

EN1-Object Counting and Display

Author: Conrad Kopala

Access faults and memory leaks are two of the most challenging problems facing the developer. Object counting and display can help find the cause of these problems.

Object counting uses a global variable `gObjectCount` to keep a running count of the current number of `TObject` derived objects. Object display prints information about each `TObject` derived object as it is created and destroyed. When an object is created, the object display appears as follows:

```
>>TObject constructor
Construct TSomeMacAppObject@ 0x2D6ACA4 Id=74 Size=108 ObjCnt = 73
#Construct TMyObject@ 0x2D6ACA4 Id=74 Size=108 ObjCnt = 73
```

Note that the address in memory, the class Id, and the class size are given along with the object count. If that object were now destroyed, the display would appear as follows:

```
#Destruct TMyObject@ 0x2D6ACA4 Id=74 Size=108 ObjCnt = 73
Destruct TSomeMacAppObject@ 0x2D6ACA4 Id=74 Size=108 ObjCnt = 73
>>TObject destructor
```

The object count hasn't changed because the `TObject` destructor hasn't been invoked yet. It is invoked last when an object is destroyed while the `TObject` constructor is invoked first when an object is created.

The current value of `gObjectCount` can be printed anytime by executing the global routine `PrintObjectCount()`. Typically, it is useful to print the object count at the beginning of `TYourApp::DoSetupMenus()`. Doing so establishes a baseline value for `gObjectCount`. Suppose a document is opened and then closed. It is expected that the value of `gObjectCount` will be the same as it was before the document was opened and closed. If it isn't, there is a memory leak.

A menu command, `Print Object Count` could also be used to print the value of `gObjectCount`.

In the case of access faults, the object display printout will be most useful. That and a good understanding of your application. The object display printout should be examined. In particular, look for objects that have been freed or objects that have never been created when they should have (that's where you need an understanding of the application.) With some effort, you should be able to identify the culprit.

Object counting can be enabled and disabled by setting the value of the flag `gObjectCountingEnabled` to `TRUE` or `FALSE` as required.

Object display printing is managed by three flags: `gPrintBaseClassInfo`, `gPrintMacAppClassInfo`, and `gPrintAppClassInfo`. The first, `gPrintBaseClassInfo` is used to control printing at the `TObject` constructor and destructor level. the second, `gPrintMacAppClassInfo` is used to control printing of `MacApp` object information while the third, `gPrintAppClassInfo` is used to control printing of your application object information.

Twist Down Lists provides menu commands allowing you to change the values of `gObjectCountingEnabled`, `gPrintBaseClassInfo`, `gPrintMacAppClassInfo`, and `gPrintAppClassInfo`. Ideally, these four menu commands should be integrated with `MacApp` and put in its `Debug` menu. However, I wanted to keep the extent of the changes to `MacApp` to a minimum and also wanted to confine them to as few places as possible.

All the changes required to implement these capabilities are confined to `UObject.h` and `UObject.cp`. The directions for implementing these capabilities follow.

I. UObject.h additions

Add the following global variables:

```
#if qDebug
extern long gObjectCount;
extern Boolean gObjectCountingEnabled;
extern Boolean gPrintBaseClassInfo;
```

```

extern Boolean gPrintMacAppClassInfo;
extern Boolean gPrintAppClassInfo;
//Three gratuitous non-essential flags. Don't include if you don't want them.
extern Boolean gAppUserFlag1;
extern Boolean gAppUserFlag2;
extern Boolean gAppUserFlag3;
#endif

```

Add the following global functions:

```

#ifdef qDebug
extern void IncrementObjectCount();
extern void DecrementObjectCount();
extern void PrintObjectCount();
//DumpClassInfo is a gratuitous non-essential function. Don't include it if you don't want //it.
extern void DumpClassInfo();
extern void DisplayMemoryInfo();
extern void InitUDebugGlobals();
#endif

```

To TObject add the following four methods:

```

#ifdef qDebug
void TObject::PrintConstructorClassInfo();
void TObject::PrintDestructorClassInfo();
void TObject::PrintAppConstructorClassInfo();
void TObject::PrintAppDestructorClassInfo();
#endif

