プログラミング基礎

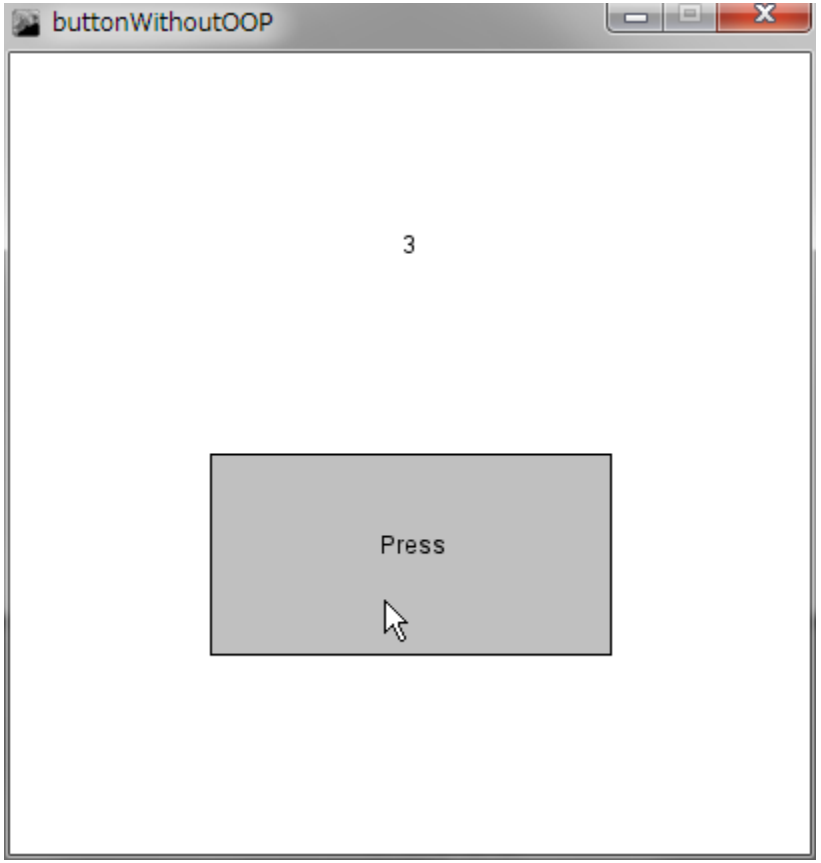
オブジェクト指向プログラミング
（前編）

今回の内容

- 復習：マウス操作に反応するプログラム
  - 例：ボタン（１個）
- ボタンを複数にするには？
  - クラスとオブジェクト
  - 例：ボタン（複数）
- 練習問題
  Drag-and-Drop（複数）を作る

例： ボタン

- 未完成のものが講義ページにあります
- 四角形のボタンを押すと数字が増える
- ボタンが押されるのは、
  - カーソルが四角形の上にいる状態で
  - マウスのボタンが押されたとき



```
boolean isPressed = false; // ボタンが押されているかどうか
boolean isHovered = false; // ボタンの上にマウスが来ているかどうか
int count = 0; // ボタンが押された回数
int bx = 100;
int by = 200;
int bw = 200;
int bh = 100;

void setup(){
  size(400, 400);
  textAlign(CENTER);
}

void draw(){
  background(255);
  // ボタンの状態に従ってボタンの色を変える
  if(isPressed){ fill(128); }
  else if(isHovered){ fill(192); }
  else{ fill(160); }

  // ボタンや回数を描画する
  rect(bx, by, bw, bh);
  fill(0);
  text("Press", bx + bw / 2, by + bh / 2);
  text(count, 200, 100);
}
```

（次のスライドに続きます）

（前のスライドから続きます）

```
// (x, y) がボタン上かどうかを判定する
boolean isOnTheButton(int x, int y){
  return x > bx && x < bx + bw && y > by && y < by + bh;
}

void mousePressed(){
  if(isOnTheButton(mouseX, mouseY)){ isPressed = true; }
}

void mouseReleased(){
  if(isPressed){ count++; }
  isPressed = false;
}

void mouseMoved(){
  if(isOnTheButton(mouseX, mouseY)){ isHovered = true; }
  else{ isHovered = false; }
}
```

