○「New MQL4（Build 600 以降）；標準クラスを使ってみる（初めの一歩)」

・アメンボです、

約半年のご無沙汰です、
当初は２か月間だけ、お休みする予定が MQL4 に劇的な変化（特に Build600）が起きて
小生も対応に追われてしまい、気が付けば６か月以上のお休みとなりました。
(投稿をお休みしていた間、MQL4 関係で色々やっていたのですが、その話はまた後ほど)

・「Build_600 以降」の件、

まずは「データフォルダ」と「コンパイラの変更」に戸惑ったことと思います。
この件は、また後ほど報告することにして、
本稿では、少し楽しい話を先にすることにしました。
Build_600 以降（以後、New_MQL4 と呼びます）では、MQL5 モドキと言っては変ですが、
様々な新機能が追加されました。(でも、参考資料の少なさには閉口しています)
その中でも「クラス」は小生のような「C++の初心者」には、一体「何に、どう使えば良い」
のか、さっぱり判らないと言うのが本音ではないでしょうか！？

・標準ライブラリ（クラス）は使えるかも！

気を取り直して、もっぱら MQL5 の資料を調べているのですが、「クラス」を理解するなら、
先ず標準で準備されている「標準ライブラリ（クラス)」を使ってみたらどうかと考えました。
案外、使い物になるかも知れないと期待半分です。

・どこから手を付ける？

それこそ「何に使ったら良いか」判らないが、Old_MQL4 では「実現が極めて困難」と思われる
「Canvas クラス」から解析を始めました。(クラスの威力を体験できるかも！と思い)

＜本稿で使用した MQL4 コード＞

※非常に単純なコードなので添付はしていません、本稿から「コピー、ペースト」で利用ください。
※本稿は「MT4；version 4.00 Build646」「MetaEditor；version 5.00 Buid934」にて確認済み。

目次：

１．標準クラスの調査進捗表

※なんせ、New_MQL4 の資料が余りに少ないので、MQL5 の資料を頼りに推測していく方法しかとれません。
調査するにしても、常に MQL5 との比較になります。
（でも、小生は MQL5 は実際に動かしたことは無いので、大変です）

＜全体像；クラスを利用するための「.mqh」ファイルが在る場所＞

MQL5；Section（ライブラリの種類）	MQL5；Location	New_MQL4;Location（ホルダ有無）
Base class	Include¥	未確認
Classes of data	Include¥Arrays¥	未確認
Classes for file operations	Include¥Files¥	未確認
Classes for string operations	Include¥Strings¥	未確認
Classes for graphic objects	Include¥Objects¥	ChartObjecyはあるけど？未確認
Class for creating custom graphics	Include¥Canvas¥	本稿；チョット使ってみた
Class for working with chart	Include¥Charts¥	未確認
Technical indicators	Include¥Indicators¥	未確認
Trade classes	Include¥Trade¥	無い！？
Trading strategy classes	Include¥Expert¥	無い！？
Classes for creation of control panels and dialogs	Include¥Controls¥	未確認

