**CECS 544**

**Semester Project**

**Due April 20, 5pm**

What to turn in:

1. a jar file of your project’s executable

2. a zip file of your project’s source code

1. What is the project:

Our project is a “lite” version of Microsoft’s ancient Windows File Manager (FM) program. This program has been re-released and is available for Windows 10 from the Microsoft store. It is not required that you download it, but it may help you understand some of our project’s functionality if you use it. Here is shot of the project:

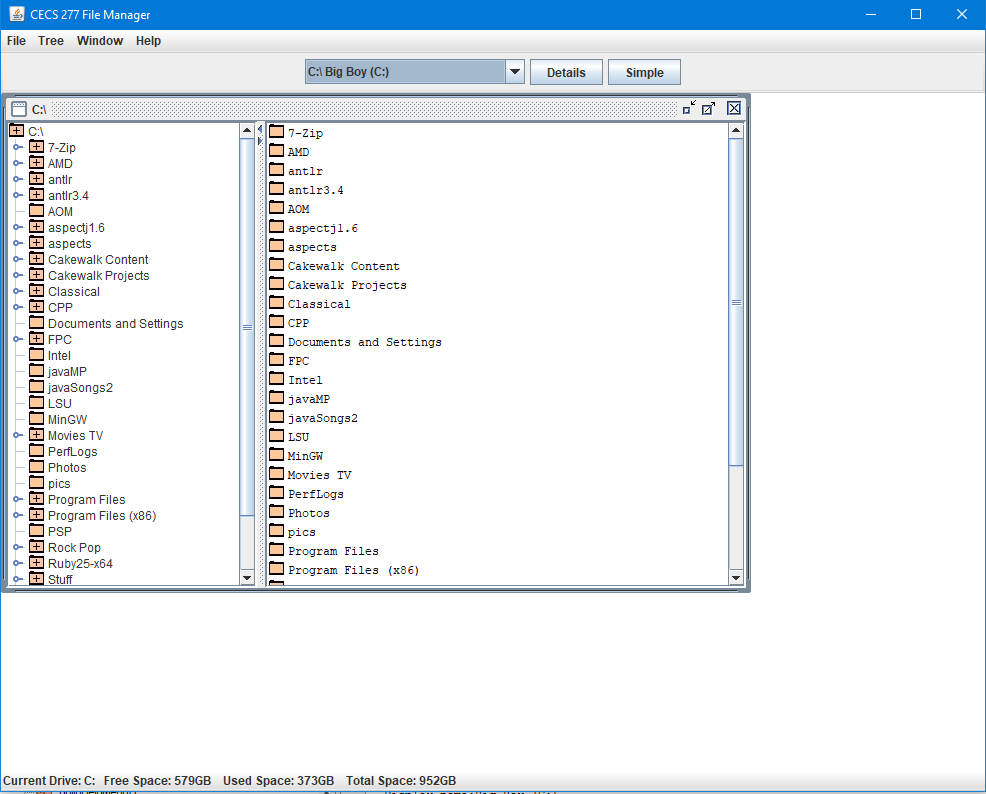


Figure 1 The Whole File Manager Enchilada

A first look at FM’s requirements and features:

1. It has a GUI. I did mine in Java Swing which is pretty easy to learn.
2. It is a multiple document interface. One or more drive windows may be open at the same time.
3. There is a menu system.
4. There is a toolbar (just below the menu).
5. There is a “statusbar” (more like an information bar) at bottom.
6. Drive windows go into the main window pane. They are movable, resizeable, can be minimized, maximized, closed.

**You are required to do this project in Java.**

2. Main Feature

The main feature of the project is the presentation of multiple frames containing a split-pane. The frame holds a splitpane, both sides of which contain scrollpanes. The left scrollpane contains a tree structure representing the folder structure for the entire drive. The right scrollpane contains a list of all the folders and files in the folder which is currently selected in the left scrollpane. Selecting a different folder in the left scrollpane causes the right scrollpane to redraw to show the folers and file sthat belong to the newly selected folder.

The frame can be closed, minimized (what Java calls iconified), maximized ( de-iconified) , and moved within the desktop of FM. The frame top bar always has the name of the current folder.

More than one frame can be open at a time. Frames are independent of one another.