押されたことを覚えておく

もう押されてない

ボタンの上に来たと覚えておく

ポイント！　描画と操作を分けて書く

- `draw()`　　は描画に徹する
- `mouse~()`　では描画はしない

- その方がわかりやすいから

オブジェクト指向プログラミング（前編）

ボタンを複数にするには？

- 複数あると言えば配列！こんな感じ？

```
boolean[] isPressed; // ボタンが押されているかどうかがたくさん
boolean[] isHovered; // ボタンの上にマウスが来ているかどうかがたくさん
int[] count; // ボタンが押された回数がたくさん
int[] bx; // ボタンのx座標がたくさん
int[] by; // ボタンのy座標がたくさん
int[] bw; // ボタンの幅がたくさん
int[] bh; // ボタンの高さがたくさん
```

こりゃ大変！

- でも自然に考えたらこんな感じ？

```
Button[] b; // ボタンがたくさんある
```

Button の配列を作りたい

↓

Button という型はない

↓

ないなら自分で作ろう！

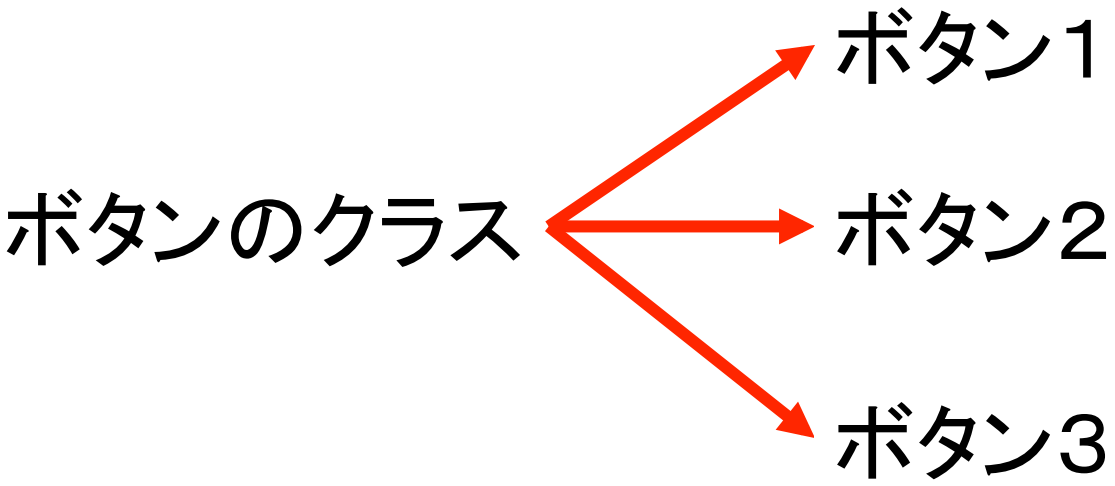
クラスとオブジェクト

- クラス： 複数のデータ（変数）・機能（関数）をひとまとめにした「型」
- オブジェクト： あるクラスの具体的なデータ

- 用語： オブジェクト指向プログラミング (Object-Oriented Programming)

