Start with the version of the Permits 3 tier solution we went over in class that is posted on Canvas.

That version uses an array to hold what should be “persistent” data, you need to modify the code to save the permit data to a disk file. So you are going save the data that is in the array to the file system so the data will persist between stopping and starting the program.

Here is the overview, just read it but **don’t** start coding yet.

* Modify your “data tier” by adding a new “DiskStore” **static** class inside a new file inside the DataModelClassLibrary **project**. This new class will be called ***only*** by the DataModel class. It should expose 2 methods

1. WriteStringArray, which takes **in** a 2 dimensional string array and writes it’s data to the disk
2. ReadStringArray, which reads the **same** disk file. It creates a 2 dimensional string array of size 10,3 then reads the data from the file and puts it in the array, and **returns that new array**. We will use this “utility static class” to save and restore our data.

* Then you will modify the DataModel class to make use of that new class;

1. Everytime the DataModel gets a call to add a new entry to the “fake db” (the array),
   * it must first call the this new ReadStringArray method to get the current values from disk updated into its in memory array,
   * then, after getting the current data into the array, it updates that array to add the new person’s permit data (as the code already does).
   * Then, before that method does a return, it needs to call the new WriteStringArray method , passing to that method the updated array, to save it to disk with the freshly updated data.
2. Whenever a report is asked for in the DataModel class, it uses this very same new ReadStringArray method to get the current version from disk and then it simply returns that to the middle tier. The middle tier already knows how to pass that data back to the UI tier.

It’s fine for this homework that the array is a fixed size, to only hold 10 data rows, and the code that reads and writes the string data can also assume a fixed number of rows (10) and columns(3). So you can hard code those values in any and all arrays and loops you use. *(Of course it is much better if your loops using arrays sizes to determine their iteration count, so that you can change the fake db array to hold only 3, or 30 entries, and the only code change is to change the size of the array.)*

You **should not** change **anything** in the UserInterface code nor in the Middle, BusLogic class. If you think you need to, stop – you are making a mistake.

======================== CODING ADVICE ============================

* Ok, now start coding.
* I suggest you start by just creating this new **public static** class in the DataModeClassLibrary: right click on that project, select add, select class, pick class, and name it DiskStore.cs That will create a new file for this new class.
* In this new DisktStore class definition, add a new **public** **static** method WriteStringArray method that *accepts* a 2 dimensional string array as a parameter and writes the data from the array to this disk, just as we did in the ReadWriteStringArray project. It does not return anything.
* In the same DisktStore class, add a 2nd new **public** **static** method ReadStringArray method that *returns* a 2 dimensional string array, but takes in no parameters.
  + Again look at the code in the ReadWriteStringArray program we went over in class, and figure out how, in the new ReadStringArray method in the DiskStore Class, to read the file you wrote, and then in a loop, stuff the array you created there with those values from the file data.
  + The code should, as we did in the program, verify the file exists before trying to read it, and if it does not, it should create a new empty one.
* Now Modify the existing SaveNewPermit method in the DataModel class, such that
  + **AFTER** it updates the array when it adds a new permit, but just before it does the return, add a new call to that new DiskStore.WriteStringArray method and pass in the array it uses (fakeDB).
  + Make sure the program still works as before. While you are making a new method call and passing the data array, since you don’t read it back yet, this change should not break the existing functionality of the program. Make sure the program still can add permits and read back a report of them all. But also, using Notepad, look at your new file on disk and see if it is saving permit data onto the disk. Get this working before moving on.
* Only when that all works, move on to modify your GetPermitsDT () method in the DataModel class.
  + Modify that GetPermitsDT () code to return to the BusLogic class the array you get back when you call DiskStore.ReadStringArray();
    - Make sure your ReadStringArray() method starts by calling the VerifyFileExists(10,3) method so that if this is the first time you are running the program, it will create the needed disk file.
  + Do this by just putting a return in front of it: return DiskStore.ReadStringArray(); And get rid of the return fakeDB code that is there now. So now you are not returning the data in the fakeDB array, you are returning the array from the new method all the way back to the UI layer.
    - Run your program. Again, verify that you can add permits and then get a report, but now the data showing in the report is coming all the way from your disk.

=====================================================================

At this point

> You are saving the data in a file,

> and you are able to retrieve it from the file.

But, if you stop the program, and restart it, the prior data is not in your fakeDB array, it is lost.

==============================================================

So, whenever we are adding a new permit, we should read the **existing** data from disk and make sure our fakeDB has the latest data before updating the array. That way if we stop the program and restart it, our old data is not lost or overwritten.

So modify the SaveNewPermit method to start off, very first line, by doing a

fakeDB = DiskStore.ReadStringArray(); // if you hard coded this method to [10,3]

or

fakeDB = DiskStore.ReadStringArray(10,3); // if your method accepts the sizes for the array

This should call your new method and retrieve the file from disk, and load the file data into the fakeDB

However, since our new method in DiskStore creates an empty file with 30 rows of the string “” (which is an empty string, not a null) in the loop, you need to change

if (fakeDB[i, 0] == null)

to

if (fakeDB[i, 0] == null || fakeDB[i, 0] == ""

At this point, everything should work, and you should be able to add a few names, stop the program, start it again, and run a report and see the names still there.