This is an atempt to document the steps I go through in creating an applicaiton using Dinkclass. Before I get into the coding steps I need to have a good idea as to what I intend on creating. To start with I will simply present my idea for an application and start fleshing out the details untill I'm ready to start putting the application together.

The application I had in mind is an application which benchmarks the through put of sending messages by using High Level and Apple events. I want to bench mark these tests against both Think C's meassage dispatcher and direct function calls. I'm also interested in the performance of these messaging scheams as the size of the measages is varied. Initialy I was thinking of implementing a simple distributed processing application, but I couldn't think of an interesting enough application and chose to just analize the preformance of the communication chanels available to me.

At this point I have started the specifide the Operational goals, the key architectural requierments, and the planned model of the applicaton's implementation. Also to date I have made 2 written attempts at documenting these 3 components of the documentation for more than 4 different design and application ideas. I had also thought a lot about the low level requierments of sending the data via the IAC chanels.

It become clear that one of the things I was doing was developing a plan of how I was going to develop the application. So I backed off and mapped out how I was going to solve the problem of developing this applicatoin.

For this application my plan is to first implement the profiling and user interface using the dirrect function calls then after refining the API and user interfaces, implement the other communications chanels and benchmarks one at a time. (not an earth shaking plan, but it dosn't need to be anything exciting to work)

After choosing a developement plan I am ready to start development.

Steps in creating a new DinkClass application:

1) Created a new folder for the project "DC HLEvents"

2) Started ThinkC and created a new project in that folder "DHLEvents.π"

3) Added the base classes and MacTraps the project

4) Using the Set Project Type... menu option I set almost all of the Size flags I could.

5) Under the Options menu option I set the compiler options to use the debugger and generate MacsBugs symbols, I also turn off all the optimizations options, set the language extensions to be ThinkC + Objects and require prototypes, and I set the compiler settings to make class method NOT virtual by default.

6) Did a link to see what I missed in the project. In this case it was ANSI-Small, PrGlue, oopsDebug and a main function.

7) Created the class declaration for the DWindow subclass by copying the Init and Draw declarations from the DScribbleWind.h file. This window subclass will not have much to do, it will only display results of a measurement performed by the DApplication subclass.

7.1) I put the guard on this .h file to prevent cyclic includes of the file at compile time. File guards are the #ifndef _DHLWINDOW__ #define _DHLWINDOW__ at the top of the file and the #endif __DHLWINDOW__ at the bottom of the class declaration file.

7.1) I put in comments as to what the methods are going to do so that I don't forget what I intended to do by the time I get to the implementation of these methods.

8) Create the class implementation file, include the .h file and stub out the Init and Draw methods using code copied from the DScribbleWind.c file. I had to cut out most of the lines in these functions because my new program doesn't use the same variables....

8.1) I put in comments as to my intentions for the Draw method, because I have not yet even declared the DHLApp subclass of DApplication..

9) I check the syntax of the DHLWindow class implementation and correct the sill errors made (I left virtual key words in the .c file and typeOed the #include in the .h file.)

10) I created the declaration of the DHLDoc subclass of DDocument. I copied the declarations from the DScribbleDoc's class declaration, added comments on my intentions for the methods I declared in DHLDoc and put in the file guard.

11) I created the class definition file for the DHLDoc class methods by copying, pasting and editing the methods I needed from DScribbleDoc.c.

11.1) I added comments indicating my intentions, and checked the syntax. I found that I forgot to close a comment block at the top of the file, eg /* */ pair had a typeOH in the "*/".

11.2) I added the .c file to my projected and did a link, just because I felt like it. The link failed because I have no main function yet. I don't want a main until I have the DHLApp subclass defined anyway.

12) Created the .h class declaration file for the DHLApp subclass of DApplciation. I put in the file guard, and copied the initial methods needed form DScribbleApp.h.

12.1) Knowing that my application's design needs to do some tests in response to user menu selections I added 3 more methods to support the user interface to these tests (which I have NOT completely thought out yet!! I just know that I'm going to place them in the DApplication subclass, and an idea of what the tests require to run, hence the SetUpMenus is needed.)

12.2) I added my comments to make my intentions clear to myself (because I'll forget what I was thinking later). I have internally NOT declared the DHLApp methods or related member variables needed to do the test this application is being created for in the first place!!! This is because I'm taking an approach of doing the things I know are needed first and filling in the details later type of approach on this stage of the development. This approach can blow up in your face if you take it too far and forget what stubs you need to fill in and you make major modifications to the organization of the application. So if you try this same approach be careful not to paint yourself into a corner, by not letting the amount of code you write in this manner get too large before you fill in the other details.

13) I created the class definition (.c) file for the DHLApplication, and filled in an initial version of the methods my copying the methods form DScribbleApp.c, for some of the methods and DScribbleDoc.c for the menu related methods (those methods are inherited from the DEventHandler subclass to BOTH DApplication and DDocument).

13.1) I edited the copied methods to indicate my intentions and comment out the parts which I wanted to use later but didn't have much to do with any application other than DScribble. I Checked the syntax and added it to the project.

14) Created the main function and added the core apple event handlers, by copying the code for the ScribleMainProgram.c file, and editing out all the Scribble references in the file and putting in the needed DHLApp and Doc references in their place. I modified the HandleODOC apple event handler to NOT to anything by coping the code in the HandlePDOC handler proc. I did a link check and ran the program. It crashed because I had not set up the program resources yet.

15) Created the resources for this HLEvents application by coping the resource file "DC Scribble.π.rsrc" from the DC Scribble folder into my DC HLEvents folder, renaming the resource file and editing it to be specific to the HLEvents application. I used resEdit to edit the about box and the menus to be what is needed for my HLEvent application.

16) Now I have the initial stub application running and I can now start going in and implementing the application specific details to make the program work correctly. It is noted that as the program stands at this stage it dosen't do much and the menu enablings are not getting set correctly (because I only stubbed that part of the code out). It is worth noting some of the things which need to be done at this time.

- need to define the constants for the application specific menus and menu items.
- need to flesh out the menu related functions in DHLApp
- need to decide where I'm going to place the data generated by the various Tests fired in response to the users menu selections.
- need to have the DHLWindow::Draw method use this info to render the data for the user.

    - need to implement the tests and all supporting code for there execution.

  It is also worth noting that I expect to make a number of changes to the classes as I implement the test functions.

To get to this stage for the time I started this coding session has been 3.5 hours. I am saving a copy of this project to use as a template ("DC Template App") , so that in the future it will be much easier to get to this stage on the next application.!!!!!!