CSCE 155
Spring 2008

Homework Assignment 5: Advanced CSE Registration System

Assigned: Monday, March 31, 2008
Due: Monday, April 14, 2008

Note: This assignment is to be completed individually - collaboration is strictly prohibited.
Points: 100 points

Objectives

The objectives of this homework assignment:

1. Familiarize with the use of **arrays**
    a. Manipulate a collection of data values using an array
    b. Compute statistics out of an array of values
    c. Declare and use an array of primitive/reference data types
2. Familiarize with the use of **exception handling** for data format and range checking
3. Familiarize with a **sorting method**
    a. Using the Heap sort on an array of object elements
4. Re-use existing classes
5. Understand basic GUI objects
6. Write a GUI application
    a. Identify the GUI objects that are necessary for the application
    b. Organize the positions and functions of these GUI objects for the application
    c. Design the application such that the user interface is sensible and user-friendly
7. Familiarize with the **`javax.swing`** and AWT packages and layout management
8. Familiarize with code documentation, compilation, and execution
9. Expose to Java syntax, programming styles, and Java classes

Problem Description

This is a continuation of your *GUI CSE Registration System* (Homework Assignment #4). Statistical information is important when analyzing student data and the department needs your help to sort and provide statistical data for the students.

You should design a good graphical user interface for the calculation and display of the information about students. The students processed in the system should be stored in a Java array. ***Note: You cannot use JCF classes or any standard classes to store the students.*** The system should use the drop down menus and other options shown below (note that when there is a conflict, the following requirements overwrite those of Homework Assignment #4 and/or any previous homework assignment):

1. Your program should have at a minimum:
   (a) `Students.java` that is provided with this homework assignment on BlackBoard. This class holds all the information about ***a*** *student* as defined in previous homework assignments. You must use the class provided on BlackBoard for you with this homework assignment so that it will read the new input file correctly.
   (b) `Courses.java` that is provided with this homework assignment on BlackBoard. This class holds all the information about ***a*** *course* as defined in previous homework assignments. You must use the class provided on BlackBoard for you with this homework assignment so that your solution will read the new input file correctly.
   (c) `RegistrationSystem.java` that provides most of the functionality of the advanced CSE registration system as required (*Hint:* You should reuse some of your `RegistrationSystem.java` class from Homework Assignment #4 or use (and acknowledge) the `RegistrationSystem.java` class provided as the solution on BlackBoard).

2. The Advanced CSE Registration System GUI should contain the following:
   (a) A menu system to control the overall system. See Figure #1 – Drop Down Menus and Options. It must include the following menus and menu items:
      1. File Menu with the following menu items:
         a. New – Create a new students file
         b. Open – Brings up an open **`JFileChooser`** for user to select students file to open and opens that file
         c. Save – Saves current students to the open students file
         d. Save As – Brings up a save **`JFileChooser`** for the user to select the students file to which to save the current students
         e. Quit – Prompts to save current students to file if they have not been saved and exits the system
      2. Edit Menu with the following menu items:
         a. Add Student – Create a new student for the set of students
         b. Delete Student – Deletes a selected student for the set of students
      3. Process Menu with the following menu items:
         a. Compute Stats – Calculates the statistics for the entire set of students
         b. Sort Students – Allows user to sort set of students

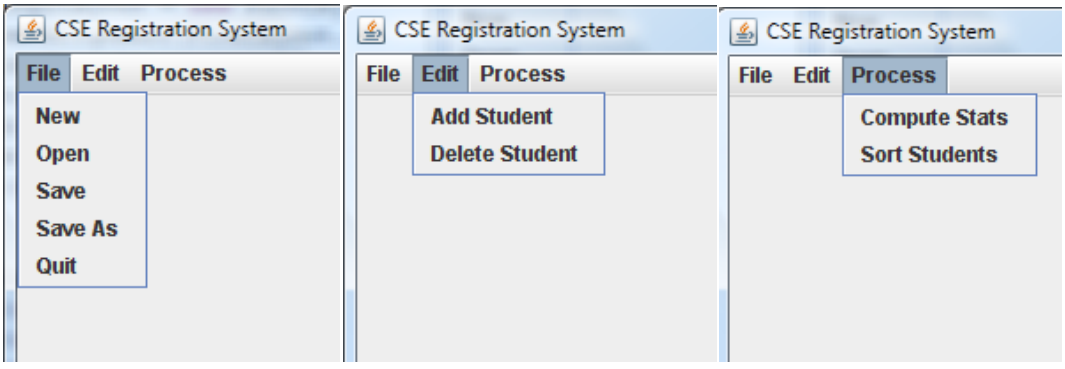

Figure #1 – Drop Down Menus and Options

(b) The system should operate with the following processes:

1. The user must first create a new list of students or open a list of students from a file. You will need to ask the user how many students are currently stored in the data file they wish to open.
2. Once the user has chosen to create a new set of students or open a list of students, then he/she may begin adding new and deleting existing students. Both processes require an additional window.
    a. Adding should be done in a separate window and should ask for all of the data for the student (e.g. Student Name, ID, Registration Date, Phone Number and address information). Then the user can click the "Search Courses" button to locate a specific course and associate it with the newly added student. The user should have buttons similar to Homework Assignment #4 where they can select to "Clear" and reset the input fields for reentering the student information. The window should also have a "Save" or similar button to commit the information to the current list of students. Once added, the main display of students should include the new student as appropriate.
    b. For deleting, consider displaying a separate window using a combo box or a list area with all of the students' names listed. When the student is selected for deletion, your application should make sure the user really wants to delete the student information by displaying a message asking "Are you sure you want to delete?" If confirmed, your application should remove the student information from the current list of students. Once deleted, the students should no longer be displayed in the main display of students.
3. Once some number of students has been added or read from the file, the user may now process the data.
    a. `Compute Stats`: When the user selects this menu item, the program will calculate the statistical information for all of the students in the set of data. The statistics should be calculated based on the **total amount due** by the students. Statistics for the students include: (a) the number of students, (b) the mean of the students' total amount due, (c) the median of the students' total amount due, and (d) the standard deviation of the students' total amount due. See the Appendix for additional information.
    b. `Sort Students`: When the user selects this menu item, the system will create a sorted array of students, sorted by their **total amount due** values in descending order using a Heap sort. ***Note: The program is not allowed to sort using any readily available Java sort libraries/functions. You must implement your own Heap sort method.*** See the Appendix for additional information.
4. The GUI should display the list of students (name, ID, associated course number, and total amount due) and the statistics for the total amount due. See Figure #2 – Screen Shot of Advanced CSE Registration System for an example of the GUI. You are required make the appearance of the interface similar (does not have to be exactly the same) to the screenshot shown below in Figure #2. When the user opens a file that contains students, all of the students should be displayed. When the user selects to sort the data, the display should be updated to display the sorted data. The text area should always show a current list of students (name and ID), associated course, and

total amount due. Allow the window to be big enough to accommodate all of your components.

