プログラミング演習２
クラスと継承の再復習

中村，高橋，小林，橋本

本日の流れ

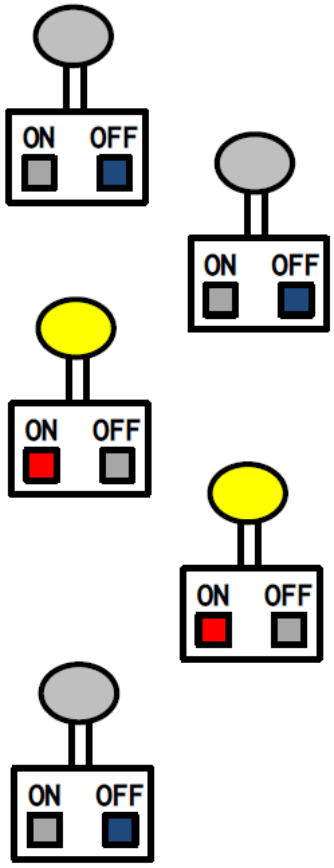
- 宿題の解説
- 座学＋演習
- 課題

- 中間課題提示（中村先生）

宿題4-1：hw_SwitchLight

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

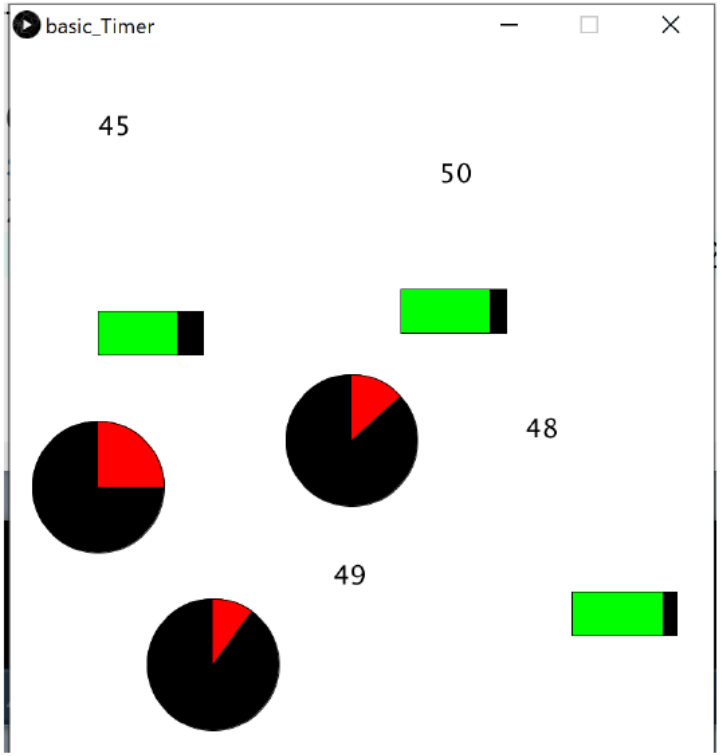
- 2つのボタンと1つのライトからなる照明を作成する．この照明をSwitchLightクラスとして作成せよ（形状については任せる）
- ボタンは左右に配置し，左側のボタンをクリックするとライトが光り，右側のボタンをクリックするとライトが消えるようにせよ
- また，SwitchLightクラスを利用して，5つの照明を画面に表示するようにせよ



宿題4-2: hw_Timer

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

- 配布する hw_Timer.zip を利用し，生成されてから60秒経過すると自動的に消滅する（何も描画されなくなる）タイマーを，TimerCoreクラスを継承して，TextTimerやProgressTimerを参考にしつつ2つ以上実現せよ
- また，マウスクリックしたときにそれぞれのタイマーがランダムに生成されるようにせよ



宿題4-3: hw_VendingMachine

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

- 自動販売機（VendingMachineクラス）では日本円の10円・50円・100円の3種類の硬貨を扱い、販売される商品は全て200円以下で、10円～200円で10円刻みとなっています。
- 自動販売機には投入金額と購入金額からおつりを計算し、おつりを返却するか、商品を購入できないことを知らせる機能が必要です。これらの機能は、以下のルールに従って動作します。
  - お釣りがない場合は「ありがとうございました。おつりはありません」と表示する。
  - お釣りがある場合は「ありがとうございました。おつりは10円3枚と50円1枚です」と表示し、おつりを返却する。
  - 投入金額が足りないまたはおつりを返却できない場合は「商品を購入できません。10円0枚、50円1枚、100円1枚を返却します」と商品を購入できないことを知らせるとともに投入金額を返却する。
  - なお投入されたお金は、おつりとしても利用することが可能です。

宿題4-3: hw_VendingMachine

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

- ヒント
  - VendingMachineクラスには，内部に10円玉，50円玉，100円玉が何枚あるかを管理する変数を用意
  - 初期の枚数をセットするメソッドを作成

    initialize(10円の数, 50円の数, 100円の数) メソッド
  - お金を投入するメソッドを作成

    insert(10円の数, 50円の数, 100円の数) メソッド
  - 購入する商品を指定するメソッドを作成

    buy(値段) メソッド
    - buy メソッドは，標準出力で結果を返すようにせよ．ただし，
    - おつりは「投入金額 - 購入金額」で計算されます。
    - おつりは自動販売機の内部にある硬貨から枚数が最も少なくなるように選んだ硬貨の組合せで返却せよ

