DSHEng4 装置通信エンジンライブラリ(GEM+GEM300)

ソフトウェア・パッケージ

DSHEng4Class クラス・ライブラリ

クラス生成・消滅トレースと表示機能について

２０１１年2月

株式会社データマップ

文書番号　DSHEng4-09-30306-00

[取り扱い注意]

- この資料ならびにソフトウェアの一部または全部を無断で使用、複製することはできません。
- 本説明書に記述されている内容は予告なしで変更される可能性があります。
- Windows は米国 Microsoft Corporation の登録商標です。
- ユーザーが本ソフトウェアの使用によって生じた遺失履歴、（株）データマップの予見の有無を問わず発生した特別損害、付随的損害、間接損害およびその他の拡大損害に対して責任を負いません。

【改訂履歴】

番号	改訂日付	項 目	概 略
1.	201１.２月	初版	
2.			
3.			
4.			

目　次

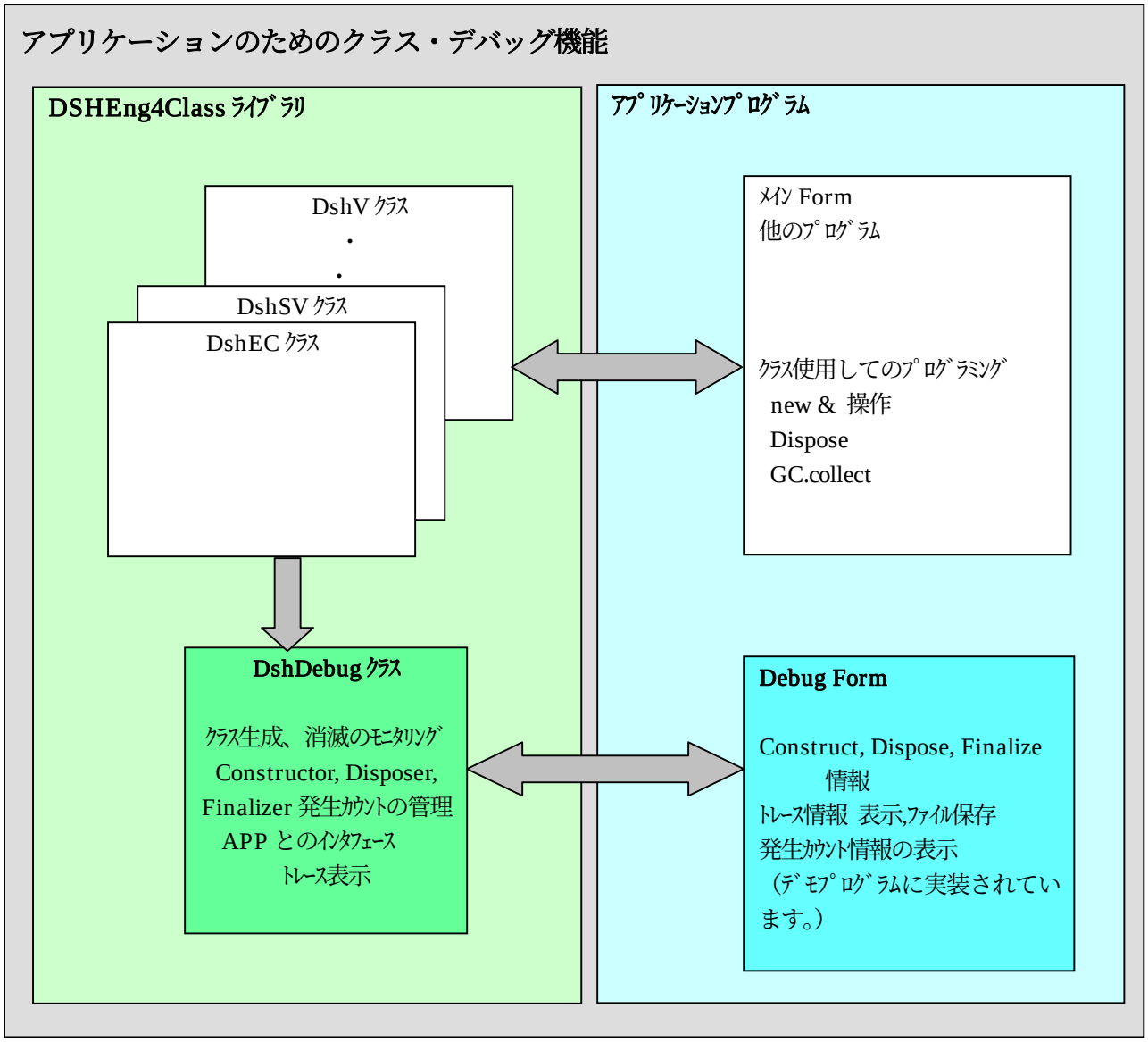
以　上

１．はじめに

DSHEng4Class クラス・ライブラリが有する各クラスについて、インスタンスの生成/消滅と、その回数をトレース管理する機能とユーザ・インタフェースを実装しました。

本機能とインタフェースを使用することによって、ユーザは、ランタイム時のクラスの生成・消滅のタイミングをトレースの内容をユーザのフォーム画面に表示することができ、また、実際にプログラムによって行われた生成・消滅の回数を知ることができます。

ユーザは、アプリケーション・プログラム開発作業の際、クラスのトレースをしながらデバッグを進めることができます。



２．クラスの Finalize と Dispose について

DSHEngClass のクラスの構成は次のようになっています。DshAlarm クラスの例で示します。

```
public class DshAlarm : IDisposable
{
    public DshAlarm( uint alid)      // Constructor
    {
        <生成時の処理>
    }
    ~DshAlarm()                      // Finalizer
