プログラマーの脳みそ

< デザインパターンとしての例外... | @nagise 結婚記念パーティー>

2010-02-02

Java変態文法最速マスター 26

02:29 | 274 users

Java基礎文法最速マスター - 何かしらの言語による記述を解析する日記をリスペクト。

Javaの変態文法・技法一覧です。Javaの基礎をある程度知っている人はこれを読めばJavaの変態をマスターしてJavaを書くことができるようになっています。簡易リファレンスとしても利用できると思いますので、これは足りないと思うものがあれば教えてください。

1.基礎

エンクロージング型内部classの作成

外部classのインスタンスに紐付くインスタンスを生成します。外部クラスのインスタンス - 内部クラスのインスタンス間に、クラス - インスタンスのような関係を持たせることができます。

```
public class Outer {
        public class Inner {
        }
}
```

というようなクラスを作った場合、

```
Outer o = new Outer();
Inner i = o.new Inner();
```

となります。new演算子の前に外部クラスのインスタンスを記述するわけです。oの代わりにthisを渡す場合はthis.newという記述になるわけですが、this.は省略できるためOuterクラスのメソッドでInnerをnewする場合には省略して単にnew Inner()と書くことが出来ます。

```
public class Outer {
        public class Inner {
                public class Inner2 {
                        public class Inner3 {
                        }
                }
        }
        public void hoge() {
                Inner i = new Inner(); //this.newを省略してnewと書ける
                Inner.Inner2 i2 = i.new Inner2();
                Inner.Inner2.Inner3 i3 = i2.new Inner3();
        }
}
```

もちろん、多階層にすることもできます。連鎖的に呼び出すと面白いでしょう。

```
new Outer().new Inner().new Inner2().new Inner3();
```

カレンダー

<< 2010/02 >>

日	月	火	水	木	金	土
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28						

最近のコメント

2010-01-22 Nagise
2010-01-22 kensir0u
2010-01-20 aoisome
2010-01-20 Nagise
2010-01-20 aoisome

最近のトラックバック

2010-02-02 ネットサービス研究室 - プログラミング基礎文法最速マスターまとめ
2010-02-02 なんとなく日記 - (基礎|変態)文法最速マスターシリーズのまとめ
2010-02-02 h-ふじた(藤田浩)の日記 - 今日のリスペクト
2010-02-02 葉っぱ日記 - JavaScript変態文法最速マスター
2010-01-16 みねこあ - つれづれ

プロフィール

Nagise
脳の変わりにチップが埋まっているらしい

mainメソッドの作成なしでのプログラムの実行

main()メソッドが呼び出される前にclassの初期化が走るため、そのあたりにコードを書くことでmainメソッドがないクラスから処理を実行することが出来ます。

```
public class Starter {
	static {
		System.out.println("Hello!");
	}
}
```

上記サンプルはオーソドックスなstatic初期化ブロックを用いた方式です。ショートコーディング、あるいはコードゴルフでは定番の手法ですね。コンソールからの実行を行うと以下のようになります。

```
C:¥hoge>java Starter
Hello!
Exception in thread "main" java.lang.NoSuchMethodError: main
```

クラスの初期化の段階でHello!が出力され、その後にNoSuchMethodErrorでpublic static void main()がない、と怒られます。

staticな初期化が行われるフェースで呼び出せば実行は可能なのでstatic初期化ブロック以外にもstaticフィールドの初期化でメソッドを呼び出すなどの手法も使えます。

```
public class Starter {
	static int i = hoge();

	private static int hoge() {
		System.out.println("Hello!");
		return 0;
	}
}
```

System.out.printlnメソッド

System.outフィールドはSystem.setOut()メソッドで置き換えることができます。

java.io.PrintStreamクラスを継承したクラスを作成してごにょごにょしてやると、System.out.println()メソッドをトリガーにいろいろ行うことが出来ます。

標準出力への書き出しをした場合に語尾に「にょ」をつけるとかやりたい放題です。

コメント

//によるコメントは行末までが有効範囲です。//コメント中にユニコードエスケープした改行文字を置くことでコメントに偽装してプログラムを記述することが出来ます。

コメント中のユニコードエスケープというトリックを使ったネタとしてはJavaでポインタ演算子を使うエイプリルフールネタなどが面白いですね。

```
public class Test {
    public static void main(String[] args) {
        int i = 5;
        // #start unsafe# http://java.sun.com¥u000a¥u002f¥u002a
        int *ip = &i;
        // ポインタの内容に3を加算
        *ip += 3;
        // 演算結果をiに代入
        i = *ip;
        // #end unsafe# http://java.sun.com¥u002a¥u002fi+=¥u0033¥u003b
        System.out.println(i);
    }
}
```

