしまうので、非常に簡単に作れて便利です。

```
int[] number = int[3];
```

しかしこの静的配列には二つ欠点があります。一つは、最初にその箱の大きさを決めてしまうため、後でもっとものを入れたいなと思ったときに箱の大きさを大きくできないという点です。もう一つは、整数の箱には整数しか入れられないという制限があるため、ぐちゃぐちゃにいろいろなものを一つの箱に入れることができないという点です。

二つ目の欠点は整理するという意味においては一つの箱には一種類という制限は妥当で、余計なことを考えなくて良いというメリッ

トはありますが、一つ目の欠点に関しては配列を多く扱うにつれて非常に不便に感じる点となってきます。

そこで出てくるのが動的配列というものです。動的配列とは、上に挙げた二つの欠点を取り払った配列のことです。つまり、まるで胃腸のように、ものを入れれば入れるほどその分だけ箱の大きさが膨らんでいき、そしてその中にはどんな種類のものを入れてもかまわない、こうした性質をもった配列のことを動的配列と呼びます。

それがArrayListと呼ばれるものです。そして、それはクラスという形で書かれています。使い方は以下のようになります。

```
1 ArrayList sampleList;
2
3 void setup(){
4   size(100,100);
5   sampleList = new ArrayList();
6   sampleList.add(30.0);
7   sampleList.add("abcde");
8   sampleList.add(0.353);
9   print(sampleList);
10 }
11
12 void draw(){
13   float d;
14   d = (Float)sampleList.get(0);
15   ellipse(width/2, height/2, d,d);
16 }
```

ArrayListのクラスにどのようなサブルーチンやファンクションが含まれているかは、以下のサイトから確認してみてください。非常に多様に配列を扱うことがこのレファレンスからでも分かると思います。

<http://download.oracle.com/javase/1.4.2/docs/api/java/util/ArrayList.html>

# PVector

PVectorというクラスはベクターを扱うクラスです。これを使う前に、まずベクターという概念を理解していただく必要がありますが、ここでは説明しません。知っていることを前提に話をすすめさせていただければと思います。以下のサイトからベクターについての説明がみれますので、そこで少し理解を深めてください。

[http://en.wikipedia.org/wiki/Euclidean\\_vector](http://en.wikipedia.org/wiki/Euclidean_vector)  
<http://ja.wikipedia.org/wiki/空間ベクトル>

まずは基本的な書き方です。以下のように書かれるのが基本です。といっても、クラスを使うということを意識すれば、ArrayListにしてもPVectorにしても書き方は同じです。

```
1 PVector vec1,vec2;
2
3 void setup(){
4   size(200, 200);
5   smooth();
6   vec1 = new PVector(40, 60);
7   vec2 = new PVector(70, 30);
8 }
9 void draw(){
10   ellipse(vec1.x, vec1.y, 20, 20);
11   ellipse(vec2.x, vec2.y, 20, 20);
12   vec2.add(vec1);
13   ellipse(vec2.x, vec2.y, 20,20);
14 }
```

6行目：vec1という名前で、xy座標上のx = 40, y = 60の位置に新しいベクターを作っています。

7行目：vec2という名前で、xy座標上のx = 70, y = 30の位置に新しいベクターを作っています。

10行目：vec1の座標位置 (vec1.xがvec1のx座標上の位置、vec1.yがvec1のy座標上の位置。ここではvec1.x = 40, vec1.y = 60となる) を使って円を描いてます。

11行目：vec2の座標位置 (vec2.xがvec2の

x座標上の位置、vec2.yがvec2のy座標上の位置。ここではvec2.x = 70, vec2.y = 30となる) を使って円を描いてます。

12行目：vec2という名前のベクターにvec1という名前のベクターを足しています。この足すという行為は、そのまま空間ベクターを足し合わせるという考え方をそのまま使えます。

13行目：新しくvec1を足されたvec2の座標を使って新たに円を描いています。

PVectorを使うことで実際何が便利かというと、位置、速度、方向性をベクターを用いて書くことができるようになるため、このクラスを使うことでものを動かすといったことが非常に簡単にできるようになります。上に挙げた例以外にも、ベクター同士を引いたり、ベクターをノーマライズ（長さを1にして方向性だけを持つようにする）したり、ベクターを数値で掛けたり（長さが掛けられる）することができます。

