Project Overview

Application data model:

- use a MySQL database called itm411DB for all of the record data.
- create the single database table with fields from the CSV file. The DB does not have
- to be created dynamically, but can be using JDBC if you wish.
- provide the capability of re-initializing the table data if the user wished to start over.

Application functional model:

- manipulate populationRecord objects from the GUI using database CRUD (Create,
- Retrieve, Update and Delete) JDBC operations.
- provide components for user interactions for all of the functions required in MP2 (e.g.
- data analytic queries, etc).
- provide GUI components to trigger serialization and deserialization.
- display GUI results in either tabular, text field or text area formats based on the userspecified output type. For example, if the user wishes to view all records, then a tabular format would be rendered. Or, if the user wished to view or manipulate a single record, then a textfield format would be used to render the data of a single record.
- display GUI status of application state and user interactions (e.g. successfully loaded
- DB). Hint: Use a JLabel to update GUI status by rewriting the contents.
- consider usability in your GUI components layout (i.e. the easier it is to use, the higher
- the grade). Ensure that the UI is intuitive to use as a 'newbie' user. Use context areas
- (functional decomposition) for similar functionality. So if the user wishes to execute
- any of the data analytics, the GUI components to execute this functionality should be
- grouped together and easy to understand at a common location on the GUI. Consider
- using tabbed frames, internal frames, pop-up sub-windows, etc.

Installation, Compile & Run-Time Requirements

Program was built using a Windows 7 desktop configured with— java version 1.6.0_26 . The source code (project files) was developed using Eclipse IDE for Java Developers / Version: Juno Service Release 1 / Build id: 20120920-0800. In addition, for database the connectivity the application relies on the secondary mysql-connector-java-3.1.14-bin.jar library.

The application accepts four sequential command line arguments. These include:

#	Parameter	Example
1	Mysql Connection string	jdbc:mysql://10.40.81.93:3306/
2	Mysql Database Name	Itm411db

3	Mysql Database login	itm
4	Mysql Database password	itm
5	Data Directory path	C:\\CHProjects\\Code\\Java\\itm411\\FMP\\data\\

To run, extract the contents of the mp3.zip file to a local folder. Open a windows command prompt, and navigate to the "dist" subdirectory. Type the following:

```
java -jar "..\dist\fmp.jar" "jdbc:mysql://10.40.81.93:3306/" "itm411db" "itm" "itm"
"C:\CHProjects\Code\Java\itm411\FMP\data\"
```

NOTE – Prior to running, you will need to create a Population_T table on the target MYSQL database. The script for creating this table is included in the sql subdirectory.

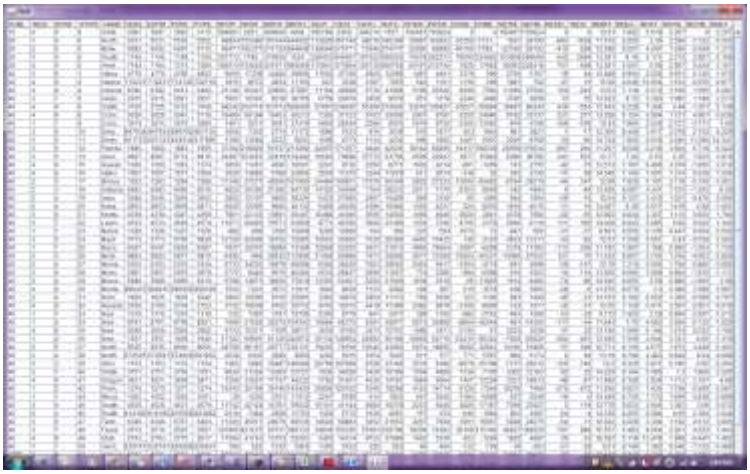
Insights and Expected Results

This application delivers the results as described earlier in this document in as much as was possible to build over the period of the last 48 hours . Additionally, I would like to state that building this type of component from scratch using Java is time consuming and not recommended. Much of this functionality could have been implemented much faster and easier using newer pre-built technologies such as Silverlight or Jqgrid. Re-inventing the wheel is something I try to avoid, and this exercise seemed to push that idea to the limit.

