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（1）

中村，高橋，小林，橋本

例：動物園を実現する

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

- 猫，犬，猿，象，熊がいる動物園を再現する
  - 表示場所に関する情報

    猫(catX, catY), 犬(dogX, dogY), 猿(monkeyX, monkeyY),
    象(elephantX, elephantY), 熊(bearX, bearY)
  - それぞれの動物の描画

    drawCat(), drawDog(), drawMonkey(), drawElephant(), drawBear();
  - それぞれの動物の移動

    moveCat(), moveDog(), moveMonkey(), moveElephant(), moveBear();
  - それぞれの動物の睡眠

    sleepCat(), sleepDog(), sleepMonkey(), sleepElephant(),
    sleepBear();
  - などを用意し，それぞれをプログラムの内部から呼び出す必要がある

かなり面倒で混乱のもと

愚直に書くとどうなる？

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

- miyashita, komatsu, fukuchi という3つのボールが動き回るような世界をプログラミングせよ

http://nkmr.io/lecture/

から

ball_original.pde

をダウンロードしよう

ball_original

```
int miyashita_posX;
int miyashita_posY;
int miyashita_speedX;
int miyashita_speedY;
int komatsu_posX;
int komatsu_posY;
int komatsu_speedX;
int komatsu_speedY;
int fukuchi_posX;
int fukuchi_posY;
int fukuchi_speedX;
int fukuchi_speedY;

void setup() {
  size( 400, 300 );
  miyashita_posX = (int)random( 0, width );
  miyashita_posY = (int)random( 0, height );
  miyashita_speedX = (int)random( 1, 5 );
  miyashita_speedY = (int)random( 1, 5 );

  komatsu_posX = (int)random( 0, width );
  komatsu_posY = (int)random( 0, height );
  komatsu_speedX = (int)random( 1, 5 );
  komatsu_speedY = (int)random( 1, 5 );

  fukuchi_posX = (int)random( 0, width );
  fukuchi_posY = (int)random( 0, height );
  fukuchi_speedX = (int)random( 1, 5 );
  fukuchi_speedY = (int)random( 1, 5 );

  fill( 255, 0, 0 );
}

void draw() {
  background( 255 );
```

動きは動物に任せたい

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

- 猫，犬，猿，象，熊を定義
  - それぞれの座標は意識したくない
    - cat.x, cat.y, dog.x, dog.y, monkey.x, monkey.y, ...
    - 内部で適当に処理してもらう
  - 描画はシンプルにしたい
    - cat.draw(), dog.draw(), monkey.draw(), elephant.draw(), ...
  - 移動もシンプルにしたい
    - cat.move(), dog.move(), monkey.move(), elephant.move(), ...
  - 睡眠も任せてしまう
    - cat.sleep(), dog.sleep(), monkey.sleep(), elephant.sleep(), ...

すべてを XXXX . 機能 という形に！

オブジェクト指向（クラス）

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（変数の状態）や処理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

たとえば

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

- 人間というクラスを定義する
  - 人間には名前という変数
  - 現在地という場所に関する変数
  - 年齢という変数

  などがある
- 人間には下記のメソッドがある
  - 移動する
  - 食べる
  - 喋る
  - 聞く

  など

たとえば

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

- 人間というクラスを使って、「小松」「宮下」というものを具現化する（インスタンス化する）
- 「小松」や「宮下」が持つ機能をインスタンスメソッドと呼ぶ
- 「小松」や「宮下」の情報（変数）をインスタンス変数と呼ぶ（人間に共通のものは静的変数）
- 人間に共通のメソッド（なんという生き物か？というメソッド）を静的メソッドと呼ぶ

Processingには静的メソッド／静的変数はない

変数を直接触らない

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

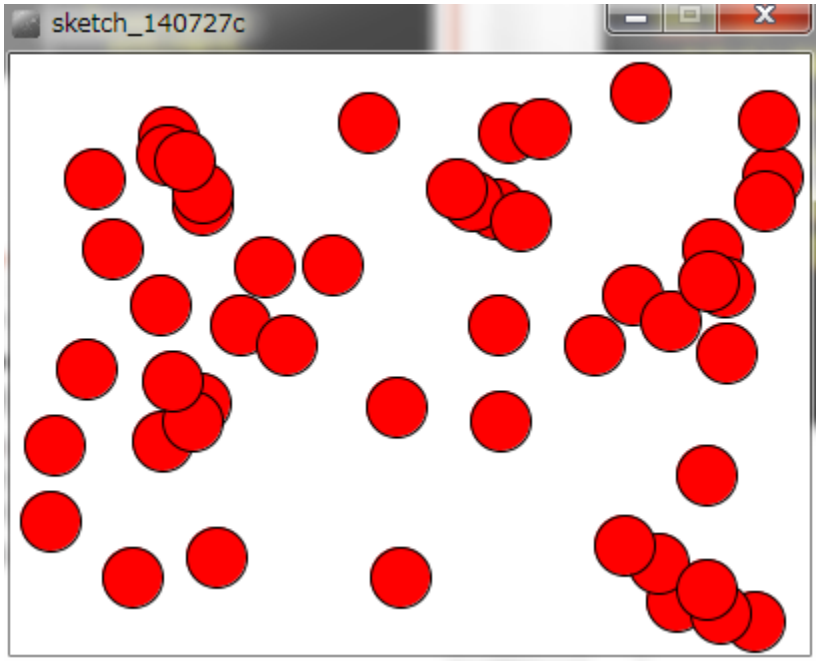
- メソッドだけで色々と制御できるように！
- 時計を実現することを考える
  - なかの制御系を直接動かさない
  - 時・分・秒を変更するために、ダイヤルを引っ張り出し、ダイヤルを回す
  - ボタンを押すことでアラームのセットなど