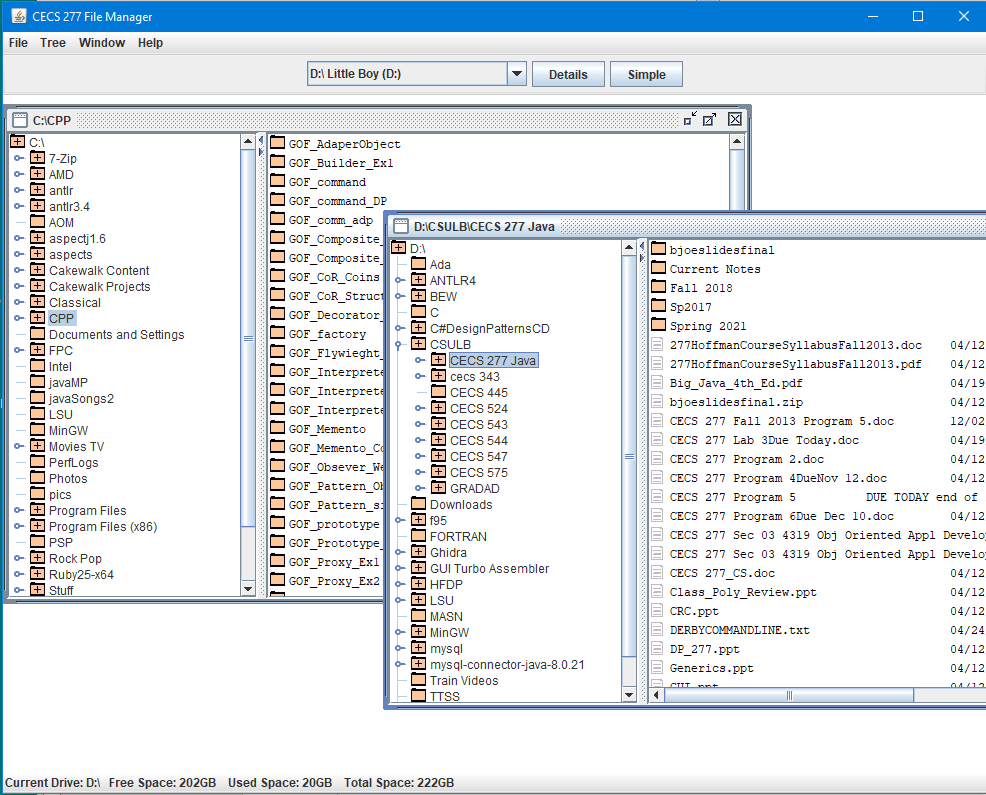


Figure 2 Multi-frames showing

As Figure 2 shows, there can be multiple frames open each containing its own directory information. Theoretically infinite.

Let’s take a closer look at the frame contents. The root of the tree in the left side is always a drive. The tree root is always expanded. Each entry in the tree is a directory (folder) on the drive. No files should ever be in the left window.

Notice that some folders have an icon with a +, others don’t. The + indicates that this directory (folder) has subdirectories. For example, in the front frame (D:\CSULB\CECS 277 Java) the CSULB directory icon has a + and can be seen to have multiple subdirectories. The selected folder D:\CSULB\CECS 277 Java has an icon with +, and as can be seen in the right scrollpane, has subdirectories (and files). We do not need to put + on the folders in the right side.

The files in a directory are presented with a different icon and additional information. The file icon is similar to that of a windows .html or .txt icon. The additional information is the file’s last modified date and its size in bytes. See Figure 3.