```

II. UObject.cp additions

If you are going to compile for 68K add the following:

```

#ifdef qDebug || qTheDebugger && !qPowerPC
    #ifndef __UCLASSDESCITERATOR__
    #include "UClassDescIterator.h"
    #endif

    #ifndef __MEMORYHEAP__
    #include "MemoryHeap.h"
    #endif

    #ifndef __BESTFITHEAP__
    #include "BestFitHeap.h"
    #endif

    #ifndef __OBJECTHEAP__
    #include "ObjectHeap.h"
    #endif
#endif

```

Add the following global variables:

```

#ifdef qDebug
long gObjectCount = 0;
Boolean gObjectCountingEnabled = FALSE;
Boolean gPrintBaseClassInfo = FALSE;
Boolean gPrintMacAppClassInfo = FALSE;
Boolean gPrintAppClassInfo = FALSE;
//Three gratuitous non-essential flags. Don't include if you don't want them.
Boolean gAppUserFlag1 = FALSE;

```

```
Boolean gAppUserFlag2 = FALSE;
Boolean gAppUserFlag3 = FALSE;
#endif
```

Add the following four methods to TObject:

```
#if qDebug
//-----
// TObject::PrintConstructorClassInfo
//-----
#pragma segment MADestructorRes
void TObject::PrintConstructorClassInfo()
{
    ClassName theClassName;
    ClassID theClassId;
    size_t theClassSize;

    if (gPrintMacAppClassInfo)
    {
        this -> GetClassName(theClassName);
        theClassId = this -> GetClassID();
        theClassSize = this -> GetClassSize();
        fprintf(stderr, "Construct %s", (char*)theClassName);
        fprintf(stderr, "@ %p", this);
        fprintf(stderr, " Id=%d", theClassId);
        fprintf(stderr, " Size=%d", theClassSize);

        if (gObjectCountingEnabled && gPrintMacAppClassInfo)
            PrintObjectCount();

        fprintf(stderr, "\n");
    }
}
#endif
```

```
#if qDebug
//-----
// TObject::PrintDestructorClassInfo
//-----
#pragma segment MADestructorRes
void TObject::PrintDestructorClassInfo()
{
    ClassName theClassName;
    ClassID theClassId;
    size_t theClassSize;

    if (gPrintMacAppClassInfo)
    {
        this -> GetClassName(theClassName);
        theClassId = this -> GetClassID();
        theClassSize = this -> GetClassSize();
        fprintf(stderr, "Destruct %s", (char*)theClassName);
        fprintf(stderr, "@ %p", this);
        fprintf(stderr, " Id=%d", theClassId);
        fprintf(stderr, " Size=%d", theClassSize);

        if (gObjectCountingEnabled)
            PrintObjectCount();

        fprintf(stderr, "\n");
    }
}
```

```

}
#endif

#ifdef qDebug
//-----
// TObject::PrintAppConstructorClassInfo
//-----
#pragma segment MADestructorRes
void TObject::PrintAppConstructorClassInfo()
{
    ClassName theClassName;
    ClassID theClassId;
    size_t theClassSize;

    if (gPrintAppClassInfo)
    {
        this -> GetClassName(theClassName);
        theClassId = this -> GetClassID();
        theClassSize = this -> GetClassSize();
        fprintf(stderr, "#Construct %s", (char*)theClassName);
        fprintf(stderr, "@ %p", this);
        fprintf(stderr, " Id=%d", theClassId);
        fprintf(stderr, " Size=%d", theClassSize);

        if (gObjectCountingEnabled)
            PrintObjectCount();

        fprintf(stderr, "\n");
    }
}

#endif

#ifdef qDebug
//-----
// TObject::PrintAppDestructorClassInfo
//-----
#pragma segment MADestructorRes
void TObject::PrintAppDestructorClassInfo()
{
    ClassName theClassName;
    ClassID theClassId;
    size_t theClassSize;

    if (gPrintAppClassInfo)
    {
        this -> GetClassName(theClassName);
        theClassId = this -> GetClassID();
        theClassSize = this -> GetClassSize();
        fprintf(stderr, "#Destruct %s", (char*)theClassName);
        fprintf(stderr, "@ %p", this);
        fprintf(stderr, " Id=%d", theClassId);
        fprintf(stderr, " Size=%d", theClassSize);

        if (gObjectCountingEnabled)
            PrintObjectCount();

        fprintf(stderr, "\n");
    }
}

}

