Summary

COLLABORATORS			
	TITLE : Summary		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		September 19, 2022	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

1 Summary **1**

1.1 A++ Class Summary 1

1.2 a++application 1

1.3 appobject 2

Chapter 1

Summary

1.1 A++ Class Summary

```
                      A++ Class summary  -  A view through the library
   ------------------------------------------------------
            ($Date: 1994/05/09 21:24:46 $)


          A++ APPObject class
                A++ Doubly Linked List classes
    A++ LivingObject class
    A++ SignalResponder class
    A++ Timer class
    A++ TimedMsgPort class

    A++ AttrList class
    A++ IntuiObject class
    A++ GraphicObject class
    A++ Gadget classes
    A++ Window classes
    A++ DrawArea classes


          How A++ applications are build up
               ------------------------------------------------------------
    -> Back to the root menu..
```

1.2 a++application

Every A++ application that uses IntuiObjects, that is every application with an Intuition® graphical user interface, has the same pattern:

```
void APPmain() // NOTE: int main(argc,argv) is defined in 'APPmain.cxx'
{
   // create objects that represent the application (any objects you like)
   // (of course you can create and destroy objects anywhere in your
   //  application.)
   ...
```

```
    // enter the main event loop
    while (running)   // control the loop yourself
    {
       SignalResponder::WaitSignal();
       // each received signal is processed within WaitSignal().
       // WaitSignal() returnes after each signal.
       // Usually you will not do anything here in this loop.
       // Action takes place in event callbacks on objects.
       // "Think object-oriented!"
    }
}
```

The C standard main function is defined in "APPmain.cxx" which is within the A++ link library ("aplusplus.lib").

By simply adding a SignalResponder for the CTRL-C_BREAK signal, that breaks the loop, the program can be terminated at any point by sending a BREAK signal to it, either with pressing CTRL-C in the standard input window (CLI) or by use of the 'break <process-no.>' command. Look at the demo programs.

1.3 appobject

APPObject class

This is a virtual base class for all A++ classes. It is used to detect constructor failures within the inheritance path of an object. Therefore each constructor of a derived class should check for the proper initialisation of its base classes and in case of failure should not allocate any resources but let the user see the occured error.

The class user who creates an object should test it's validity with 'Ok()'. The macro 'APPOK()' checks a given object pointer being NULL before applying 'Ok()' to it.

On object deletion, the APPObject base class sets the object status to APPOBJECT_INVALID, thus causing obj->Ok() returning FALSE.

```
myProcedure( )
{
   DerivedObject *obj = new DerivedObject( );

   if (obj)          // check for memory allocation failure
      if (obj->Ok())    // check validity of the created object
   // both if stmnts can be replaced by 'if (APPOK(obj)'
   {
         ....  // work on the object
   }
   else
   {
      _dout(cerr << "Initialisation error occured: " << obj->IError() << endl;)
      delete obj;    // free the memory allocated for the object data
   }
   }
```

```
}
```

Class implementors are recommended to check the base class validity within their constructor:

```
class MyClass : private InheritedClass, virtual public APPObject
{
   public:
      MyClass( )
      {
         if (Ok())   // has an error already occured ?
         {
            // at this point the object has the class ID of the last class in ↩
               the
            // inheritance list.
            if (initialise( )==FALSE)    // class initialisation
            {
               #define MYCLASS_SOMETHING_FAILED (MY_CLASS+1)
               _ierror(MYCLASS_SOMETHING_FAILED);
               // set the error variable to a value and
               // print the error string "MYCLASS_SOMETHING_FAILED" to stderr
            }
            else setID(MY_CLASS);
         }
         else  // initialise only for SAFE destruction, no resource allocation
         {
         }
      }
}
```

And add your class to the A++ classes list with its personal class ID:

ADD IN FILE: APlusPlus/environment/Classes.h

```
#define MY_CLASS xxx
```

(Will probably be changed to a runtime type info system without class ID’s..)