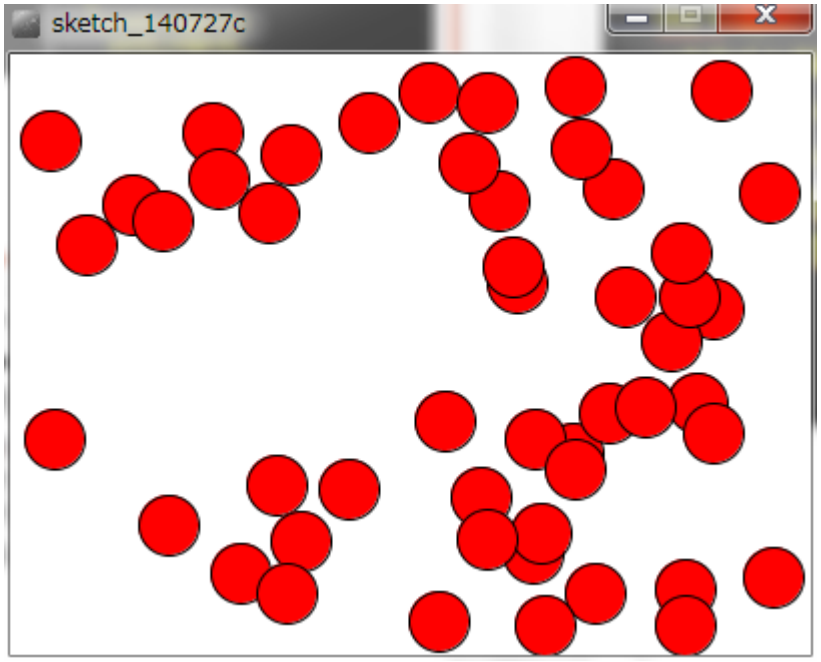
端で跳ね返る50個のボール

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

400x300の画面内にX座標Y座標ともにランダムに配置された50個の円を描き，それぞれのX，Y方向へのスピードに応じて移動するようにせよ

- posX，posY という座標の配列と，speedX，speedY というXおよびY方向の速度をもつ配列を導入！





```
int  []  posX = new int[50];
int  []  posY = new int[50];
int  [] speedX = new int[50];
int  [] speedY = new int[50];

void setup()
{
  size( 400, 300 );
  int i=0;
  while( i<50 )
  {
    posX[i] = (int)random(0,width);
    posY[i] = (int)random(0,height);
    speedX[i] = (int)random(1,5);
    speedY[i] = (int)random(1,5);
    i++;
  }
}
```

```
void draw()
{
  background( 255 );
  fill( 255, 0, 0 );
  int i=0;
  while( i<50 ){
    posX[i] += speedX[i];
    posY[i] += speedY[i];
    if( posX[i] > width-15 ){
      posX[i] = width-15;
      speedX[i] = -speedX[i];
    }
    if( posX[i] < 15 ){
      posX[i] = 15;
      speedX[i] = -speedX[i];
    }
    if( posY[i] > height-15 ){
      posY[i] = height-15;
      speedY[i] = -speedY[i];
    }
    if( posY[i] < 15 ){
      posY[i] = 15;
      speedY[i] = -speedY[i];
    }
    ellipse( posX[i], posY[i], 30, 30 );
    i++;
  }
}
```

問題ないけれど・・・

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

- 変数（配列）が沢山でちょっと分かりにくい
  - posX[n] と posY[n] と speedX[n] と speedY[n] がセットだけれど，それぞれ配列として独立しているし，なんだかよくわからない
  - 条件分岐もいろいろあって draw 内がわかりにくい

クラスの定義

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 クラスを定義

  class クラス名 {

  　クラスの諸要素に関する定義

  }

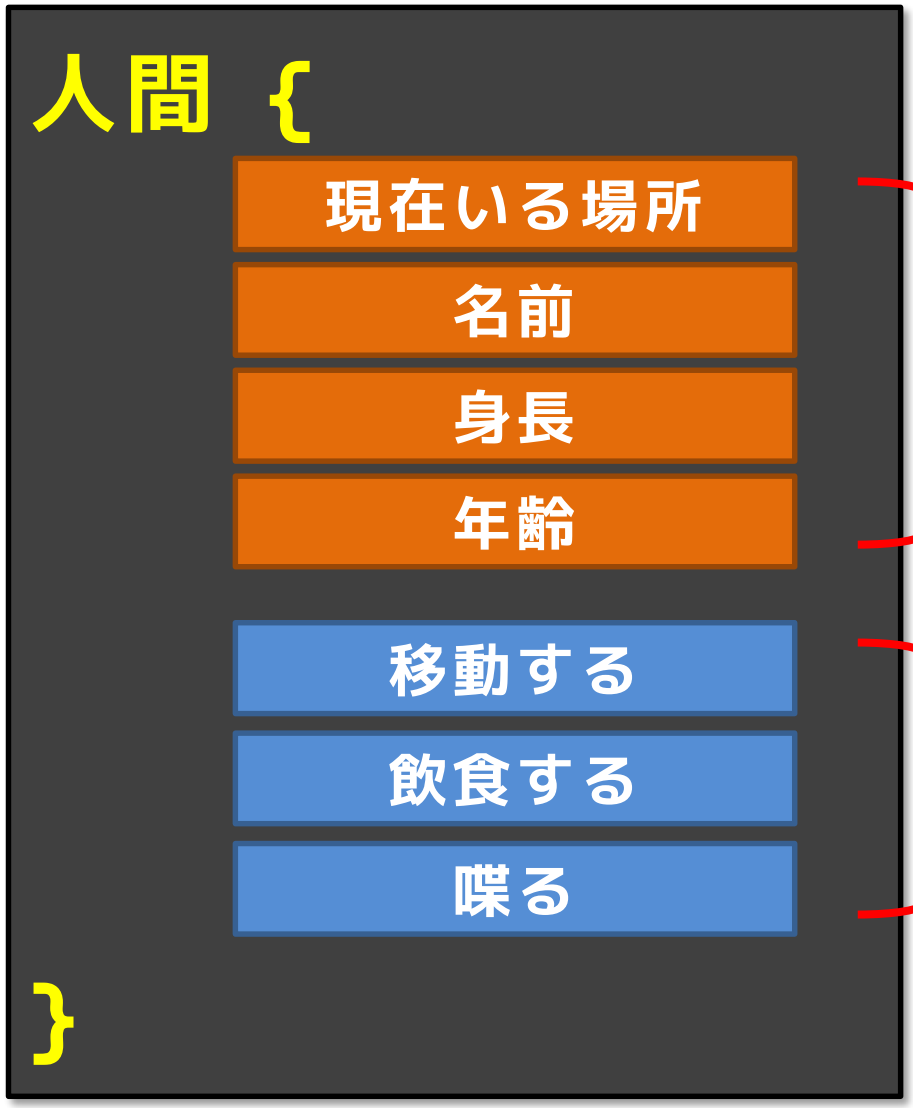
```
class Ball {

}
```

```
class Human {

}
```

```
class Animal {

}
```