```

```
#endif
```

Add the following four global functions:

```
#if qDebug
//-----
// IncrementObjectCount
//-----
#pragma segment MAGlobalRes
void IncrementObjectCount()
{
    if (gObjectCountingEnabled)
        gObjectCount++;
}
//-----
// DecrementObjectCount
//-----
#pragma segment MAGlobalRes
void DecrementObjectCount()
{
    if (gObjectCountingEnabled)
        gObjectCount--;
}
//-----
// PrintObjectCount
//-----
#pragma segment MAGlobalRes
void PrintObjectCount()
{
    if (gObjectCountingEnabled)
        fprintf(stderr, " ObjCnt = %d", gObjectCount);
}
//-----
// DumpClassInfo:
//-----
//The following function is a gratuitous non-essential function. Do not include if you //don't want it.

#pragma segment MAGlobalRes
void DumpClassInfo()
{
    MA_ClassReference aClassDesc;
    CClassIterator iter;

    for (aClassDesc = iter.FirstClassDesc(); iter.More(); aClassDesc = iter.NextClassDesc())
    {
        ClassName theClassName = aClassDesc -> GetClassName();
        ClassID theClassId = aClassDesc -> GetClassID();
        size_t theClassSize = aClassDesc -> GetClassSize();

        fprintf(stderr, " %s", (char*)theClassName);
        fprintf(stderr, " classId = %d", theClassId);
        fprintf(stderr, " classSize = %d", theClassSize);
        fprintf(stderr, "\n");
    }
}
//-----
// DisplayMemoryInfo:
//-----
#pragma segment MAGlobalRes
void DisplayMemoryInfo()
{
```

```

long availableMemory = 0;
long heapSize = 0;
Size tempSize = 0;
Size szTemporaryReserve;
Size szMemReserve;

GetReserveSize(szTemporaryReserve,szMemReserve);

availableMemory = gObjectHeap -> BytesFree();
heapSize = gObjectHeap -> HeapSize();
unsigned long freeMemory = FreeMem();

fprintf(stderr, " heapSize = %d", heapSize);
fprintf(stderr, " availableMemory = %d", availableMemory);
fprintf(stderr, " used = %d", heapSize - availableMemory);
fprintf(stderr, "\n");
fprintf(stderr, " freeMemory = %d", freeMemory);
fprintf(stderr, "\n");
fprintf(stderr, " szTemporaryReserve = %d", szTemporaryReserve);
fprintf(stderr, " szMemReserve = %d", szMemReserve);
fprintf(stderr, "\n");
fprintf(stderr, "\n");
}
//-----
// InitUDebugGlobals
//-----
#pragma segment MAGlobalRes
void InitUDebugGlobals()
{
gObjectCount = 0;
gObjectCountingEnabled = FALSE;
gPrintBaseClassInfo = FALSE;
gPrintMacAppClassInfo = FALSE;
gPrintAppClassInfo = FALSE;

//If you've chosen not to use the following three gratuitous non-essential flags remove //them.
gAppUserFlag1 = FALSE;
gAppUserFlag2 = FALSE;
gAppUserFlag3 = FALSE;
}
#endif

```

III. UObject.cp changes

Add the following snippet of code to TObject::TObject()

```

#ifdef qDebug
if (gPrintBaseClassInfo)
    fprintf(stderr, " >>TObject constructor\n");
IncrementObjectCount();
#endif

```

Add the following snippet of code to TObject::~TObject()

```

#ifdef qDebug
if (gPrintBaseClassInfo)
    fprintf(stderr, " >>TObject destructor\n");
DecrementObjectCount();
#endif

```

Add the following snippet of code to TObject* TObject::ShallowClone()

```

#if qDebug
IncrementObjectCount();
if (gPrintAppClassInfo)
{
    ClassName theClassName;
    ClassID theClassId;
    size_t theClassSize;
    this -> GetClassName(theClassName);
    theClassId = this -> GetClassID();
    theClassSize = this -> GetClassSize();
    fprintf(stderr, ">>Cloned %s", (char*)theClassName);
    fprintf(stderr, "@ %p", result);
    fprintf(stderr, " Id=%d", theClassId);
    fprintf(stderr, " Size=%d", theClassSize);

    if (gObjectCountingEnabled)
        PrintObjectCount();

    fprintf(stderr, "\n");
}
#endif

```

IV. MacApp Object changes

To each MacApp object, add the following to its constructor:

```

#if qDebug
this -> PrintConstructorClassInfo();
#endif

```

and the following to its destructor:

```

#if qDebug
this -> PrintDestructorClassInfo();
#endif

```

The above statements should always be added to the constructors and destructors in the same relative place: for example either at the end or the beginning. If some are added at the beginning and some are added at the end or somewhere in between, confusion will arise when you attempt to interpret the printout. I prefer the end.

