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

プログラミング演習2
クラスに関する補足

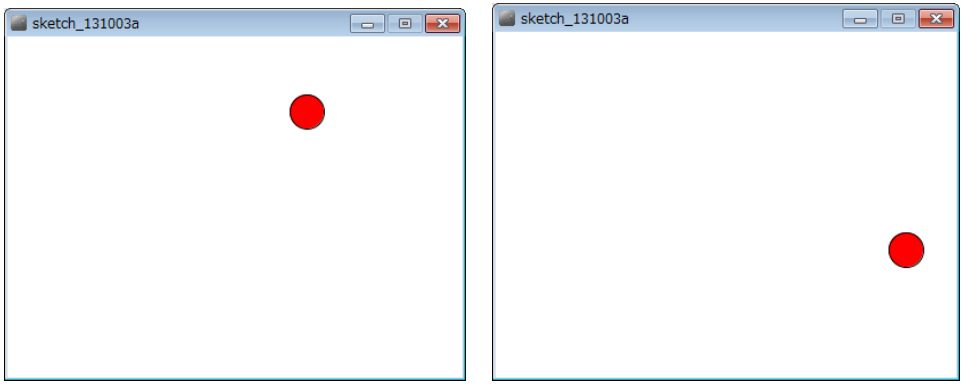
中村，小松，菊池

端で跳ね返る円を描く

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

(Q1) 400x300のウインドウ内で，画面中央から毎フレームx方向に2ピクセル，y方向に3ピクセルずつ移動する直径が30の赤い円が右端・左端・上端・下端に来ると跳ね返るようにするには？

- 考え方
  - 右端・左端・上端・下端で衝突する時の条件を整理
  - 衝突した時の速度を反転させる
    - speedX = -speedX;
    - speedY = -speedY;



端で跳ね返る円を描く

```
int x;
int y;
int speedX;
int speedY;

void setup() {
    size( 400, 300 );
    fill( 255, 0, 0 );

    x = 200;
    y = 150;
    speedX = 2;
    speedY = 3;
}
```

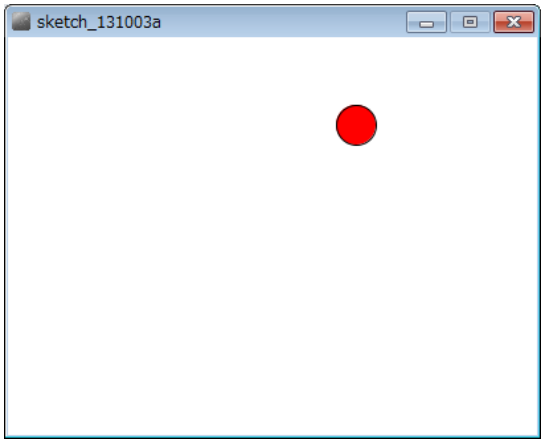
```
void draw() {
    background(255);
    x = x + speedX;
    y = y + speedY;
    if ( x+15 > width ) {
        x = width - 15;
        speedX = -speedX;
    }
    if( x - 15 < 0 ){
        x = 15;
        speedX = -speedX;
    }
    if( y + 15 > height ){
        y = height - 15;
        speedY = -speedY;
    }
    if( y - 15 < 0 ){
        y = 15;
        speedY = -speedY;
    }
    ellipse( x, y, 30, 30 );
}
```

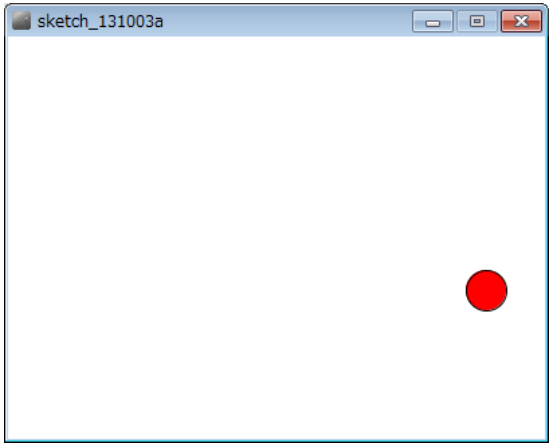
端で跳ね返る円を描く

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

(Q2) (Q1)を改良し，円の位置を画面上任意の場所から毎フレームx，y方向に任意の速度になるようにせよ

- 考え方
  - 任意の場所や速度は random() を使うことで求めることができる





端で跳ね返る円を描く

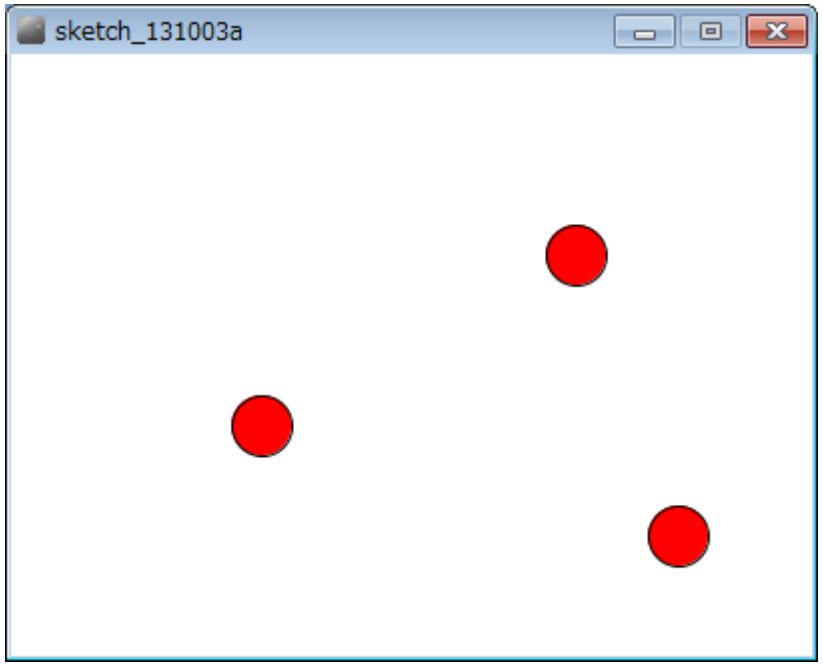
```
int x;
int y;
int speedX;
int speedY;

void setup() {
    size( 400, 300 );
    fill( 255, 0, 0 );

    x = (int)random(width);
    y = (int)random(height);
    speedX = (int)random(5);
    speedY = (int)random(5);
}
```

