計算機プログラミングI

第9回　2002年12月12日(木)

- クラスの設計
  - ◊インスタンス変数
  - ◊コンストラクタ
  - ◊インスタンスメソッド
  - ◊親クラス

……どのように定義するか?

なぞなぞ

なぞなぞ

- 「1と2」と「4と3」がありました
  この2つを掛けたらいくつ?

なぞなぞ

- 「1と2」と「4と3」がありました
  この2つを掛けたらいくつ?
- 答え:「-2と11」

なぞなぞ

- 「1と2」と「4と3」がありました
  この2つを掛けたらいくつ?
- 答え:「-2と11」
- 「aとb」と「cとd」を掛けると
  「ac-bdとad+bc」になります
  このような数はどんな数?

なぞなぞ

- 「1と2」と「4と3」がありました
  この2つを掛けたらいくつ?
- 答え: 「-2と11」
- 「aとb」と「cとd」を掛けると
  「ac-bdとad+bc」になります
  このような数はどんな数?
- 答え: 複素数

なぞなぞ

- 「1と2」と「4と3」がありました
  この2つを掛けたらいくつ?
- 答え: 「-2と11」
- 「aとb」と「cとd」を掛けると
  「ac-bdとad+bc」になります
  このような数はどんな数?
- 答え: 複素数
- 複素数を1つの値として扱う
  → 複素数**オブジェクトの設計**

複素数 (complex number)

- 複素数のベクトル表現
  $x+yi$ 実部と虚部の実数
- プログラム中での扱い
  - ◊実数(double)型の変数を2つ用意して計算、or
  - ◊一つの「複素数」オブジェクトとして扱う

複素数: クラスの定義

- 複素数オブジェクトの性質の定義
  = クラスの定義
- 名前 : Complex

```
/** 複素数 */
class Complex {
  インスタンス変数・
  インスタンスメソッド・
  コンストラクタの定義
}
```

複素数: インスタンス変数の定義

- 複素数のベクトル表現: x+yi
  → 2つの実数
  → 2つの実数型インスタンス変数

```
class Complex {
   public double real;   // 実部
   public double imag;   // 虚部
   インスタンスメソッド・
   コンストラクタの定義
}
```

```
Complex c = new Complex();
c.real = 1;
c.imag = 2;
```

複素数:コンストラクタの定義

- Complex c = new Complex(1,2); のようにしたい

```
class Complex {
  public double real;  // 実部
  public double imag;  // 虚部
  /** 実部，虚部から複素数を作る */
  public Complex(double r, double i) {
    real = r;
    imag = i;
  }
}
```

複素数:インスタンスメソッドの定義

- 複素数を使った計算: z_1+z_2, z_1*z_2, $|z|$, ...
- 計算方法: $(x_1+y_1i)+(x_2+y_2i) = (x_1+x_2)+(y_1+y_2)i$
  $(x_1+y_1i)*(x_2+y_2i) = ...$
  これを毎回書くのは大変
- メソッドを使うと「定義」と「使用」を分けられる
- オブジェクト指向でのやり方:
  - ◊複素数クラスに足し算を「定義」　～**メソッド定義**
  - ◊複素数オブジェクトに足し算を指示して「使用」　～**メソッド呼び出し**

複素数:インスタンスメソッドの定義

```
class Complex {
  インスタンス変数・
  コンストラクタの定義
   public Complex add(Complex y) {
      double r = real + y.real; //実部の計算
      double i = imag + y.imag; //虚部の計算
      return new Complex(r,i);  //新しい複素数を作って返す
   }
}
```

```
Complex c1 = new Complex(1,2);
Complex c2 = new Complex(3,4);
Complex c3 = c1.add(c2);
```

練習9-1*: 複素数クラス

- Complexクラスを完成させる

  インスタンスメソッドの追加

  - 掛け算をする multiply ← 結果は複素数
  - 絶対値を求める magnitude ← 結果は実数
  - その他(引き算・割り算・・・)

- Complexクラスをテストする

  a = 1+2*i*, b = 3+4*i*, c=5+6*i* のときの

  (ア) b+c　(イ) ab　(ウ) ac

  (エ) ab+ac　(オ) a(b+c)

(参考プログラム Complex.java, ComplexTest.java)

複素数の応用

- マンデルブロ集合
  - ◊複素関数 $f(z)=z^2+c$ の非発散領域 c の集合
  - ◊自己相似形

(実演)

- プログラミング
  - ◊複素数オブジェクトを使って $f(z)=z^2+c$ を定義
  - ◊タートルグラフィクスライブラリを使って作図

練習

- 9-2(マンデルブロ集合)
  - ◊ Complexクラスに f, divergeNumber を追加。
  - ◊ Mandelクラスを完成 (参考: Mandel.java)
- 9-3 (マンデルブロ集合の自己相似性)
  集合の一部を拡大
- 9-4 (他のマンデルブロ集合)
  f を $f(z) = z^3+c$ に変える
- 9-5 (ジュリア集合)
  $f(z) = z^2 - a$ のときに
  $f^k(c)$ が発散しないような
  c の集合
- 9-6 (オブジェクトを使う意義)
  Complex オブジェクトを使わず計算するプログラムの読み易さ