クラスの定義

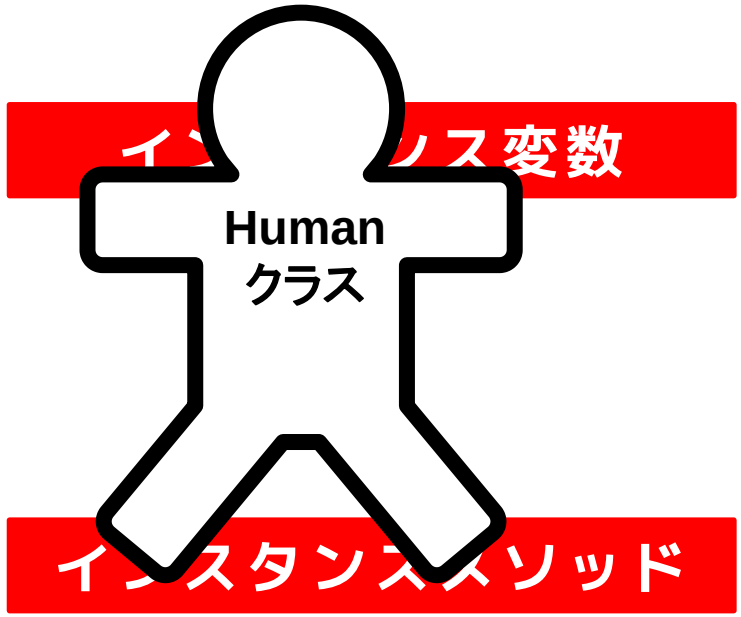
明治大学総合数理学部
先端メディアサイエンス学科
中村研究室



クラスの定義

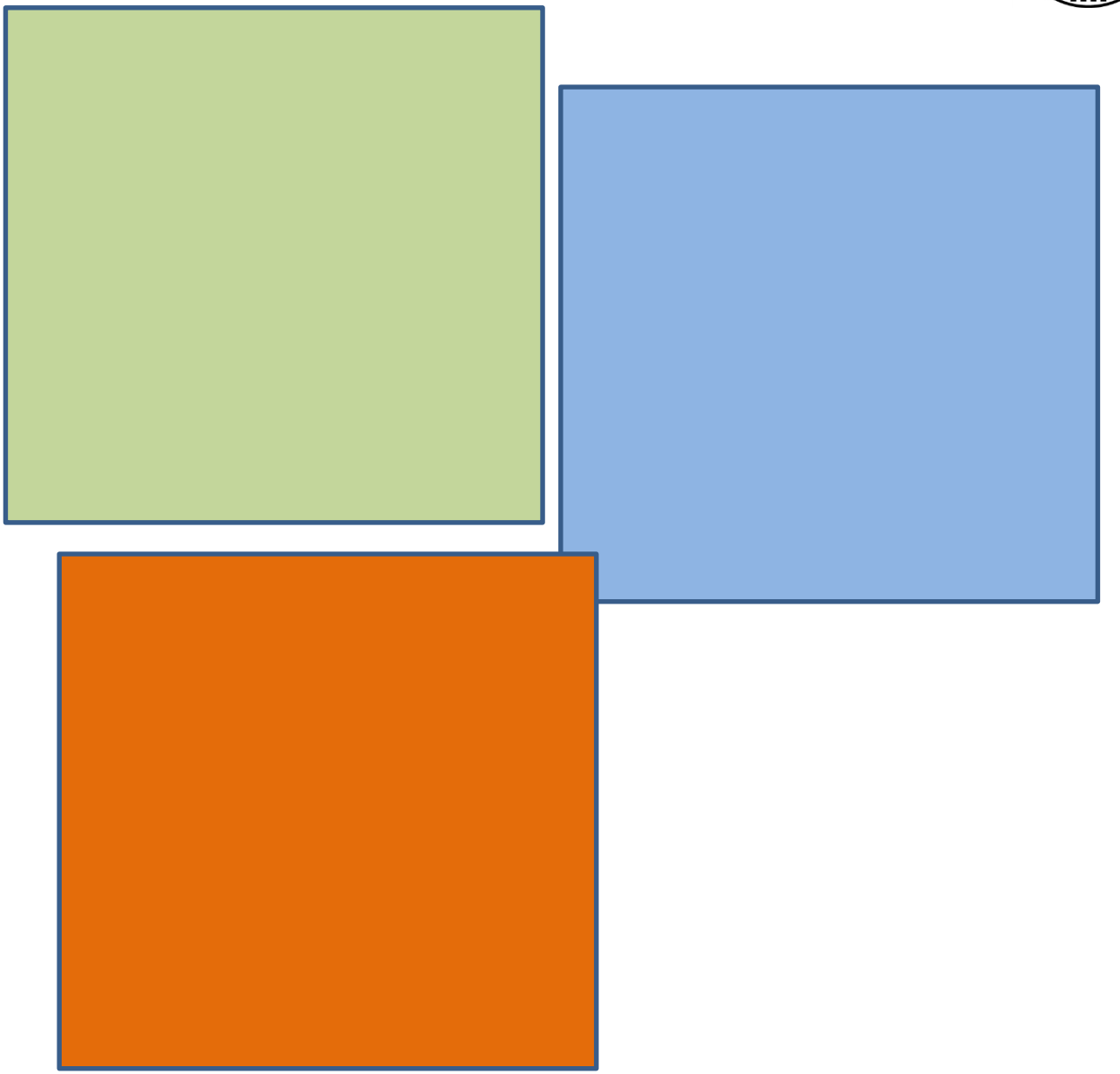
明治大学総合数理学部
先端メディアサイエンス学科
中村研究室

```
class Human {
    int lon, lat;
    String name;
    float bodyHeight;
    int age;

    void move();
    void eat();
    String say();
}
```



インスタンス化

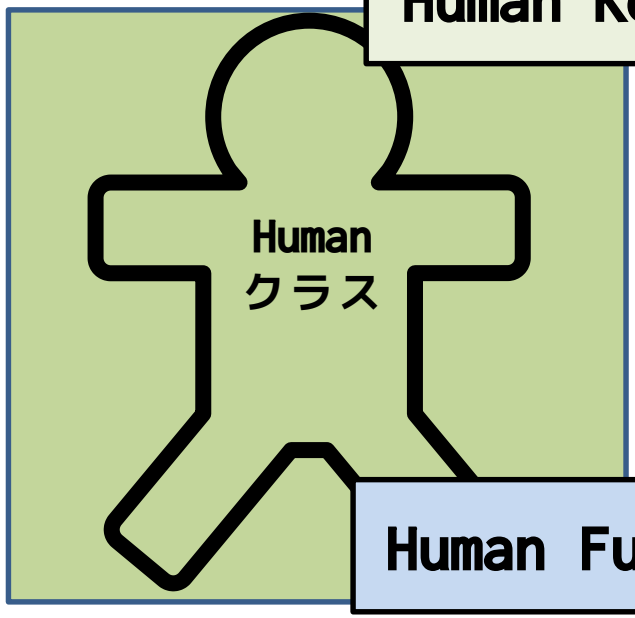
明治大学総合数理学部
先端メディアサイエンス学科
中村研究室

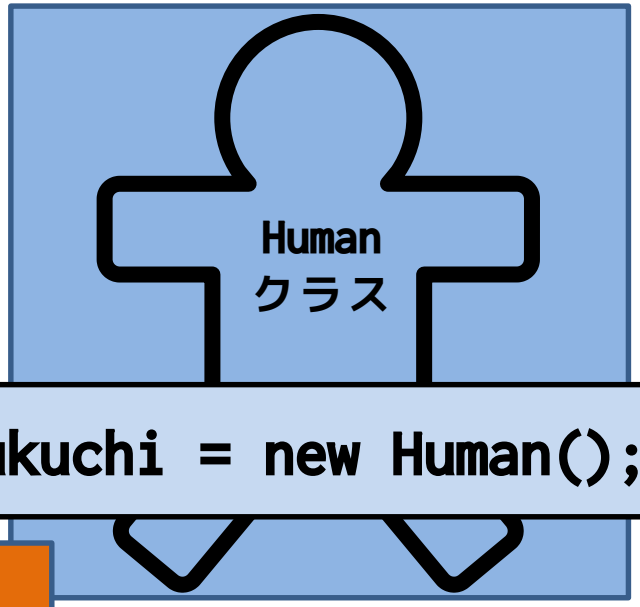


インスタンス化

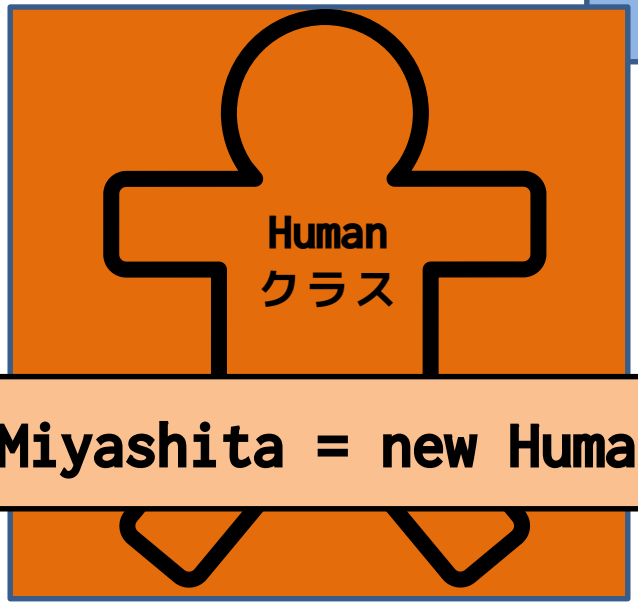
明治大学総合数理学部
先端メディアサイエンス学科

```
Human Komatsu = new Human();
```






```
Human Fukuchi = new Human();
```