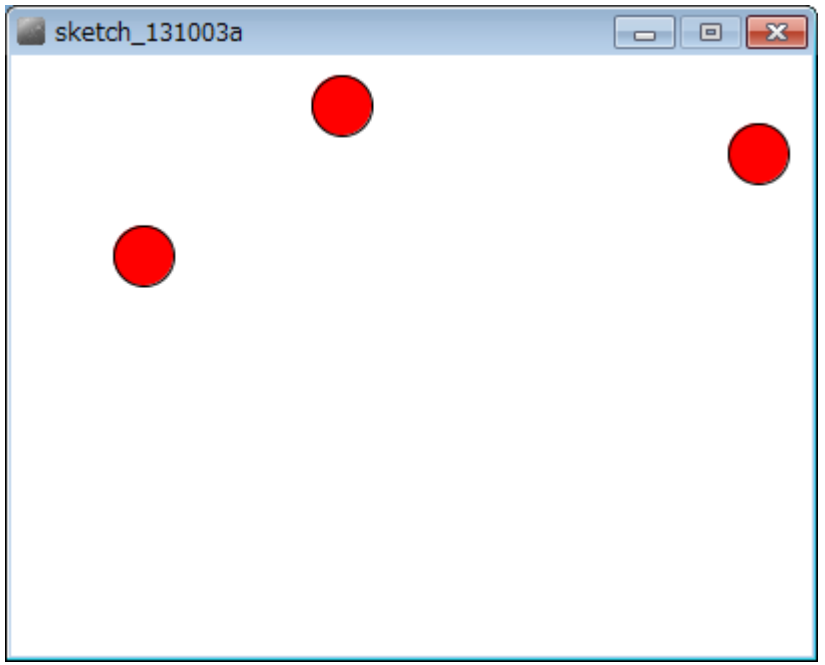
```
void draw() {
    background(255);
    x = x + speedX;
    y = y + speedY;
    if ( x+15 > width ) {
        x = width - 15;
        speedX = -speedX;
    }
    if( x - 15 < 0 ){
        x = 15;
        speedX = -speedX;
    }
    if( y + 15 > height ){
        y = height - 15;
        speedY = -speedY;
    }
    if( y - 15 < 0 ){
        y = 15;
        speedY = -speedY;
    }
    ellipse( x, y, 30, 30 );
}
```

端で跳ね返る3つの円を描く

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

(Q3) (Q2)を改良し，動きまわる円を3つにせよ





端で跳ね返る3つの円を描く

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

(Q2)をそのまま改良すると・・・

- int miyashitaX;
- int miyashitaY;
- int miyashitaSpeedX;
- int miyashitaSpeedY;
- int komatsuX;
- int komatsuY;
- int komatsuSpeedX;
- int komatsuSpeedY;
- int kikuchiX;
- int kikuchiY;
- int kikuchiSpeedX;
- int kikuchiSpeedY;

(komatsuX, komatsuY)
速度: komatsuSpeedX, komatsuSpeedY

多分凄く速い

(miyashitaX, miyashitaY)
速度: miyashitaSpeedX, miyashitaSpeedY

(kikuchiX, kikuchiY)
速度: kikuchiSpeedX, kikuchiSpeedY

こんな感じで
たくさん変数を置くことに

```
int miyashitaX;
int miyashitaY;
int miyashitaSpeedX;
int miyashitaSpeedY;
int komatsuX;
int komatsuY;
int komatsuSpeedX;
int komatsuSpeedY;
int kikuchiX;
int kikuchiY;
int kikuchiSpeedX;
int kikuchiSpeedY;
void setup() {
  size( 400, 300 );
  fill( 255, 0, 0 );
  miyashitaX = (int)random(width);
  miyashitaY = (int)random(height);
  miyashitaSpeedX = (int)random(5);
  miyashitaSpeedY = (int)random(5);
  komatsuX = (int)random(width);
  komatsuY = (int)random(height);
  komatsuSpeedX = (int)random(5);
  komatsuSpeedY = (int)random(5);
  kikuchiX = (int)random(width);
  kikuchiY = (int)random(height);
  kikuchiSpeedX = (int)random(5);
  kikuchiSpeedY = (int)random(5);
}
```

```
void draw() {
  background(255);
  miyashitaX = miyashitaX + miyashitaSpeedX;
  miyashitaY = miyashitaY + miyashitaSpeedY;
  komatsuX = komatsuX + komatsuSpeedX;
  komatsuY = komatsuY + komatsuSpeedY;
  kikuchiX = kikuchiX + kikuchiSpeedX;
  kikuchiY = kikuchiY + kikuchiSpeedY;

  if ( miyashitaX+15 > width ) {
    miyashitaX = width - 15;
    miyashitaSpeedX = -miyashitaSpeedX;
  }
  if ( miyashitaX-15 < 0 ){
    miyashitaX = 15;
    miyashitaSpeedX = -miyashitaSpeedX;
  }
  if ( komatsuX+15 > width ){
    komatsuX = width - 15;
    komatsuSpeedX = -komatsuSpeedX;
  }
        :
        :
        :
   ellipse( miyashitaX, miyashitaY, 30, 30 );
   ellipse( komatsuX, komatsuY, 30, 30 );
   ellipse(  kikuchiX, kikuchiY, 30, 30 );
}
```

条件だけで48行！

ごちゃごちゃに

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

- setup() や draw() の中身がごちゃごちゃになる
  - setup内が14行，draw内が58行に！
- 配列を使う？
  - int [] x = new int [3];
  - int [] y = new int [3];
  - int [] speedX = new int [3];
  - int [] speedY = new int [3];

	0	1	2
x	100	65	45
y	58	150	100
speedX	-4	3	5
speedY	-3	2	-2

  - などのように配列を定義することも考えられるが，それぞれの配列がセットなのに，x[1], y[1], speedX[1], speedY[1] のように横断したものがセットになりわかりにくくなる

クラスで考える

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

- 座標，スピードを持ったオブジェクトを作る
  - ここではx, y座標とspeedX，speedYを持つmiyashitaというBallを考える

定義する

```
class Ball {
    int x;
    int y;
    int speedX;
    int speedY;
}
```

変数を定義

インスタンス化する（使えるようにする）

```
Ball  miyashita;
miyashita = new Ball();
miyashita.x = 200;
miyashita.y = 150;
miyashita.speedX = 5;
miyashita.speedY = 7;
```