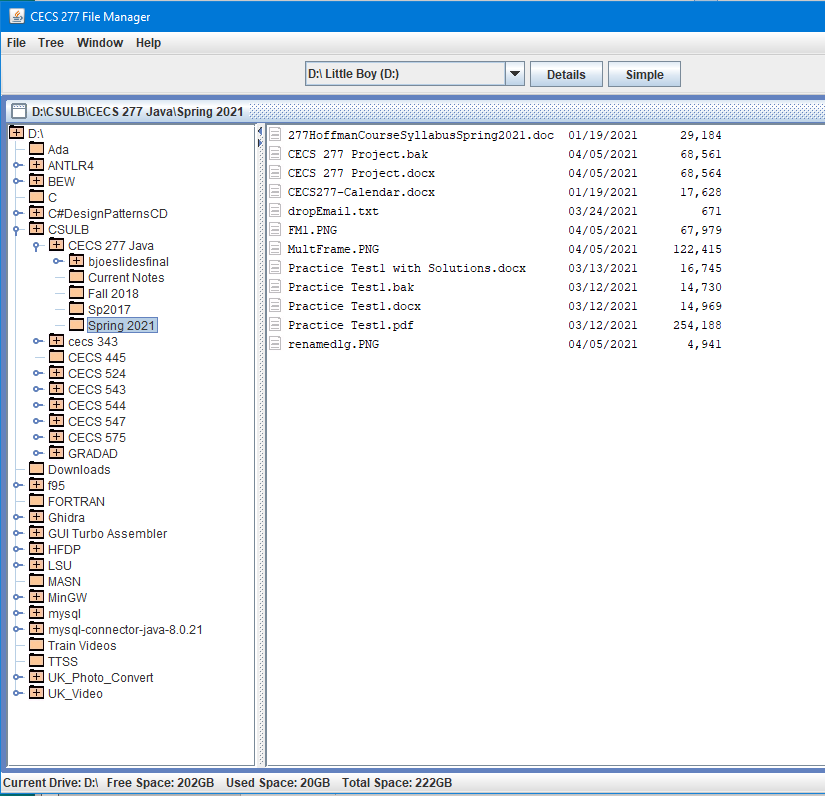


Figure 3 – An Open Directory Panel

Notes on the frames.

1. In the tree, don’t read the entire drive into the tree at once. That will take far too long and use too much memory. It is necessary to read at least one level further into a directory to determine if it has subdirectories unless there’s a package that does that for us.
2. Double-clicking the folder in the tree “expands” that branch (and only that branch - does not expand the subdirectories if they exist). Each folder must be double clicked to expand it (except the root as noted).
3. As a tree branch is clicked (single or double) its corresponding content is displayed in the right scrollpane.

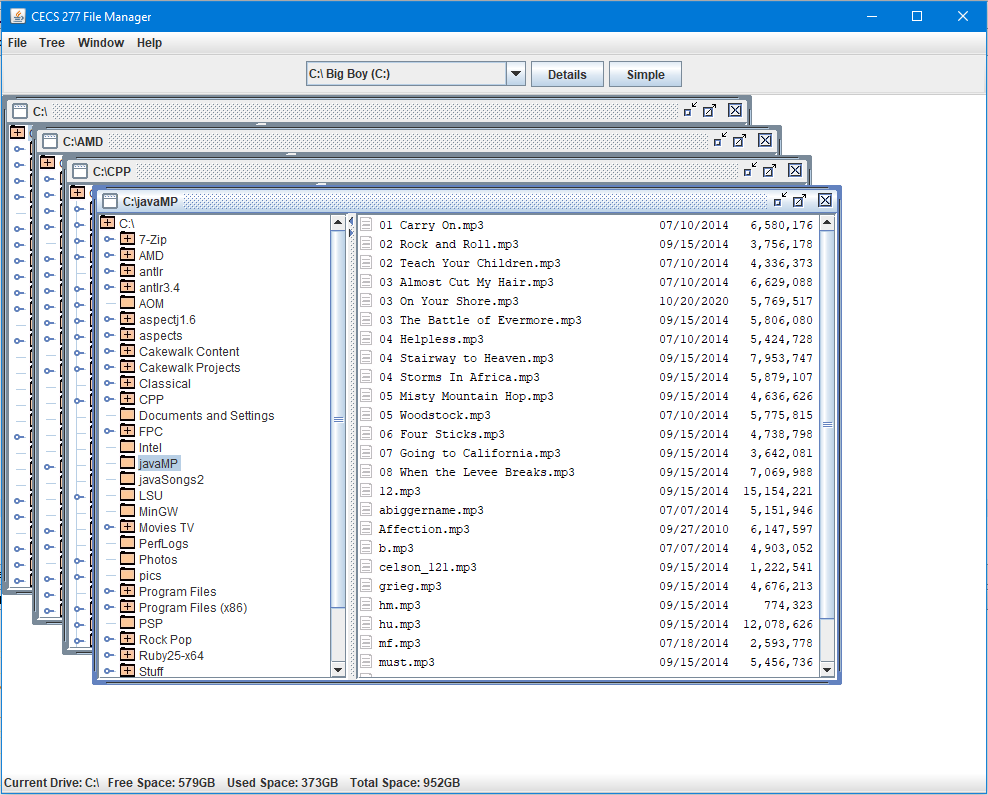


Figure 4 - 4 Frames on 4 different directories

3. Secondary Features

File managers exist to manage files. Just looking at the contents of adrive is not all that useful. Here are the secondary but essential features of the project.