＜New_MT4 の「Include」フォルダ内＞

２．とにかく、「Canvas クラス」を使ってみた（その１）

・MQL5 の記事が参考になるというものの、暗闇の中を手探りで進むような心境で調査中です。

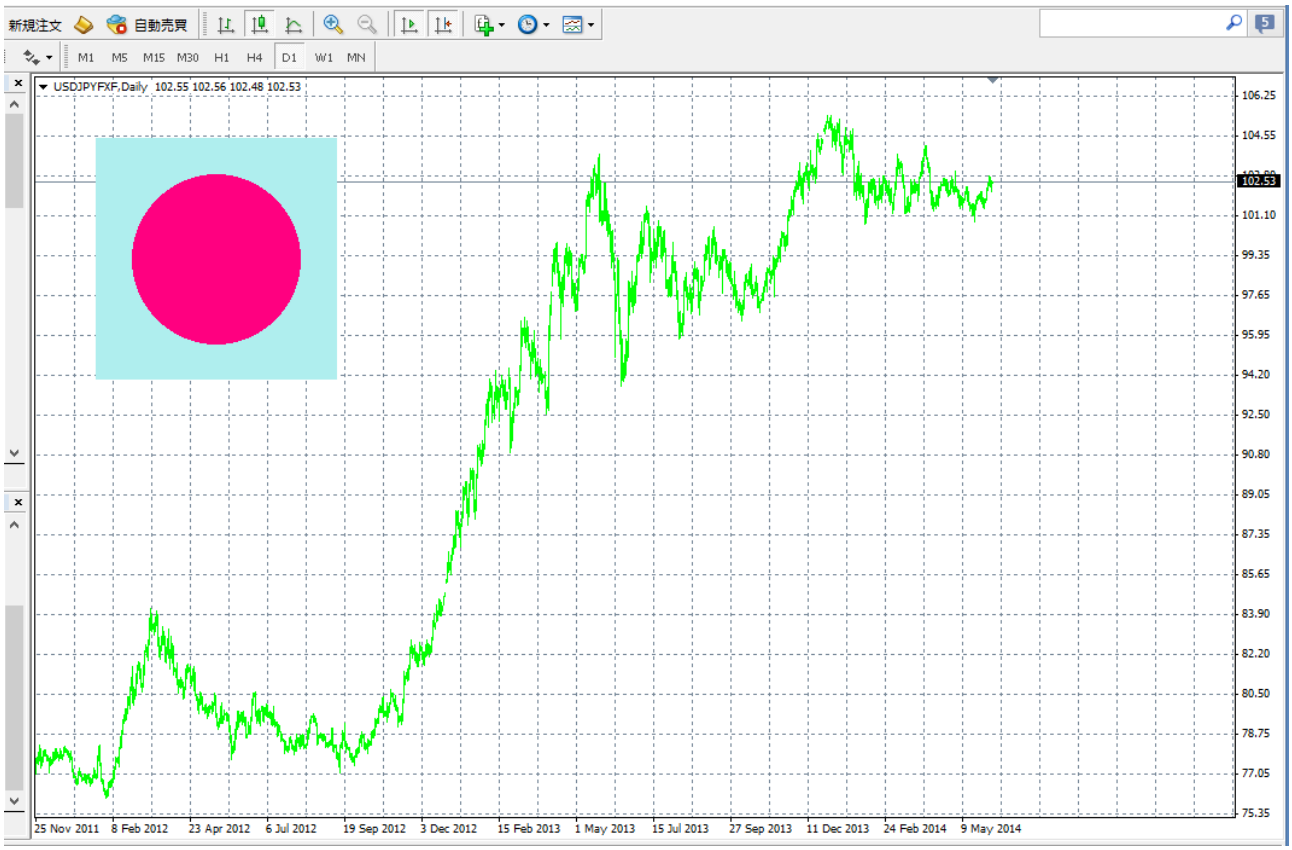
（１）コード例・・これは「スクリプト」です

※簡単なコードなので、「コピー＆ペースト」で試してください。

```
//+------------------------------------------------------------------+
//|                                              class_canvas_01.mq4 |
//|                                                          amenbo  |
//|                                                  泉の森の弁財天池 |
//+------------------------------------------------------------------+
#property copyright "amenbo"
#property link      "泉の森の弁財天池"
#property version   "1.00"
#property strict
//
#include <Canvas¥Canvas.mqh>
//
void OnStart()
  {
   //キャンバスを設定
   CCanvas can;
   //ビットマップ・ラベルを生成
   //・・チャート・ウィンドウ左上を原点（X=0,Y=0）とする・・
   can.CreateBitmapLabel("MySample",50,50,200,200,COLOR_FORMAT_ARGB_RAW);

   //「ビットマップ・ラベル（配列)」初期化
   can.Erase(XRGB(0xAF,0xEE,0xEE));
   can.Update();

   //ビットマップ・ラベル上に描画
   //・・ラベル左上が描画の原点（x=0,y=0）となる・・
   //can.FillRectangle(50,50,150,150,XRGB(255,0,128));//OK
   can.FillCircle(100,100,70,XRGB(255,0,128));//OK
   can.Update();

   Sleep(2000);

   //オブジェクトの消去とメモリ解放
   ObjectDelete(0,"MySample");
   can.Destroy();
   //
   PlaySound("alert2");

   }
```

