

# #14: CPlusTESample

Written by: Andrew Shebanow

| | | |
|---|---|---|
| **Versions:** | 2.0 | May 1990 |
| **Components:** | `AppLib.h` | May 1, 1990 |
| | `AppLib.r` | May 1, 1990 |
| | `Application.cp` | May 1, 1990 |
| | `Application.h` | May 1, 1990 |
| | `Document.cp` | May 1, 1990 |
| | `Document.h` | May 1, 1990 |
| | `Exceptions.cp` | May 1, 1990 |
| | `Exceptions.h` | May 1, 1990 |
| | `List.h` | May 1, 1990 |
| | `List.cp` | May 1, 1990 |
| | `TECommon.h` | May 1, 1990 |
| | `TEDocument.cp` | May 1, 1990 |
| | `TEDocument.h` | May 1, 1990 |
| | `TESample.cp` | May 1, 1990 |
| | `TESample.make` | May 1, 1990 |
| | `TESample.h` | May 1, 1990 |
| | `TESample.r` | May 1, 1990 |
| | `TESampleGlue.a` | May 1, 1990 |
| **From MacApp 2.0:** | `UMAFailure.a` | May 1, 1990 |
| | `UMAFailure.h` | May 1, 1990 |
| | `UMAFailure.inc1.p` | May 1, 1990 |
| | `UMAFailure.p` | May 1, 1990 |

---

This program requires C, C++, and Asm from MPW 3.1 or greater to build. It also uses the Failure handling routines, provided here, from MacApp 2.0.

For use with MPW 3.2, follow the instructions in the “TESample.make” file.

---

This version of TESample has been substantially reworked in C++ to show how a “typical” object-oriented program could be written. To this end, what was once a single source code file has been restructured into a set of classes which demonstrate the advantages of object-oriented programming.

There are four main classes in this program. Each of these classes has a definition (.h) file and an implementation (.cp) file.

The `TApplication` class does all of the basic event handling and initialization necessary for Macintosh toolbox applications. It maintains a list of `TDocument` objects and passes events to the correct `TDocument` class when appropriate.

The TDocument class does all of the basic document handling work. TDocument objects are objects that are associated with a window. Methods are provided to deal with update, activate, mouse-click, key down, and other events. Some additional classes which implement a linked list of TDocument objects are provided.

The TApplication and TDocument classes together define a basic framework for Macintosh applications, without having any specific knowledge about the type of data being displayed by the application's documents. They are a (very) crude implementation of the MacApp application model, without the sophisticated view hierarchy.

The TESample class is a subclass of TApplication. It overrides several TApplication methods, including those for handling menu commands and cursor adjustment, and it does some necessary initialization. Note that we only need to override a few basic routines—the rest of the work is done in a generic way by TApplication (isn't OOP great?).

The TEDocument class is a subclass of TDocument. This class contains most of the special-purpose code for text editing. In addition to overriding most of the TDocument methods, it defines a number of additional methods which are used by the TESample class to get information on the document state.

The UMAFailure files are a hacked up version of MacApp 2.0's UFailure unit. The Exceptions files are a set of C++ macros that make recovering from errors easier.

**Segmentation Strategy**

This program has only one segment, since it isn't really big enough to make multiple segments worthwhile. We do unload the data initialization segment at start time, which frees up some memory.

**_SetPort Strategy**

Toolbox routines do not change the current port. In spite of this, in this program we use a strategy of calling _SetPort whenever we want to draw or make calls which depend upon the current port. This makes us less vulnerable to bugs in other software which might alter the current port (such as the bug (feature?) in many desk accessories which change the port on _OpenDeskAcc). Hopefully, this also makes the routines from this program more self-contained, since they don't depend on the current port setting.

**Clipboard Strategy**

This program does not maintain a private scrap. Whenever a Cut, Copy, or Paste occurs, we import or export from the public scrap to TextEdit's scrap immediately, using the TEToScrap and TEFromScrap routines. If we did use a private scrap, the import or export would be in the activate or deactivate event and suspend or resume event routines, respectively.

**Version 2.0 Changes**

Version 2.0 adds the ability to open and save documents. It also allows the user to open multiple documents. For error checking, it uses the Failure unit now instead of the previous if-then-else method.