JAPLA研究会資料 2013/3/23

Ｊユーザへのオブジェクト指向(OOP)の接し方

西川　利男

オブジェクト指向(OOP＝Object Oriented Progrmamming)とは、現在、少しでもコンピュータ・プログラミングをやろうとすれば、いやでも目に入ってくる「はやり」のことばである。

オブジェクト指向を真に理解しようとすると、その哲学からはじまりどうにも分からなくなってしまう。自分の身に即したほどほどのところにとどめるのがよい。要はどう接していくかが、Ｊユーザとしての私の知恵というものである。

世の中には２種類の人がいる。ものへの接し方を、出来る限り理解して、納得して使おうという人と、自分ではその中身は理解したくない、使えればよいという人である。その境目を決めるのがオブジェクト指向というものである。

オブジェクト指向とは、哲学でもなんでもない。カネでなんでもやろうという資本主義経済のなせるコマーシャルの文句にすぎない。さて、哲学はやめてＪでの具体的な例で始めよう。

たとえばある数３の平方根を求めたいとする。
その昔であれば、開平法というのを学校で習ったので、紙とエンピツでシコシコ計算する。現在では、１００円でも買える電卓で√のキーを押せばすぐ得られる。どういう計算で求められるのか、知ろうとする人もいまやごく少ない。

コンピュータをやる人でも開平法をプログラミングを組んでやる人はまずいないであろう。Ｊのユーザであれば、%: とするところだろう。しかし、オブジェクト指向の信奉者は、カネをはらってでも電卓ソフトを買いたがる。

１．Ｊのオブジェクト指向とは

これは、Ｊシステム上の画面の実行ではつぎのようになる。
ふつうのＪユーザではつぎのようにする。

```
   %: 3
1.73205
```

一方、オブジェクト指向の信奉者のＪユーザではつぎのようにする。

```
   load 'user¥classes¥ptestoop'
   ins =: conew 'ptestoop'
```

これで、OOPの市販者(クラス)から電卓入りのソフト(インスタンス)を手に入れた。

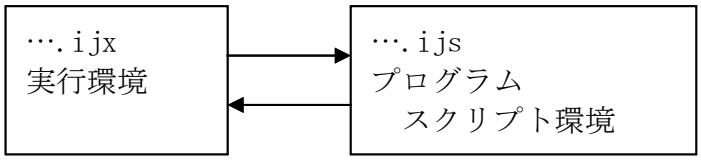
```
   sqrt__ins 3
1.73205
```

sqrtという「ことば」を使えば「√」キーと同様に平方根がすぐ求められる。しかし、平方根がどういう内容の計算かを知ることは出来ないし、その必要はない。

オブジェクト指向は、もっと大きな複雑な処理を行いたいときのものである。オブジェクト指向で行うか、直接プログラムするか、それぞれにメリットとデメリットがある。また、オブジェクト指向のプログラムを作るとなると大変だが、使うだけならそれほどのこともない。その接し方をもう少し見ていこう。

２．Ｊのオブジェクト指向のしくみ

Ｊのふつうの基本となる環境は次のようになっている。



さて、この実行環境で、簡単な名詞と動詞を定義してみる。

```
  DA =: 123
  double =: +:
```

これを実行してみる。

```
  DA
123
  double DA
246
```

そして、どういう名詞、動詞が定義されているかを見る。

```
   names ''
DA      double
```

Ｊシステムでは値(数値、文字、文字列)やプリミティブ(+:など)はあらかじめ備えられている。しかし、namesのような命令はどうやって実行されたのだろうか？
実はあらわには見えないが、Ｊでは「locale」という機能でつながっている。

この状態は、つぎのようにコマンドconlを使って見ることができる。

```
   conl 0    クラスオブジェクトのレベル0のネーム・リストを表示
+----+-+-+
|base|j|z|
+----+-+-+
```

起動した最初の実行環境はbaseという名のlocaleであり、表には見えないが標準ライブラリなどのシステムプログラムはj-locale, z-localeとしてつながっている。