変数の定義（箱を作る）

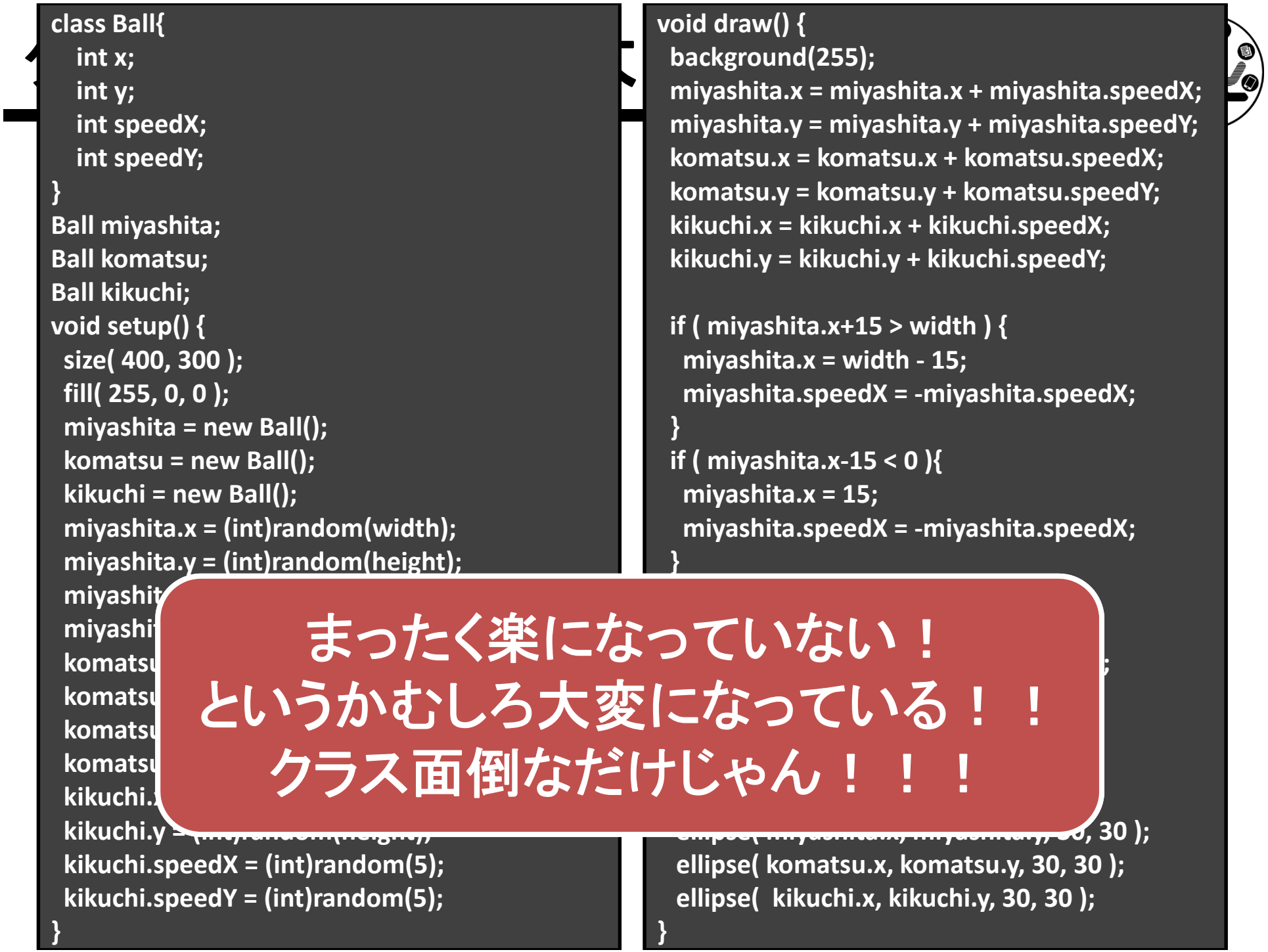
newで具体化

値を代入

描画してみる

```
ellipse( miyashita.x, miyashita.y, 30, 30 );
```

オブジェクト変数名 . インスタンス変数名


class Ball{
int x;
int y;
int speedX;
int speedY;
}
Ball miyashita;
Ball komatsu;
Ball kikuchi;
void setup() {
size(400, 300);
fill(255, 0, 0);
miyashita = new Ball();
komatsu = new Ball();
kikuchi = new Ball();
miyashita.x = (int)random(width);
miyashita.y = (int)random(height);
kikuchi.speedX = (int)random(5);
kikuchi.speedY = (int)random(5);
}
void draw() {
background(255);
miyashita.x = miyashita.x + miyashita.speedX;
miyashita.y = miyashita.y + miyashita.speedY;
komatsu.x = komatsu.x + komatsu.speedX;
komatsu.y = komatsu.y + komatsu.speedY;
kikuchi.x = kikuchi.x + kikuchi.speedX;
kikuchi.y = kikuchi.y + kikuchi.speedY;
if (miyashita.x+15 > width) {
miyashita.x = width - 15;
miyashita.speedX = -miyashita.speedX;
}
if (miyashita.x-15 < 0){
miyashita.x = 15;
miyashita.speedX = -miyashita.speedX;
}
ellipse(komatsu.x, komatsu.y, 30, 30);
ellipse(kikuchi.x, kikuchi.y, 30, 30);
}
まったく楽になっていない！
というかむしろ大変になっている！！
クラス面倒なだけじゃん！！！

インスタンスメソッド

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

- 移動をインスタンスメソッドにしてしまう
  - 全ての円は場所や速度は違うけれど，同じルールで動いているのでまとめることが可能！
  - 内部で勝手に振る舞うメソッド（関数）にしてしまう

