＜クラスとオブジェクトの利用方法＞

ここでは、オブジェクト指向プログラミングについて述べます。MachiKania では、クラスベースのオブジェクト指向プログラミングが可能です。次のような特徴があります。

１）オブジェクトは、フィールド(public もしくは private)とメソッド(public のみ)を持つ。
２）クラスの記述は別ファイルに於いて行ない、１ファイルに１クラスの記述とする。
３）複数のクラスを利用出来る。また、クラス中で別のクラスを利用出来る。
４）クラスの継承は、未実装。
５）フィールド名とメソッド名には、２文字以上６文字以内の英数字が指定出来る。

別ファイルとしてクラスを記述し、このファイルをクラス指定する事で、複数のファイルからなるプログラムを作成する事が可能です。クラス間の接続はフィールドとメソッドを介してのみ行なわれます。したがって、同じ名前の変数やラベルが複数のファイルに於いて存在していたとしても、それぞれファイルごとに制御が行なわれますので、名前の衝突が起きる事はありません。

＜クラスの記述の仕方＞

クラスを作成するには、"クラス名.BAS"の名でテキストファイルを準備します。ただし、クラス名は２文字以上６文字以内の英数字です。クラスファイルでは、FIELD, STATIC, METHOD 命令が必須です(どれか１種類だけでも使えます)。

FIELD [PUBLIC] x[,y[,z[, ...]]]
　　クラスファイル中で、パブリックフィールドを宣言する。"PUBLIC"は省略可。x,y,z 等はフィールド名を 6 文字以内の英数字で指定。ただし、フィールドを文字列型で使う場合は、「$」をフィールド名の後ろに付ける。
FIELD PRIVATE x[,y[,z[, ...]]]
　　クラスファイル中で、プライベートフィールドを宣言する。x,y,z 等はフィールド名を 6 文字以内の英数字で指定。ただし、フィールドを文字列型で使う場合は「$」をフィールド名の後ろに付け、配列として使う場合は「()」を付ける。
STATIC [PUBLIC] x[,y[,z[, ...]]]
　　クラスファイル中で、パブリックスなタティック変数を宣言する。"PUBLIC"は省略可。
STATIC PRIVATE x[,y[,z[, ...]]]
　　クラスファイル中で、プライベートなスタティック変数を宣言する。USEVAR と同じ。
METHOD x
　　クラスファイル中で、メソッドを宣言する。x は、メソッド名を 6 文字以内の英数字で指定。

クラスファイル中でフィールド値（パブリック・プライベートの両方）にアクセスする場合は、FIELD 指定した長い名前の変数に直接アクセス(読み込み及び書き込み)して下さい。「THIS」などの表記は必要ありません。

オブジェクトのフィールドではないスタティックな変数を扱いたい時は、STATIC もしくは USEVAR 命令を使って下さい。ここで使用した長い名前の変数は、他のファイルでの同一名の変数の値に影響しません。ただし、変数 A-Z の読み出し・変更は、他のファイルで使用する値に影響します。

METHOD 命令で宣言されたメソッドは、すべてパブリックです。プライベートなサブルーチン呼び出しには、LABEL 命令と GOSUB 命令/関数を用いて下さい。メソッド呼び出しの際の引数は、ARGS()関数で取り出す事が可能です。メソッド中で自身のオブジェクトへのポインターが必要な場合は、ARGS(-2)で取り出す事が出来ます。

コンストラクターを指定する場合は、「METHOD INIT」として下さい。このように INIT メソッドが指定された場合、オブジェクト作成時に、このメソッドが呼ばれます。INIT

メソッドに引数を与える事も可能です(引数は必須ではありません)。

クラス中でオブジェクトへのポインターが必要な場合は、ARGS(-2)で取り出す事が出来ます。

記述例(CLASS1.BAS の名前で保存)：

```
  FIELD PUBLIC TEST1,TEST2
  FIELD PRIVATE TEST3
  STATIC PUBLIC TEST7
  METHOD INIT
    IF ARGS(0)=2 THEN
      TEST1=ARGS(1)
      TEST2=ARGS(2)
    ENDIF
  RETURN
  METHOD TEST4
    TEST1=ARGS(1)
    TEST2=ARGS(2)
  RETURN
  METHOD TEST5
    TEST3=TEST1+TEST2+TEST7
  RETURN
  METHOD TEST6
  RETURN TEST3
  METHOD TEST8
  RETURN TEST7
```

