プログラミング基礎

オブジェクト指向プログラミング

(Object-Oriented Programming; OOP)

今回の内容

- クラスとオブジェクト
- 複数キャラクターのアニメーション
- 課題：アニメーション作品

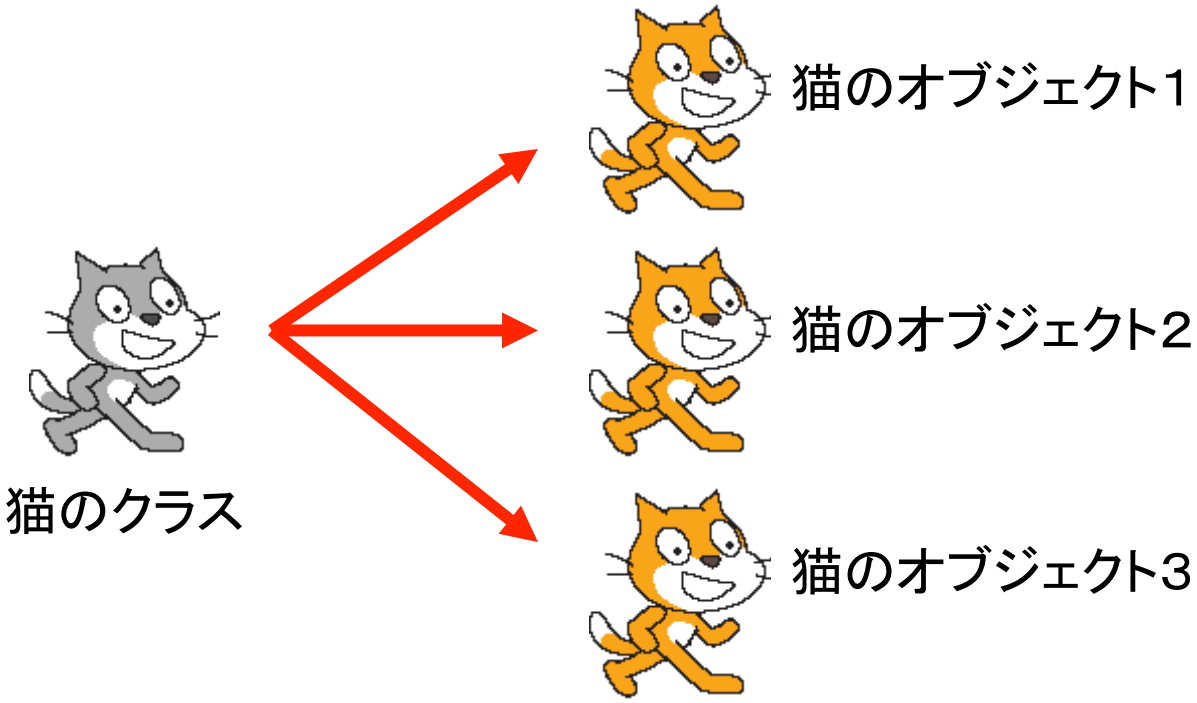
クラスとオブジェクト

- オブジェクト (Object)
  - 複数のデータ（変数）や機能（関数）を
    ひとまとめにしたもの
- クラス (Class)
  - オブジェクトの元となる設計図のようなもの

- 用語
  - オブジェクト指向プログラミング
  - オブジェクト指向言語

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



（参考）Scratch との比較

- Scratch の「スプライト」がオブジェクトに相当
  - 「猫のX座標」、「猫のY座標」
- Scratch には「クラス」はない
  - たとえば猫を複数表示したかったら、プログラムをまるまるコピペしないといけない
  - Processingでは猫のプログラムを1回だけ書く

例：クラスを作ってみよう

```
class Cat{
  int x;
  int y;

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

  void drawCat(){
    image(catImage, x - catImage.width / 2, y - catImage.height / 2);
  }
}
```

【書式】 class クラス名{ … }

クラスの中身には
変数や関数を書く

（実行するには次のスライドのプログラムも必要）

例:オブジェクトを作って使ってみよう

（前のスライドのプログラムからつづけて）

```
PImage catImage;
Cat c;

void setup(){
  size(400, 400);
  catImage = loadImage("cat1-a.gif");
  c = new Cat();
}

void draw(){
  background(255);

  c.moveTo(mouseX, mouseY);
  c.drawCat();
}
```

作ったクラスは型と同じように使える
(Cat 型の変数 c を作っている)

Cat 型のオブジェクトを作って
変数に覚えておく

【書式】 new クラス名()

関数を外から使う場合
（詳細は後述）

メンバ変数・メンバ関数

- 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
  - 他と同様 {} の中、つまりクラスの中に限定
  - クラス外から使う場合には . (dot) を使う
  - 特別な変数 “this” が使える
  - 詳細は次のスライド

メンバ変数と変数の有効範囲

```
class Cat{
  int x;
  int y;

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

  void drawCat(){
    image(catImage, x - catImage.width / 2, y - catImage.height / 2);
  }
}
```