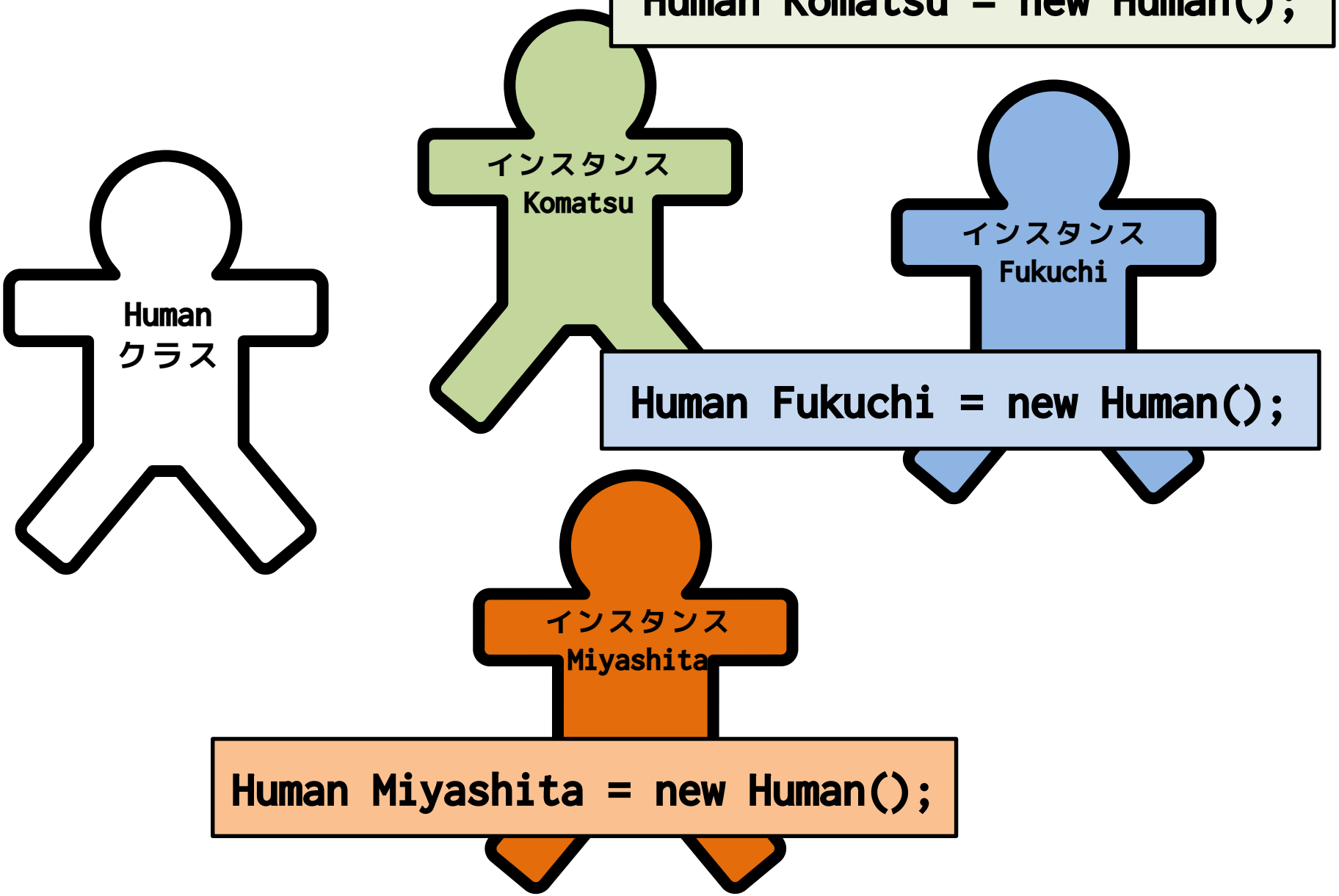



```
Human Miyashita = new Human();
```

インスタンス化

明治大学総合数理学部
先端メディアサイエンス学科

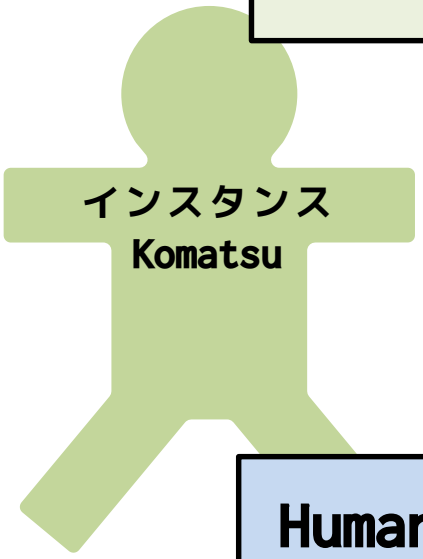
```
Human Komatsu = new Human();
```

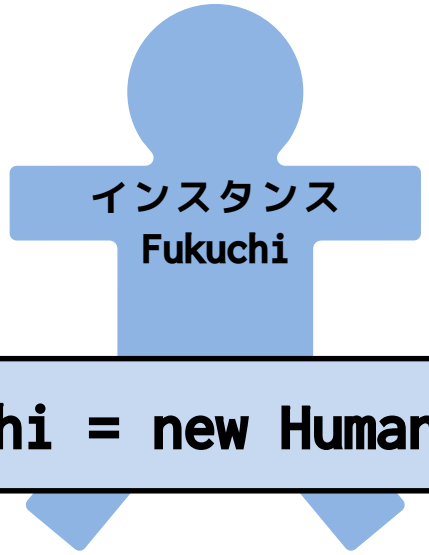


インスタンス化

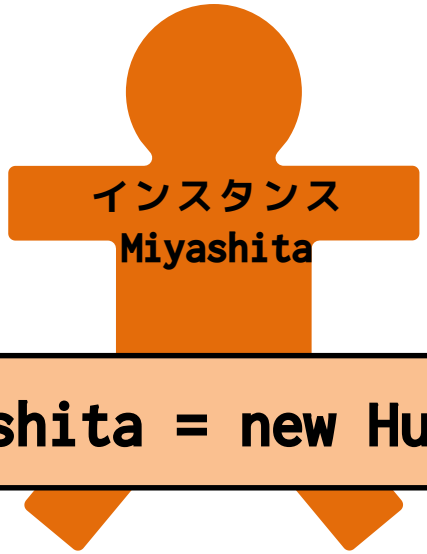
明治大学総合数理学部
先端メディアサイエンス学科


```
Human Komatsu = new Human();
```