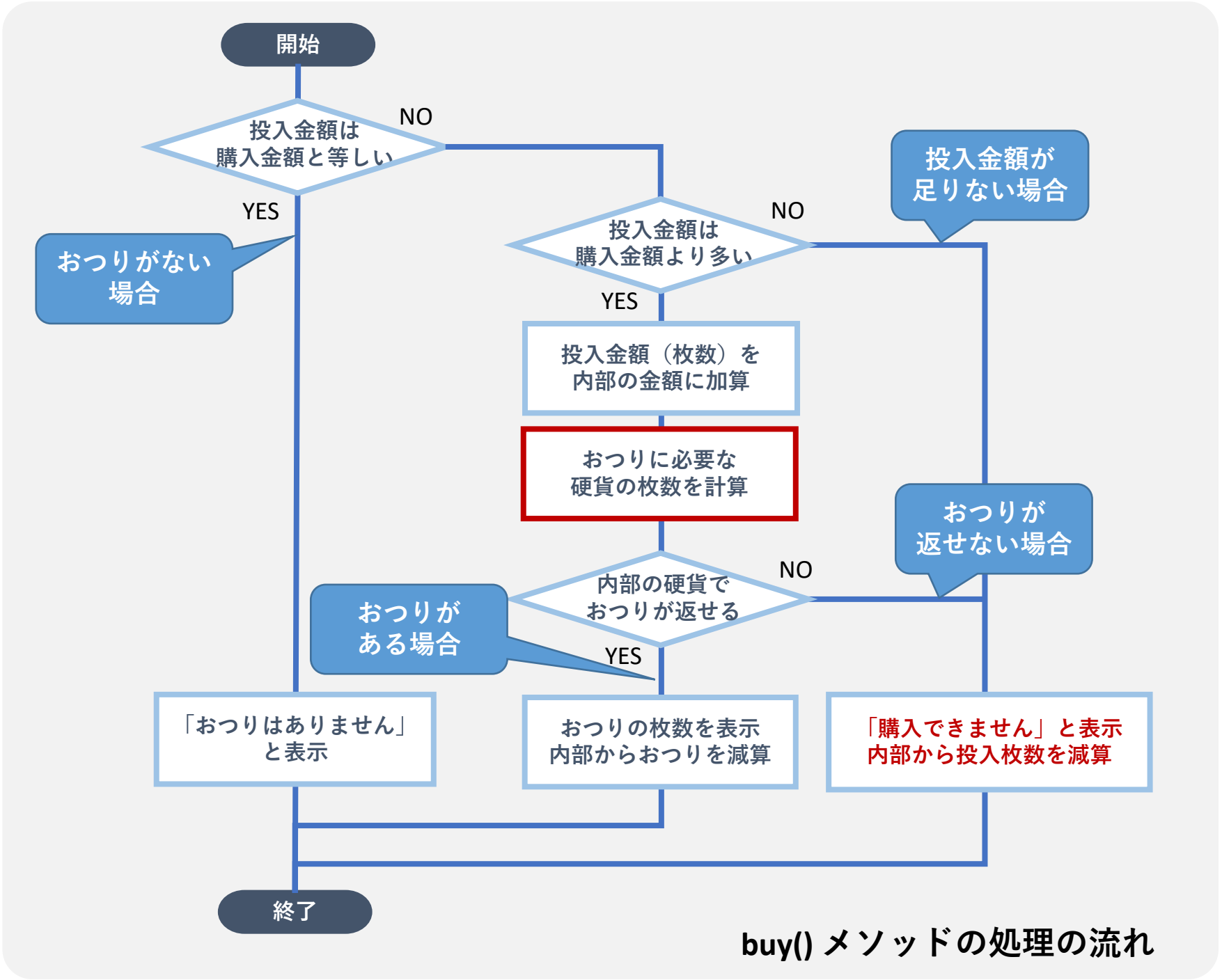
宿題4-3: 動作チェック

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

- 色々なパターンを用意して問題ないかを確認しよう

```
vMachine.initialize( 5, 5, 5 );
vMachine.insert( 0, 1, 1 );
vMachine.buy( 130 );
vMachine.insert( 0, 0, 2 );
vMachine.buy( 110 );
vMachine.insert( 0, 0, 2 );
vMachine.buy( 140 );

vMachine.initilize( 9, 8, 7 );
vMachine.insert( 0, 0, 2 );
vMachine.buy( 110 );
vMachine.insert( 0, 0, 2 );
vMachine.buy( 120 );
vMachine.insert( 0, 0, 2 );
vMachine.buy( 130 );
vMachine.insert( 2, 0, 2 );
vMachine.buy( 180 );
```

ありがとうございました。おつりは10円2枚と50円0枚と100円0枚です
商品を購入できません。10円0枚、50円0枚、100円2枚を返却します
ありがとうございました。おつりは10円1枚と50円1枚と100円0枚です
ありがとうございました。おつりは10円4枚と50円1枚と100円0枚です
ありがとうございました。おつりは10円3枚と50円1枚と100円0枚です
ありがとうございました。おつりは10円2枚と50円1枚と100円0枚です
商品を購入できません。10円2枚、50円0枚、100円2枚を返却します



buy() メソッドの処理の流れ

クラスの再復習

オブジェクト指向，難しいですよね

- すごく便利！と言われてもピンとこない人も多いはず
- かくいう高橋も，クラスがわかるようになったのは
  大学院生になってからでした

以下の流れで復習しましょう

- プログラミング中に何を考えているか
- クラスを意識してみる
- クラス書き方をもう一度

プログラミング中に何を考えているか

授業中の課題や宿題を解くとき

- おそらく課題の要件を満たすことに必死
- （ある意味）ゴールが明確なので多少無茶でも書いちゃう
  「同じものを4つ，5つ描く」ぐらいならひたすらコピペ，など

→ **プログラミング言語の文法について学ぶ授業なので，ひとまず今はこういう思考でもOK**（だと思う）

でも，本来頭を使うべきところは…？

- コサイン類似度って？何に使える？
- タイマーのデザインもっとこだわりたい
- 他人に理解してもらえるか？ などなど

言語についてではなく，何をどう作るかを考えたい

考えながらプログラムを書けるようになるには

書き方を覚える

- 条件分岐，ループ，配列，関数…
- クラスの構成，インスタンス化，継承…
- 初見の命令を調べる，エラーをしっかり読む

いろいろなプログラムを読み書きして慣れる

書き方を工夫する

気にしなくても良い部分を作る

- 変数や関数をクラス（オブジェクト）としてまとめる
- 継承を使って必要最低限の部分だけ書く
- （もちろん，命名やインデント，コメントに気をつける，なども）

考えるためにリソースを割ける！

なぜクラスを使うの？（1/2）

座標（x, y）と
大きさ（w, h）

```
int[] b_x = new int[10];
int[] b_y = new int[10];
int[] b_w = new int[10];
int[] b_h = new int[10];
```

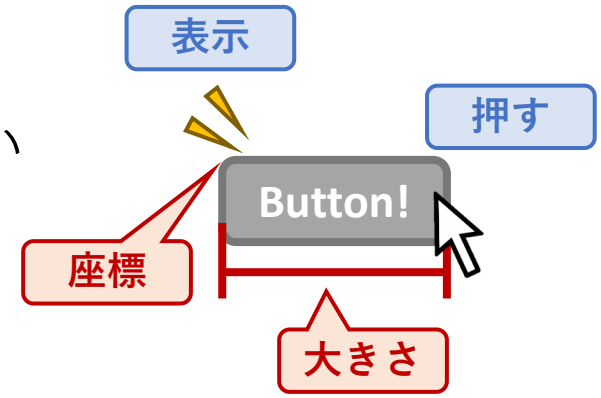
データの扱い方を意識

- たとえば，10個のボタンを作るとき，右のように配列でも書ける
- 各ボタンは**変数ごと**にまとまった状態
  - 10個の x 座標のまとまり，10個の y 座標のまとまり…
  - どのボタンと対応しているかはプログラマが管理
    配列の場合は，要素の番号（インデックス）に従って管理する