（2）実行結果

※実行すると、こんな表示が現れます。(Canvas のトレードへの応用は全く未知ですが？)



※「２秒」経過すると、アラーム音がして図形は消えます。

（3）少し「コード」解説すると

※現状で小生が理解している内容を記述しておきます。

小生は「クラス」の初心者ですので、間違い等あればご容赦。(間違いは教えてください)

```
#include <Canvas¥Canvas.mqh>
```

標準ライブラリ（クラス）を参照します

```
CCanvas can;
```

「can」を Canvas クラスとして宣言

```
can.CreateBitmapLabel("MySample",50,50,200,200,COLOR_FORMAT_ARGB_RAW);
```

ビットマップ・ラベルを生成します。

名前；　"MySample"に設定しました。

座標系；　チャート・ウインドウの左上を原点（X=0,Y=0）とし、
右方向が X 軸、下方向が Y 軸となります。
①まず、原点から（X=50,Y=50）ピクセルのところを「基点」とします
②次に、「基点」からラベルの幅（ΔX=200,ΔY=200）を指定しています

カラー指定；
「COLOR_FORMAT_ARGB_RAW」は、ユーザー（プログラマー）が指定するという意味。

```
can.Erase(XRGB(0xAF,0xEE,0xEE));
can.Update();
```

ビットマップ・ラベルの初期化です。

カラー指定；　XRGB(0xAF,0xEE,0xEE)　で指定可能です。

「0x；ゼロ、エックス」をつけると 16 進、何もつけなければ 10 進での指定

例

配色	10 進表示	16 進表示
white	(255, 255, 255)	(FF, FF, FF)
black	(0, 0, 0)	(00, 00, 00)
blue	(0, 0, 255)	(00, 00, FF)
green	(0, 255, 0)	(00, FF, 00)
red	(255, 0, 0)	(FF, 00, 00)
PaleTurquoise	(175, 238, 238)	(AF, EE, EE)

※「can.Update();」は（描画）データのリフレッシュで、指定の度に必要なようです。

（念のため書いとくほうが良いでしょう）

```
can.FillCircle(100,100,70,XRGB(255,0,128));//OK
can.Update();
```

円を描いて、中を塗りつぶします。

座標系；　ややこしいのですが、「ビットマップ・ラベル」の左上が「基点」となります。

（チャート・ウインドウの左上ではありません）

①ラベル左上「基点」から（X1=100,Y1=100）のところを「円の中心」とします。

②「円の中心」で、半径 r=70 の円を描画し、中を塗りつぶします。

（単位はすべて「ピクセル」です）

カラー指定；

XRGB(255,0,128)で指定してみました。(配色名は、確認してみてください)

※最後に、(描画) データのリフレッシュを行っておきます。

```
ObjectDelete(0,"MySample");
can.Destroy();
```

最終の処理を行います。

①まず、(描画) オブジェクト類を消去します

②最後に、使用した「クラス」を解放しておきます。

※「解説」の必要は無いかもしれませんが、ついでに補足します。

```
Sleep(2000);　　(ミリ秒)間、お休みします。
PlaySound("alert2");　　スクリプトが終了したことを音で知らせます。
```

※やっぱり、補足しときます；

```
//can.FillRectangle(50,50,150,150,XRGB(255,0,128));
```

「//」を外して実行すると判りますが、ラベル上で描画する場合は、

「150」は幅ではなく、ラベルの左上「基点」からの位置になります。(小生は混乱しました)

以　上