オブジェクト指向と
ゲームプログラミング

○　Java 基礎編 － 第３回　コンストラクタ　○

コンストラクタ

クラスには、コンストラクタ(constructor：構築子)という特別なメソッドがあります。コンストラクタは、クラスと同じ名前を持ち、値を返さない(返せない)メソッドで、オブジェクトの生成時に自動的に呼び出されます。通常のメソッドのように引数を受け取り、オーバーロードで複数定義することができます。

コンストラクタは、名前どおり、オブジェクトを生成するための処理を記述する関数です。おもに、フィールドの初期化といったオブジェクトを生成するための適切な初期化作業を行います。コンストラクタの処理がすべて終わったとき、オブジェクトが完全に生成されます。コンストラクタをきちんと書くことで、オブジェクトの状態を生成時から保証することができるのです。

●コンストラクタの特徴
- クラス名と同じ名前を持つ
- オブジェクトをインスタンス化(実体化)したときに呼び出される
- 引数を受け取ることができる
- 戻り値を返すことができない
- オーバーロードできる

コンストラクタでオブジェクトを初期化する以外に、初期化のためのメソッド(仮にinitメソッドとします)を用意する方法も考えられます。この方法では、Initメソッドが呼び出されない限り、オブジェクトの初期化が行われません。するとそのオブジェクトは初期化されるまで意味のないものとなり、他のメソッドが呼び出されたときの動作が予測できなくなってしまいます。このような理由から、オブジェクトの初期化はコンストラクタで行うようにします。

しかし、コンストラクタは戻り値を返せないなどの理由から、エラーが起きる可能性のある初期化作業(クラスライブラリの初期化)は、初期化のための関数を作成した方が無難です。このような場合でも、コンストラクタで最低限の初期化作業は必要です。

前回のCharacterクラスにコンストラクタを追加してみましょう。

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

    double      posX    = 0.0;                  // x座標
    double      posY    = 0.0;                  // y座標
    int         animeNo = 0;                    // アニメーション番号

    // コンストラクタ
    public Character(double x, double y) {
        posX = x;
        posY = y;
    }

    // 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
}
```

Characterメソッドがコンストラクタです。クラス名と同じ名前を持っており、voidなどの型の指定がありません。
このようにコンストラクタを書くと、以下のように、コンストラクタを呼び出してインスタンスを生成することができます。

```
Character  hinomoto = new Character(120.0, 0.0);
```

この例では、hinomotoオブジェクトの生成時にコンストラクタが呼び出され、フィールドposX, posYは引数で与えられた120.0と0.0、animeNoは0で初期化されます。

デフォルトコンストラクタ

前述のように引数を取るコンストラクタを１つでも定義した場合、

```
Character   hinomoto = new Character();
```

のように、引数を与えずにオブジェクトを生成することができなくなってしまいます。これは、「データを与えずにオブジェクトを生成させない」という場合には便利な機能ですが、「データを与えなくてもオブジェクトを生成したい」という場合は、不便に感じます。このような場合は、以下のように引数を取らないコンストラクタを定義します。

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

    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() {
        posX = 0.0;
        posY = 0.0;
    }

    // コンストラクタ
    public Character(double x, double y) {
        posX = x;
        posY = y;
    }

    // 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
}
```

引数を取らないコンストラクタをデフォルトコンストラクタと呼びます。デフォルトコンストラクタにより、引数を与えないオブジェクトの生成が可能になります。
また、コンストラクタを１つも定義しない場合は、コンパイラによって自動的にコンストラクタが作成されるようになっています。これもデフォルトコンストラクタと呼びますが、その内容は何もしないというものです(ただし、各フィールドは0, false, nullの適切な値で初期化されます)。

練習問題

1　以下のプログラムを入力し、コンストラクタがどのように実行されるかを確認しましょう。

```
// テストクラス
class Test {
    Test() {
        System.out.println("Testのコンストラクタが呼ばれました");
    }
}

public class TestProg {
    // メインメソッド
    public static void main(String[] args) {
        System.out.println("*** mainメソッド実行");
        Test   test = new Test();
        System.out.println("*** testが生成されたました");
    }
}
```

2　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)コンストラクタは、メソッドの一種である。
(2)コンストラクタの名前は、クラス名と同じでなければならない。
(3)コンストラクタは、戻り値を返せないので、戻り値のデータ型をvoidにしなければならない。
(4)引数を受け取るコンストラクタを定義することはできない。
(5)１つのクラスにつき１つのコンストラクタしか定義することができない。

3　次のように整数値の座標を表すPointクラスを作成しましょう。ただし、座標の範囲はｘが０～６４０、ｙが０～４８０となるようにしましょう。

フィールド
ｘ座標…posX
ｙ座標…posY

メソッド

```
void setX(int x);          // ｘ座標を設定する
void setY(int y);          // ｙ座標を設定する
int getX();                // ｘ座標を得る
int getY();                // ｙ座標を得る
```

コンストラクタ

```
Point();                   // ｘ，ｙ座標ともに０で初期化する
Point(int x, int y);       // 初期座標を指定する
```