The following assume a file has been selected (single clicked)

1. Renaming a file. The selected file can be renamed in its directory.
2. Copying a file. The selected file can be copied to its directory (by renaming and copying) or copied to a different directory. Or by DnD.
3. Deleting a file. The selected file can be deleted from its directory.
4. Double clicking a file will either:
5. execute the file is it is executable
6. open the registered application for that type of file and load the file. For example, if the file is a .doc, the program registered with the OS for that file type (probably Word) will be launched and the file opened in Word.
7. Don’t worry. All it takes to execute the file is one declaration and ONE method call!

How does FM rename, copy and delete? Two ways to do the same thing.

1. File menu entries for Rename. Copy and Delete.
2. A popup menu that opens over the file.

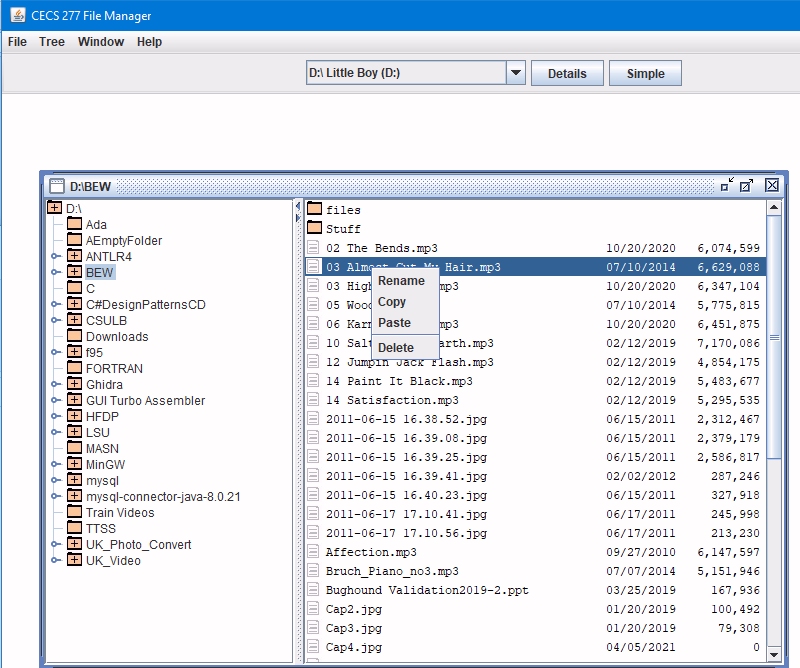
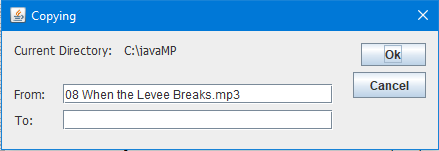
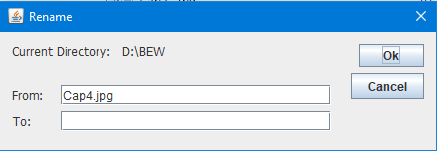


Figure 5 - Popup Menu

The code that does 1 can be used to do 2 and vice versa. An action listener can be written to respond to the commands. For Rename and Copy FM will need a small dialog box to get information. For Delete FM will use a JOptionPane to comfirm the deletion.

A small dialog can be reused for Rename and Copy because both require the same information. We can just change the dialog title to reflect the purpose of each.

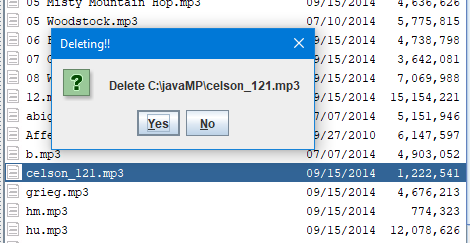




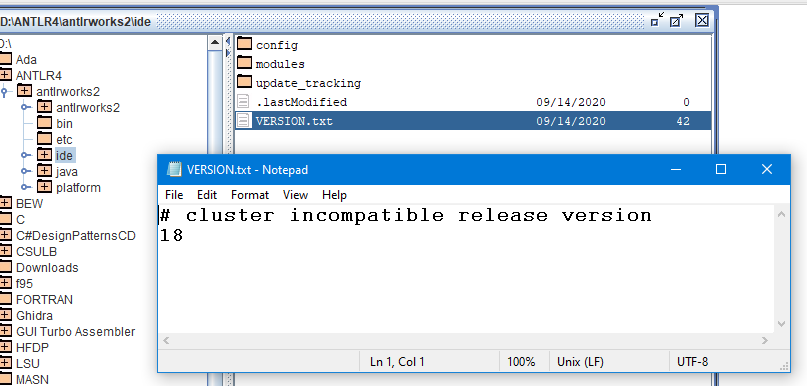
Same dialog, different names. Copying or renaming requires the user (in a very 1990’s sort of way) to provide the complete path and file name for the “To:” textbox.