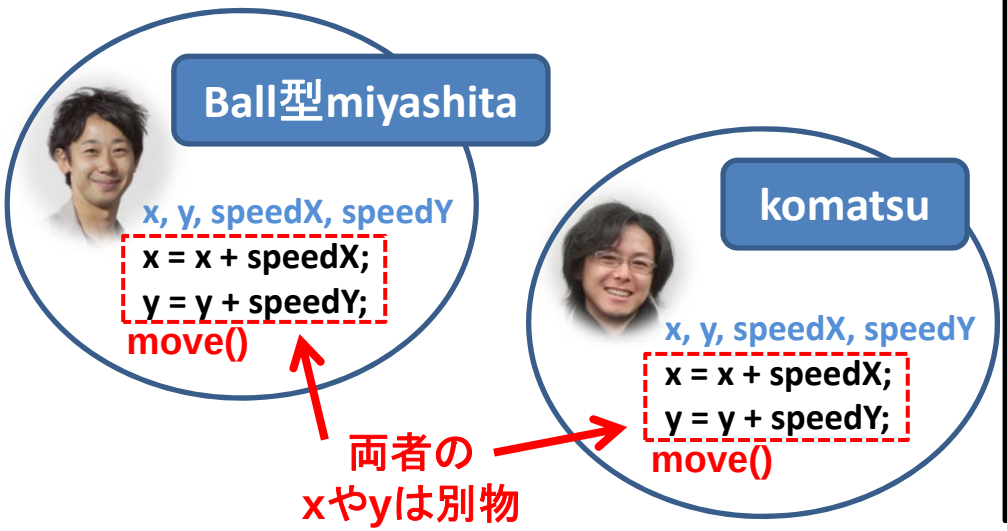
下記のように指定するだけで動くように！

```
miyashita.move();
komatsu.move();
kikuchi.move();
```

void move() を作る

- 移動用メソッドを追加
  - 位置を変更
  - 端に来ると跳ね返る

- インスタンスメソッドではクラス内変数を活用




```
class Ball{
   int x;
   int y;
   int speedX;
   int speedY;
   void move(){
     x = x + speedX;
     y = y + speedY;
     if ( x+15 > width ) {
        x = width - 15;
        speedX = -speedX;
     }
     if( x - 15 < 0 ){
        x = 15;
        speedX = -speedX;
     }
     if( y + 15 > height ){
        y = height - 15;
        speedY = -speedY;
     }
     if( y - 15 < 0 ){
        y = 15;
        speedY = -speedY;
     }
   }
}
```

クラス内で定義されているxやy speedX，speedYを利用したり変更したりできる

改良したBallクラスを使うと

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


```
Ball miyashita;
Ball komatsu;
Ball kikuchi;
void setup() {
  size( 400, 300 );
  fill( 255, 0, 0 );
  miyashita = new Ball();
  komatsu = new Ball();
  kikuchi = new Ball();
  miyashita.x = (int)random(width);
  miyashita.y = (int)random(height);
  miyashita.speedX = (int)random(5);
  miyashita.speedY = (int)random(5);
  komatsu.x = (int)random(width);
  komatsu.y = (int)random(height);
  komatsu.speedX = (int)random(5);
  komatsu.speedY = (int)random(5);
  kikuchi.x = (int)random(width);
  kikuchi.y = (int)random(height);
  kikuchi.speedX = (int)random(5);
  kikuchi.speedY = (int)random(5);
}
```

```
void draw() {
   background(255);
   miyashita.move();
   komatsu.move();
   kikuchi.move();

   ellipse( miyashita.x, miyashita.y, 30, 30 );
   ellipse( komatsu.x, komatsu.y, 30, 30 );
   ellipse(  kikuchi.x, kikuchi.y, 30, 30 );
}
```

draw() がかなり
短くなった！

インスタンスメソッド続き

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

- 最初の位置を設定する部分もインスタンスメソッドにしてしまおう！
  - 初期位置の設定方法は
    - XXXXX.x = (int)random(width);
    - XXXXX.y = (int)random(height);
    - XXXXX.speedX = (int)random(5);
    - XXXXX.speedY = (int)random(5);

void init() で初期化

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

```
class Ball{
   int x;
   int y;
   int speedX;
   int speedY;
   void init(){
      x = (int)random(width);
      y = (int)random(height);
      speedX = (int)random(5);
      speedY = (int)random(5);
   }
```

```
   void move(){
      x = x + speedX;
      y = y + speedY;
      if ( x+15 > width ) {
         x = width - 15;
         speedX = -speedX;
      }
      if( x - 15 < 0 ){
         x = 15;
         speedX = -speedX;
      }
      if( y + 15 > height ){
         y = height - 15;
         speedY = -speedY;
      }
      if( y - 15 < 0 ){
          y = 15;
          speedY = -speedY;
      }
   }
}
```

改良したBallクラスを使うと

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


```
Ball miyashita;
Ball komatsu;
Ball kikuchi;
void setup() {
  size( 400, 300 );
  fill( 255, 0, 0 );
  miyashita = new Ball();
  komatsu = new Ball();
  kikuchi = new Ball();
  miyashita.init();
  komatsu.init();
  kikuchi.init();
}
```

```
void draw() {
  background(255);
  miyashita.move();
  komatsu.move();
  kikuchi.move();

  ellipse( miyashita.x, miyashita.y, 30, 30 );
  ellipse( komatsu.x, komatsu.y, 30, 30 );
  ellipse(  kikuchi.x, kikuchi.y, 30, 30 );
}
```

setup() もかなり
短くなった！

コンストラクタ！

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

- コンストラクタは new されたときに呼び出される場所．init() はそこで呼び出したら良いのでは？

```
class Ball{
    int x;
    int y;
    int speedX;
    int speedY;
    Ball(){

    }
}
```

コンストラクタ！

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

```
class Ball{
   int x;
   int y;
   int speedX;
   int speedY;
   Ball(){
      init();
   }

   void init(){
      x = (int)random(width);
      y = (int)random(height);
      speedX = (int)random(5);
      speedY = (int)random(5);
   }
```

コンストラクタで
initメソッドを呼び出す！

```
   void move(){
      x = x + speedX;
      y = y + speedY;
      if ( x+15 > width ) {
         x = width - 15;
         speedX = -speedX;
      }
      if( x - 15 < 0 ){
         x = 15;
         speedX = -speedX;
      }
      if( y + 15 > height ){
         y = height - 15;
         speedY = -speedY;
      }
      if( y - 15 < 0 ){
          y = 15;
          speedY = -speedY;
      }
   }
}
```