一方，みなさんが注目したいのはボタン自体では？

- 「あるボタン」の**座標**が知りたい
- 「あるボタン」を画面に**表示**したい
- 「あるボタン」を**押す**動作を作りたい

→ **こうした変数や機能を**
オブジェクト単位でまとめる



なぜクラスを使うの？（2/2）

プログラム内で使用するものを抽象的に捉える手段

例：大きさ 10 の正方形を（100, 100）に表示

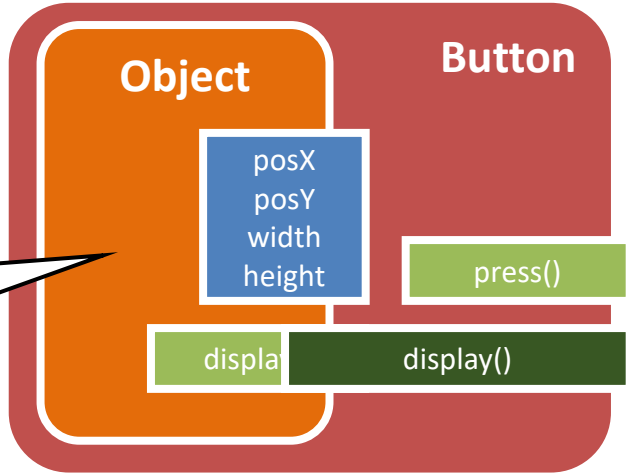
例：直径 10 の円を（200, 200）に表示

これらを抽象的に

→ **「ある大きさ」**の**「なにか」**を**「どこか」**に**「なにかする」**

この抽象的なものを先に仕様としてまとめる

- 具体的なものを作る場合はこの仕様に従う
- 仕様に反することは「できない」
- 「ここだけ作れば動く」仕組み

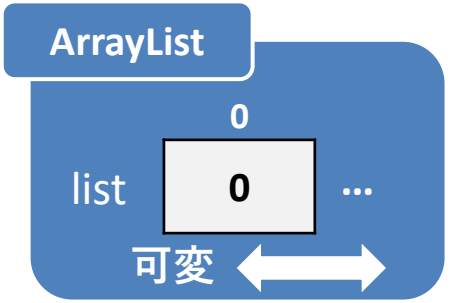


たとえば，Buttonクラスをのインスタンスを作ると，Buttonクラスで定義されたこと以外は「できない」

すでにかなり恩恵を受けています

たとえば，ArrayListクラスの中身は配列

- new ArrayList() と書くと，
  内部で決まった大きさの配列が作られる
- add() で配列の要素外になる場合は，
  配列自体を大きめに作り直す
- 配列とは別に要素数を管理する変数，
  要素数を返す関数size()がある