In general, it is not a good idea to make the above changes to TEvent and TToolboxEvent as the mere act of printing this information generates TToolboxEvents and you just keep switching between your application and the debugger log window.

Note that the following MacApp objects do not have constructors:

| | |
|-----------|-------------|
| UMenuMgr | TMenuTable |
| UMenuMgr | TMenuListID |
| UMenuMgr | TCmdTable |
| UVUAssist | TGridItem |
| UVUAssist | TVUAssist |

and the following do not have destructors:

| | |
|-------------------|---------------------|
| UCommand | TInvalCursorCommand |
| UMailableDocument | TMailableDocument |

V. MYourApplication.cp

In MYourApplication.cp, right after

```
main() {
```

add the following

```
#if qDebug
    InitUDebugGlobals();
//InitUDebugGlobals() sets the following global.
    gObjectCount = 0;
//InitUDebugGlobals() sets the following globals to FALSE. You may want to change
//their initial values as I do with gObjectCountingEnabled.
    gObjectCountingEnabled = TRUE;
    gPrintBaseClassInfo = FALSE;
    gPrintMacAppClassInfo = FALSE;
    gPrintAppClassInfo = FALSE;
    gAppUserFlag1 = FALSE;
    gAppUserFlag2 = FALSE;
    gAppUserFlag3 = FALSE;
#endif
```

VI. Your Application Objects

In the constructor of each of your application objects add the following:

```
#if qDebug
this -> PrintAppConstructorClassInfo();
#endif
```

and in the destructor add the following:

```
#if qDebug
this -> PrintAppDestructorClassInfo();
#endif
```

The above statements should always be added to the constructors and destructors in the same relative place: for example either at the end or the beginning. If some are added at the beginning and some are added at the end or somewhere in between, confusion will arise when you attempt to interpret the printout. I prefer the end.

If you want to provide menu control for the global variables that have been added continue with the following steps:

VII. YourApplication.r

Define the following command numbers. Use whatever values you want.

```
#if qDebug
#define cCountObjects          1901 //menu command number
#define cResetObjectCount      1902 //menu command number
#define cPrintBaseClassInfo    1903 //menu command number
#define cPrintMacAppClassInfo  1904 //menu command number
#define cPrintAppClassInfo     1905 //menu command number
#define cAppUserFlag1          1906 //menu command number
#define cAppUserFlag2          1907 //menu command number
#define cAppUserFlag3          1908 //menu command number
#define cAllowUnlimitedDocs     1909 //menu command number
#endif
```

Define the following menu using whatever value you choose.

```
#if qDebug
#define mTestMenu              7
#endif
```

```
#if qDebug
```



```

resource 'CMNU' (mTestMenu,
#if qNames
"Test",
#endif
nonpurgeable) {
    mTestMenu,
    textMenuProc,
    EnablingManagedByMacApp,
    enabled,
    "Test",
    {
/* [ 1] */ "AppUserFlag1",          nolcon,noKey,noMark,plain,cAppUserFlag1;
/* [ 2] */ "AppUserFlag2",          nolcon,noKey,noMark,plain,cAppUserFlag2;
/* [ 3] */ "AppUserFlag3",          nolcon,noKey,noMark,plain,cAppUserFlag3;
/* [ 4] */ "-", nolcon, noKey,noMark,plain,nocommand;
/* [ 5] */ "Count Objects",          nolcon,noKey,noMark,plain,cCountObjects;
/* [ 6] */ "Reset Object Count",     nolcon,noKey,noMark,plain,cResetObjectCount;
/* [ 7] */ "-", nolcon,noKey,noMark,plain,nocommand;
/* [ 8] */ "Print Base Class Info",  nolcon,noKey,noMark,plain,cPrintBaseClassInfo;
/* [ 9] */ "Print MacApp Class Info", nolcon,noKey,noMark,plain,cPrintMacAppClassInfo;
/* [10] */ "Print App Class Info",   nolcon,noKey,noMark,plain,cPrintAppClassInfo;
/* [11] */ "-", nolcon,noKey,noMark,plain,nocommand;
/* [12] */ "Allow Unlimited Docs",  nolcon,noKey,noMark,plain,cAllowUnlimitedDocs    };
    };
#endif