The result is that the “From:” file is renamed to the current directory or copied as it is named to the new directory name provided by the user.

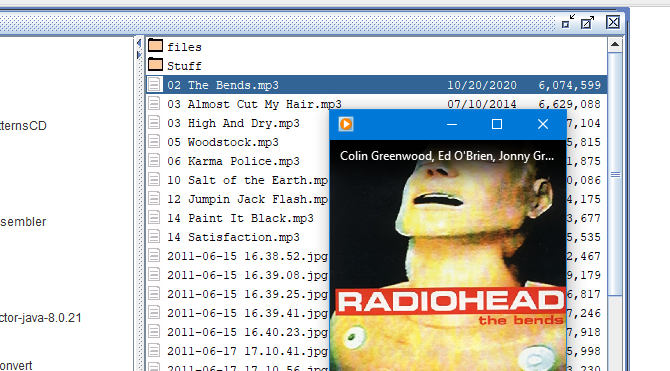
Deleting is a little different. Starts the same way by clicking a file and then selecting Delete, but doesn’t require additional information, just confirmation.



Double clicking a file will cause it to execute or cause the OS registered default program for that file type to be launched and the file opened in it. If there is no default program, the OS will ask the user what to use. Here’s an example for .txt files.



And an example for .mp3 files.

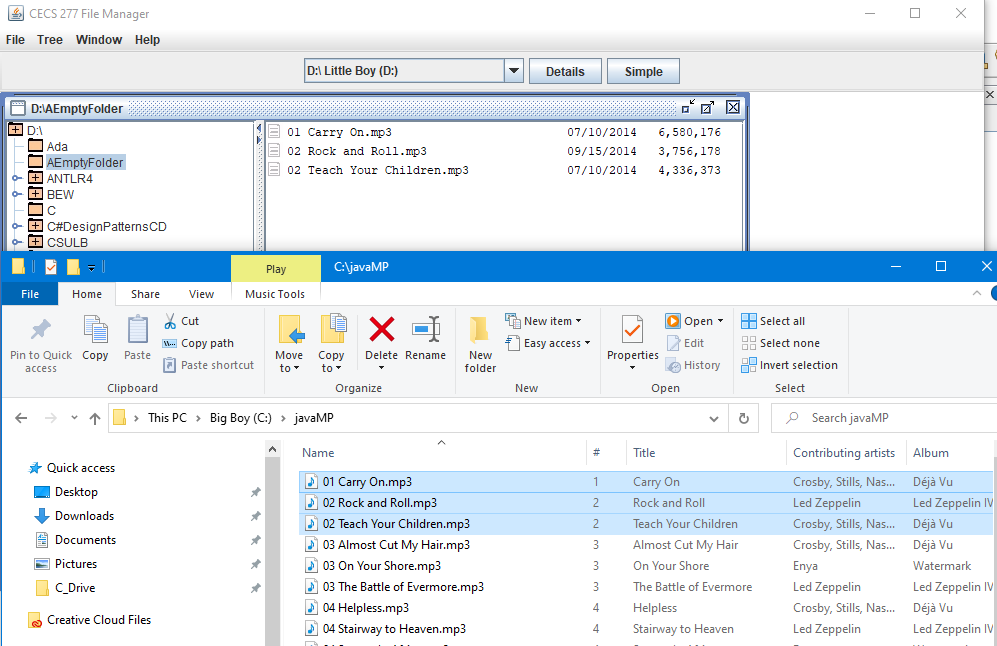


It takes one line of code to execute any file.

Copying by Drag and Drop. Another secondary feature of FM is copying by Drag and Drop. This is another one of those things that sound impossible a novice programmer to do but is not really difficult.

