オブジェクト指向とゲームプログラミング

○　Java 基礎編 － 第５回　this，静的メンバ　○

this

ある処理を行うとき、自分自身を参照したいことがあります。たとえば、あるメソッドでほかのメソッドを呼び出す際に、引数で自分自身のオブジェクトを渡すような場合です。同じように、メソッドが呼び出し元に対して、操作中のオブジェクト自身を返す必要が生じる場合があります。

そういった場合に、Javaではキーワードthisを使ってオブジェクト自身を参照することができます。thisを宣言したり、何か特別なことを行う必要はありません。thisは常に使用可能であり、現在のオブジェクトに対する参照を返します。

thisは、通常のクラス型変数と同じように使うことができます。以下の文では、現在のオブジェクトのvisitメソッドを呼び出し、引数として自分自身を渡しています。

```
this.visit(this);          // 現在のオブジェクトのvisitメソッドの呼び出し
```

同様に、現在のオブジェクトのフィールドにアクセスするために、フィールド名の前にthisを付けることができます。

```
this.posX = 89.3;    // 現在のオブジェクトのメンバ変数posXに89.3を代入
```

ほとんどのメソッドでは省略しているthisを、明示的に指定することができます。したがって、以下の２つの構文は同じ意味です。

```
this.posX = 89.3;    // 現在のオブジェクトのposXに89.3を代入
posX      = 89.3;    // メンバ変数posXに89.3を代入(thisを省略した場合)
```

上記のようにthisを使うことが有益な状況としては、クラス内のローカル変数とメソッドが同じ名前を持っている場合が考えられます。thisを使うことによって、これら２つの変数を区別することができます。なお、staticを使って宣言された静的なメソッド内では、thisを使うことはできません。

また、コンストラクタ内でthisにカッコをつけると、任意のコンストラクタを呼び出すことができます。thisによるコンストラクタの呼び出しは、複数のコンストラクタがあるとき、あるコンストラクタから別のコンストラクタを呼び出し、その後でなにか処理を行いたい、という場合に使用します。

```
this(引数, 引数, …);    // 任意のコンストラクタの呼び出し
```

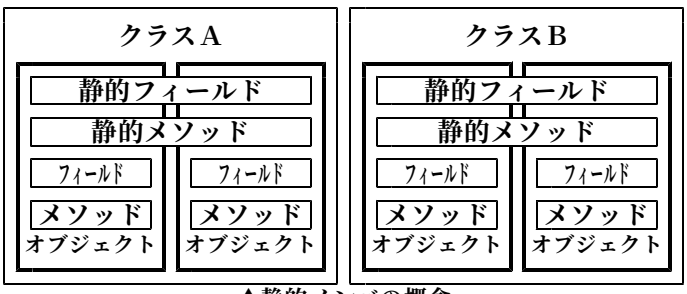
静的メンバ

同じクラスのオブジェクトが複数存在する際､オブジェクト間でデータを共有したい場合があります。このような場合には、キーワードstaticを使ってフィールドを宣言します。

クラスのオブジェクトをいくつ生成しても、各オブジェクトのフィールドは、ほかのオブジェクトからは独立した値を持っています。しかし、staticを使って宣言された変数は、同じクラスのすべてのオブジェクト間でその領域を共有します。したがって、あるオブジェクトがstaticなフィールドの値を変更すると、ほかのオブジェクトのstaticなフィールドもその値に変更されます。

staticなフィールドは、オブジェクトを生成しなくても使用することができます。また、オブジェクトが生成されるごとに初期化しても意味がないので、コンストラクタで初期化しないようにします。

メソッドにもstaticを指定することができます。staticなメソッドは、オブジェクトを生成しなくても呼び出すことができます。ただし、staticなメソッドの中では、そのクラスのstaticでないメンバにアクセスすることができません。どうしてもアクセスしたい場合は、そのメソッドにthisを渡します。



▲静的メンバの概念

static指定されたメンバを静的メンバと呼びます。静的メンバは、インスタンスを生成しなくても使用でき、クラス名を使ってアクセスします。

```
class Character {
    public final static int   MAX_ANIME = 5;        // アニメーション最大数

    static int   count = 0;                          // 総キャラクタ数

    public static void incCount() { count++; }       // キャラクタ数インクリメント
    public static void decCount() { count--; }       // キャラクタ数デクリメント
    public static int  getCount() { return count; } // キャラクタ数取得

    double      posX;                                // ｘ座標
    double      posY;                                // ｙ座標
    int         animeNo = 0;                         // アニメーション番号

    // コンストラクタ
    public Character() {
        this(0.0, 0.0);
    }

    // コンストラクタ
    public Character(double x, double y) {
        incCount();     // キャラクタが生成されるごとにカウンタを１増やす
        posX = x;
        posY = y;
    }

    // ファイナライザ
    protected void finalize() {
        decCount();     // メモリから消滅したら、カウンタを１減らす
    }

    // 移動
    public void move(double addX, double addY) {
        posX += addX;
        posY += addY;
    }

    // アニメーション
    public void animetion() {
        animeNo++;
        if(MAX_ANIME <= animeNo)
            animeNo = 0;
    }

    // アクセスメソッド
    public double getX() { return posX; }   // ｘ座標取得
    public double getY() { return posY; }   // ｙ座標取得
}
```

たとえば、上記のクラスでは、静的なフィールドとしてMAX_ANIMEとcount、静的なメソッドとしてincCount, decCount, getCountが定義されています。これらには、以下のようにクラス名でアクセスします(自身のクラス内では、クラス名を省略することができます)。

```
Character.getCount();   // キャラクター総数の取得
```

静的なフィールドは、オブジェクト固有でなくクラス固有であることから、クラスフィールドとも呼ばれます。また、同じように静的なメソッドをクラスメソッドと呼びます。これに対し、静的でないフィールドは、オブジェクト(インスタンス)ごとに独立していることから、インスタンスフィールドと呼びます。同じように静的でないメソッドをインスタンスメソッドと呼びます。

練習問題

1　以下の文章の内容が正しい場合には○、間違っている場合には×を付けましょう。

(１)インスタンスメソッドの中では、静的なメンバにアクセスすることはできない。
(２)クラスメソッドの中では、静的でないメンバにアクセスすることはできない。
(３)メソッドの中でstatic宣言された変数も、同じクラスのほかのオブジェクトと共有される。

2　キーワードthisについて簡潔に説明しましょう。