    {
        <必要な処理>
        Dispose();
    }
    public void Dispose()            // Disposer
    {
        <必要な処理>                 // アンマネージドメモリの開放など
        GC.SuppressFinalize(this); // これで Finalizer が呼び出されない。
    }
```

ユーザが使用する DSHEng4 クラス・ライブラリのクラスには、**IDisposable** インターフェイスが追加されており、そして、Dispose()メソッドが実装されています。

Dispose()内では、**GC.SuppressFinalize(this);** によって、当該オブジェクトに、ファイナライザーを呼び出さないことをシステムに要求しています。

クラスのトレース対象として、生成(Constructor)、消滅(Finalizer)、使用済処理(Disposer)の３つがあります。

Finalizer と Disposer との関係をまとめると次のようになります。

・ユーザによって Dispose()が呼び出されれば，システムによって Finalizer()は呼びだされない。
（Dispose()内の **GC.SuppressFinalize()**による。）

・Finalizer(~Class 名)は、システムの GC(Garbage Collection()時に呼び出され、Dispose()を呼び出す。
（Finalizer は、システムが判断するタイミングまたは、ユーザによる GC.Collect()メソッドによって呼び出される。

[注] ユーザがクラスライブラリのクラスを使用した際、そのクラスのオブジェクト（インスタンス）について使用済みになった時点で、ユーザプログラムが明示的に Dispose()を実行しておけば、Dispose()が、内部のオブジェクトで使用した資源を解放してくれます。そして、Finalizer は呼び出されることはありません。
このことは、使用済処理を行うために、システムからの GC を待つ必要がないことになります。