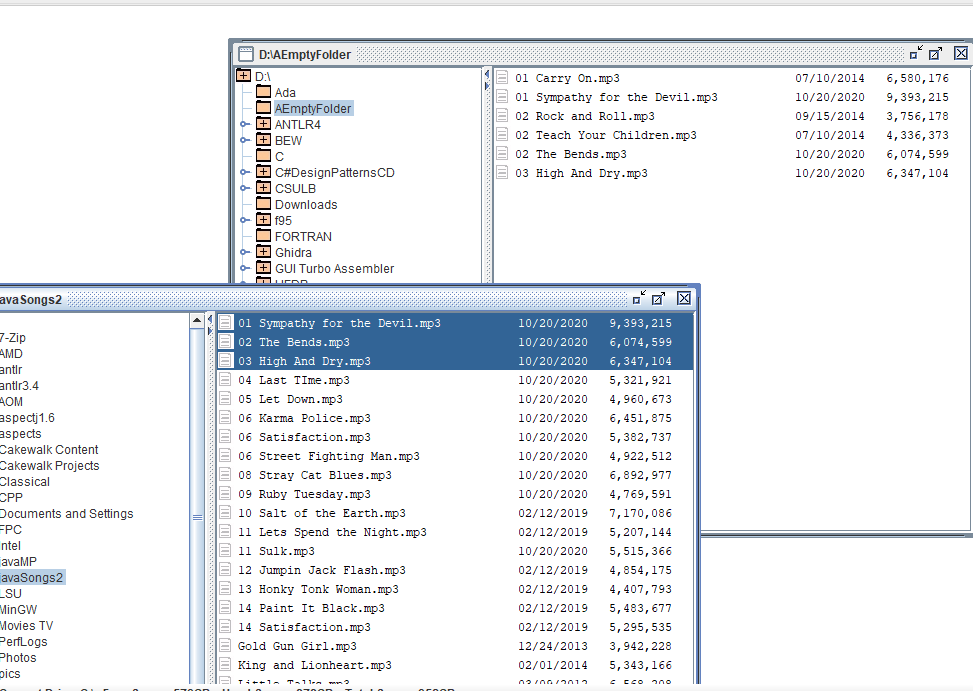
FM must be capable of receiving files from other sources such the current default Windows folder manager thing is called. FM must also be capable of copying by DnD from one of its frames to another of its frame. That is, internally.

Here is DnD externally.



The three highlighted files in Windows Explorer have been dragged on to the right scrollpane and added to that directory. All this operation requires is a DropTarget class in your project.

To do internal DnD, the project must add additional classes for drag gesture recognition and transfer file preparation. Here is an example:



The top three songs from C:\JavaSongs2 have been copied to the now inappropriately name D:\AEmptyFolder.

4. Miscellaneous Features.

Let’s start at the top. In the menu system there are these choices:

**File: Rename, Copy, Delete, Run, Exit**

**Tree: Expand Branch, Collapse Branch**

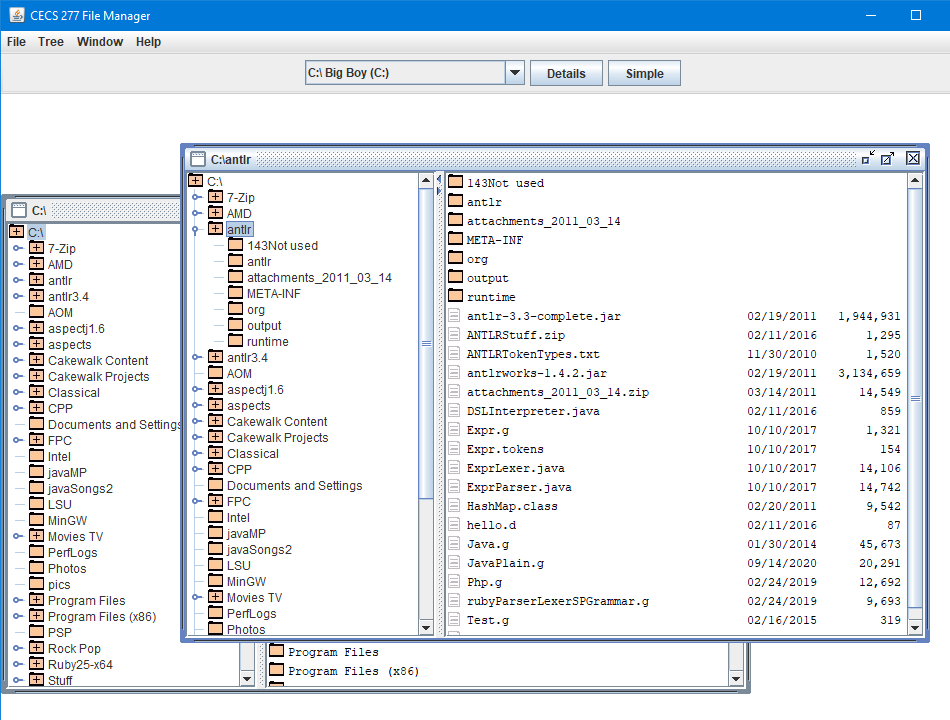
**Window: New, Cascade**

**Help: Help, About**

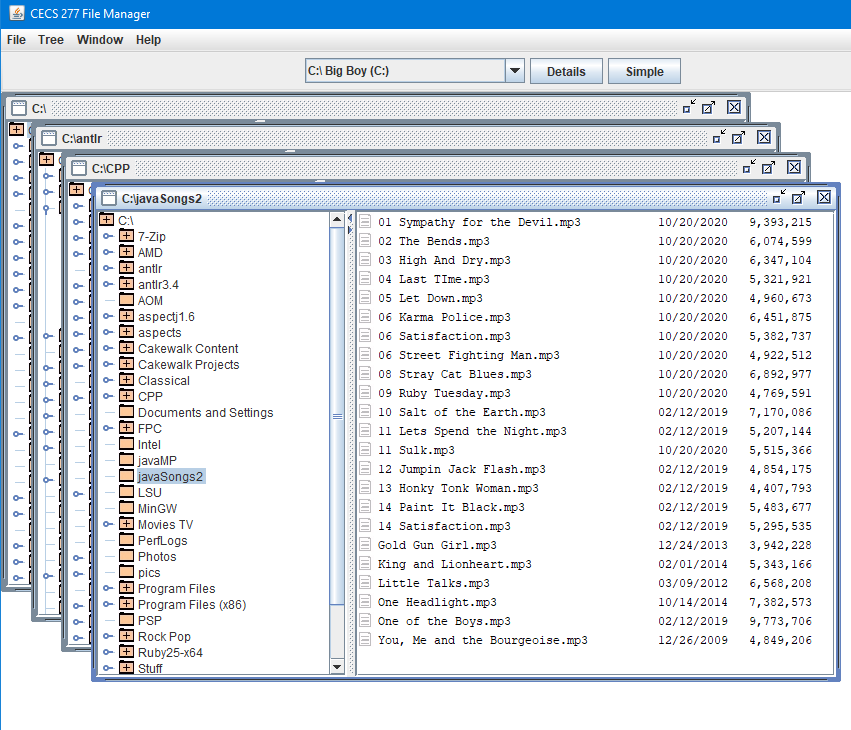
Rename, copy, delete and run can all share the code from the popup menu.

Tree: Expand Branch and Tree: Collapse Branch do as they say: if a branch in the directory tree is selected and it as subdirectories and Expand Branch is clicked, that one branch is exapnded one level only. If a branch is expanded and Collapse is clicked, that branch is collapsed. Note that if subbranched of the collapsed branch are expanded, then upon expanding the collapsed branch the subbranches are expanded, too.

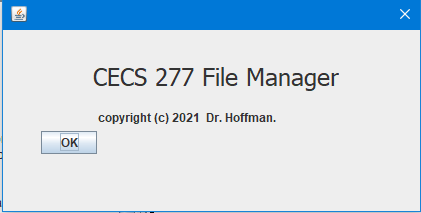
Window: New creates a new internal frame and places it at location 0,100 within the desktop pane. It defaults to C:. Any number of new frames can be created.



Window: Cascade does what cascading windows do everywhere. It stacks the open frames in a nice, neat overlapping style.



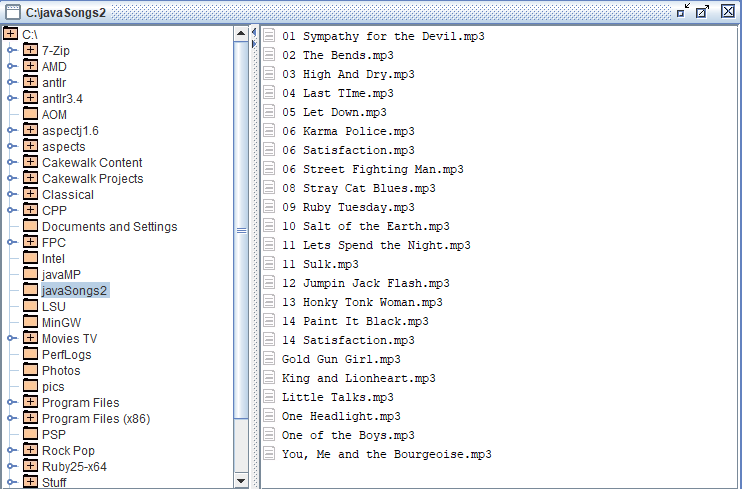
Continuing in the menu system, Help: Help is left to your imagination. Help: About show this is no news dialog:



4.1 The Toolbar

At the top of FM just under the menu but above the desktop is the toolbar. The toolbar has at a minimum a combobox for drive selection. Every drive on the computer must be found and listed, even USB drives that can be disconnected, jump drives, cd-rom/dvd-rom drives, fixed hard drives. Every drive. FM does not have to recognize drive changes dynamically. If a drive is added or removed while FM is running it does not need to respond (but would be nice if it did).