＜オブジェクトの作成方法＞

オブジェクトを作成するためには、まず USECLASS 命令で使用するクラスを指定し、次に NEW 関数を用います。

USECLASS x[,y[,z[, ...]]]
　　クラスの利用を宣言する。x,y,z 等は、クラス名を 6 文字以内の英数字で指定。
NEW(x[,y[,z[, ...]]])
　　クラスオブジェクトを作成し、オブジェクトへのポインターを返す。x はクラス名を指定。y,z 等はコンストラクターがある際に利用される引数。

記述例１：

```
  USECLASS CLASS1
  A=NEW(CLASS1)
```

記述例２（コンストラクターに引数を指定する場合）：

```
  USECLASS CLASS1
  A=NEW(CLASS1,123,456)
```

＜オブジェクトの破棄方法＞

作成したオブジェクトは、DELETE 命令を用いて破棄する事も出来ます。ただし、デストラクターは実装していませんので、必要ならば、それ用のメソッドを作成して破棄する直前に呼び出して下さい。

オブジェクトのフィールドに文字列もしくは配列を用いている場合、オブジェクトを DELETE するだけではこれらの領域は破棄されません。その場合、まずメソッド内でこれら

のフィールドを破棄(DELETE 命令が使えます)してから、オブジェクトを破棄するようにして下さい。

記述例：
```
USECLASS CLASS1
A=NEW(CLASS1)
DELETE A
```

＜フィールドへのアクセス方法＞

パブリックフィールドへアクセスする場合は、オブジェクトを含む変数に続けて「.」とフィールド名を記述して下さい。フィールドが文字列の場合は「$」を、実数の場合は「#」を、それぞれ後ろに付けて下さい。

記述例：
```
USECLASS CLASS1
A=NEW(CLASS1)
A.TEST1=123
PRINT A.TEST2
```

＜メソッドへのアクセス方法＞

パブリックメソッドへアクセスする場合は、オブジェクトを含む変数に続けて「.」とメソッド名、続けて「( )」を記述して下さい。メソッドが戻り値として文字列を返す場合、メソッド名の後ろに「$」を付けて下さい（実数の場合は「#」）。メソッドは関数と同じ扱いですが、戻り値を利用しない場合は、CALL 命令により呼び出す事も出来ます。

CALL x
　　x で指定されたオブジェクトのメソッドを呼び出す。

記述例（579 が表示される）：
```
USECLASS CLASS1
A=NEW(CLASS1)
CALL A.TEST4(123,456)
CALL A.TEST5()
PRINT A.TEST6()
```

＜スタティック変数・スタティックメソッドの利用＞

クラス中で"STATIC PUBLIC"で指定された変数は、外部から値を参照したり、値を変更したりすることが出来ます。この場合、クラス名に続けて「::」と変数名を記述して下さい。

また、それぞれのメソッドは、スタティックメソッドとして利用する事も出来ます。この場合、クラス名に続けて「::」とメソッド名、続けて「( )」を記述して下さい。

記述例（123 が表示される）：
```
USECLASS CLASS1
CLASS1::TEST6=123
PRINT CLASS1::TEST7()
```

＜ライブラリーについて＞

クラスファイルは、カレントディレクトリーに「クラス名.BAS」ファイルが有れば、読み

出されます。もしカレントディレクトリーに無ければ、「\LIB\クラス名\クラス名.BAS」を探し、有ればそれが読み出されます。従って、「\LIB」ディレクトリーにクラス群を配置しておけば、いつでも使用可能なライブラリーとして使えます。

＜バージョン履歴＞
・KM-1207/KM1302 2019 年 3 月公開。