特別な変数 this を使うと
メンバ変数の方を示す

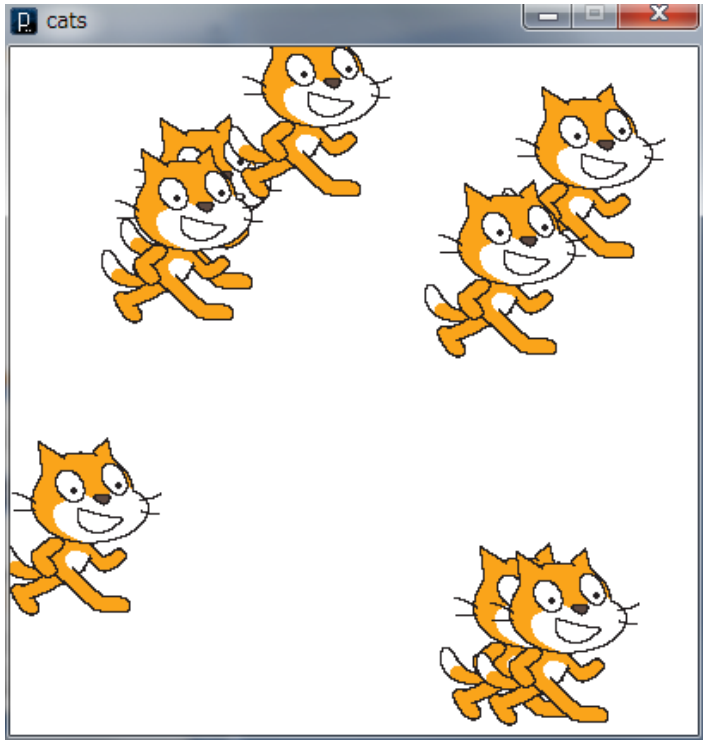
紛らわしくない場合は
this を付けなくてもいい

練習

- 練習１： Cat 型のオブジェクトを2つ作って、異なる場所に描画してみよう
  - Cat 型の変数を2つ (たとえば c1 と c2) を作る
- 練習２: Cat に拡大縮小機能を追加しよう
  1. 拡大率を覚えるメンバ変数 scale を追加
  2. 関数 drawCat を、scale を使うように修正
  3. scale を変更する関数 zoom を追加
- 練習３: 練習２の Cat を使って、マウスが近づくと大きくなって離れると小さくなる猫を作ろう

複数キャラクターのアニメーション

- 複数の猫が画面内を動き回るプログラム
  1. キャラクターのクラスを作る
  2. そのクラスからオブジェクトを複数作る
     - 作ったら配列に入れておく
  3. すべてのオブジェクトを動かす＆描画する



Cat クラスに追加

```
class Cat{
  int x, y;
  int vx, vy;

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

  void setVelocity(int vx, int vy){
    this.vx = vx;
    this.vy = vy;
  }
```

- 速度を覚える変数 vx, vy
- 速度を設定する関数 setVelocity

```
  void move(){
    x = constrain(x + vx, 0, width);
    y = constrain(y + vy, 0, height);
  }

  boolean isInsideX(){ return x > 0 && x < width; }
  boolean isInsideY(){ return y > 0 && y < height; }

  void bounceX(){ vx = - vx; }
  void bounceY(){ vy = - vy; }
```

- 速度通りに動かす関数
- 画面内か判定する関数
- 跳ね返る(速度反転)関数

setup と draw

```
PImage catImage;
Cat[] cats;

void setup(){
  size(400, 400);
  catImage = loadImage("cat1-a.gif");
  cats = new Cat[8];
  for(int i = 0; i < cats.length; i++){
    cats[i] = new Cat();
    cats[i].moveTo(ceil(random(width)), ceil(random(height)));
    cats[i].setVelocity(ceil(random(10)), ceil(random(10)));
  }
}
```

Cat の配列を作って、
初期位置・速度を乱数で初期化

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

  for(int i = 0; i < cats.length; i++){
    Cat c = cats[i];
    c.move();
    if(!c.isInsideX()){
      c.bounceX();
    }
    if(!c.isInsideY()){
      c.bounceY();
    }
    c.drawCat();
  }
}
```

1回drawする毎に猫をすべて動かす
画面外に出た猫は跳ね返らせる

（参考）クラスを使わないと…

- 4つの配列を使って作ることになる

```
int[] x, y;
int[] vx, vy;
```

- Cat クラスを作ると配列は1つだけでOK

課題4：アニメーション作品

- 以下の３要件をすべて満たすアニメーション作品を製作する
  - 画像を2枚以上使用する
  - 配列を使用する
  - クラスを使用する
- 例
  - 複数の動く猫がそれぞれ歩いている
  - 背景がパラパラ漫画で前景が複数の猫

提出方法 今までと違うので注意！

- プロジェクトのフォルダを zip 圧縮する
  1. プロジェクトを保存
  2. Tools > Archive Sketch
  3. (名前).zip という名前のファイルが出来る
- 上記手順で作成した zip ファイルを
  課題提出システムから提出

ヒント：色々な動き（１）

- 等速で動く
  - 右方向　　vx > 0, vy == 0
  - 左方向　　vy < 0, vy == 0
  - 【考えてみよう】上下方向に動く
- 速さが変化する
  - 加速　1回動くごとに vx, vy を増やす
  - 【考えてみよう】減速する
  - 【考えてみよう】最初は加速して、減速して止まる

ヒント:色々な動き(2)

- ジャンプ
  1. 最初
     - 上方向に動いてほしいので vy < 0
  2. 重力: 1回動くごとに vy を増やす
     - vy > 0 になると落ちてくる
  3. 着地
     - y 座標が地面に到達したら vy = 0
     - 再びジャンプしてほしい場合は 1. に戻る
  - 【考えてみよう】走りながらジャンプする