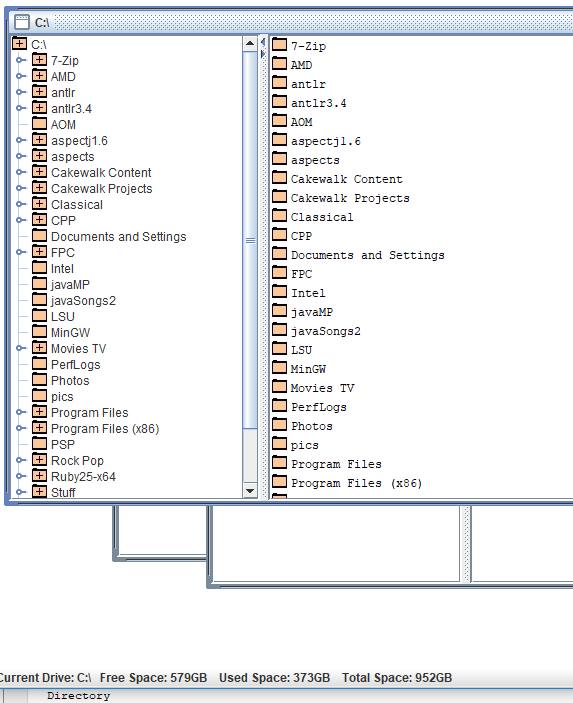
The next required toolbar components are Simple and Details. Clicking Simple will redraw the right scrollpane without file sizes and dates. Details restores file sizes and dates. Here is Simple:



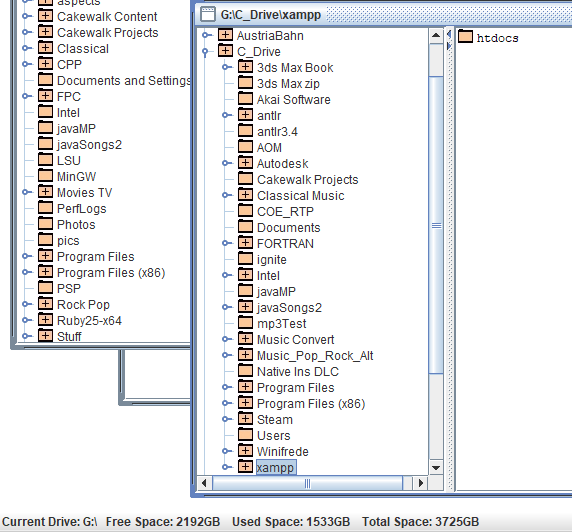
You may add other buttons to the toolbar to link to other FM features if you wish.

4.2 Statusbar (Information bar)

At the very bottom of the application is a smal string the provides drive information. It must reflect which drive is currently displayed in the frame that has the focus of input. That means if two windows are open and one is on drive C: and the other on F: (a jump drive maybe) then when the frame with C: is selected (has focus), C: data is shown. When F: is selected, F: data is shown. Acertaining which frame has the “focus” is trivial.



Here the C: drive has the focus.



G: is in focus and its information is displayed. Wow! 3725GBs!

That’s it for features.

How to start.

1. Start with the main frame. I will provide some videos on how to use Java Swing classes.
2. Add the menu system (with no operations linked), the toolbar, statusbar.
3. Develop the desktop pane. I will do a video.
4. Develop the tree side first (left). Then do the file list(right).
5. Figure out the multiple frame concepts. I will do a video.
6. Add the popup menu. Code the separate functions in their own handler classes.
7. Link the handler classes to the menu system.
8. Finish off with expand/collapse, cascading.

You need to scratch write this. You can use the Swing UI builder for dialogs, but that’s it.

**Testing**

The point behind this project and doing it in Java is to test it. I mentioned before that there are somes pretty goof Java testing tools for free. The early due date is to allow at least three weeks of testing.

We’ll cross the testing bridge when we get to it. I would advise you to do this in Netbeans or Eclipse.