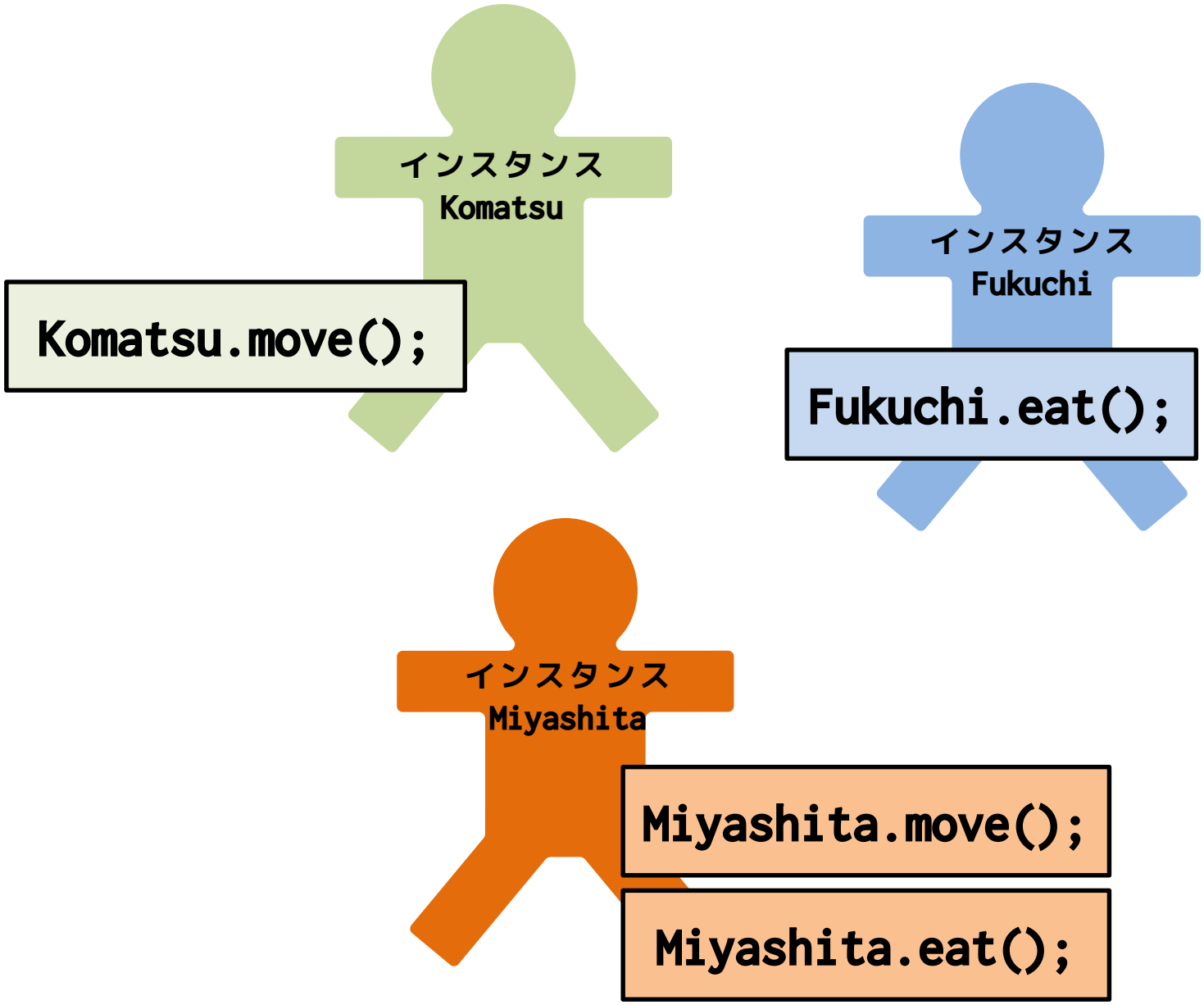

```
Human Fukuchi = new Human();
```




```
Human Miyashita = new Human();
```

使い方

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



他の例

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

- 時計というクラスを作る
  - 時計のインスンタンス変数
    - 現在時間（時分秒）の情報
    - 目覚ましのON/OFF状態管理変数
    - 目覚ましの設定時間
  - 時計のインスタンスメソッド
    - 分変更メソッド
    - 時計停止メソッド（秒針停止）
    - 目覚まし設定時間変更メソッド
    - 目覚ましのON/OFF切り替えメソッド

定義したクラスの使い方

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

- クラスの中で変数を定義すると，そのクラス内の変数として使うことができる
  - クラス内で変数を定義
  - クラスを使う場合は new！
  - クラス内の変数を使う場合はドットでつなぐ

```
class Ball
{
  int posX;
  int posY;
  int speedX;
  int speedY;
}

Ball ball;
ball = new Ball();
ball.posX += ball.speedX;
ball.posY += ball.speedY;
```

インスタンス化

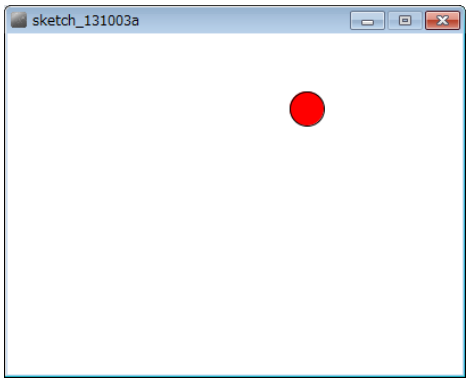
クラス名　変数名 = new クラス名();

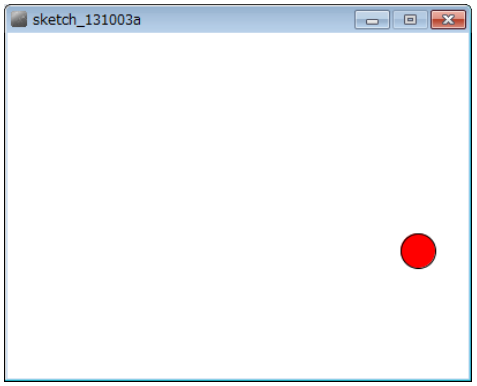
端で跳ね返るオブジェクト

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;`





```
int posX;
int posY;
int speedX;
int speedY;

void setup()
{
  size( 400, 300 );
  posX = (int)random( 0, width );
  posY = (int)random( 0, height );
  speedX = (int)random( 1,5 );
  speedY = (int)random( 1,5 );
  fill( 255, 0, 0 );
}
```

今までの知識で
プログラムを組むと

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

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(posX，posY)とspeedX，speedYを持つBallクラスを考え，その変数を定義する．

定義する

```
class Ball {
    int posX;
    int posY;
    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 ball;
ball = new Ball();
ball.posX = 200;
ball.posY = 150;
ball.speedX = 5;
ball.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( ball.posX, ball.posY, 30, 30 );
```

オブジェクト変数名 . インスタンス変数名

クラスを使ってみる

明治大学総合数理学部
先端メディアサイエンス学科

```
class Ball
{
  int posX;
  int posY;
  int speedX;
  int speedY;
}

Ball ball;

void setup()
{
  size( 400, 300 );
  ball = new Ball();
  ball.posX = (int)random( 0, width );
  ball.posY = (int)random( 0, height );
  ball.speedX = (int)random( 1 ,5 );
  ball.speedY = (int)random( 1, 5 );
  fill( 255, 0, 0 );
}
```

```
void draw()
{
  background( 255 );
  ball.posX += ball.speedX;
  ball.posY += ball.speedY;
  if ( ball.posX > width-15 ) {
    ball.posX = width-15;
    ball.speedX = -ball.speedX;
  }
  if ( ball.posX < 15 ) {
    ball.posX = 15;
    ball.speedX = -ball.speedX;
  }
  if ( ball.posY > height-15 ) {
    ball.posY = height-15;
    ball.speedY = -ball.speedY;
  }
  if ( ball.posY < 15 ) {
    ball.posY = 15;
    ball.speedY = -ball.speedY;
  }
  ellipse( ball.posX, ball.posY, 30, 30 );
}
```

