モバイルシステム演習

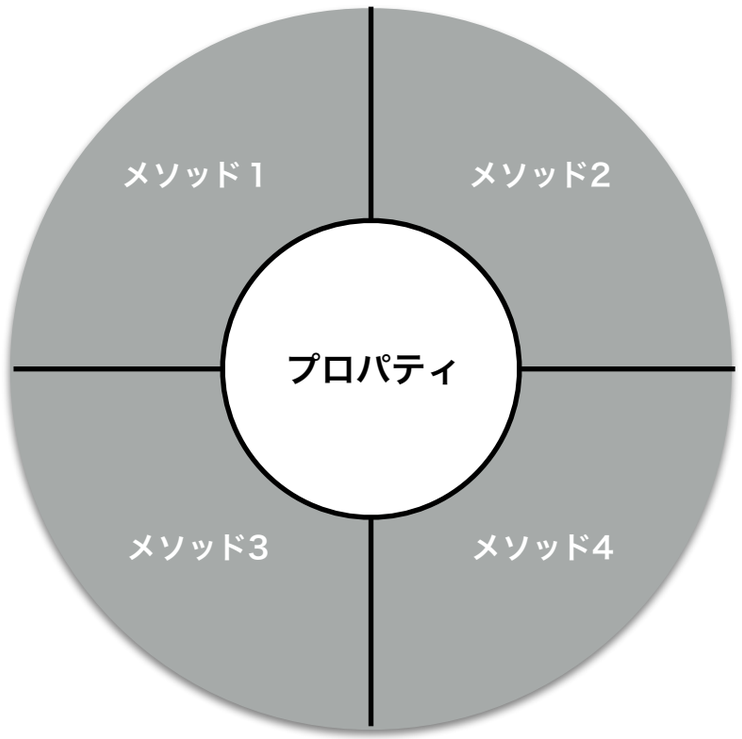
Processing

- オブジェクト指向プログラミング -

株式会社GOCCO.
watanabe@gocco.co.jp

オブジェクト指向プログラミング(OOP)

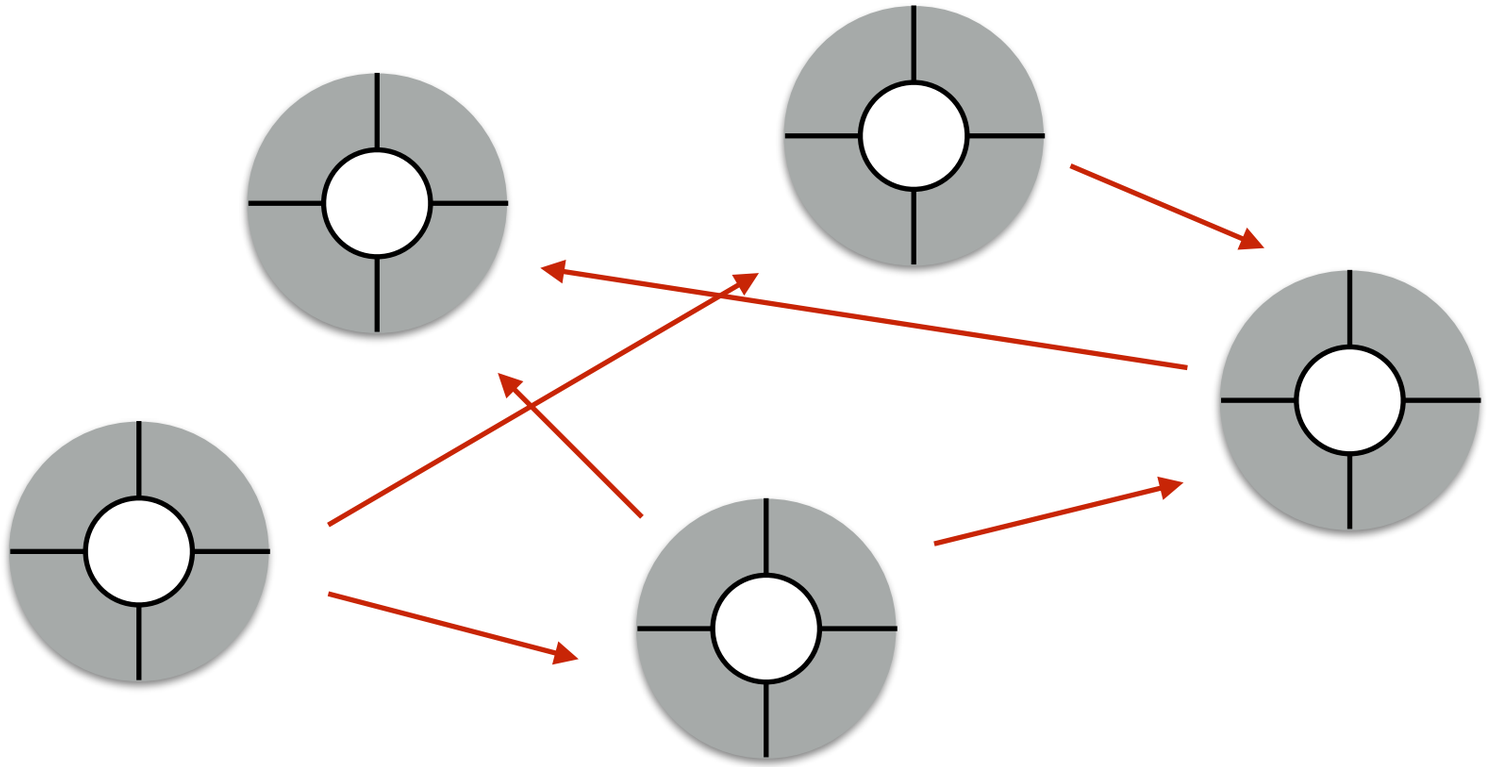
オブジェクトはプロパティ（性質、状態)と
メソッド（動作、ふるまい）から構成される