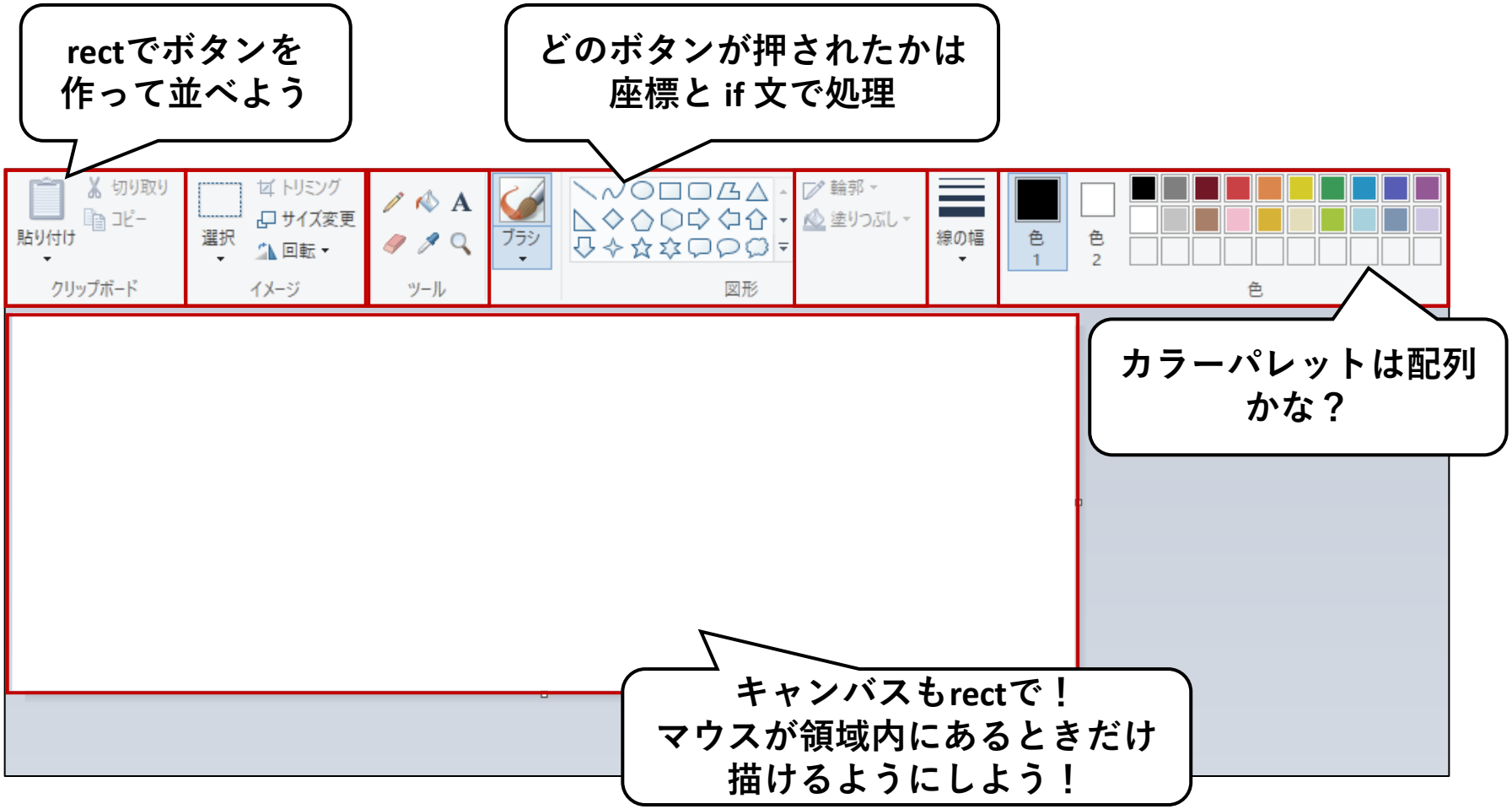


我々は中身を気にせずに，
「色々なものをいくつでも追加可能な便利なクラス」
として使うことができる

参考：Processingの IntListクラス のコード

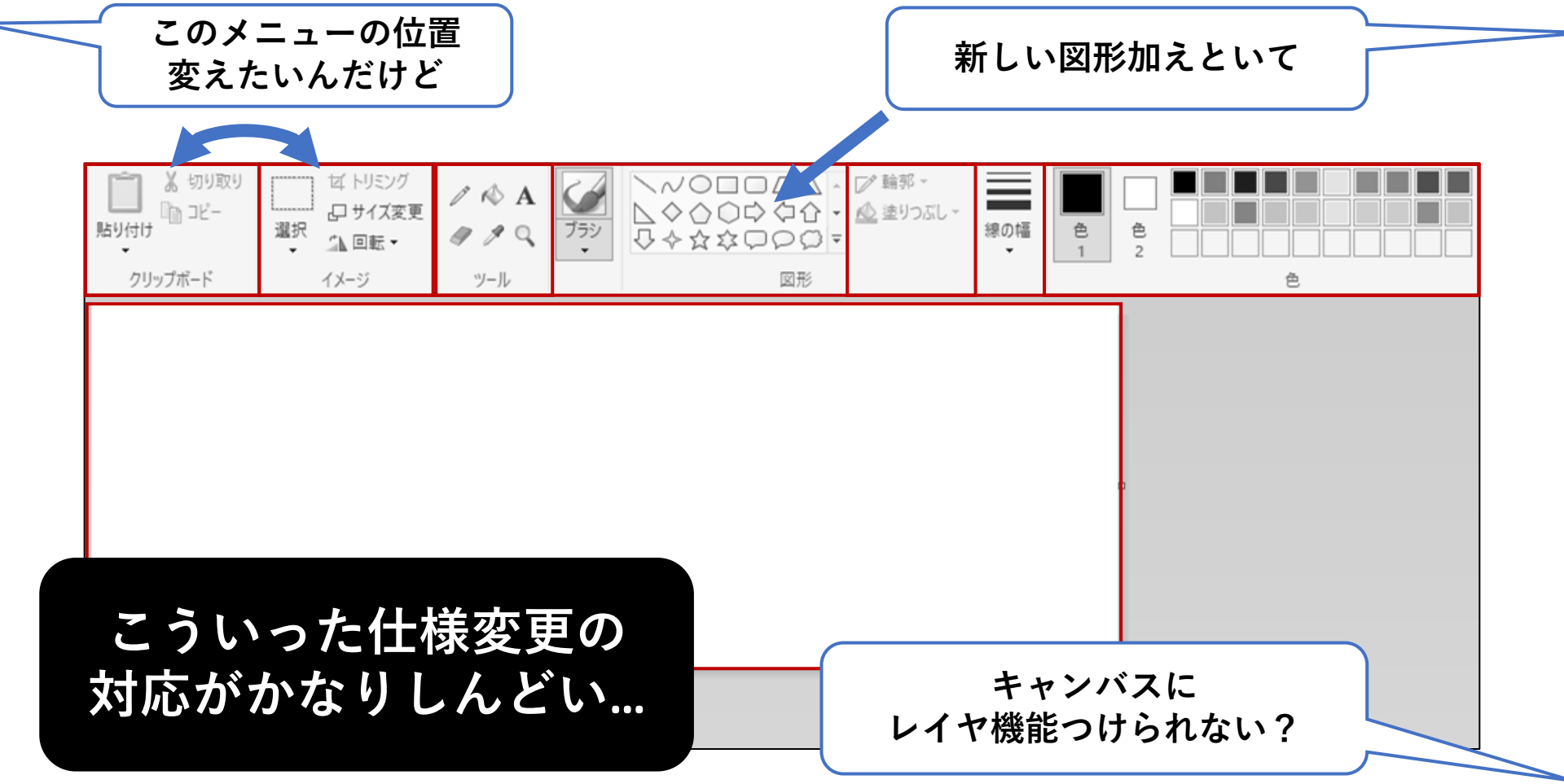
例）ペイントをどう作る？

クラスを使わずに（P演習１の知識で）書こうとすると，
プログラム中の変数や処理の流れはどうなる？



例）ペイントをどう作る？

後からレイアウトを変更したり，
機能を追加したりしたいときはどうか？



クラスを意識して考えてみる

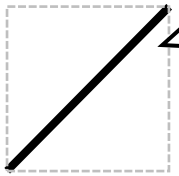
全体の構造や各要素に必要な変数・機能,
使い回しができそうな部分について考えてみる