ボールを3つにすると？

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

- 3つの変数を定義
  - new で具体化
    - miyashita
    - komatsu
    - fukuchi

    というボールにする！
  - そのそれぞれの値を参照できるようにする

```
class Ball{
  int posX;
  int posY;
  int speedX;
  int speedY;
}

Ball miyashita;
Ball komatsu;
Ball fukuchi;

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

  miyashita = new Ball();
  komatsu = new Ball();
  fukuchi = new Ball();
}
```

```
class Ball
{
  int posX;
  int posY;
  int speedX;
  int speedY;
}

Ball miyashita;
Ball komatsu;
Ball fukuchi;

void setup()
{
  size( 400
  fill( 25
  miyashit
  komatsu
  fukuchi
  miyashit
  miyashit
  miyashita
  miyashita.speedY = (int)random(1,5);
  komatsu.posX = (int)random(width);
  komatsu.p
  komatsu.
  komatsu.
  fukuchi.
  fukuchi.
  fukuchi.s
  fukuchi.speedY = (int)random(1,5);
}
```

```
void draw()
{
  background(255);
  miyashita.posX = miyashita.posX + miyashita.speedX;
  miyashita.posY = miyashita.posY + miyashita.speedY;
  komatsu.posX = komatsu.posX + komatsu.speedX;
  komatsu.posY = komatsu.posY + komatsu.speedY;
  fukuchi.posX = fukuchi.posX + fukuchi.speedX;
  fukuchi.posY = fukuchi.posY + fukuchi.speedY;

  if ( miyashita.posX > width-15 ) {
    miyashita.posX = width - 15;
  }
}
```

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！！！

データ型としてしか使っていないため
面倒で当然

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();
fukuchi.move();
```

オブジェクト変数名 . インスタンスメソッド名

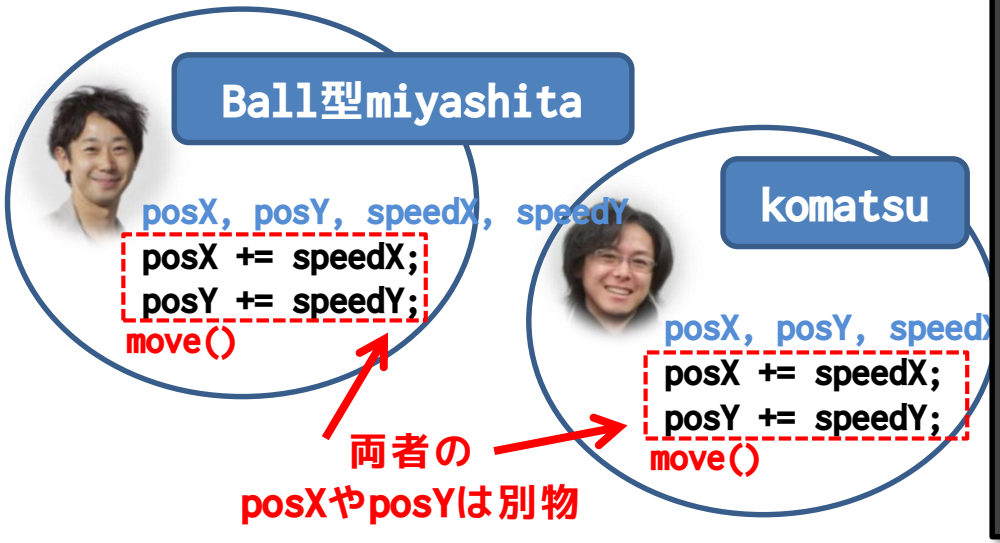
void move() を作る

明治大学総合数
先端メディアサ
中村研究室

- 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

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

```
class Ball{
  int posX;
  int posY;
  int speedX;
  int speedY;
  void move(){
    this.posX += this.speedX;
    this.posY += this.speedY;
    if (this.posX > width-15 ) {
      this.posX = width - 15;
      this.speedX = -this.speedX;
    }
    if( this.posX < 15 ){
      this.posX = 15;
      this.speedX = -this.speedX;
    }
    if( this.posY > height-15){
      this.posY = height - 15;
      this.speedY = -this.speedY;
    }
    if( this.posY - 15 < 0 ){
      this.posY = 15;
      this.speedY = -this.speedY;
    }
  }
}
```

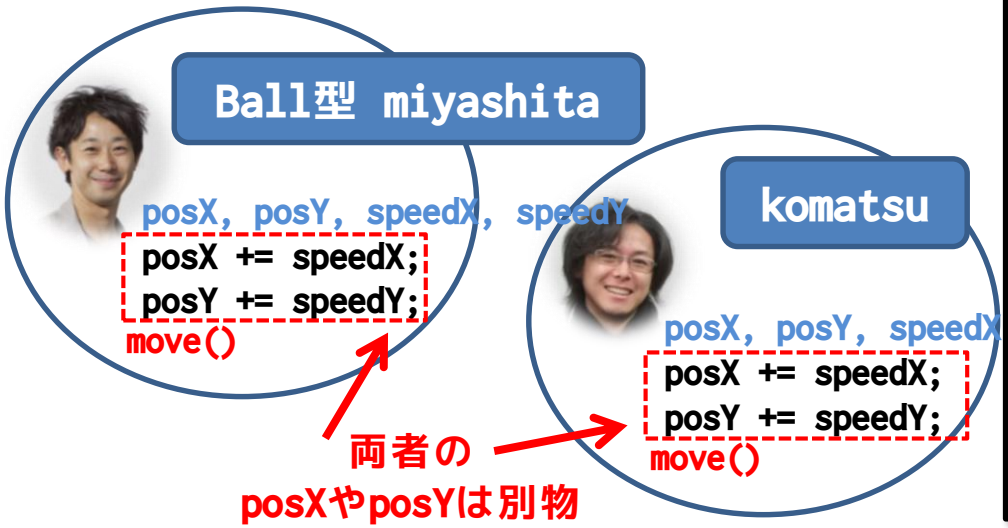


void move() を作る

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

this.は省略可能



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 fukuchi;

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

  miyashita = new Ball();
  komatsu = new Ball();
  fukuchi = new Ball();

  miyashita.posX = (int)random(width);
  miyashita.posY = (int)random(height);
  miyashita.speedX = (int)random(5);
  miyashita.speedY = (int)random(5);
  komatsu.posX = (int)random(width);
  komatsu.posY = (int)random(height);
  komatsu.speedX = (int)random(5);
  komatsu.speedY = (int)random(5);
  fukuchi.posX = (int)random(width);
  fukuchi.posY = (int)random(height);
  fukuchi.speedX = (int)random(5);
  fukuchi.speedY = (int)random(5);
}
```

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

  ellipse( miyashita.posX, miyashita.posY, 30, 30 );
  ellipse( komatsu.posX, komatsu.posY, 30, 30 );
  ellipse( fukuchi.posX, fukuchi.posY, 30, 30 );
}
```

draw() がかなり短くなった！

プログラムを動かそう！

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

- http://nkmr.io/lecture/ の第2回講義資料にある Ball.txt を利用しよう
  - Ballクラスの部分は別のタブに！（次ページで説明）

クラスを作るときは別タブで

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

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.posX = (int)random(width);`
    - `XXXXX.posY = (int)random(height);`
    - `XXXXX.speedX = (int)random(1,5);`
    - `XXXXX.speedY = (int)random(1,5);`

initialize() 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 posX;
  int posY;
  int speedX;
  int speedY;

  void initialize()
  {
    posX = (int)random(width);
    posY = (int)random(height);
    speedX = (int)random(1,5);
    speedY = (int)random(1,5);
  }
```