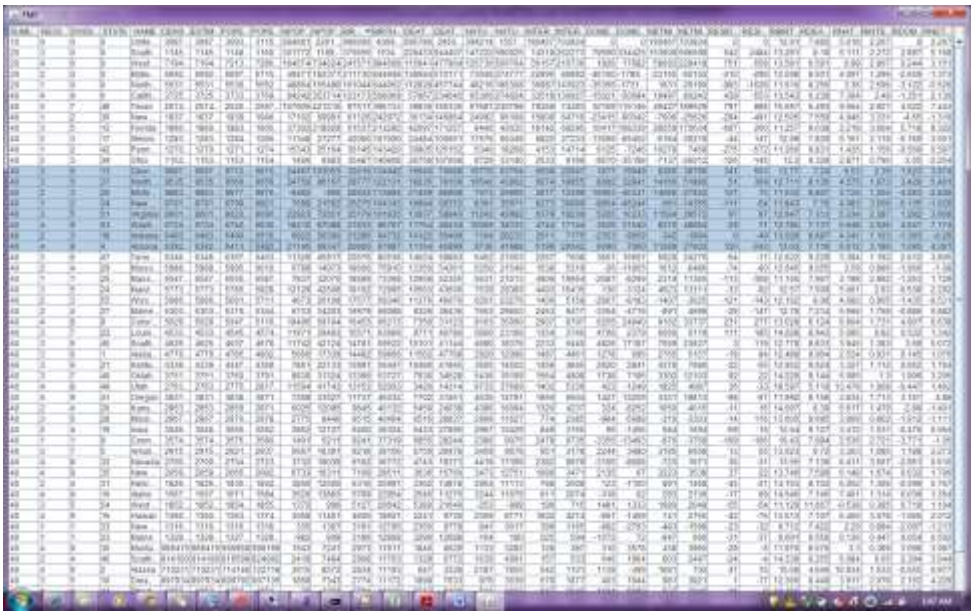
Qualifications / soap box out of the way, the program provides a mixed mode approach to persistence. At startup, the target MySql database is truncated and repopulated from the NST_EST2011_ALLDATA.csv file. After words, the data is then extracted and render using a JFrame control to host a JTable control. CREATE / UPDATE / DELETE functionality is provided to user via Keystroke Listeners.

Screen Captures

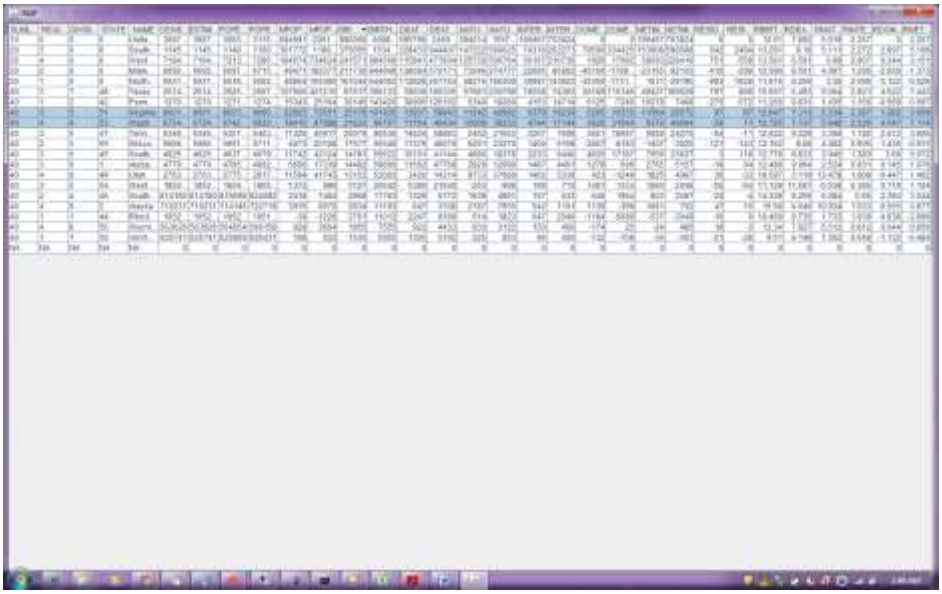
On startup, the application output the original MP2 output. Performing the various Analytic operations outlined in this project. Once this initial logic completes, the application opens the following form window—