オブジェクト

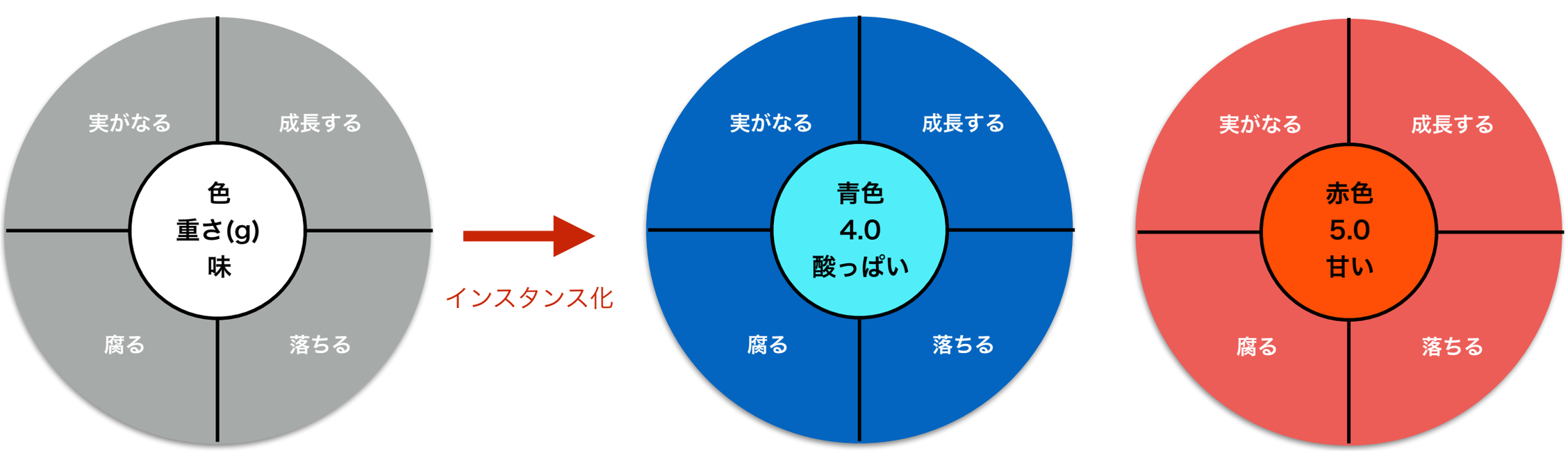
オブジェクト指向プログラミング(OOP)

オブジェクトの集まりとしてプログラムを構成
オブジェクト同士がメッセージを送り合う



クラス

クラスとはオブジェクトの設計図
クラスをインスタンス化(実体化)することで、
インスタンス(オブジェクト)となる



リンゴ
(クラス)

青りんご
(インスタンスオブジェクト)

ふじ
(インスタンスオブジェクト)

Proessingで
オブジェクト指向プログラミング

クラス定義

クラス名

```
class Particle {
  float x;
  float y;
  float step;

  Particle(float xx, float yy) {
    // 初期化
    x = xx;
    y = yy;
    step = 20;
  }

}
```

プロパティ

コンストラクタ(特殊関数)