Group クラス
Buttonをまとめるクラス,
このGroupの中で位置や大きさを調整

ToolBar クラス
Groupをまとめるクラス,
表示・非表示できると便利？

Button クラス
当たり判定，押されたときの動作,
継承して異なる機能を作るかも

Stroke クラス
描かれたものを個別に管理,
移動や大きさ変更などの幾何変換ができる



あくまで一例ですが,
こういう考え方をしておくと,
プログラムの管理・拡張が楽！

Canvas クラス
基本は長方形，マウスで描画できる,
大きさも可変にしておくとよいかも？
複数のキャンバス（レイヤ）に対応しておく？

【演習】クラスの復習

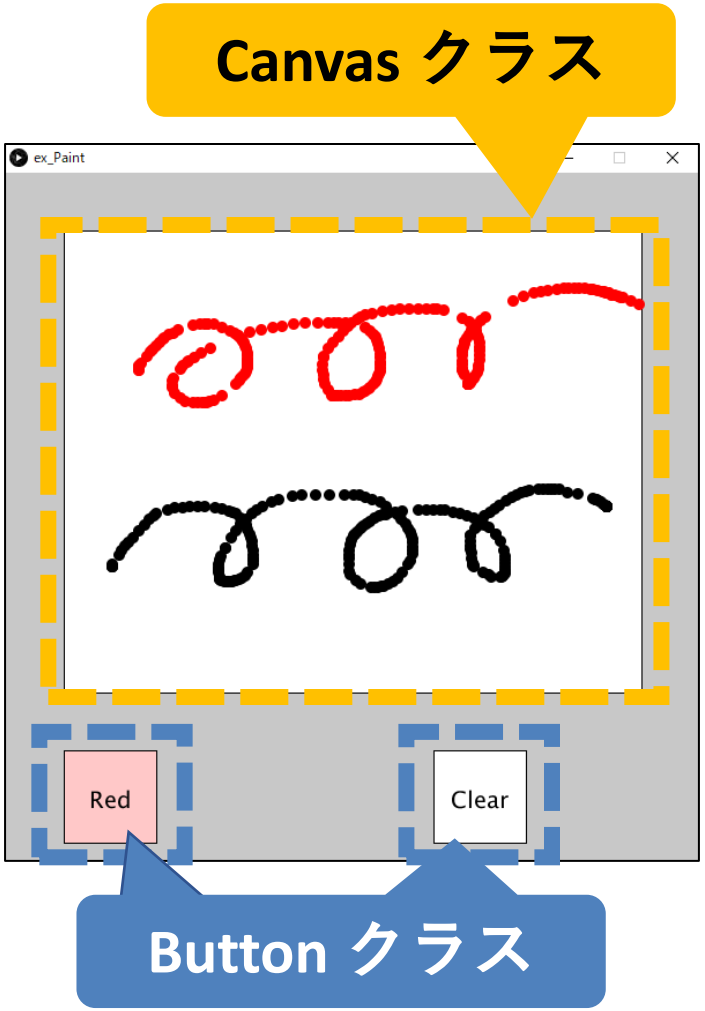
そのまま課題に
使います

簡単なペイントアプリを作ってみましょう

- クラスを作りながら復習
- クラスの構成，用語など
  覚えましょう

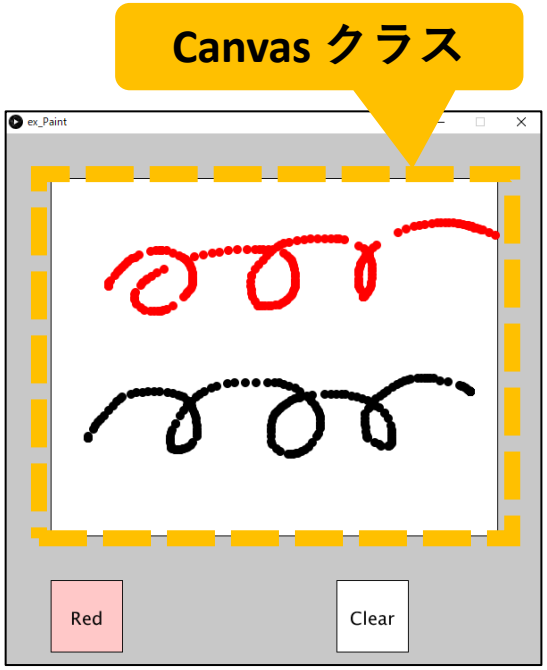
クラス

- Canvas
- Button
  - 色を変えるボタンと
    キャンバス初期化ボタン
  - こちらを継承して別の
    ボタンを作る予定



Canvasクラス

rectで描画領域を
作成する想定




```
class Canvas {
  int x, y;
  int w, h;
  color background;

  Canvas(int _x, int _y, int _w, int _h, color _bg) {
    x = _x;
    y = _y;
    w = _w;
    h = _h;
    background = _bg;
    clear();
  }

  void clear() {
    fill(background);
    rect(x, y, w, h);
  }

  boolean onMouse(int mx, int my) {
    boolean check_x = x < mx && mx < x + w;
    boolean check_y = y < my && my < y + h;

    if (check_x && check_y) {
      return true;
    } else {
      return false;
    }
  }
}
```

Canvasクラス

まずはクラス名を決める

- 適当に命名しない
- { } で全体を囲う

インスタンス変数を定義

- 座標やサイズ（rectを想定）
- 背景色

コンストラクタを書く

- インスタンス化したときに最初に呼ばれる関数
- インスタンス変数に具体的な値を入れる

```
// キャンバスクラス
class Canvas {
  int x, y; // キャンバスの左上座標
  int w, h; // 幅，高さ
  color background; // 背景色

  // コンストラクタ
  Canvas(int _x, int _y, (略) ) {
    x = _x;
    y = _y;
    w = _w;
    h = _h;
    background = _bg;
    clear(); // キャンバスの初期化
  }

 . . .
```

```
Canvas canvas = new Canvas( 座標x , 座標y , 幅，高さ，背景色 );
```

Canvasクラス

インスタンスメソッド

- **clear()**
  - 背景色を設定
  - rectを書き直す
- **onMouse(int mx, int my)**
  - マウスが自分自身に乗っているか判定
  - True か False を返す

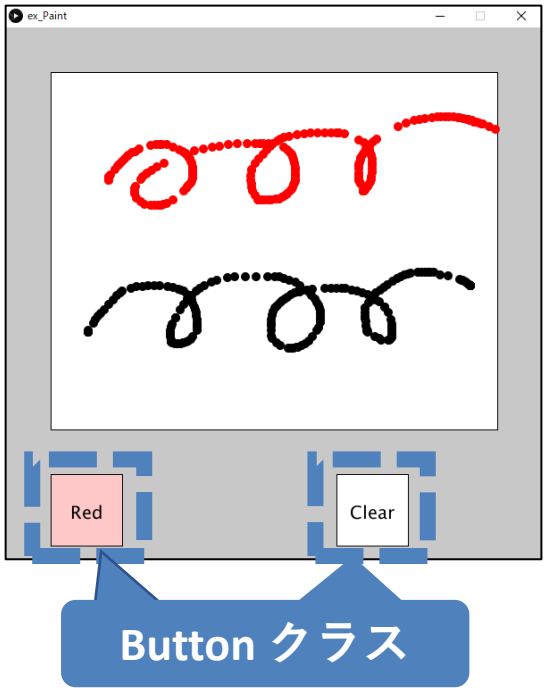
```
 . . .
// clear()：キャンバス初期化
void clear() {
 fill(background);
 rect(x, y, w, h);
}

// onMouse()：マウスが乗っているか
boolean onMouse(int mx, int my) {
 boolean check_x = x < mx && mx < x + w;
 boolean check_y = y < my && my < y + h;

 if (check_x && check_y) {
  return true;
 } else {
  return false;
 }
}
}
```

指定した位置・大きさ・色でキャンバスを作る準備完了
Canvasには「初期化する機能」と
「マウスがキャンバス上にあるか判定する機能」がある…
これだけを覚えれば，Canvasクラスの中身は一旦忘れてOK

Buttonクラス

コピペした場合は
「Ctrl+T」で整えてください

rectでボタンを書く
文字列でラベルを付ける




```
class Button {
  int x, y, w, h;
  String label;
  boolean state;

  Button(String _label, int _x, int _y, int _w, int _h) {
    x = _x; y = _y; w = _w; h = _h;
    label = _label; state = false;
  }

 void display() {
    textAlign(CENTER, CENTER);
    textSize(20);
    stroke(0);
    strokeWeight(1);

    if (state == true) { fill(255, 200, 200); }
    else { fill(255); }
    rect(x, y, w, h);
    fill(0);
    text(label, x+w/2, y+h/2);
  }

  boolean onMouse(int mx, int my) {
    boolean check_x = x < mx && mx < x + w;
    boolean check_y = y < my && my < y + h;
    if (check_x && check_y) { return true; }
    else { return false; }
  }

  boolean press(int mx, int my) {
    if (onMouse(mx, my)) {
      state = !state;
      return true;
    }
    return false;
  }
}
```