３．クラス・カウント機能とインタフェース

DSHEng4Class ライブラリが提供するインタフェースとして２つあります。

・クラスメソッドに対する直接インタフェース
・DshDebug クラスに対するトレース制御と各クラス情報アクセスインタフェース_

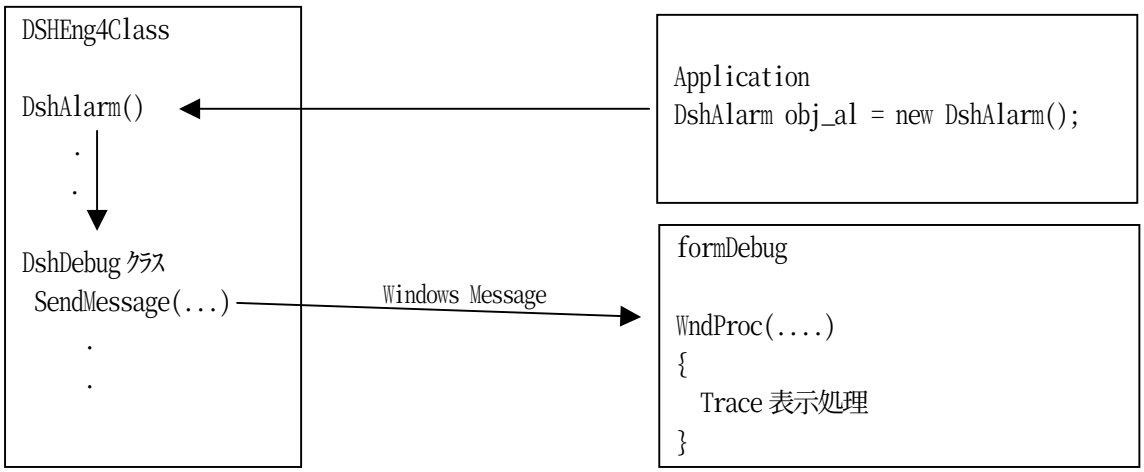
３．１　DshDebug クラスに対する初期セットアップとトレース選択設定

トレース表示機能を有効にするための初期設定メソッドです。

DshDebug クラス・メソッド	機能
void init_DshDebug()	DshDebug ｸﾗｽの初期化、ｸﾗｽ名などの初期設定など。 **ｸﾗｽ・ﾗｲﾌﾞﾗﾘが自動的に実行します。**
void set_trace_form_handle(IntPtr handle)	Application のﾄﾚｰｽ情報を表示する form の ﾊﾝﾄﾞﾙを設定します。(Windows Msg 送信用) handle がﾌｫｰﾑﾊﾝﾄﾞﾙです。
void set_trace_WM(int wm)	ﾄﾚｰｽ情報を表示する form へ送る Windows Message ID を設定します。 wm がﾒｯｾｰｼﾞ ID です。
void enable_trace_log(bool flag)	Trace 表示を有効にするかどうかを指定します。 (随時、設定変更できます。) flag = ture で有効、 =false で無効にします。

トレース表示情報は、DshDebug クラスから、アプリケーションの Debug 用 Form のハンドルに対し、Windows API 関数の SendMessage()関数を使って、Debug Form に準備された WndProc()に送信します。

WndProc()の詳細については、EngCsDemo プログラムの formDebug.cs を参照してください。



３．２　クラス直接のインタフェース

クラスの生成、資源の解放として、Constructor と Dispose()メソッドをクラスに対し直接呼出します。

クラスの Cosntruct, Dispose, Finalize カウンタの考え方は次のようになります。
それぞれの事象のカウントを nc, dc, fc とすると次のようになります。

（1）Constructor を呼び出されると、nc がカウントアップされます。

（2）Dispose メソッドが呼び出されると, dc がカウントアップされます。

（3）Finalizer がシステムから呼び出されると Dispose メソッドを呼び出し、fc がカウントアップされます。

Finalizer の前にユーザによって Dispose メソッドが呼び出されたら、Finalizer は呼び出されません。このことは、ユーザから Dispose が呼び出されなかったときにだけ、システムから呼び出されて fc がカウントアップされることになります。

このことは、nc = dc ならば、Construct されたクラスがすべて Dispose されたことになります。

クラスの Cosntruct, Dispose, Finalize カウントの取得、リセットは、３．３の DshDebug クラスを使って実行することができます。

３．３　DshDebug クラスのインタフェース

クラスのConstructor, Finalizer, Disposerのトレースカウント(呼び出された回数)の取得ならびにクラスの表記名を取得するためにDshDebugクラスを使用できます。

なお、何れのメソッドも、クラスインデクスを指定してメソッドを呼び出します。
インデクスは、DshDebug クラスに定数として定義されており、クラス名の頭に **DBG_** が付きます。

