# CS6326 Human-Computer Interactions

Fall 2020

**Assignment 2 – Rebate Form**

Since this is a course in user interface design, your assignment is to write a simple program to maintain a file of data. This is a very common thing to write in the “real world,” but it’s not as easy as it looks. Consider a company that has a department that processes rebates\*. The buyer of a particular product sends in a form with various information and a clerk enters it into a program to be stored in a database. (This model is changing. Companies are now getting users to do some of the data entry by using a Web site.) Rebates are available only to people living in the United States. Your job is to write that program. This is a “sovereign app,” so design it as such. Specifications are as follows:

1. The file (and therefore the screen) will contain the following fields:
   1. First name (20)
   2. Middle initial (1) (may be blank; entry not required)
   3. Last name (20)
   4. Address line 1 (35)
   5. Address line 2 (35) (may be blank; entry not required)
   6. City (25)
   7. State (2)
   8. Zip code (9)
   9. Gender (1) (M or F)
   10. Phone number (21)
   11. E-mail address (60)
   12. Proof of purchase attached (Yes/No)
   13. Date received (default to today but changeable)
2. The following fields will be in the file but are added by your program, not the user, and are not visible to the user. Add them only when adding new records, and do not change them if a user modifies a record.
   1. The time at which the user first entered a character into the first field, accurate to the second in hh:mm:ss format.
   2. The time at which the user pressed the Save button.
   3. The number of times the user pressed the *backspace* key during data entry. This can be obtained by processing a keystroke event, and will be explained.
3. Your program must provide three functions:
   1. **Add** a new record to the end of the file.
   2. **Modify** an existing record and write it back to the file. This means that you can modify **all** fields of the record, including the name, but not the three invisible fields.
   3. **Delete** a record from the file.
4. Your program must not be able to add a record that contains the same first name, last name, and phone number as a record already in the file. You must not be able to modify an existing record such that the modification violates this rule, either. In general, your program must not be able to enter bad data into the file. That is, dates must be checked for validity, for example. On the other hand, there is no way to check a person’s name for validity.
5. **Data file handling:**  Below are the requirements (not suggestions) for the data file.
   1. Since this is not a database class, you will use a flat text file with one record per line. Fields are separated by tabs.
   2. If the file does not exist, your program must create it.
   3. The name of the file is **CS6326Asg2.txt**. You will lose five (5) points for using any other file name. **Do not** have any information other than the file name. For example, using a file with a name like c:\users\jxc064000\CS6326\CS6326Asg2.txt is **not** acceptable and will cost you 20 points.
   4. The fields must be written in the order given above, with two additional fields given below that are not entered by the user.
   5. When the program starts, read the entire file into memory. When a user saves data, write the entire file.
6. “But C# has database functionality. Can we use that?” No.
7. You may not read everything into a datagrid or something like it and treat it as a spreadsheet. You must have separate fields for the various data elements. (One issue with using a datagrid is that the user will have to side-scroll, which is poor design. A second issue is that in a real system there could be thousands of records, but a real system would have functionality I’m not requiring and take too long to program.)
8. You should show the full name and phone number **only** (not every field) in a multi-column scrollable list. Clicking on a list item should put all of its data elements into the fields for modification. That is, if a name is selected from the list, the clerk should be able to save the changes or discard them.
9. All three functions must be done from the same main screen. There should be no menu and no multiple screens.
10. You have a choice of language. You may write in C# using only packages that are installed as part of Visual Studio. Do not use any third-party controls that we (the grader and I) must load in order to compile and run your program. This is a desktop app, meaning you should use WinForms or Windows Presentation Foundation (WPF.) You may also write in Java using JavaFX and/or Swing, but again, no third-party controls or packages. This is more difficult than C#.
11. Your program must apply good object-oriented methodology. That is, the user interface, application logic (what little there is in this program) and the technical services layer must be separate classes.
12. This is a user interface assignment, so pay particular attention to how your program looks and how easy it is to use. Minimize the number of keystrokes necessary. Apply principles you have learned from the reading and lectures.
13. This is an individual assignment. All code other than standard libraries and APIs must be your own.
14. Program naming conventions: Your project must have the name Asg2-<netid>. That is, if I were to hand it in I would name it Asg2-jxc064000. The executable will then also have that name. You will lose 10 points for not following this convention.

I strongly suggest that you ask yourself many “what if?” kinds of questions about how this program may be used and the kinds of errors a clerk might make. Write it and test it. If you finish the code on the day it is due you will almost certainly not get full credit.

**This is a complex assignment.** Read it carefully before you start, then refer back to it as you design and write, and finally, once the program is complete, make sure you have met all of the requirements. In particular, this requires user interface design. Keep in mind principles such as minimalist design, filling the screen, and so on.

**If you have questions about this assignment**, please feel free to bring them up in class, e-mail them to me so we can discuss them in class, or come to talk with me during office hours.

**Visual Studio** is available free through DreamSpark, or you can download Visual Studio Community from Microsoft at https://visualstudio.microsoft.com/vs/express/. If you get Visual Studio Professional, make sure you use VS2017 or VS2019, nothing earlier. If you do not own a computer that runs Windows, such machines are available in the lab and have all required software loaded. Mac users, check this out: <https://www.macworld.co.uk/how-to/mac-software/code-c-sharp-mac-3640347/>

**To hand in through eLearning:**  **Your entire Visual Studio project** and a text file with at least two names and addresses you have entered, created by the program. This should contain the executable in the bin\Release directory.

|  |  |
| --- | --- |
| **Grading (total points: 100)**: | |
| Meets technical requirements and OO design | 30% |
| Does not crash on invalid input | 20% |
| Clean user interface: follows various conventions, minimizes keystrokes and use of the mouse. | 40% |
| Program comments (header comments is 5%) | 10% |

|  |  |
| --- | --- |
| Grading Rubric | |
| Doing I/O in the user interface module instead of a separate class. | -10 |
| Separate screens for add, modify and delete (should all be done on the same screen.) | -30 |
| Program requires mouse to operate | -20 |
| Program crashes with valid input | -20 |
| Program violates design guidelines given in various lessons and slide sets. | -5 to -30 |
| Program has hard-coded path along with the file name such that it won’t run on the grader’s machine | -20 |
| Program allows invalid input to be entered into the file, such as null values for required fields, dates in the future, etc. | -10 |
| Focus not on first field | -3 |
| Use of pop-up dialogs for error messages for errors that could have been caught by ongoing validation of data. | -10 |
| Missing list of entries. | -5 |
| Not following the naming conventions. | -10 |
| Data file not named correctly | -5 |

\*Rebates are money sent back to someone who purchases something. Let’s say you go to Electron Hut and buy a $40.00 disk drive that gives you a $5.00 rebate. You fill out the form, similar to the one above, and send it in along with proof that you actually bought the flash drive. Many weeks later, you receive a check for $5.00 in the mail.