改良したBallクラスを使うと

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


```
Ball miyashita;
Ball komatsu;
Ball kikuchi;

void setup() {
  size( 400, 300 );
  fill( 255, 0, 0 );
  miyashita = new Ball();
  komatsu = new Ball();
  kikuchi = new Ball();
}
```

```
void draw() {
  background(255);
  miyashita.move();
  komatsu.move();
  kikuchi.move();

  ellipse( miyashita.x, miyashita.y, 30, 30 );
  ellipse( komatsu.x, komatsu.y, 30, 30 );
  ellipse(  kikuchi.x, kikuchi.y, 30, 30 );
}
```

setup() がさらに
短くなった！

折角なので

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

- 描画もインスタンスメソッドにしてしまおう
  - ellipse(miyashita.x, miyashita.y, 30, 30);
  - ellipse(komatsu.x, komatsu.y, 30, 30);
  - ellipse(kikuchi.x, kikuchi.y, 30, 30);

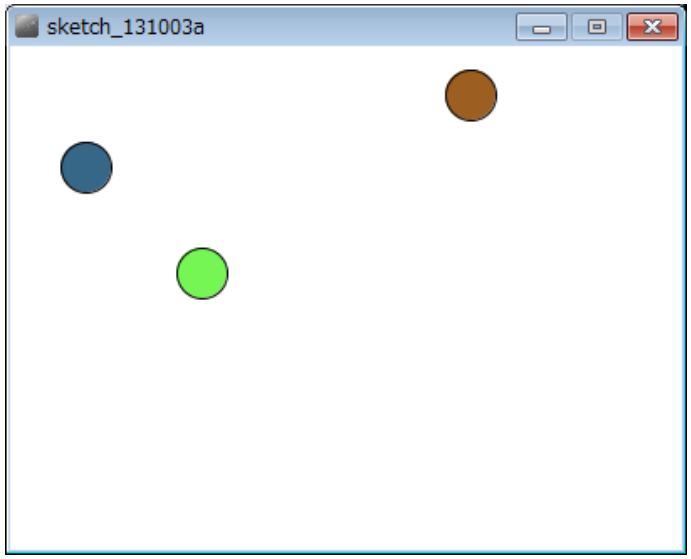
```
class Ball{
   int x;
   int y;
   int speedX;
   int speedY;
   void display(){
      ellipse( x, y, 30, 30 );
   }
        :
```

```
void draw() {
   background(255);
   miyashita.move();
   komatsu.move();
   kikuchi.move();

   miyashita.display();
   komatsu.display();
   kikuchi.display();
}
```

各々で色を変えてみよう！

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

- 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;




```
class Ball{
   int x;
   int y;
   int speedX;
   int speedY;
   int red;
   int green;
   int blue;
   void display(){
      fill( red, green, blue );
      ellipse( x, y, 30, 30 );
   }
   void init(){
      x = (int)random(width);
      y = (int)random(height);
      speedX = (int)random(5);
      speedY = (int)random(5);
      red = (int)random(255);
      green = (int)random(255);
      blue = (int)random(255);
   }
```

名前を表示してみよう

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

- 名前用のインスタンス変数を追加
  - String name;
- 名前をセットするインスタンスメソッドを追加
  - void setName(String s);

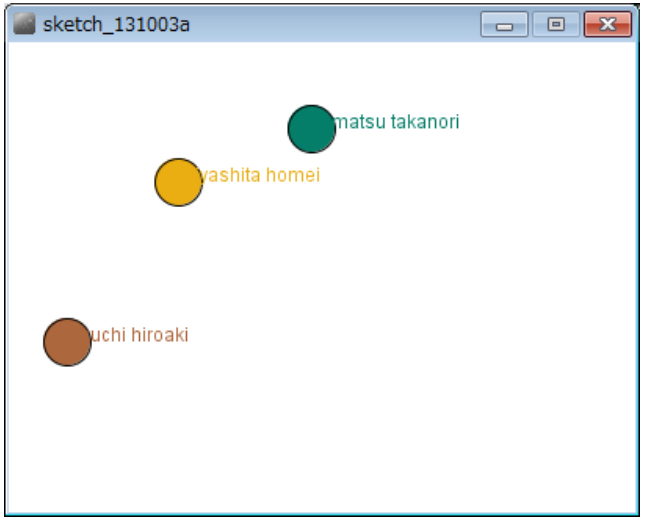
```
class Ball{
  int x;
  int y;
  int speedX;
  int speedY;
  int red;
  int green;
  int blue;
  String name;
  void display(){
    fill( red, green, blue );
    ellipse( x, y, 30, 30 );
    text( name, x, y );
  }
  void setName( String s ){
    name = s;
  }
```

名前を表示してみよう

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

- setName メソッドを利用して名前を設定

```
Ball miyashita;
Ball komatsu;
Ball kikuchi;
void setup() {
  size( 400, 300 );
  fill( 255, 0, 0 );
  miyashita = new Ball();
  komatsu = new Ball();
  kikuchi = new Ball();
  miyashita.setName( "miyashita homei" );
  komatsu.setName( "komatsu takanori" );
  kikuchi.setName( "kikuchi hiroaki" );
}
```



コンストラクタで名前を設定

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

- setName で名前を設定したが，コンストラクタ（最初に呼び出される部分）で名前を設定することも可能

```
Ball miyashita;
Ball komatsu;
Ball kikuchi;
void setup() {
   miyashita = new Ball("miyashita homei");
   komatsu = new Ball("komatsu takanori");
   kikuchi = new Ball("kikuchi hiroaki");
}
```

```
class Ball{
   int x;
   int y;
   int speedX;
   int speedY;
   int red;
   int green;
   int blue;
   String name;

   Ball( String s ){
      name = s;
      init();
   }
```