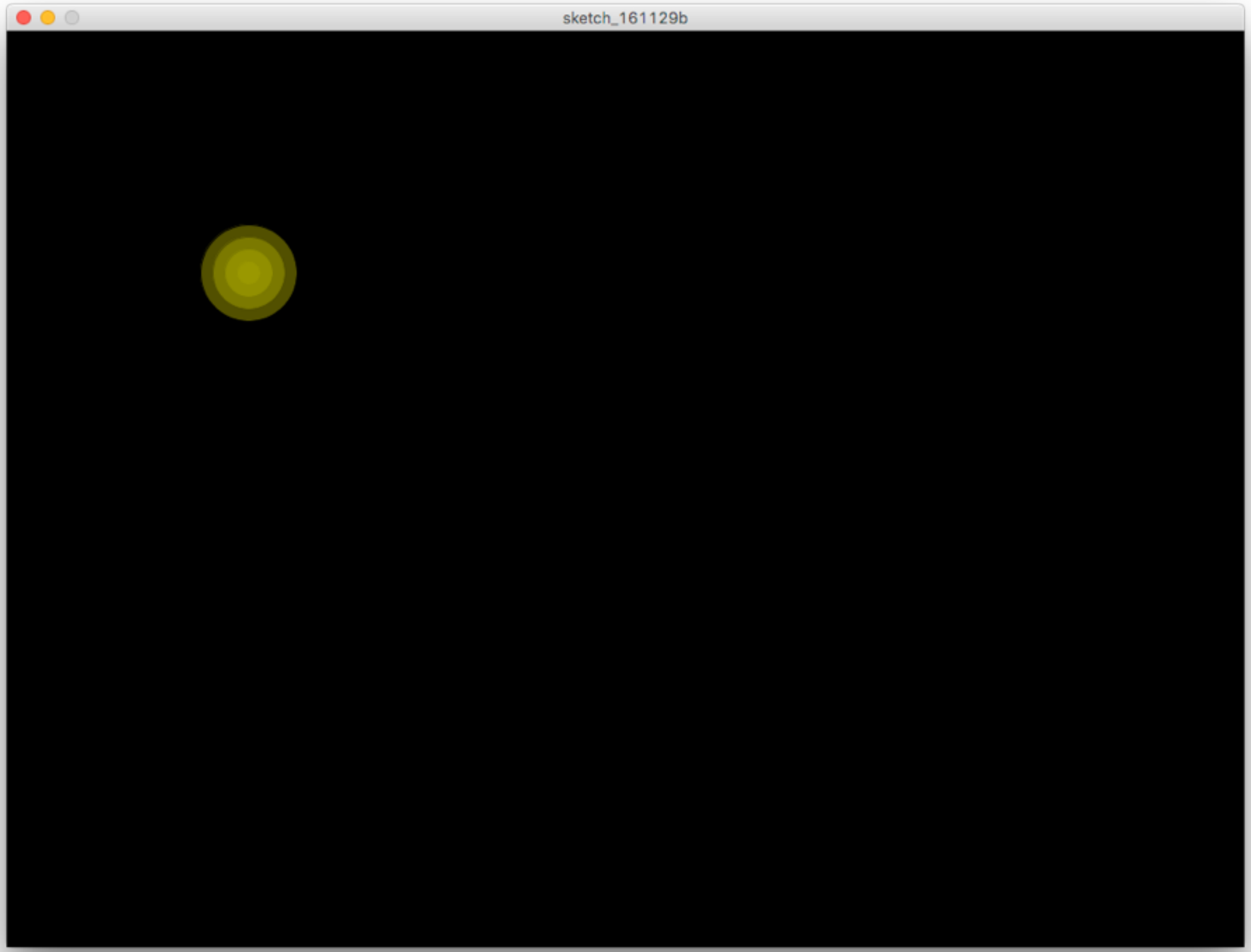
- 結果の型はない(voidとか)
- 引数のパラメータは任意に定義できる
- 必ずクラス名と同じ名前
- 実体化する時に呼ばれる

前回定義した関数を組み込む

```
class Particle {
  float x;
  float y;
  float step;

  Particle(float xx, float yy) {
    // 初期化
    x = xx;
    y = yy;
    step = 20;
  }

  void drawCircle() {
    pushMatrix();
    translate(x, y);
    for (int i = 0; i < 5; i++) {
      fill(255, 255, 0, i * step);
      ellipse(0, 0, i * step, i * step);
    }
    popMatrix();
  }
}
```

クラスからオブジェクトを生成するまで
(インスタンス化)

```
Particle p;        //オブジェクト宣言 Particleクラスのオブジェクトpを宣言

void setup() {
  size(1024, 758);
  p = new Particle(200, 200);
}                          // Particleクラスからオブジェクトを作成

void draw() {
  background(0);
  p.drawCircle();    // drawCircle関数を実行
}
```


sketch_161129b

Particleクラスを拡張

オブジェクトを動かすにはどうしたらよいか？

プロパティに
forceを加えて動くオブジェクトを作成する

```
class Particle {
  float x;
  float y;
  float step;
  float vx;
  float vy;
```