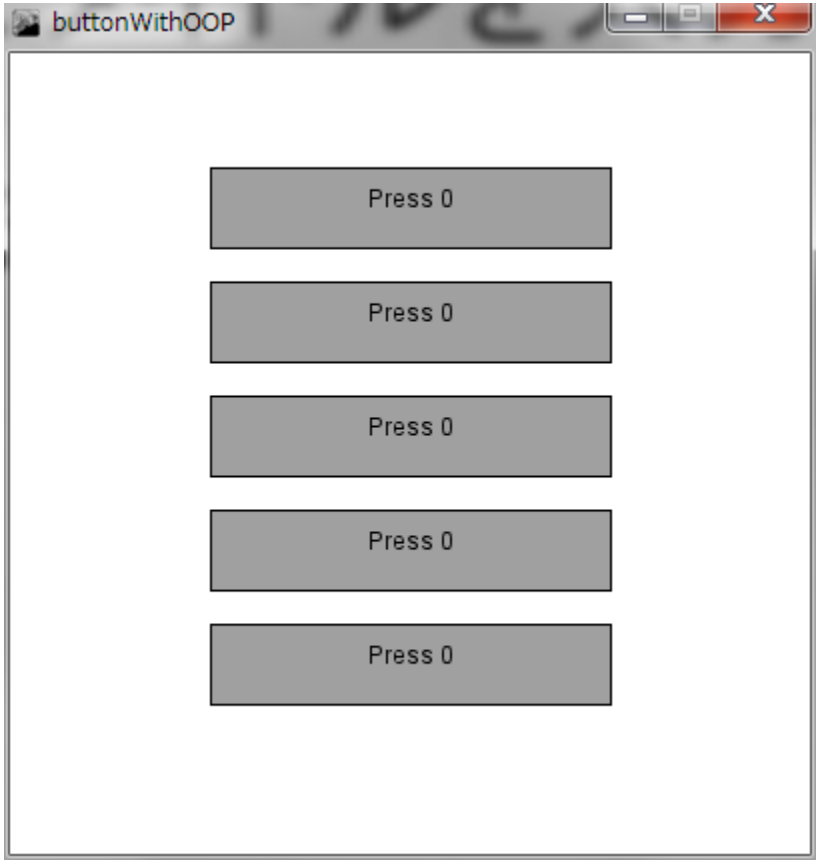
クラスとオブジェクト（つづき）

- ひとつの設計図（クラス）から
  複数のオブジェクトを作ることができる



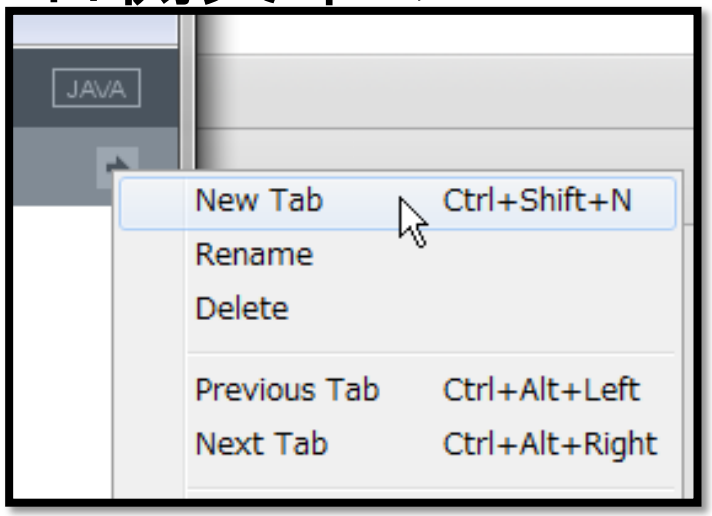
ボタン（OOP使用版）

- 各ボタンを押した回数がカウントされる
- Button クラスを作成、使用



クラスの作成

1. 右側矢印のメニューから New Tab を実行



2. 作成したいクラスの名前を入力

Button クラス

1. 新しいタブに Button クラスを作る

```
class Button {
  boolean isPressed = false; // ボタンが押されているかどうか
  boolean isHovered = false; // ボタンの上にマウスが来ているかどうか
  int count = 0; // ボタンが押された回数
  int x;
  int y;
  int w = 200;
  int h = 40;
}
```

【書式】 class クラス名{ … }

中身にはひとまとめにしたいものを書く

2. 作った Button クラスを使う

```
Button[] b; // ボタンがたくさんある

void setup(){
  size(400, 400);
  textAlign(CENTER);

  // ボタンをたくさん作る
  b = new Button[5];
  for(int i = 0; i < b.length; i++){
    b[i] = new Button();
    b[i].x = 100;
    b[i].y = height / (b.length + 2) * (i + 1);
  }
}

void draw(){
  background(255);
  for(int i = 0; i < b.length; i++){
    if(b[i].isPressed){ fill(128); }
    else if(b[i].isHovered){ fill(192); }
    else{ fill(160); }
    rect(b[i].x, b[i].y, b[i].w, b[i].h);
    fill(0);
    text("Press " + b[i].count, b[i].x + b[i].w / 2, b[i].y + b[i].h / 2);
  }
}
```

作ったクラスは他の型と同じように使える

Button 型のオブジェクトを作る

【書式】 new クラス名()

オブジェクト中の変数を使う

【書式】 オブジェクト.変数名

（次のスライドに続きます）

（前のスライドから続きます）

```
// (x, y) がボタン b 上かどうかを判定する
boolean isOnTheButton(int x, int y, Button b){
  return x > b.x && x < b.x + b.w && y > b.y && y < b.y + b.h;
}

void mousePressed(){
  for(int i = 0; i < b.length; i++){
    if(isOnTheButton(mouseX, mouseY, b[i])){ b[i].isPressed = true; }
  }
}

void mouseReleased(){
  for(int i = 0; i < b.length; i++){
    if(b[i].isPressed){ b[i].count++; }
    b[i].isPressed = false;
  }
}

void mouseMoved(){
  for(int i = 0; i < b.length; i++){
    if(isOnTheButton(mouseX, mouseY, b[i])){ b[i].isHovered = true; }
    else{ b[i].isHovered = false; }
  }
}
```