コンストラクタの
引数を変更

コンストラクタの不思議

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

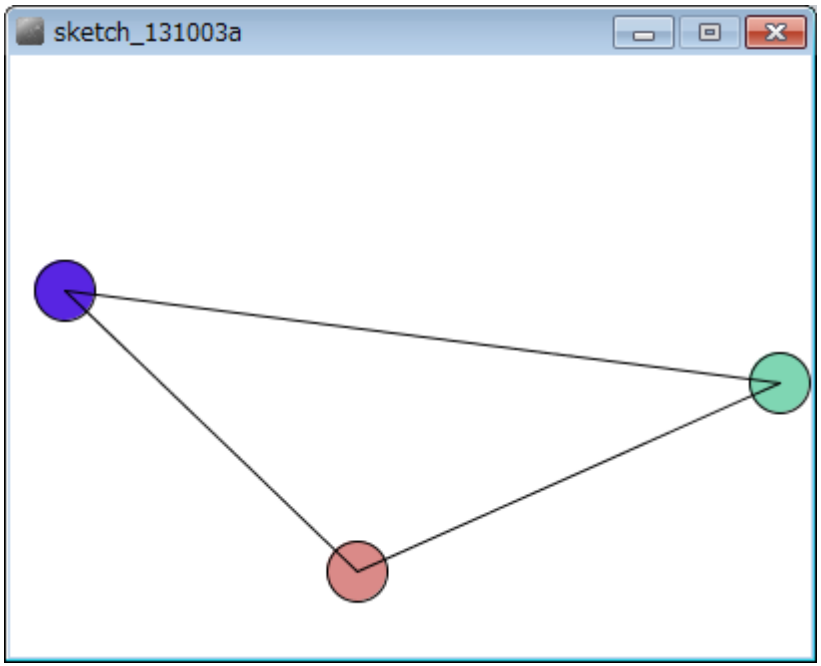
- 何故 void Ball(){ … } じゃないの？
  - コンストラクタは，そもそも返り値（returnで返されるもの）が存在しない．そのため，返り値に関する設定が不要
- Ball(){…}とBall(int x, int y){…}とBall(String s){…} どれが正しいの？
  - どれでもOK
  - new のときに，どう呼び出すかの違い
    - Ball() のコンストラクタは Ball b = new Ball();
    - Ball(int x, int y) は Ball b = new Ball(500, 100);
    - Ball(String s) は Ball b = new Ball("miyashita");

    で，それぞれ呼び出される．

静的メソッドで線をつなぐ

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

- 3つの円を繋ぐ線を描画してみよう
  - 返り値の無い void drawEdges(Ball a, Ball b, Ball c); を作って，呼び出す




```
void drawEdges( Ball a, Ball b, Ball c ){
  line( a.x, a.y, b.x, b.y );
  line( b.x, b.y, c.x, c.y );
  line( c.x, c.y, a.x, a.y );
}

void draw() {
  background(255);
  miyashita.move();
  komatsu.move();
  kikuchi.move();
  miyashita.display();
  komatsu.display();
  kikuchi.display();
  drawEdges( miyashita, komatsu, kikuchi );
}
```

沢山の円を描くには？

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

```
class Ball{
  int x, y;
  int speedX, speedY;
  int red;
  int green;
  int blue;
  Ball(){
    init();
  }
  void display(){
    fill( red, green, blue );
    ellipse( x, y, 30, 30 );
  }
  void init(){
    x = (int)random(width);
    y = (int)random(height);
    speedX = (int)random(5);
    speedY = (int)random(5);
    red = (int)random(255);
    green = (int)random(255);
    blue = (int)random(255);
  }
```

String name; と
setName() は削除
コンストラクタも変更

```
  void move(){
    x = x + speedX;
    y = y + speedY;
    if ( x+15 > width ) {
      x = width - 15;
      speedX = -speedX;
    }
    if( x - 15 < 0 ){
      x = 15;
      speedX = -speedX;
    }
    if( y + 15 > height ){
      y = height - 15;
      speedY = -speedY;
    }
    if( y - 15 < 0 ){
      y = 15;
      speedY = -speedY;
    }
  }
}
```

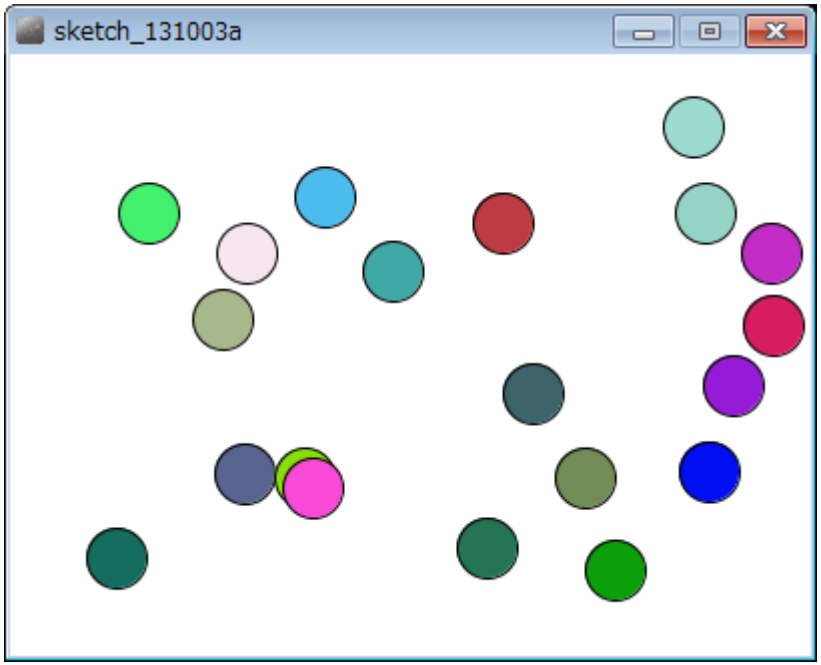
オブジェクト＋配列

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

- 20個の丸を動かしてみる

```
Ball [] balls = new Ball [20];

void setup() {
  size( 400, 300 );
  fill( 255, 0, 0 );
  for( int i=0; i<20; i++ ){
    balls[i] = new Ball();
  }
}

void draw(){
  for( int i=0; i<20; i++ ){
    balls[i].move();
    balls[i].display();
  }
}
```

型 [] 配列名 = new 型 [要素数];