速度

```
  void update() {
     x += vx; // x = x + vxと同じ
     y += vy; // y = y + vyと同じ
  }
}
```

現在位置を更新

生成したオブジェクトに速度プロパティを代入
追加した関数を呼び出して、位置を更新

```
Particle p;

void setup() {
  size(1024, 758);
  p = new Particle(200, 200);
  p.vx = 5;
  p.vy = 2;
}

void draw() {
  background(0);
  p.update();
  p.drawCircle();
}
```

全体のコード

```
Particle p;

void setup() {
  size(1024, 758);
  p = new Particle(200, 200);
  p.vx = 5;
  p.vy = 2;
}

void draw() {
  background(0);
  p.update();
  p.drawCircle();
}
```

```
class Particle {
  float x;
  float y;
  float step;
  float vx;
  float vy;

  Particle(float xx, float yy) {
    // 初期化
    x = xx;
    y = yy;
    step = 20;
  }

  void drawCircle() {
    pushMatrix();
    translate(x, y);
    for (int i = 0; i < 5; i++) {
      fill(255, 255, 0, i * step);
      ellipse(0, 0, i * step, i * step);
    }
    popMatrix();
  }

  void update() {
    x += vx; // x = x + vxと同じ
    y += vy; // y = y + vyと同じ
  }
}
```

クラス設計図

Particle

プロパティ
x, y, step, vx, vy

コンストラクタ
Particle(float xx, float yy)

メソッド
update()

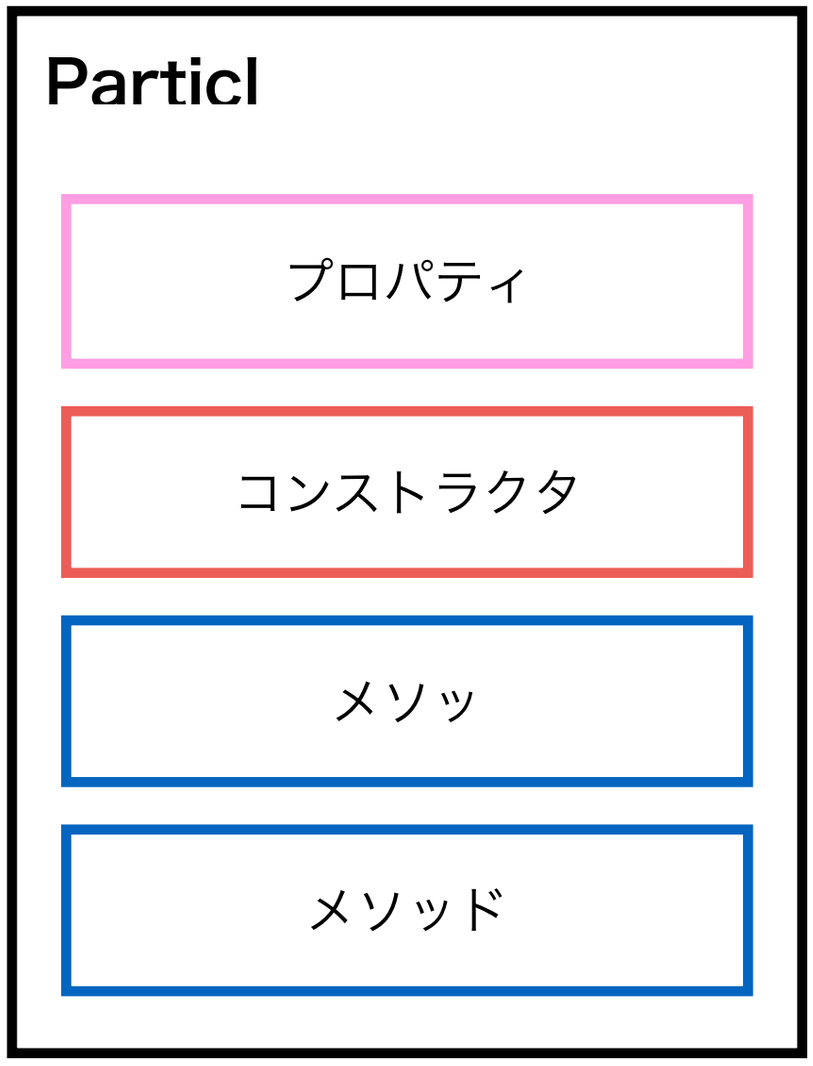
メソッド
drawCircle()

練習1

Particleオブジェクトを増やして、画面に
大きさ、色、速度の違うparticleオブジェクトを10個
表示させてみよう

ヒント

- 色プロパティを追加する
- drawCircle関数内でfillを使う



練習2

上下左右で衝突判定と跳ね返りをするオブジェクトを
表示させてみよう

ヒント

- クラスのupdate関数の中に跳ね返りを定義する