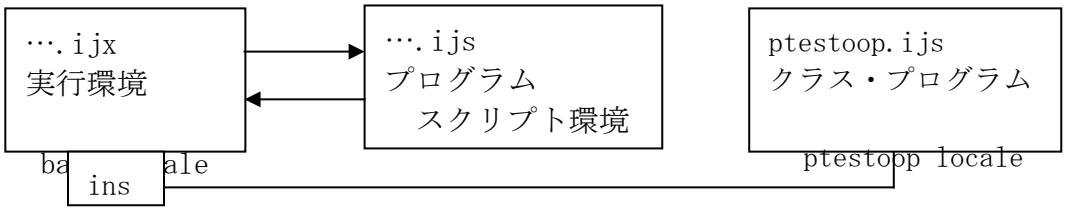
さて、以上のことを頭においた上で、オブジェクト指向のクラスプログラムをロードする。なお、ロードといっても通常のスクリプト・プログラムのロードとは異なり、localeに登録されるのである。この状態のlocaleをconlコマンドで見てみる。

```
   load 'user¥classes¥ptestoop'
   conl 0
+----+-+--------+-+
|base|j|ptestoop|z|
+----+-+--------+-+
```

次に、このクラス・プログラムを実行環境でも使えるようにする。それにはこのクラス・プログラムからconewによりインスタンスを作成する。

```
   ins =: conew 'ptestoop'
```

ここまでの状態を図示すると以下のようになる。



クラス・プログラムの中には次のものが入っている。

```
   names_ptestoop_ ''
COCLASSPATH DA          create      destroy     sqrt
```

たとえば、sqrt という動詞がある。これを実行するには、次のようにする。

先に定義した名詞 DA(=123)に対して

```
   sqrt__ins DA
11.0905
```

とすれば、DA の平方根が得られる。しかし、names で定義を見るとここにはない。代わりにインスタンス ins が追加されている。

```
   names ''
DA      double ins
```

なお、クラス ptestoop には、名詞 DA も定義されているが、これは実行環境の名詞 DA とは異なる。

```
   DA__ins
789
```

オブジェクト指向の用語では、クラスにある名詞をフィールド、動詞をメソッドともいう。また、このようにしてクラス内の名前の衝突、クラスの隠蔽、保護などもはかられる。

つまり、クラス・プログラムでのメソッド sqrt やフィールド DA など定義の詳細は知らなくても、使えることは出来るのである。これがオブジェクト指向による最大のメリットである。

オブジェクト指向のキーワードであるクラス、インスタンスとはこういう意味である。なお、インスタンス(例)のことをオブジェクトと呼ぶことも多いが、オブジェクトとはあまりに一般的なことばであり、私は使いたくない。

いままでで、J のオブジェクト指向の使い方のみを述べた。クラス・プログラムの作り方は参考までに、コーディングは付けたが、ここでは述べない。もっと大規模な実際例により、別項で説明したいと思う。

以前の筆者のレポートも参考にされたい。

[1] 西川利男「J のオブジェクト指向プログラミングー１、簡単な例でやってみる」
JAPLA シンポジウム資料 2005/12/10
[2] 西川利男「J のオブジェクト指向プログラミングー２、J のスプレッドシート
(Grid)と数独パズルへの適用」JAPLA シンポジウム資料 2005/12/10

参考

```
NB. test OOP
corequire 'user¥classes¥ptestoop'

ooprun =: 3 : 0
ins =: conew 'ptestoop'
'** OOP: loaded OK **'
)

SQRT =: 3 : 0
sqrt__ins y.
)
```

```
   load'f:¥j402¥user¥testoop.ijs'
   ooprun ''
** OOP: loaded OK **
   SQRT 3
1.73205
```

クラスプログラム

```
NB. Simple Test Object Oriented Method
coclass'ptestoop'

create=:3 : 0
0
)

sqrt=: 3 : '%: y.' NB. method
DA =: 789          NB. field

destroy=:codestroy
```