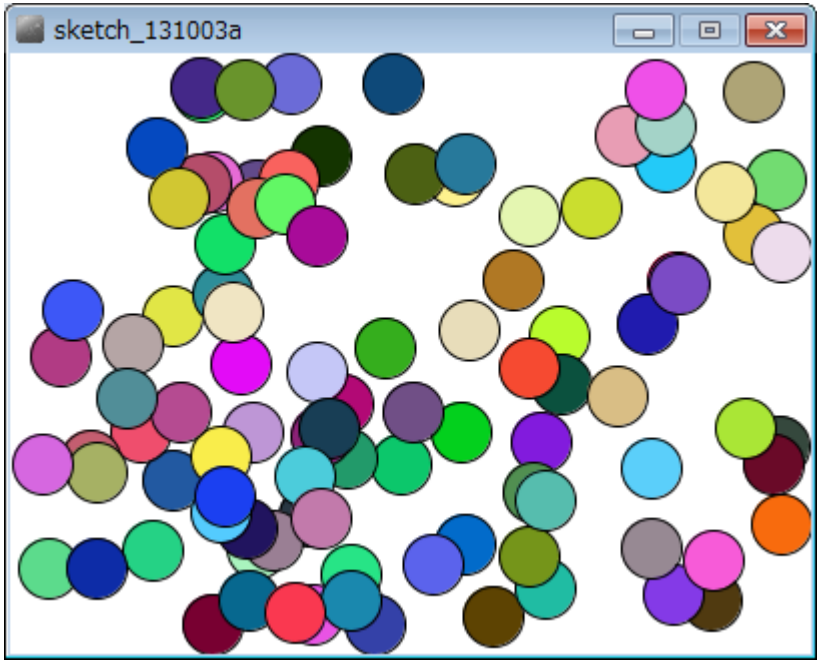
オブジェクト＋配列

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

- 100個の丸を動かしてみる

```
Ball [] balls = new Ball [100];

void setup() {
  size( 400, 300 );
  fill( 255, 0, 0 );
  for( int i=0; i<balls.length; i++ ){
    balls[i] = new Ball();
  }
}

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

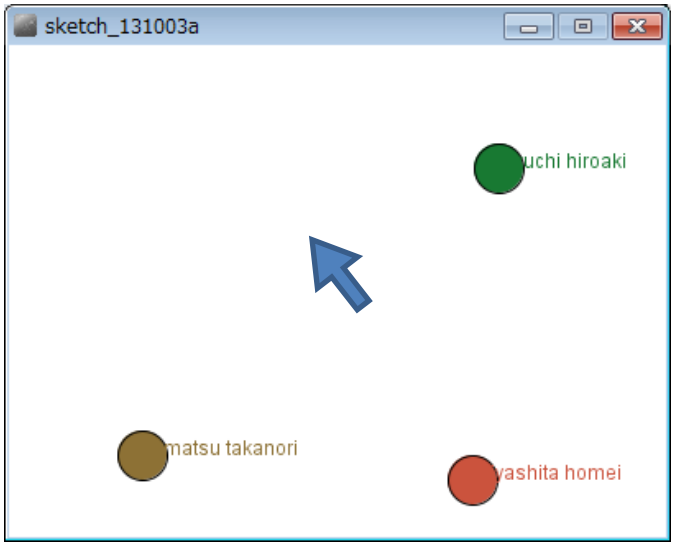
配列変数名.length
で配列の長さを取得



静的メソッドとは？

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

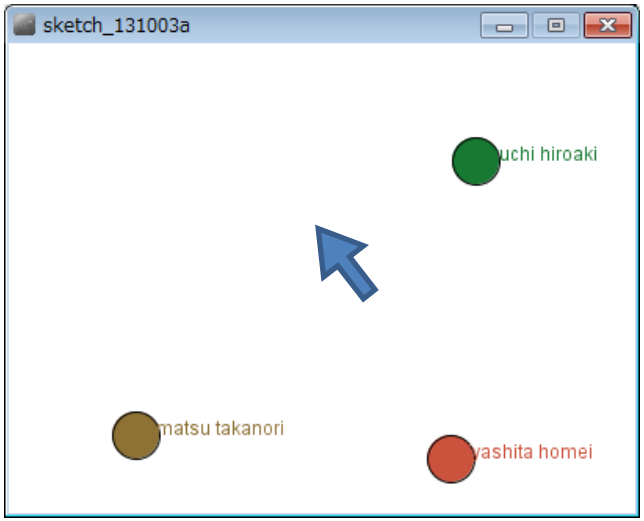
- インスタンスメソッドは，主にそのオブジェクト自体の振る舞いを記述するもの
  - オブジェクトの一人称で実行されるメソッド
- 静的メソッドは，主にその世界の振る舞いを記述するもの
  - オブジェクト集合を客観的に見て実行されるメソッド



メソッドの違い

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

- マウスカーソルと各オブジェクトの関係を考える
  - インスタンスメソッド：オブジェクト自身とマウスカーソルとの距離を計算する
    - miyashita.distance(mouseX, mouseY);
  - 静的メソッド：マウスカーソルからどのオブジェクトが最も近いかを計算する
    - nearest(mouseX, mouseY, miyashita, komatsu, kikuchi);



インスタンス/静的メソッド

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

- インスタンスメソッドは各々で動作する
  - 独自の振る舞いをもたせる時に便利
  - 「適当に移動して」と指示する感覚に近い

- 静的メソッドは統一的に動作する
  - 統一的な振る舞いをもたせたり，その世界の中における色々なものを合わせて処理する時に便利
  - 「こう動いて」と命令する感覚に近い

インスタンス/静的メソッド

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

- 円の移動（跳ね返りなし）のみを考えた場合

インスタンスメソッド

```
class Ball {
   int x;
   int y;
   int speedX;
   int speedY;
   void move(){
      x += speedX;
      y += speedY;
   }
}
      :
Ball ball;
void draw(){
   ball.move();
}
```

静的メソッド