Morphing再び

- 課題で作ったMorphingのためのコード

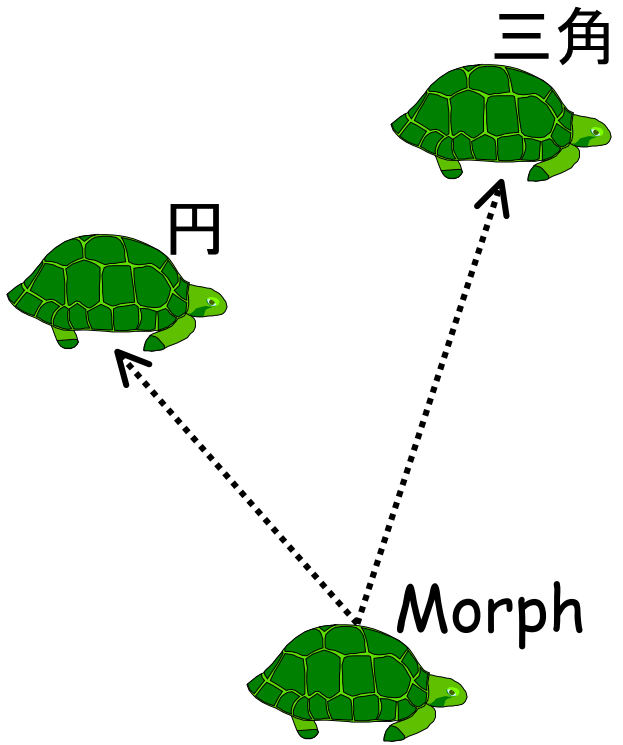
```
for(i = 0; i < steps; i++){
    mc (円)を移動
    mt (三角形)を移動
    mm.moveTo((mc.getX()+mt.getX())/2,
              (mc.getY()+mt.getY())/2,
              0);
}
```

- 監督者が中間の位置を計算し、
  その位置へタートルを移動させる

オブジェクト指向によるMorphing

- 中間の図形を描くタートルは
  - ◊追いかけるタートルを知っている
  - ◊追いかけろ、と指示を受けると
    自ら中間地点へ移動

```
for(i = 0; i < steps; i++){
    mc (円)を移動
    mt (三角形)を移動
    mm.track();
}
```



Morphクラスの設計

```
public class Morph
                extends Turtle {
  // 追いかける対象
  public Turtle m1, m2;
  コンストラクタ・インスタンスメソッドの定義
}
```

- Turtleクラスを拡張
- 追いかける対象
  →インスタンス変数

Morph: コンストラクタ

- Morph m = new Morph(mCircle, mTriangle);
  のようにしたい

```
public class Morph extends Turtle {
  public Turtle m1, m2;
   public Morph(Turtle m1, Turtle m2) {
     this.m1 = m1;
     this.m2 = m2;
   }
  インスタンスメソッドの定義
}
```

Morph:インスタンスメソッド

- m.track() とすると、m1 と m2 の中間へ移動

```
public class Morph extends Turtle {
  public Turtle m1, m2;
  コンストラクタ(略)
   public void track() {
       自身をm1とm2の中間に移動
   }
}
```

Morphクラスの使用

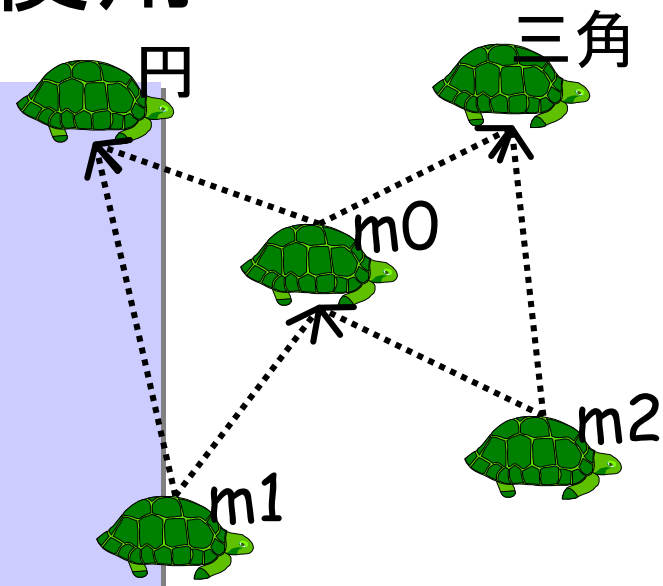
```
Turtle mc = new Turtle();
Turtle mt = new Turtle();
Morph mm = new Morph(mc,mt);

for(i = 0; i < steps; i++){
    mc (円)を移動
    mt (三角形)を移動
    mm.track();
}
```

- 使う側は「どうやって中間の位置を
  追いかけているか」を気にしなくてよい

Morphクラスの使用

```
Turtle mc = new Turtle();
Turtle mt = new Turtle();
Morph m0 = new Morph(mc,mt);
Morph m1 = new Morph(mc,m0);
Morph m2 = new Morph(m0,mt);
for(i = 0; i < steps; i++){
    mc (円)を移動
    mt (三角形)を移動
    m0.track(); m1.track(); m2.track();
}
```



- 「Morphオブジェクトを追いかけるMorphオブジェクト」も可能 ← MorphはTurtleの子クラス

練習

- 9-7 (Morphオブジェクト)*
  - ◊Morph クラスを完成
  - ◊課題プログラムをMorph オブジェクトを使うように
- 9-8 (段階的なMorphing)
  - ◊Morphを追いかけるMorph