```

VIII. UYourApplication.h

In your UYourApplication.h define the following command numbers:

```

#if qDebug
const CommandNumber cCountObjects          = 1901;    //menu command number
const CommandNumber cResetObjectCount      = 1902;    //menu command number
const CommandNumber cPrintBaseClassInfo    = 1903;    //menu command number
const CommandNumber cPrintMacAppClassInfo  = 1904;    //menu command number
const CommandNumber cPrintAppClassInfo     = 1905;    //menu command number
const CommandNumber cAppUserFlag1         = 1906;    //menu command number
const CommandNumber cAppUserFlag2         = 1907;    //menu command number
const CommandNumber cAppUserFlag3         = 1908;    //menu command number
const CommandNumber cAllowUnlimitedDocs    = 1909;    //menu command number
#endif

```

IX. UYourApplication.cp

In TYourApplication::DoSetupMenus() add the following code:

```

#if qDebug
    if (gObjectCountingEnabled == TRUE)
        EnableCheck(cCountObjects, TRUE, TRUE);
    else
        EnableCheck(cCountObjects, TRUE, FALSE);

    Enable(cResetObjectCount, TRUE);

    if (gPrintBaseClassInfo == TRUE)
        EnableCheck(cPrintBaseClassInfo, TRUE, TRUE);
    else
        EnableCheck(cPrintBaseClassInfo, TRUE, FALSE);

    if (gPrintMacAppClassInfo == TRUE)
        EnableCheck(cPrintMacAppClassInfo, TRUE, TRUE);
    else

```

```

        EnableCheck(cPrintMacAppClassInfo, TRUE, FALSE);

    if (gPrintAppClassInfo == TRUE)
        EnableCheck(cPrintAppClassInfo, TRUE, TRUE);
    else
        EnableCheck(cPrintAppClassInfo, TRUE, FALSE);

    if (gAppUserFlag1 == TRUE)
        EnableCheck(cAppUserFlag1, TRUE, TRUE);
    else
        EnableCheck(cAppUserFlag1, TRUE, FALSE);

    if (gAppUserFlag2 == TRUE)
        EnableCheck(cAppUserFlag2, TRUE, TRUE);
    else
        EnableCheck(cAppUserFlag2, TRUE, FALSE);

    if (gAppUserFlag3 == TRUE)
        EnableCheck(cAppUserFlag3, TRUE, TRUE);
    else
        EnableCheck(cAppUserFlag3, TRUE, FALSE);

    if (gAllowUnlimitedDocs == TRUE)
        EnableCheck(cAllowUnlimitedDocs, TRUE, TRUE);
    else
        EnableCheck(cAllowUnlimitedDocs, TRUE, FALSE);
#endif

```

In TYourApplication::DoMenuCommand(CommandNumber aCommandNumber) add the following code to the switch statement:

```

#ifdef qDebug
    case cCountObjects:
    {
        if (!gObjectCountingEnabled)
            gObjectCount = 0; //If we're turning object counting on, reset it first
//There is no need to reset it if we're turning it off.

        gObjectCountingEnabled = !gObjectCountingEnabled;
    }
    break;

    case cResetObjectCount:
        gObjectCount = 0;
        break;

    case cPrintBaseClassInfo:
        gPrintBaseClassInfo = !gPrintBaseClassInfo;
        break;

    case cPrintMacAppClassInfo:
        gPrintMacAppClassInfo = !gPrintMacAppClassInfo;
        break;

    case cPrintAppClassInfo:
        gPrintAppClassInfo = !gPrintAppClassInfo;
        break;

    case cAppUserFlag1:
        gAppUserFlag1 = !gAppUserFlag1;
        break;

```

```
case cAppUserFlag2:
    gAppUserFlag2 = !gAppUserFlag2;
    break;

case cAppUserFlag3:
    gAppUserFlag3 = !gAppUserFlag3;
    break;

case cAllowUnlimitedDocs:
    gAllowUnlimitedDocs = !gAllowUnlimitedDocs;
    break;
#endif
```

7/1/96