Buttonクラス

クラス名を決める

- 適当に命名しない
- { } で全体を囲う

インスタンス変数を定義

- 座標やサイズ（rectを想定）
- ラベル（文字列）
- ボタンの状態

```
// ボタンクラス
class Button {
  int x, y; // ボタンの左上座標
  int w, h; // ボタンの幅，高さ
  String label; // ボタンに表示する文字列
  boolean state; // ボタンの状態

 // コンストラクタ
  Button(String _label, int _x, （略）) {
    x = _x;
    y = _y;
    w = _w;
    h = _h;
    label = _label; state = false;
  }
  . . .
```

コンストラクタを書く

- インスタンス化したときに最初に呼ばれる関数
- インスタンス変数に具体的な値を入れる

```
Button button = new Button( ラベル , 座標x , 座標y , 幅，高さ );
```

Buttonクラス

インスタンスメソッド

- **display()**
  - ボタンを描画
    ※状態に応じて色を変える
  - rect → text の順に書く
- **onMouse(int mx, int my)**
  - Canvasクラスと同じ
- **press(int mx, int my)**
  - 自身が押されたかどうか
  - onMouse()を活用して,
    ボタンが押されたらTrueに
    なるような仕組み

```
 . . .
// display()：ボタンを表示する
void display() {
   textAlign(CENTER, CENTER);
   textSize(20);
   stroke(0);
   strokeWeight(1);

   if (state == true) { // ボタンの状態で色を変更
     fill(255, 200, 200);
   } else {
     fill(255);
   }
   rect(x, y, w, h);
   fill(0);
   text(label, x+w/2, y+h/2);
 }

 // onMouse()：マウスが乗っているか
 boolean onMouse(int mx, int my) {
   boolean check_x = x < mx && mx < x + w;
   boolean check_y = y < my && my < y + h;

   if (check_x && check_y) { return true; }
   else { return false; }
 }

 // press()：ボタンを押す
 boolean press(int mx, int my) {
   if (onMouse(mx, my)) {// onMouseを活用
     state = !state;
     return true;
   }
   return false;
 }
}
```

メイン部分

インスタンス化

初めて具体的なモノとして
プログラム上に作られる

draw()の処理

- ClearボタンがTrue
  - キャンバス初期化
- キャンバスに描画
  - mousePressedと,
    キャンバスのonMouseで
  - 描画するときは,
    ボタンの状態で色変更
- ボタンを描画

mousePressed()の処理

- 全ボタンのpress()を
  呼んでしまう

```
ArrayList<Button> buttons;
Button btn_red, btn_clear;
Canvas canvas;

void setup() {
  size(600, 600);
  frameRate(60);
  background(200);

  buttons = new ArrayList<Button>();
  btn_red = new Button("Red", 50, height-100, 80, 80);
  btn_clear = new Button("Clear", 370, height-100, 80, 80);
  buttons.add(btn_red);
  buttons.add(btn_clear);

  canvas = new Canvas(50, 50, 500, 400, color(255));
}

void draw() {
  if (btn_clear.state) {
    btn_clear.state = false;
    canvas.clear();
  }

  if (mousePressed && canvas.onMouse(mouseX, mouseY)) {
    noStroke();
    fill(255 * int(btn_red.state), 0, 0);
    ellipse(mouseX, mouseY, 10, 10);
  }

  for (int i = 0; i < buttons.size(); i++) {
    buttons.get(i).display();
  }
}

void mousePressed() {
  for (int i = 0; i < buttons.size(); i++) {
    buttons.get(i).press(mouseX, mouseY);
  }
}
```

位置や大きさは適当に
決めてください

課題5：basic_Paint

１～３のそれぞれが達成条件
できたところまで「動作する状態」
で提出しましょう

演習で実装したペイントアプリに機能を追加せよ

（1）キャンバスを2つにしてみよう

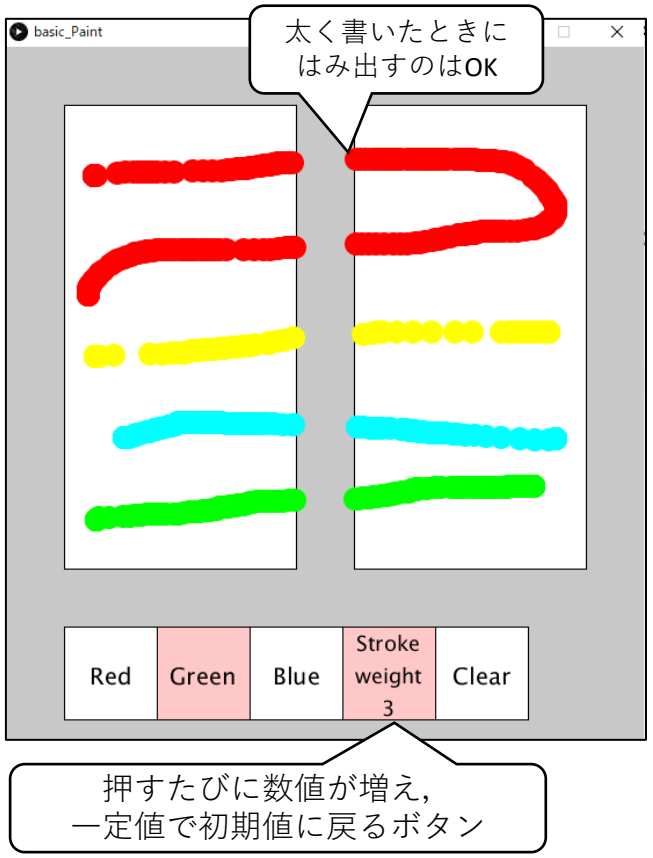
- Canvasクラスのインスタンスをもうひとつ作って並べる
- Clearボタンは両方のキャンバスを初期化

（2）ボタンを増やそう

- Redの他にGreenとBlueを作成
- 各ボタンの状態に応じて色を変更する
  - すべて押されているときは白
  - True，Falseで255, 0 を切り替える