演算子を用いてJavaでポインタを扱う

変数の宣言

変数の宣言にもアノテーションをつけることが出来ます。まず、@Targetに
LOCAL_VARIABLEを含むアノテーションを作成します。

```
import java.lang.annotation.Target;
import java.lang.annotation.ElementType;

@Target(ElementType.LOCAL_VARIABLE)
public @interface Hoge {

}
```

このアノテーションはローカル変数の宣言につけることができるので

```
@Hoge Hoge Hoge = Hoge();
```

のようなコードを書くことができるようになります。*1

フィールド宣言

transient キーワードやvolatile キーワードをつけることで、半可通をギョッとさせる程度の軽微ないたずらもありますが、ジェネリクスの型変数が汎用性が高く効果的だと思います。

```
public class Hoge<Hoge> {
        Hoge Hoge;
}
```

ここで定義されるHogeフィールドは型Hogeではなく、型変数Hogeであるところがポイントですね。

プログラムのコンパイル

javacコマンドには幾つかオプションがあります。javac - Java プログラミング言語コンパイラ

オプションに-g:noneを使用するとデバッグ情報が出力されないため、スタックトレースで行番号がunknownになります。大抵の新人はこの環境でデバッグをやらせると泣かせることができます。

オプション-processor でアノテーションプロセッサを動かすなどすると普通に有益なことができます。アノテーションプロセッサを悪用して無益なことをするアイデアを練るのもよいでしょう。

プログラムの実行

javaコマンドにも幾つかのオプションがあります。java - Java アプリケーション起動ツール

オプション-Dを利用してシステムプロパティをいじることができます。おすすめはpath.separatorの書き換えです。これで本来の設定ファイルとは違うパスのファイルを読み込ませるなどするとデバッグ作業は大いに混乱することでしょう。

オプション-javaagentを利用してエージェントを起動させ、いろいろと悪さをする、というのも楽しいでしょう。

2.数値

数値の表現

数値表現で注目すべきは8進数リテラルでしょう。

```
System.out.println(012);
```

```
10
```

特別な記号を用いたりしないため起きている事象を検索しにくく、ツボに入れば長考させることが出来ます。

四則演算

覚えておきたいのはDouble.NaNです。double型の値の1種で、日本語では非数と呼ばれます。0.0 / 0.0 といった演算を行うと発生します。NaNと他の値を演算すると結果はNaNとなります。処理フロー中で値にNaNが紛れ込むと、以後の値がすべてNaNになるので原因箇所を突き止めるのに難儀します。

また、NaNとNaNを==で比較した場合、falseとなるのも嵌りどころですね。これを利用してJAVA PUZZLERSでは以下のような出題がされています。

```
パズル29：ループ職人の花嫁(Bride of Looper)
このループが無限ループとなるiを宣言しなさい。
while(i != i) {
}
```

インクリメントとデクリメント

前置のインクリメントは特記事項はないですが、後置のインクリメントは初心者殺しですね。インクリメントが評価の後に行われる、というのが処理順序のゲシュタルト崩壊を誘います。

```
int i = 0;
i = i++ + ++i;
System.out.println(i);
```

3. 文字列

文字列とObjectの+演算子による連結などではObject#toString()メソッドが用いられます。Eclipseなどのデバッガで変数の値を確認する際にもこの値が用いられますので、不適切にオーバーライドしておくことでデバッグを混乱させることができます。

4. 配列

Java5以降では配列を使う機会もめっきり減りましたが、可変長引数で用いられたり*2と使う機会はまだ残っていますね。

```
public class Hoe<E> {
    private Class<E> type;

    public Hoe(E... e) {
        @SuppressWarnings("unchecked")
        Class<E> type = (Class<E>) e.getClass().getComponentType();
        this.type = type;
    }

    public Class<E> getType() {
        return type;
    }
}
```

こうすると、具体的な型パラメータを取得できる。
例えば

```
Hoe<Fuga> hoe = new Hoe<Fuga>();
```

という文があった場合、hoe.getType()はFuga.classになる。

型パラメータつきクラスの中から指定された型パラメータを知る方法 - Skirnirnismal

こちらのエントリで紹介されている技法は、コンストラクタに型変数の可変長引数をとることで、new Hoe<Fuga>()の呼び出しで大きさが0のFuga[]がコンストラクタ引数に渡され、そこから型変数の具体的な型であるFuga型を得る、という手法です。

なお、配列とは関係ありませんが、継承のさなかで型変数に何型がバインドされたかを得るにはジェネリックなクラスの階層を遡って適用された具象型を得るで紹介されているようなリフレクションを用いる手法が使えます。

5.ハッシュ

Object#hashCode()とObject#equals()の実装次第では標準APIのjava.util.HashMapなどを動作不良に追い込むことができます。

java.lang.Comparableあるいはjava.util.Comparatorの実装が不適切で等価ではないオブジェクトに対して0を返すような実装になっていた場合、java.util.TreeMapはputしたオブジェクトを消滅させます。