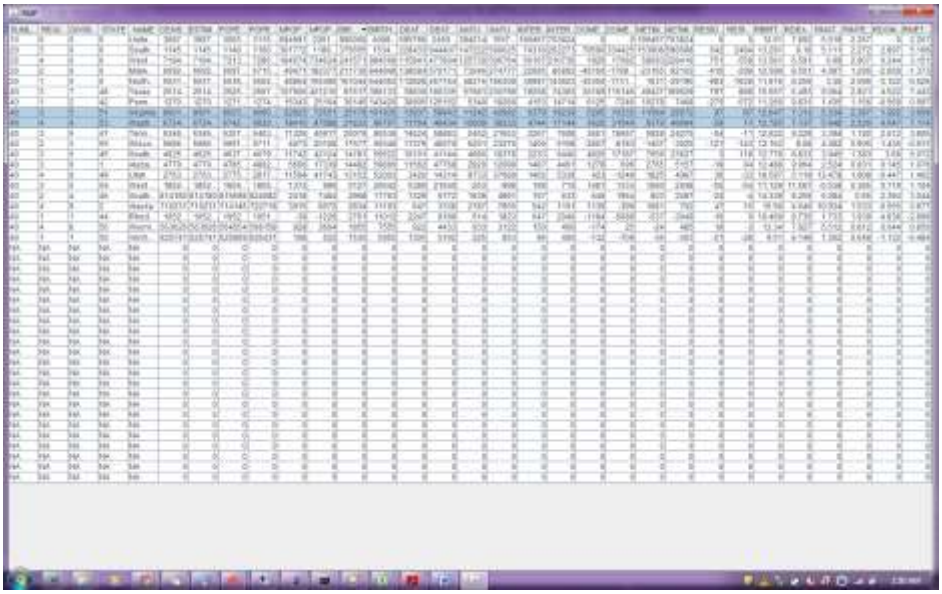
The default window is sized to fill the user's screen. Once loaded, the user can resize, re-order the columns based on their preferences. Row selections are highlighted—



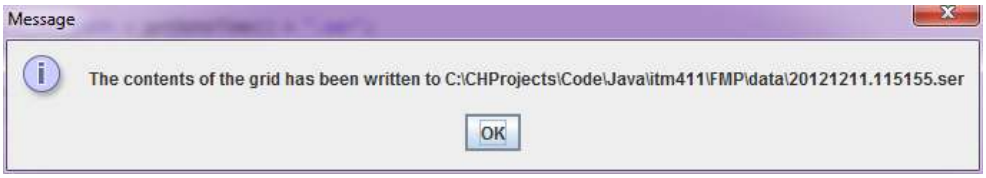
To Delete rows, the user simply needs to select and press [DEL]—



To Insert rows, the user presses the [INS] (or Insert) button—



Finally, to save their changes, they press “CTRL-U”. The application will write the contents of the grid to serialized file in the data directory passed via command line parameter at startup. The file name convention is based on current system and follows the pattern: “yyyyMMdd.hhmmss”. Once this process completes, the application displays the path of the file that was created—



Additional Info

For additional information, please refer to the Javadoc generated html pages published in the \docs sub directory. Alternately, please visit my ITM411 github repository @ https://github.com/bradyhouse/ITM411. This code base as well as all others created will attending this class are published on this site.