DshDebug クラス・メソッド	機能
bool set_trace_flag(int x, bool flag)	指定されたｸﾗｽのﾄﾚｰｽ表示有効/無効の設定をします。 x はｸﾗｽを指定する index です。 flag=true で有効、=false で無効の設定になります。 戻り値は、正常に設定できたら true を返します。
bool get_trace_flag(int x)	指定されたｸﾗｽのﾄﾚｰｽ表示有効/無効の設定状態を取得します。 x はｸﾗｽを指定する index です。 戻り値は flag=true で有効、=false で無効を意味します。
void reset_trace_count(int x)	ｘで指定されたｸﾗｽのﾄﾚｰｽ・ｶｳﾝﾀをﾘｾｯﾄ(=0)にします。 Construct, Dispose, Finalize の３つのｶｳﾝﾀ
int get_trace_count(int x, ref int nc, ref int fc, ref int dc)	ｘで指定されたｸﾗｽのﾄﾚｰｽ・ｶｳﾝﾄを取得します。 nc=生成回数、fc=Dispose 回数, dc=消滅回数に値を返します。戻り値は nc の値です。
string get_class_name(int x)	x で指定されたｸﾗｽ名を取得します。 戻り値がｸﾗｽ名です。

クラスを指定するインデクスは、DshDebug クラス内に定数として定義されています。

インデクスの定数名は、DBG_DshAlarm, DBG_DshCE などのように DBG_の後にクラス名を付けます。

たとえば、DshAlarm のクラスカウントをリセットする場合、次のように行います。

```
DshDebug.reset_trace_count( DshDebug.DBG_Alarm );
```

それから、DSHEng4Class に含まれるクラスの最大数は、DBG_DshCount 定数になっています。

４．クラスのトレース表示機能

クラスの Construct, Dispose, Finalize のタイミングで、そのクラスのトレースをアプリケーションプログラムによって表示させることができます。

４．１　トレース表示の条件とトレースの制御

クラスのトレース表示条件としては、３．で説明しましたが、アプリケーション側では、以下の設定が必要です。

たとえば、formDebug の Windows Form プログラムからプログラミングした場合は次のようになります。

（1）初期設定

```
private  constant int WM_trace_log = 3001;            // Windows MessageID

DshDebug. init_DshDebug();

DshDebug.set_trace_form_handle( This.Handle );

DshDebug.set_trace_WM( WM_trace_log );
```

<以上はトレース表示するために必ず実行する必要があります。

（2）Application 開始後(DSHEng4Class 使用中)

以下、必要に応じて実行することができます。以下、例を示します。

```
DshDebug.enable_trace_log( true );          // 全ｸﾗｽを Trace したくない場合は、false

DshDebug.set_trace_flag ( DshDebug.DBG_DshAlarm, true );

bool state = DshDebug.get_trace_flag ( DshDebug.DBG_DshAlarm );

int nc = 0;                 // new (construct) count
int dc = 0;                 // dispose count
inf fc = 0;                 // finalize count
DshDebug.get_trace_count( DshDebug.DBG_DshAlarm,  ref nc, ref fc, ref  dc );

string class_name = DshDebug.get_class_name( DshDebug.DBG_DshAlarm );
```

実際のトレース・カウント（Construct,Dispose, Finalize）のカウントは、全体の有効/無効に拘らず、その事象が発生すればカウントされます。

トレース表示が無効になっている場合は、アプリケーションの Form への Windows メッセージ送信は行われません。

４．２　アプリケーションのフォームに送信されるトレース・表示情報

DshDebug クラスの set_trace_form_handle()で設定された form に set_trace_WM()で指定されたメッセージと表示情報を送信します。

表示情報は、次のような string 型の文字列で、クラスのイベントを識別文字と発生回数が含まれます。

Construct 時	“+++++ <ｸﾗｽ名>　new　= nnn”	（ｸﾗｽﾗｲﾌﾞﾗﾘ起動後の nnn はカウント数）
Dispose 時	“>>>>> <ｸﾗｽ名>　dis　= nnn”	
Finalize 時	“----- <ｸﾗｽ名>　fin　= nnn”	