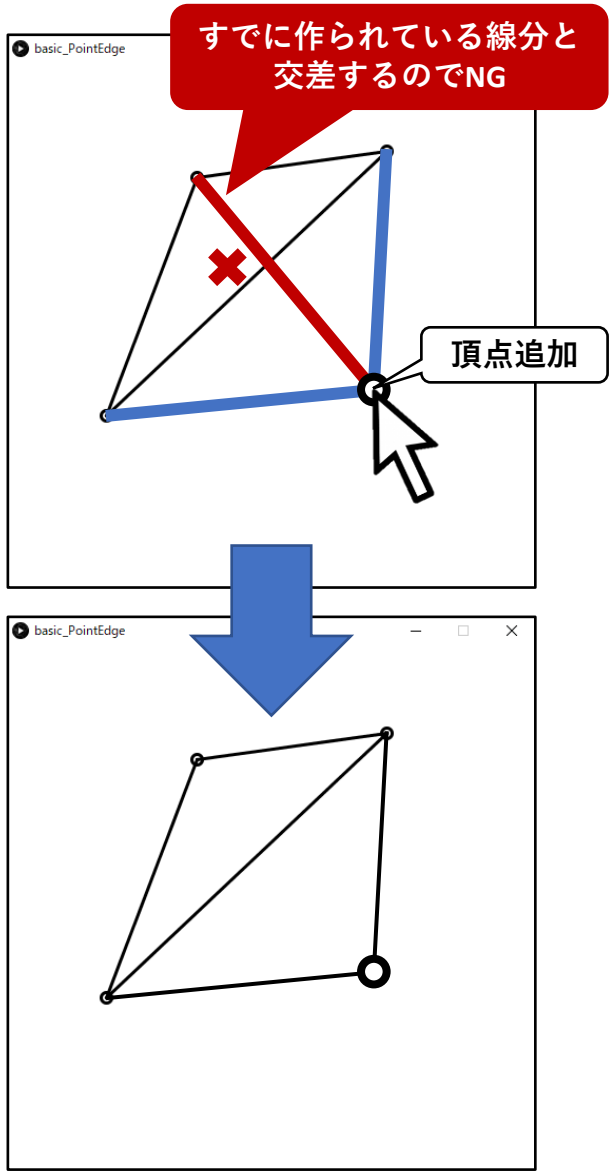
（3）新しいボタンを作ろう

- CountButtonクラス
  - Buttonクラスを継承
  - 押すたびに内部の数値が増える
    - 最大値を設定してループするようにすること
    - 例）0, 1, 2, 3, 0, 1, 2 ,3, …
  - これを描画時の太さに割り当てる



宿題5-1：hw_PointEdge

- クリックするたびに，頂点と辺を追加するプログラムを実装しよう（それぞれArrayListに追加していく）．ただし，すでに作られている辺と交差する辺は追加しないこととする．
- 頂点と辺を管理するクラスとしてPointクラスとEdgeクラスを用意してあります．
  - Point はx, y座標をインスタンス変数として持っており，コンストラクタで与えられた位置に頂点を作る．
  - Edgeは２つのPoint型のインスタンス変数を持っており，それぞれ１つの辺の両端の頂点に相当する．コンストラクタでPointを与えて辺を作る．
- Edgeクラスのインスタンスメソッドに，引数で与えられた辺との交差判定を行うメソッドintersect()を用意したのでこれを活用するとよい．
- なお，配布時のプログラムは，クリックした位置に頂点を追加するだけのものになっている．mousePressed()に，正しく辺を追加する処理を書き加えよ．



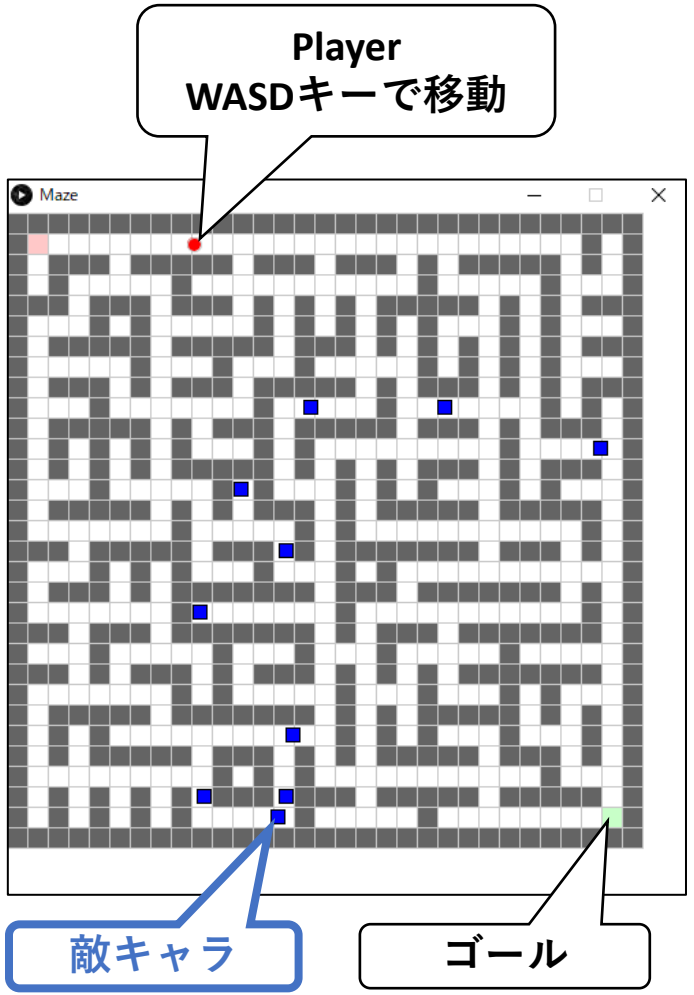
宿題5-2：hw_Maze

- 迷路の中を動き回る敵キャラクタの動作を設計しよう
- 配布時のプログラムの敵キャラクタはEnemyクラスのインスタンスで，「X軸方向にのみ移動して約1%の確率で向きを変える」という動作．
- このクラスを継承してMyEnemyクラスを作り，新しい敵キャラの種類を追加しよう．
- クラスを作ったら，敵キャラクタのインスタンス化部分の変更も忘れずに