```
class Ball {
   int x;
   int y;
   int speedX;
   int speedY;
}

Ball ball;
void move( Ball obj ){
    obj.x += obj.speedX;
    obj.y += obj.speedY;
}
void draw(){
   move( ball );
}
```

オブジェクト指向とは

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

- ざっくり説明すると，色々な値や機能をもつもの

  （例）シューティングゲーム上の敵

  - 現在位置（X座標，Y座標）
  - 何らかの移動機能（移動の関数）
  - 何らかの描画機能（描画の関数）

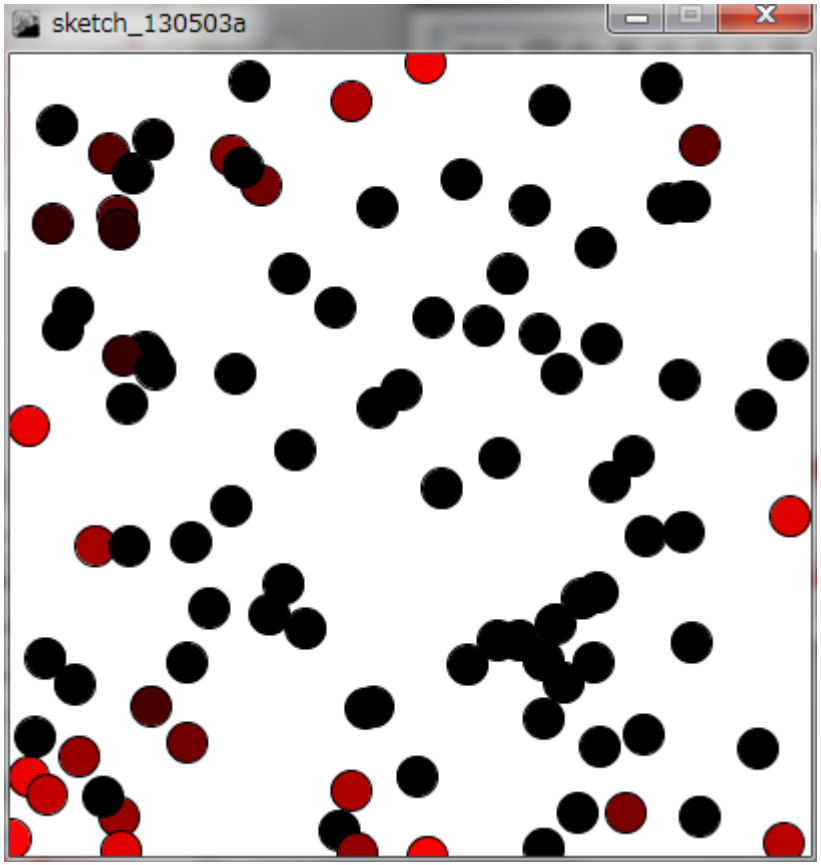
  をもっており，プログラムから移動しろ，描画しろと命令を送るだけで，その中身がどうなっているかを意識せずに利用可能

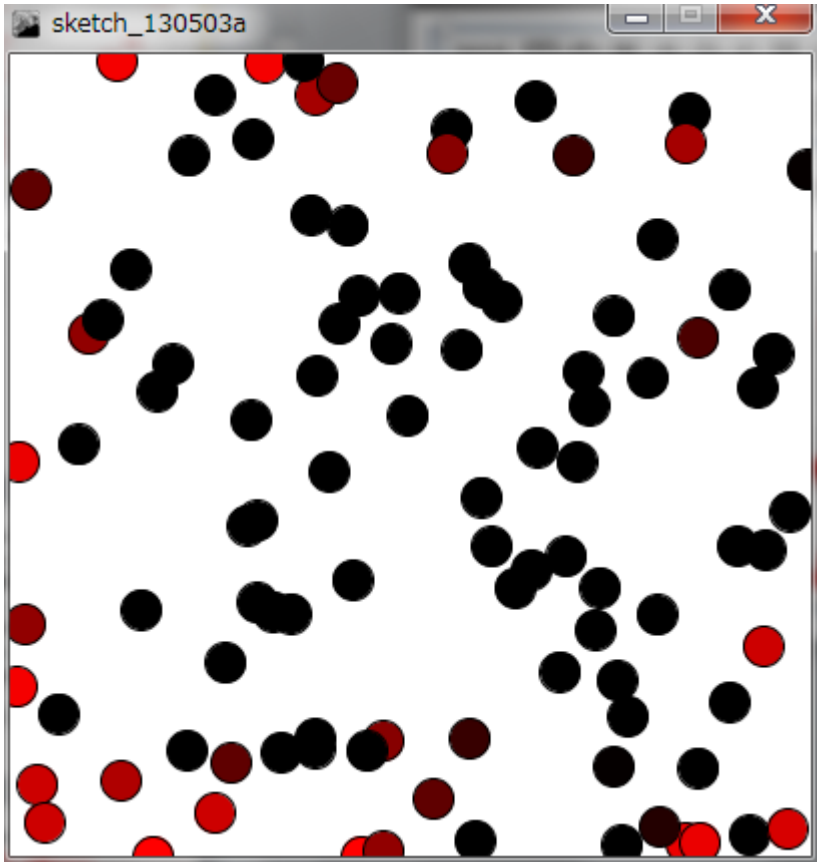
- 他人が何をどう考え実行するかを気にせず，「～をやっておいて」とお願いする感覚

色を変化させる

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

(Q) 動きまわる円の基本色を黒色にし，画面の端に衝突したら塗り色を赤色に変えるようにする





色を変化させる

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

- 考え方
  - 黒色はRGBで（0, 0, 0）と赤色はRGBで（255, 0, 0）なので，衝突したらRの量を255にし，徐々に0へと近づけていく
  - 円の描画 paint() 内で fill 関数を利用して塗りつぶし色を指定する
    - fill(fillR, 0, 0);
  - 赤色の塗りつぶし量を記憶する整数型の fillR という変数を用意し，初期値は 0 にしておく
  - 衝突したら fillR の値を 255 にする
  - 描画される度に，fillR が 0 より大きければ fillR を徐々に減算する（-10ずつ，-5ずつなど）

```
class Circle {
    float  posX;
    float  posY;
    float  speedX;
    float  speedY;
    int  fillR;
    Circle( float x, float y, float sx, float sy ){
        posX = x;
        posY = y;
        speedX = sx;
        speedY = sy;
        fillR = 0;
    }
    void  paint(){
        fill( fillR, 0, 0 );
        ellipse( posX, posY, 20, 20 );
        if( fillR > 0 ){
            fillR -= 5;
        } else if( fillR < 0 ){
            fillR = 0;
        }
    }
    void move() {
        posX += speedX;
        posY += speedY;
        if ( posX > width ) {
            posX = width;
            speedX = -speedX;
            fillR = 255;
        } else if ( posX < 0 ) {
            posX = 0;
            speedX = -speedX;
            fillR = 255;
        }
        if ( posY > height ) {
            posY = height;
            speedY = -speedY;
            fillR = 255;
        } else if ( posY < 0 ) {
            posY = 0;
            speedY = -speedY;
            fillR = 255;
        }
    }
}
```

予習問題

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

- 先述のプログラムを改良し，端に衝突すると黒くなり，徐々に薄くなるようにしましょう
  - ヒント：白は（255, 255, 255），黒は（0, 0, 0）

- 棒人間クラスを作ってみよう
  - 棒人間クラスに移動と，描画の機能を導入しよう
  - 棒人間が端まで行くとしばらく止まり，再度別方向に動き出すようにしてみましょう
    - 動かない時間をカウントする変数 wait などを導入！

予習問題

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

- 過去に作ったロボットをクラス化しましょう！
  - ロボット全体をクラスにして，動かす！と命令を送るだけで，柔軟に動くようにしましょう
- 過去に作ったゲームでそれぞれのオブジェクトをクラス化しましょう！
  - 敵と味方の分離など