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

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

  miyashita = new Ball();
  komatsu = new Ball();
  fukuchi = new Ball();
  miyashita.initialize();
  komatsu.initialize();
  fukuchi.initialize();
}
```

```
void draw()
{
  background(255);

  miyashita.move();
  komatsu.move();
  fukuchi.move();

  ellipse( miyashita.posX, miyashita.posY, 30, 30 );
  ellipse( komatsu.posX, komatsu.posY, 30, 30 );
  ellipse( fukuchi.posX, fukuchi.posY, 30, 30 );
}
```

setup() もかなり短くなった！

先ほどのプログラムを改良して動かしてみよう！

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．initialize() はそこで呼び出したら良いのでは？

```
class Ball
{
  int posX;
  int posY;
  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 posX;
  int posY;
  int speedX;
  int speedY;
  Ball(){
    initialize();
  }

  void initialize(){
    posX = (int)random(width);
    posY = (int)random(height);
    speedX = (int)random(1,5);
    speedY = (int)random(1,5);
  }
```

コンストラクタで
initizalizeメソッドを
呼び出す！

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

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

  miyashita = new Ball();
  komatsu = new Ball();
  fukuchi = new Ball();
}
```

```
void draw() {
  background(255);

  miyashita.move();
  komatsu.move();
  fukuchi.move();

  ellipse( miyashita.posX, miyashita.posY, 30, 30 );
  ellipse( komatsu.posX, komatsu.posY, 30, 30 );
  ellipse( fukuchi.posX, fukuchi.posY, 30, 30 );
}
```

setup() がさらに短くなった！

先ほどのプログラムを改良して動かしてみよう！

描画もインスタンスメソッドに

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

- 下みたいなのはあまり好ましくない
  - ellipse(miyashita.posX, miyashita.posY, 30, 30);
  - ellipse(komatsu.posX, komatsu.posY, 30, 30);
  - ellipse(fukuchi.posX, fukuchi.posY, 30, 30);

```
class Ball{
  int posX;
  int posY;
  int speedX;
  int speedY;
  void display(){
    ellipse( posX, posY, 30, 30 );
  }
```

データはなるべく隠蔽したい（カプセル化）

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

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

先ほどのプログラムを改良して動かしてみよう！

端で跳ね返る50個のボール

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

400x300の画面内にX座標Y座標ともにランダムに配置された50個の円を描き，それぞれのX，Y方向へのスピードに応じて移動するようにせよ

- posX，posY という座標の配列と，speedX，speedY というXおよびY方向の速度をもつ配列を導入！

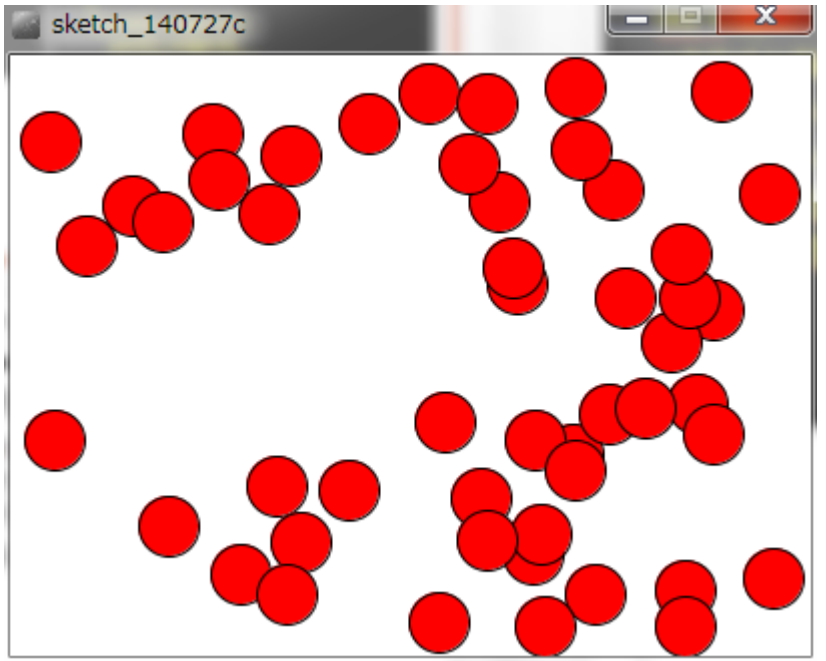
オブジェクト＋配列

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

- 50個の丸を動かすには配列を使う！

```
Ball [] balls = new Ball [50];

void setup() {
  size( 400, 300 );
  fill( 255, 0, 0 );
  for( int i=0; i<50; i++ ){
    balls[i] = new Ball();
  }
}

void draw(){
  background( 255 );
  for( int i=0; i<50; i++ ){
    balls[i].move();
    balls[i].display();
  }
}
```

型 [] 配列名 = new 型 [要素数];



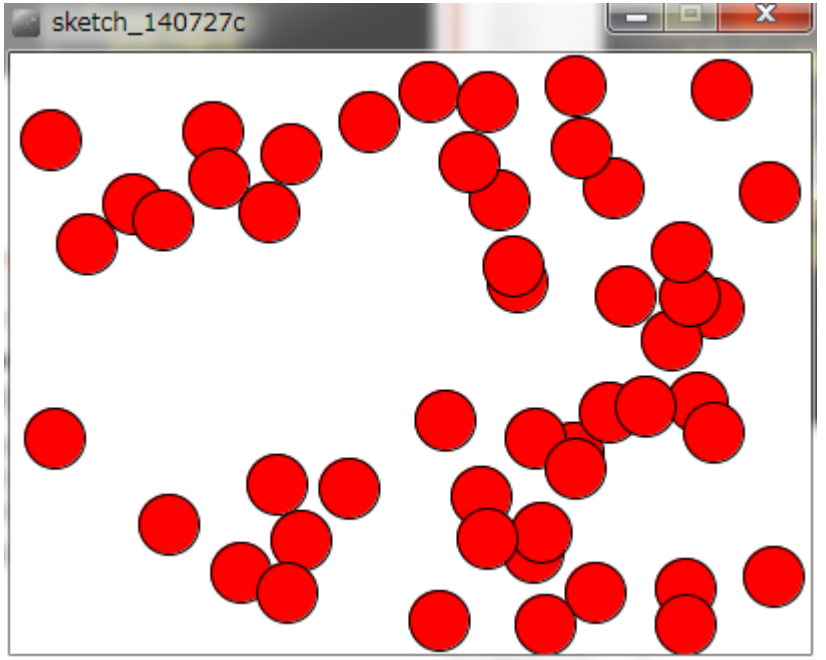
オブジェクト＋配列

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

- 50個の丸を動かすには配列を使う！

```
Ball [] balls = new Ball [50];

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

void draw(){
  background( 255 );
  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



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

- 300個の丸を動かすには配列の定義を変更

```
Ball [] balls = new Ball [300];

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

void draw(){
  background( 255 );
  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

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){…} どれが正しい？
  - どれでもOK
  - new のときに，どう呼び出すかの違い
    - Ball() は Ball b = new Ball();
    - Ball(int x, int y) は Ball b = new Ball(500, 100);