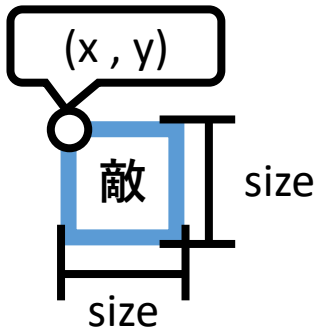
  basic_Mazeタブの47行目（付近）

  ＞　**enemies_list.add(new Enemy(…));**

※迷路は実行のたびに変化し，極稀にクリアできない形状になります．また，最初から敵キャラがプレイヤに被っていることもあります．これらの場合は再実行してください．



Enemyクラス



Characterクラスを継承して作成

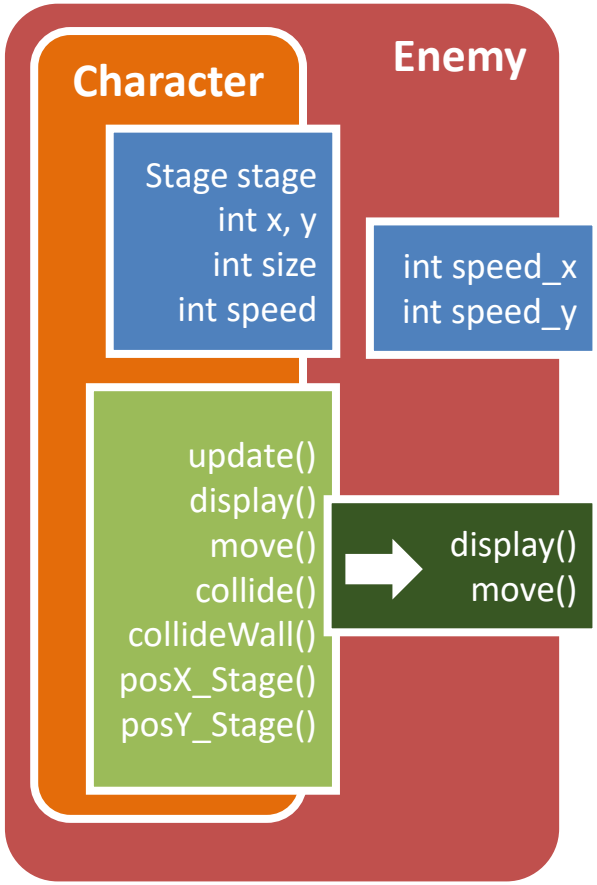
インスタンス変数

- 位置，大きさ，速度，ステージ
- 継承時にx，y 方向の速度を追加

インスタンスメソッド

基本は
ここを改良すればOK

- move()：キャラクタの動きを制御
- display()：キャラクタを描画
- update()： move()，display()を順番に呼ぶ
- collide(int x, int y)
  - 引数(x, y)の座標が自分自身と衝突しているか判定
- collideWall(int x, int y)
  - 引数(x, y)の座標がステージの壁と衝突しているか判定
  - 移動前にその座標を調べると良い
- posX_Stage()，posY_Stage()
  - いま自分がいるステージのセルのインデックスが返る



必要に応じて，変数・メソッドを追加しても良い

補足：プログラム全体の構成

各キャラクタはStageの状態を知りたい

- どに壁があるか，移動できるか，ゴールに辿り着いたか，など

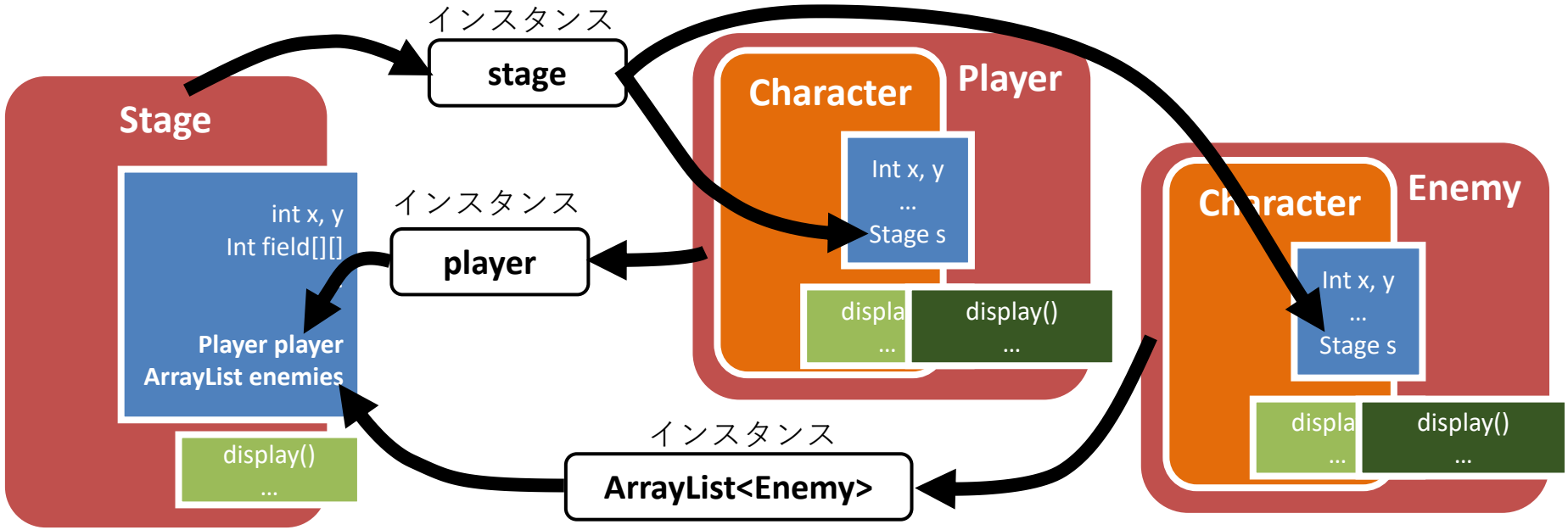
→ **CharacterクラスにStage型のインスタンス変数をもたせる**

Stageは各キャラクタの状態を知りたい

- 敵とプレイヤーがぶつかっているか，ステージ中のどこにいるか，など

→ **StageクラスにPlayer型，Enemy型のインスタンス変数をもたせる**

敵キャラは複数いるのでリストにしています



補足：HashMap

- HashMap型は，何かをキーとして値を格納する辞書のようなものであり，キーを利用して値を取り出せる

  HashMap<キーの型，格納する値の型> dic;

  dic = new HashMap <キーの型，格納する値の型>()

- HashMapに要素を追加
  - dic.put(key, value);
- HashMapから要素を取得
  - dic.get(key);

Mazeのプログラムでは，
HashMapを使ってキーボードの
押下状態を管理しています

```
// 文字列のキーに文字列の値を格納
HashMap<String,String> aadic;

void setup() {
  aadic = new HashMap<String, String>();
  aadic.put( "haa", "(ﾟДﾟ)ﾊｧ?" );
  aadic.put( "ase", "(；　Д　)" );
  aadic.put( "kita", "ｷﾀｰ(ﾟ∀ﾟ)ｰ!" );
}

void draw() {
  background(255);
  println( aadic.get("haa") );
  println( aadic.get("ase") );
  println( aadic.get("kita") );
}
```