引数が増えているのに注意

ボタンが複数になったので、
全てのボタンを繰り返しで処理

さらに、関数もひとまとめにする

```
class Button {
  boolean isPressed = false; // ボタンが押されているかどうか
  boolean isHovered = false; // ボタンの上にマウスが来ているかどうか
  int count = 0; // ボタンが押された回数
  int x;
  int y;
  int w = 200;
  int h = 40;

  void drawButton(){
    if(isPressed){ fill(128); }
    else if(isHovered){ fill(192); }
    else{ fill(160); }
    rect(x, y, w, h);
    fill(0);
    text("Press " + count, x + w / 2, y + h / 2);
  }
}
```

```
void draw(){
  background(255);
  for(int i = 0; i < b.length; i++){
    b[i].drawButton();
  }
}
```

オブジェクト中の関数を使う

自分の変数を使うときはそのまま使える

用語の確認： メンバ (member)

- あるオブジェクトに属する変数や関数のことをメンバ (member) と呼ぶ
- メンバ変数・メンバ関数の有効範囲
  - {} の中、つまりクラスの中に限定
  - クラスの外から使うには . を使う
  - 特別な変数 “this”（例：次スライド）

関数をさらに追加する

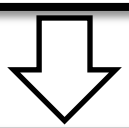
```
class Button {
  int x;
  int y;
  int w = 200;
  int h = 40;

  // 略

  // (x, y) がこのボタンの上かどうかを判定する
  boolean isOnTheButton(int x, int y){
    return x > this.x && x < this.x + w && y > this.y && y < this.y + h;
  }
}
```

this を使うとメンバ変数の方を指すようになる

```
if(isOnTheButton(mouseX, mouseY, b[i]))
```



```
if(b[i].isOnTheButton(mouseX, mouseY))
```

特別なメンバ関数、コンストラクタ

```
class Button {

  // 略

  Button(int x, int y){
    this.x = x;
    this.y = y;
  }

  // 略
}
```

コンストラクタ
クラス名と同名のメンバ関数
new したときに実行される

void と書かないことに注意

```
b[i] = new Button(100, height / (b.length + 2) * (i + 1));
```

new するときに引数を渡せる

練習問題（提出不要）

- 複数のrectを、drag-and-dropで動かせるプログラムを作成する
- 未完成版が講義ページにあります

```
boolean isGrabbed = false; // ドラッグ中かどうか
int ox = 100;
int oy = 200;
int ow = 100;
int oh = 100;
int gx;
int gy;

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

void draw(){
  background(255);
  if(isGrabbed){ fill(192); }
  else{ fill(160); }
  rect(ox, oy, ow, oh);
}

// (x, y) がボタン上かどうかを判定する
boolean isOnTheButton(int x, int y){
  return x > ox && x < ox + ow && y > oy && y < oy + oh;
}
```

※ gx, gy については次スライドの図を参照

ここまでのところはボタンとだいたい同じ

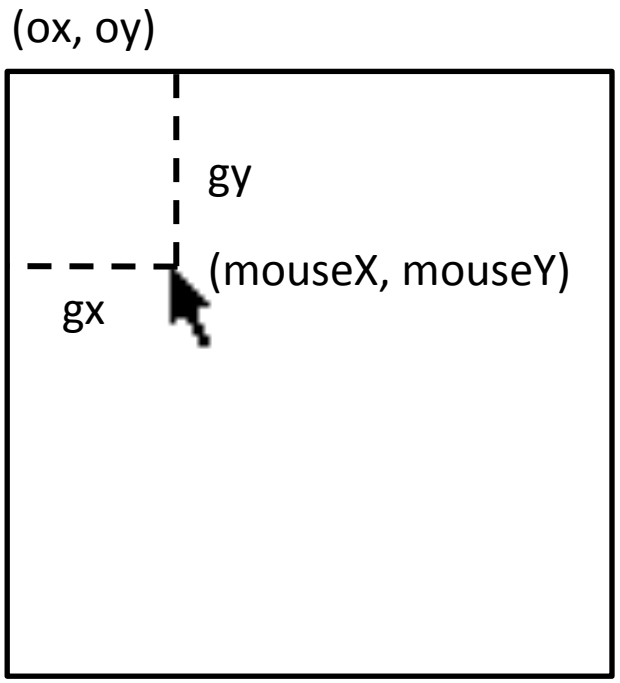
（次のスライドに続きます）

（前のスライドから続きます）

```
void mousePressed(){
  if(isOnTheButton(mouseX, mouseY)){
    isGrabbed = true;
    gx = mouseX - ox;
    gy = mouseY - oy;
  }
}

void mouseDragged(){
  if(isGrabbed){
    ox = mouseX - gx;
    oy = mouseY - gy;
  }
}

void mouseReleased(){
  isGrabbed = false;
}
```



※ ox, oy, gx, gy は上図の通り