詳しいレファレンスは以下のページにあります。用途に応じていろいろ試してみてください。

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

# 外部ライブラリ

使えるクラスはProcessing標準のクラスだけとは限りません。他の人が作って公開しているクラスなども、自分のツールとして取り込んでしまうことが可能です。この、Processingに標準で入っていない、外部の人が作ったクラス、あるいはクラス集のことを、ライブラリ (library) と呼びます。Processing以外の言語でも、他の人が作ったクラス（ツールボックス）集のことをライブラリと呼びます。今回は使ったほうがいい便利なライブラリを少しだけ紹介するだけにとどめておきます。

## サウンド

ttplib (Processing上で声をシミュレートすることができます。英語ですけれど)  
<http://www.local-guru.net/blog/pages/ttllib>

## アニメーション

gifAnimation (スケッチをG | Fアニメに出力したり、再生したりできます)  
<http://www.extrapixel.ch/processing/gifAnimation/>

## テキスト

Geomerative (TrueType fontを使えるようにしたり、2Dジオメトリを扱いやすくします)  
<http://www.ricardmarxer.com/geomerative>

## 画像認識

OpenCV (顔認識等ができる画像解析ライブラリです。もともとC++で書かれたものです)  
<http://ubaa.net/shared/processing/opencv/>

## 3D

PeasyCam (3D画面を動きやすくしてくれるカメラツールです)  
<http://mrfeinberg.com/peasycam/>

## シミュレーション

Traer Physics (物理演算ライブラリです。スプリングやパーティクルなどがあります)  
<http://www.cs.princeton.edu/%7ETraer/physics/>

fisica (これも物理演算ライブラリです。2D上での物理シミュレーションが簡単にできます)  
<http://www.ricardmarxer.com/fisica/>

kGeom (PVectorよりも多くの機能をもったベクターライブラリです)  
[http://code.google.com/p/kgeom/downloads/detail?name=kGeom\\_1.1.16.zip](http://code.google.com/p/kgeom/downloads/detail?name=kGeom_1.1.16.zip)

## インターフェース

controlP5 (スライダーとかボタンなどのセットが用意されています)  
<http://www.sojamo.de/libraries/controlP5/>

## ネットワーク

oscP5 (OSCというネットワークプロトコルを扱うためのライブラリです)  
<http://www.sojamo.de/libraries/oscP5>

ほかにもいろいろあります。以下のページへ行つていろいろ自分で試してみてください。

<http://processing.org/reference/libraries/>

# 課題

まずは自分で書きはじめるのが重要なことです。ということで、それぞれやりたいことに合わせて、簡単な課題を出そうと思います。

山本さん：

山本さんがつくっていくものは、Swarm Intelligenceと呼ばれるものです。例えば鳥の動きです。鳥一羽一羽があるルールの元に飛んだとき、まるでそこに大きな規則があるよう見えます。ということで、まずは鳥でも蟻でも羊でも何でもいいのですが、一つ生物をピックアップしてもらって、その生物がどのようなルールで動いているかということをリサーチしてください。たとえば鳥であれば、

1. 先頭の鳥にある一定の距離をおいてついていく
2. 先頭の鳥は巣のある方向に向かって飛ぶ
3. 天敵が来たら天敵と真反対に飛ぶ

等のルールを洗い出してください。自分で予想して作ってもらってもかまいません。シンプルであればあるほど良いです。

ライブラリとしては、ベクターを扱うためのライブラリを使っていただきます。まずは、PVectorかKGeomというライブラリをマスターしてください。課題としては、そのどちらかのライブラリを使って、2D画面上を飛び回る円を書いてみてください。どういった規則で、ということはここでは問いません。

大野さん：

大野さんはネットワークとつなげ、ネットワーク上の情報をProcessing上で変換して最後はフィジカルなものとしてアウトプットするということで、Processingだけにとどまらず様々なツールを使うことになります。そしてそれは、その応用性を考えると非常に有意義です。

そこではまず、Twitter4jかTweet Streamというツイッターを扱うライブラリの使い方を覚えてください。課題としては、まずはツイッター上のつぶやきをProcessing上に描写するということをやってみてください。そしてできれば、Processing上の情報をツイッター上につぶやくということも試してください。

Alex：

アレックスのテーマは、最初画像を使ってなにかしらのものを作りたいということでしたが、画像だけならすぐできてしまうので、今回は動画を扱ってもらおうと思います。youtubeの動画を使ってもいいですし、カメラでリアルタイムに撮っている動画をつかっての作品も面白いかもしれません。まずはその制限のもと、なにをしたいかを考えてみてください。

アレックスに使ってほしいライブラリはOpenCVです。まずはこのライブラリの使い方をマスターしてください。そして何か一つ、このライブラリを使った作品を作ってみてください。まずはwebcamで動画をとって、その動画をOpenCVを使って解析し、なにかエフェクトを動画につけてみてください。