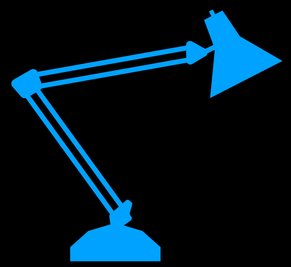
Design Programming

Spring 2018 - 9

2018.6.13 Keio University, SFC

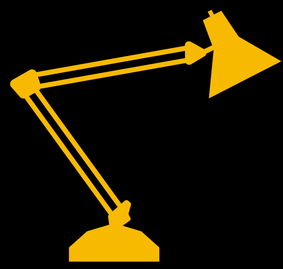
オブジェクト指向とは

オブジェクト指向とは，物を実体化させ、物に情報を与え，機能させることで
プログラミングする手法のことである.



電気スタンド（物）を作りたい
（オブジェクト）

設計図
（クラス）



実際に作ったもの
（インスタンス）

オブジェクト指向とは

クラス＝物の設計図をプログラムで表現したもの

クラスはメソッド（メンバ関数）とフィールド（メンバ変数）で構成される

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

    void display() {
        ellipse(x, y, d, d);
    }
}
```

クラス（＝物）名を宣言する

（頭文字は大文字だと他の変数や関数などと区別しやすい）

メンバ変数，フィールド（クラス内の変数）＝物の情報

メンバ関数，メソッド（クラス内の関数）＝物の機能

09/particle/particle.pde

粒子という物（オブジェクト）のクラスを定義した

クラスからインスタンスを生成して使用する

```
Particle p;

void setup() {
    size(500, 500);
    noStroke();
    p = new Particle();
    p.x = 133.0;
    p.y = 150.0;
    p.d = 50.0;
}

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

クラス以下略

09/particle/particle.pde

Particleクラスのオブジェクトpを宣言

pにParticleクラスからインスタンスを生成

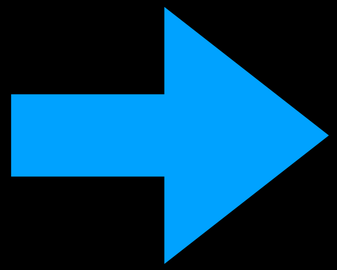
pの変数x, y, dに値を代入

pの関数display()を実行させ，入力通りに描画させる

クラスの拡張

現在地（x, y）直径d

関数display()



現在地（x, y）直径d

速度vx, 速度vy

関数display()

関数update()

クラスの拡張

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

    void display() {
        ellipse(x, y, d, d);
    }
    void update() {
        x += vx;
        y += vy;
    }
}
```

現在地（x, y）直径d

速度vx, 速度vy

関数display()

関数update()

09/particleUpdate/particleUpdate.pde

クラスの拡張

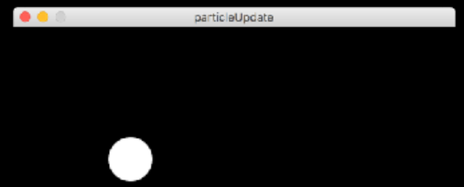
```
Particle p;

void setup() {
    size(500, 500);
    noStroke();
    p = new Particle();
    p.x = 133.0;
    p.y = 150.0;
    p.d = 50.0;
    p.vx = 5.0;
    p.vy = 2.0;
}

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

追加したvx, vyに値を代入

追加したupdate()関数を実行



09/particleUpdate/particleUpdate.pde

クラスにコンストラクタを追加

```
Particle p;

void setup() {
    size(500, 500);
    noStroke();
    p = new Particle();
    p.x = 133.0;
    p.y = 150.0;
    p.d = 50.0;
    p.vx = 5.0;
    p.vy = 2.0;
}

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

クラスからインスタンスを生成時に

変数を渡してまとめることが可能

```
p = new Particle(133.0, 150.0, 50.0, 5.0, 2.0);
```

現在地（x, y）直径d

速度vx, 速度vy

関数display()

関数update()

クラスにコンストラクタを追加

コンストラクタとは，インスタンス生成時に

実行される関数のような存在

・クラス名とコンストラクタ名は一致させる

・返り値は持てない

現在地（x, y）直径d

速度vx, 速度vy

コンストラクタ

関数display()

関数update()

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

    Particle(float _x, float _y, float _d, float _vx, float _vy) {
        x = _x;
        y = _y;
        d = _d;
        vx = _vx;
        vy = _vy;
    }

    メンバ関数以下省略

}
```

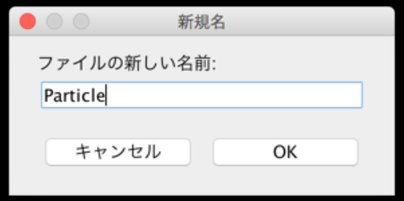
09/particleConstructor/particleConstructor.pde

ファイル分割について

クラスは独立した設計図で，メインプログラムと

ファイルの切り離しが可能.

新規タブ



クラス名で保存

クラス名.pdeが生成

汎用的なクラスを作っておけば，毎回作る必要がない

09/particleSeparate/particleSeparate.pde

演習 1

前回の跳ね返りをParticleクラスに実装しなさい

演習 1

```
void update() {
        x += vx;
        y += vy;
        if (x >= width - d / 2 || x <= d / 2)
            vx = -vx;
        if (y >= height - d / 2 || y <= d / 2)
            vy = -vy;
    }
```

09/practice1/practice1.pde

配列と組み合わせる

```
int n = 100;
Particle[] p = new Particle[n];

void setup() {
    size(500, 500);
    noStroke();
    for (int i = 0; i < p.length; i++) {
        float d = random(20.0);
        p[i] = new Particle(
            random(d / 2, width - d / 2),
            random(d / 2, height - d / 2),
            d,
            random(5.0),
            random(5.0));
    }
}

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

09/particleArray/particleArray.pde

課題

- 今日学習した「オブジェクト指向プログラミング」を使用してスケッチを１つアップロードしなさい.
  アップロード先は「09 object」とする.
  ※今日学習していない分野でもすでに自分が知っている技術は使用可能とする.

  〆切：6月19日24時まで

次回

- 応用表現１