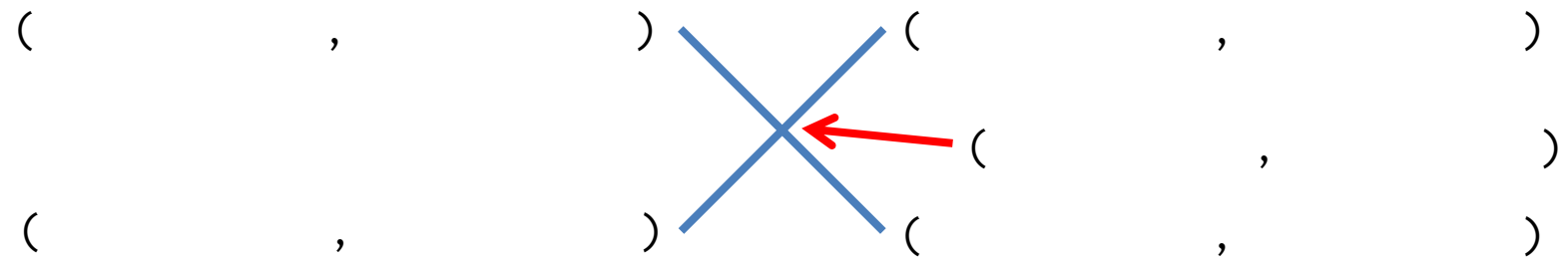
    で，それぞれ呼び出される．

xが動き回るクラスを作ろう

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

Ballクラスを利用して、Crossクラスを作ろう！

- Ballクラスと、Crossクラスの違いは、表示される図形が「○」か「x」かなだけ！！
- Crossクラスのタブを作成し、Ballクラスをコピペ！
- BallをCrossに書き換える！
- 表示だけを変更したいので、displayの中身を変更する！
- メインのプログラムで Cross を使っていこう！



それ以外のクラス

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

- 文字列を扱うクラス: String クラス
- 画像を扱うクラス: PImage クラス

などなど

PImage / 画像型

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

- 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
```

http://www40.atwiki.jp/spellbound/pages/1800.html
http://processing.org/reference/PImage_filter_.html

課題2-1：boundAll2

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 クラスを改良し，xが動き回るCrossクラスと△が動き回るTriangleクラスを作成せよ．
- またこれを利用して， 5個の〇と，4個のxと，3個の△が動き回るプログラムを作成せよ
  - ただし，その速度は目視可能なものとせよ
  - 1個程度動いていないものがあってもよい
  - また，可能であれば〇は壁で跳ね返り，xと△は跳ね返らずに反対側から出てくるようにせよ
  - 前回クラスを使っている人は、そのまま個数だけ変更して提出してもOK

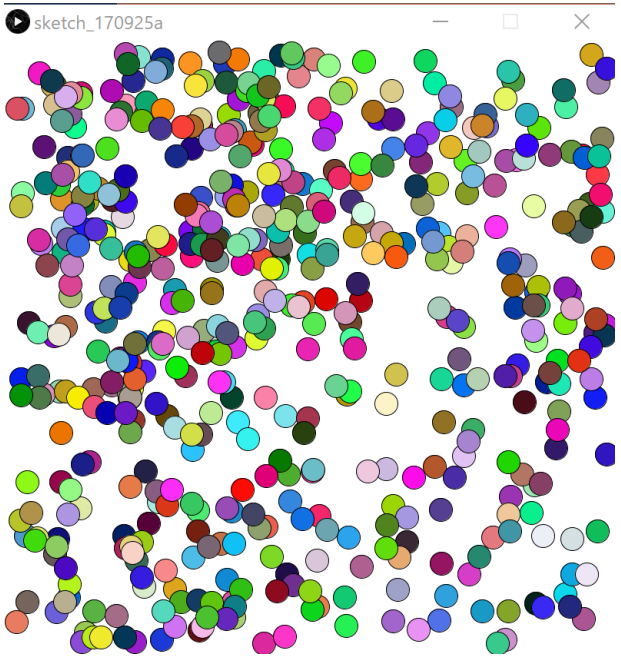
課題2-2: random555

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

- ウインドウ800x800内に555個の丸を動かし上下左右の端で跳ね返るプログラムを作成せよ
- ただし，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

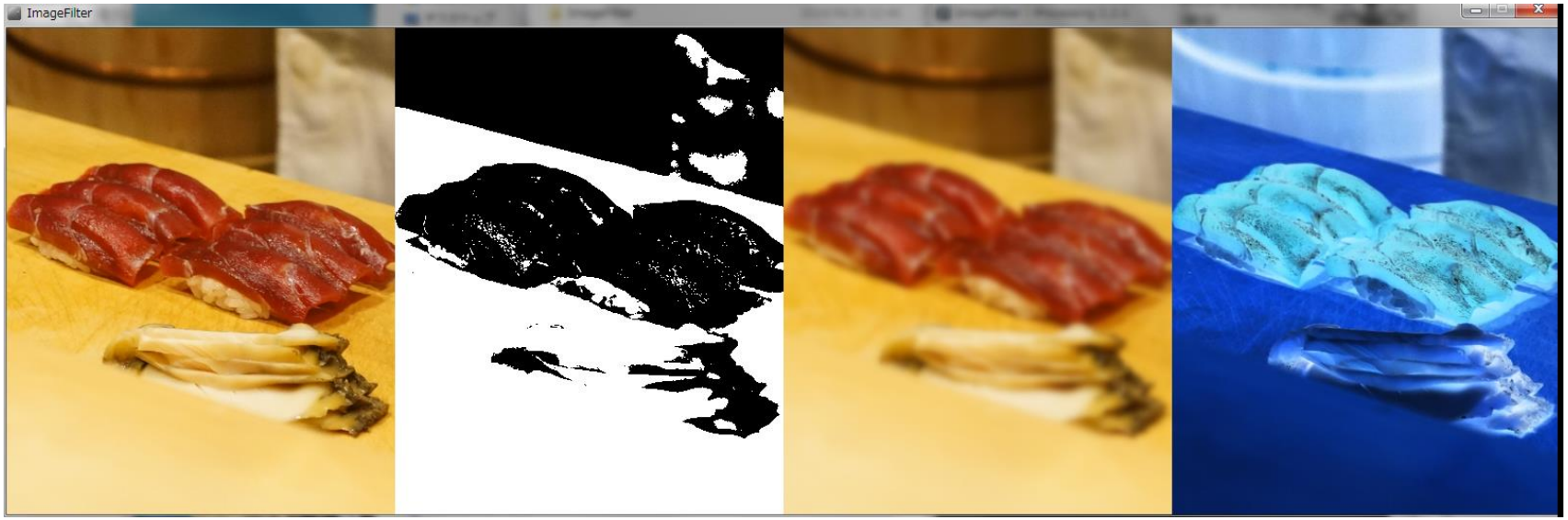
  - int red;
  - int green;
  - int blue;



課題2-3: imageProc

小テスト対象外

- PImage クラスを使って画像処理をしてみよう
  - 適当な画像をダウンロードし，その画像に対してフィルタを掛けてみましょう！
  - 一番左にオリジナル画像，次にTHRESHOLDで二値化したもの，さらに次にBLURでぼかしたもの，一番右にINVERTでネガポジ反転したものを表示しよう



宿題：charaClass

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

- 移動するキャラクタを描画するクラスを作成せよ
- なお，クラスでは下記の仕様を満たすようにせよ
  - インスタンス変数として中央の座標，XY方向の速度という値をもつ
    - centerX, centerY, speedX, speedY
  - 移動に関するインスタンスメソッド move （引数と返り値なし）
    - void move();
  - 表示するインスタンスメソッド display （引数と返り値なし）
    - void display();
    - キャラクタについては好きなものを描画するようにせよ
  - 起動時に初期位置とスピードが変わるようにせよ

まとめ

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

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