

**BCIS 3680**  
**Enterprise Java**  
**Problem #3: Database Manipulation and File Integration**

In Problem #2, you created a “front-end” to your application (splash screen) that contained a number of selectable choices under the heading “Processes.” For this application, you are to provide the mechanisms necessary to fulfill the “Maintain Expense Records” option. In addition, for this application, you are to complete the functionality of the “Enter Expense Transactions” choice (for which you started the development in Problem #1). Specifically, the enhancements to be added include linkage to an Expense database for purposes of retrieving an expense description, a scrollable list from which to select the country code, recording accumulated transactions to a output file, and updating the Expense database to reflect alterations to individual expense records.

To begin implementing these enhancements, the first step is likely to be the creation of an expense database. This database should use the same name as your eagle-mail ID. (If you eagle-mail ID is xyz1234, then that should be the name of your database.) This database should be developed in MS Access and should consist of an Expense table. This table should provide for the following attributes: expense code (3 digits), expense description (up to 30 characters), date of latest transaction (a date; you decide which format makes sense to you), number of transactions (an integer), and total amount (double). The initial data for the expense codes and descriptions can be imported from the Excel spreadsheet [Expenses.xls](#), which accompanies this assignment. The initial value for your date can be whatever makes sense for your selected format, but should not reflect a date any nearer than January 1, 2000. The values for transaction count and amount should be initialized to zero. Expense code should be the key to this table. Your column names should be reflective of the fields use, and each should have a prefix of your initials (as shown in your eagle-mail ID).

For our transaction data entry part of the application, you should be able to retrieve a matching expense description for the expense codes provided. This description should be placed in the previously unused “expense name” field. But, you need to hang on to the record. Once the user has entered the data and performed the conversion, you are to update the record once the “Add Transaction” button (or menu choice) has been invoked. Thus, in addition to adding the transaction to the existing table, you are to add 1 to the number of transactions field and add the US Dollar amount to the total amount field. You are also to replace the transaction date in the Expense (database) record with the transaction date entered by the user (provided it is a more recent date than already recorded for that expense).

Other changes for the Expense Transaction Entry portion of the application also include an alteration to the mechanism used to select country codes. In the past, the user has had to remember these codes. We wish to improve accuracy as well as release the user from the burden of having to remember these codes by providing a scrollable list of the country codes. Since it is possible that the list of countries in which we operate could change over time, these codes are provided in a sequential text file ([Country.txt](#)), which accompanies this assignment. While presently, there are less than 60 countries in which we operate, you should anticipate having up to 100 countries in this file. The area in which the country code text field exists should be replaced with a JList showing 5 country codes. You are to default the list to “None,” which should be the first choice in the list. You should reset to “None” when the reset button is pressed.

Finally, you should provide one last button labeled “Write Transactions.” When this button is pressed, all transactions that have been placed in the on-screen table should be written to a sequential file. The individual records in this file should match the number of records (and columns) reflected in the on-screen table. The file extension for this file should be “.txt” and the default file name should be a combination of the employee number and the current date. For example, if the employee 30000 entered transactions on September 30, 2008, the default file name would be something like “30000-20083009.” However, it is possible that a single employee might want to enter expense transactions for two (or more) separate trips in one day. So, you should use the FileChooser class to provide the user with both the ability to alter the name of the file to be saved as well as the ability to navigate to an appropriate subdirectory and disk drive. (You may default the path to My Documents on the C drive.)

The last enhancement is to provide a fully functional part of the application capable of providing maintenance to the expense database. You should be able to display each of the attributes (fields) of an individual record in the database, along with a descriptive label that helps to identify that data. Your application should provide basic navigation through the database (first, last, next, previous) so that any individual record can be displayed on the screen. If a field value is modified, then you should provide the capability of writing that updated record to the database. In addition, you should provide the capacity to add and delete individual records. If a record is added, the user should be able to supply the expense code and expense description. You should default the date to the current system date, while the other fields should be initialized to zero. You should not permit the addition of duplicate expense codes. You should provide a means of exiting this part of the application, which should return control to the “splash screen” part of the application.

Finally, you should properly document your solution using both class diagrams and Javadoc. Class diagrams should demonstrate the relationships between the classes.

You are to supply standard elements (hardcopy, softcopy, sample output generated, etc.) for this problem plus hardcopy of your expense database layout (both data and structure). Remember that your output for this application also includes one or more expense transaction files for individual users. In addition, you are to supply the requested documentation to support this application. Your “startup” code should be executable from a file designated as “**Prob3Driver**”.

Make sure that your submitted solution not only contains the Java elements (in a Netbeans project), but your database, Expense.txt file, and at a softcopy sample of an expense transaction file.

**Due Date: October 14<sup>th</sup>**

**Value: 10 percent**