例を示します。（Dispose は、GC によって行われた場合の例です。）

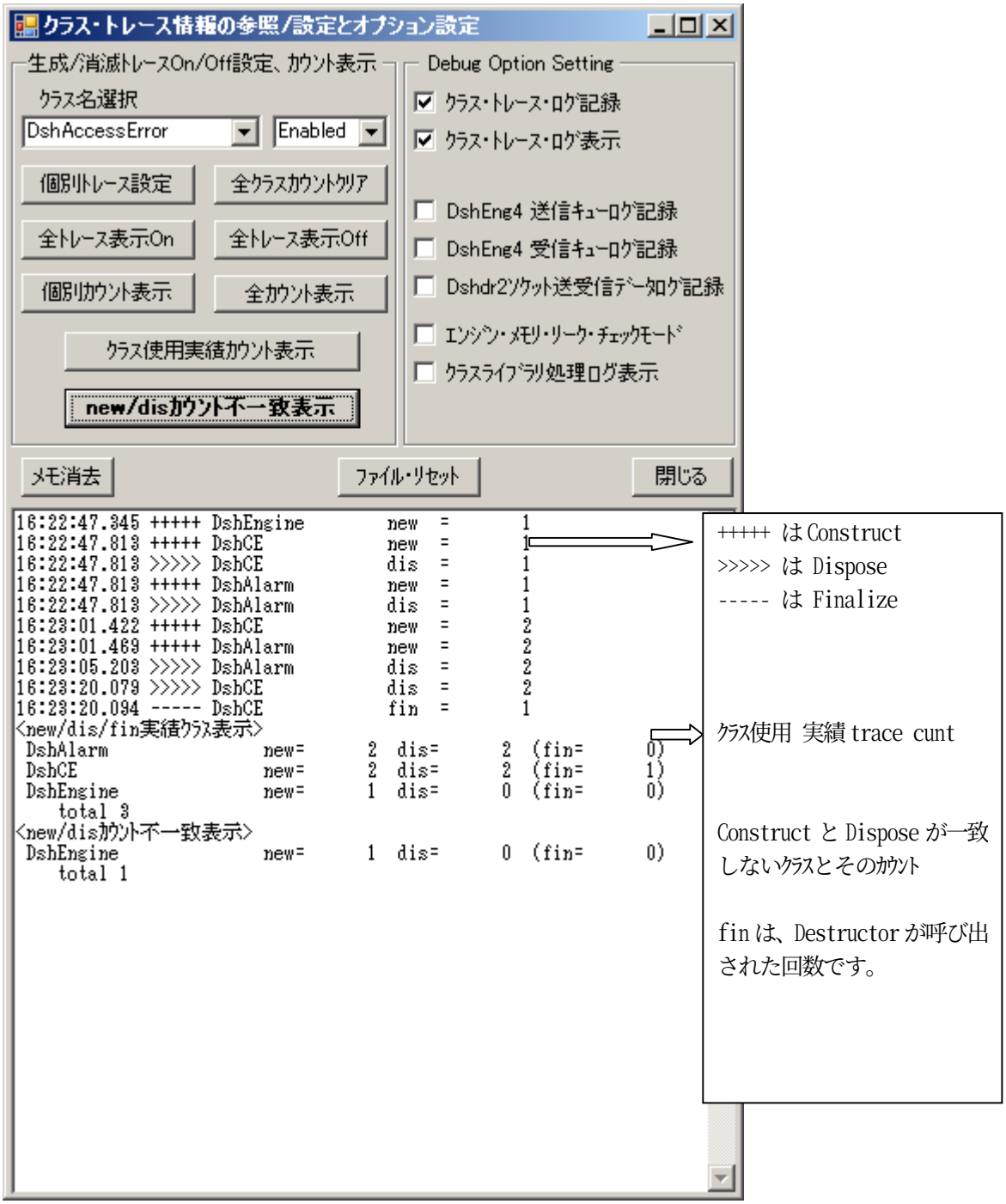
```
+++++ DshRecipe new  = 1
>>>>> DshRecipe dis  = 1
----- DshRecipe fin  = 1
```

先に説明しましたように、トレース表示はユーザプログラムによる次の処理によって実現できます。

（1）表示用フォームの form Handle と、form へ送信する Windows Message ID の設定
（2）フォームへのトレース表示出力の有効設定
（3）クラス単位でのトレース表示の有効設定

５．デモプログラムのデバッグ画面のサンプル

EngCsDemo.exe デモプログラムに含まれるクラストレース情報表示画面のサンプルを下に示します。
（formDebug.cs または doemDebug.vb を参照してください。）



formDebug.cs のフォームは、formMain によって生成され、常時、クラスライブラリからの Windows メッセージを受信できるように常時 Active の状態になっています。

表示するしないは、formMain のボタンのクリックで Visible=true にされて表示され、formDebug 画面の閉じるボタンで、Visible=false にして表示を消すようにしています。