このような挙動はデバッグしにくく、この性質を利用したコードは解読困難となることでしょう。

6.制御文

if文

条件式に^演算子(XOR)を用いることができます。マッチするシチュエーションが少ないので利用のチャンスを逃さないようにしてください。

```
if (a == null ^ b == null) {
    throw new NullPointerException();
}
```

for文

制御文で知名度が低いものはラベルでしょう。for文やwhile文ではbreakやcontinueが利用できますが、多重ループの際にはどのループに対してbreak・continueするかを指示するためにラベルを用います。

```
yLoop: for (int y=0; y<height; y++) {
    xLoop: for (int x=0; x<width; x++) {
        if (map[x][y] == Type.BOMB) {
            break yLoop;
        }
    }
}
```

ちょうどURLの書式がラベル＋//コメントの形式になるので

```
http://example.jp/
for (int i=0; i<size; i++) {
}
```

といったような記述ができます。

ブロック

breakはループじゃなくとも利用可能で、単なるブロックにラベルをつけてbreakさせることができます。

```
hoge: {
        piyo: {
                break hoge;
        }
}
```

7.メソッド

strictfpキーワードをつけて首を傾げさせることができます。

同じように見えてもsynchronizedキーワードの場合はダサいのでやめましょう。*3

8.例外処理

finally節でのreturnやthrowはよく訓練されたJavaプログラマでないと挙動を予測できないことでしょう。

9.リフレクション

黒魔術の基礎のひとつです。java.lang.reflectパッケージを利用することで、動的なフィールド参照やメソッド呼び出しが行えます。setAccessibleメソッドを利用すればprivateメソッドですら外部から呼び出すことができてしまいます。

10.クラスローダー

クラスローダーを自作することで、実行時にクラスの内容を書き換えてしまうことができます。

defineClassメソッドでbyte配列の状態からClassを生成するのですが、オーバーライドしてbyte配列に細工をいれることが可能です。

注意事項

これらの技法は通常用いるべきではない変態的なテクニックを多く含みます。ご利用は自己責任でお願いします。

関連エントリ

最速マスターシリーズの本家

Perl基礎文法最速マスター - Perl入門～サンプルコードによるPerl入門～

リスペクト元

Java基礎文法最速マスター - 何かしらの言語による記述を解析する日記

最速マスターシリーズのまとめはこちらが解り易いと思います。

はてな的プログラミング言語人気ランキング - Life like a clown

*1：通常、Javaの命名規約では変数名は小文字始まりだが、ここでは敢えてHogeとすることでアノテーション名、型名、変数名、コンストラクタ名と、4つのHogeに別々の役割をさせている

*2：Java5以降ではジェネリクスを主体として使うので配列よりListを用いるのが通常だが、可変長引数は配列での受け渡しとなっているので甚だ使い勝手が悪い

*3：synchronizedはメソッドにキーワードとしてつけるとロックオブジェクトが露出するので防衛的プログラミングをするならメソッド内部でsynchronizedブロックを用いる。なのでsynchronizedキーワードを使うと「プ、こいつ並列処理わかってねえw」とか言われかねません

コメントを書く

トラックバック - http://d.hatena.ne.jp/Nagise/20100202/1265131791

葉っぱ日記 - JavaScript変態文法最速マスター

h-ふじた(藤田浩)の日記 - 今日のリスペクト

なんとなく日記 - (基礎|変態)文法最速マスターシリーズのまとめ

ネットサービス研究室 - プログラミング基礎文法最速マスターまとめ

リンク元

253 http://b.hatena.ne.jp/hotentry

211 http://reader.livedoor.com/reader/

131 http://b.hatena.ne.jp/hotentry/it

129 http://twitter.com/

120 http://www.google.co.jp/reader/view/

95 http://www.google.com/reader/view/

89 http://www.google.co.jp/reader/view/?hl=ja&tab=wy

72 http://d.hatena.ne.jp/hasegawayosuke/20100203/p1

64 http://b.hatena.ne.jp/entry/d.hatena.ne.jp/Nagise/20100202/1265131791

64 http://d.hatena.ne.jp/tt_clown/20100202/1265096776

おとなり日記

2010-02-01 iPhoneSDK他いろいろ 開発メモ 4/15 26%

2010-02-01 nazonoDiary 4/30 13%

2010-02-02 おもしろ日記パワー 4/40 10%

2010-02-02 プログラミング言語を作る日記 6/90 6%

2010-02-02 犬も歩けば棒も歩く 4/61 6%

2010-02-02 $koherent->diary 4/74 5%

< デザインパターンとしての例外... | @nagise 結婚記念パーティー>