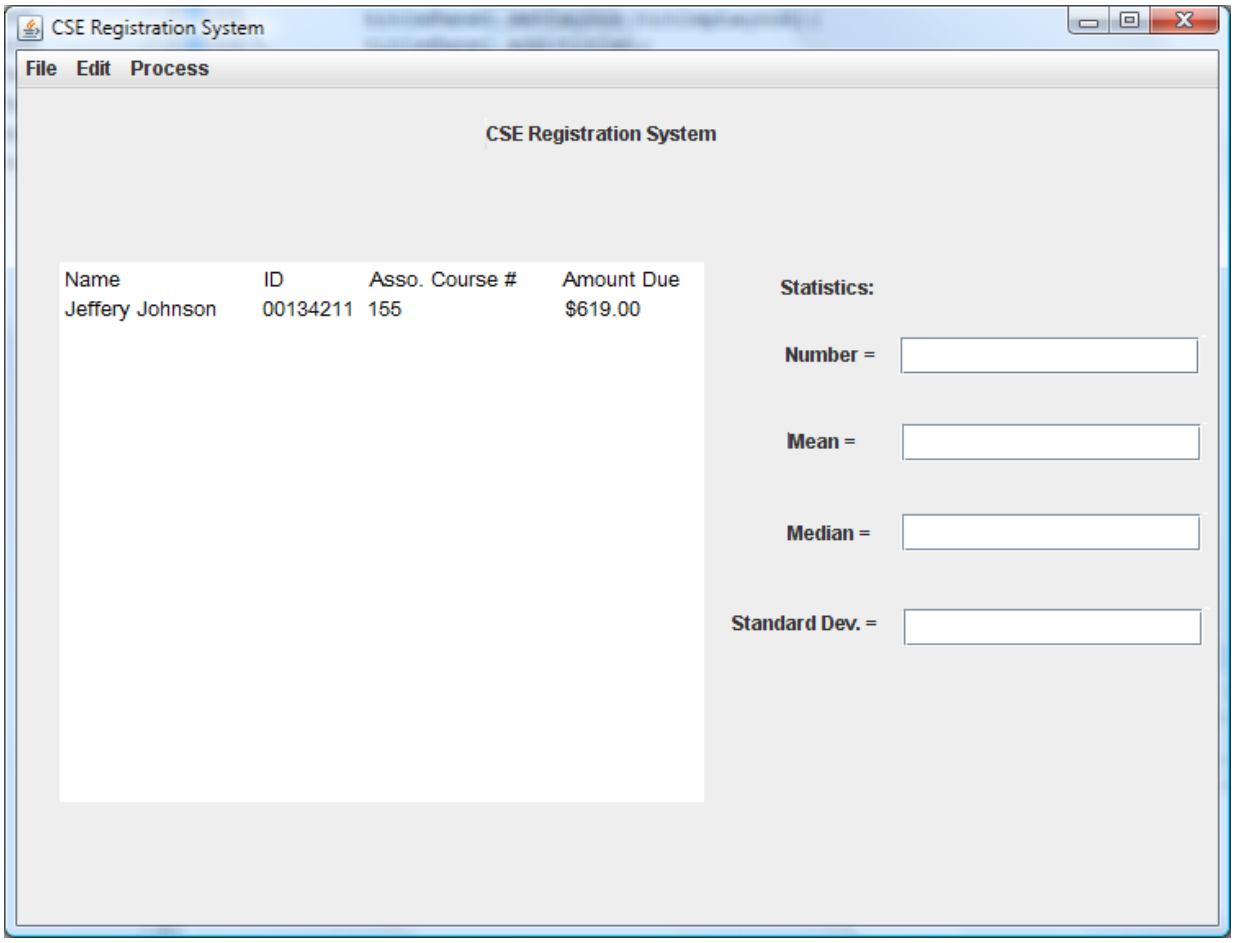

Figure #2 – Screenshot of Advanced CSE Registration System

5. Finally, the application should save the data. If the user opened a file to get the list of students, then when they select the "Save" menu item, the system should save the updated data to that file. If the user selected to create a new list of students and the first time they select the "Save" menu item, the system to respond as if the "Save As" button was selected. If the user selects the "Save As" button, the system should prompt the user with a save `JFileChooser` object to get the file to save the data. The saved file should be in the same format as the input file so that it can be opened within the program later. The example data file of students is new for this homework assignment. This new file of students has the students stored as Students objects with their associated Courses object. You must store your students as objects for this homework assignment. The example data file has thirteen (13) students currently in it.

3. Additional information:
   (a) For this homework, your system is not required to save or record the total amount due calculations for the students to a file.
   (b) Your program must handle exceptions and be robust.
   (c) For all other details except displaying the information, please refer to the Homework Assignment #4.

Supplied Solution Components

The Registration System is the main application for the program. You may use the partially completed classes from the previous four homework assignments. You may also use the solutions from the previous four homework assignments as long as you acknowledge the original author of the work appropriately. You must use the **`Students.java`** and **`Courses.java`** class files provided for this homework assignment on BlackBoard in order to read the new object based input data file provided for this assignment. The main (application) class for the program should be called "**`RegistrationSystem`**".

A new object based example data input file is provided for this assignment and is called **`students.dat`**. When reading the file, you will need to find out from the user how many records are stored in it. The example file on BlackBoard originally contains thirteen (13) student records. When reading this file, you will need to explicitly cast the objects read into Students objects to store them correctly.

Submission Procedure

This assignment is due **Monday, April 14, 2008 at the start of class (1:30 PM). Assignments five minutes late will not be accepted**. It is highly recommended you read the grading policy and grading guidelines on BlackBoard for a complete explanation of how assignments will be graded. Remember, your program should follow good programming style, include plenty of comments (including appropriate *javadocs*), and perform all of the functionality outlined above.

After completing the assignment, you must "handin" the following files online. Please hand in the files as individual files and not as a compressed/combined file.

1. Source files: **`RegistrationSystem.java`** and all other programmer-defined source files created/used for the program.
2. Compiled files: **`RegistrationSystem.class`** and all other compiled files required for the program.
3. Readme file: **`README.TXT`** (See Homework #1 for what should be included)
4. Testing file: **`TEST`** (See Homework #1 for what should be included. Be sure to include multiple tests and an explanation (test case) for each of the tests you run.)
5. Student Input File: **`students.dat`**

In addition, you must submit a *stapled* paper copy of your **`README`** file and **`TEST`** file. Both of these steps must be done by the start of the class on the day the assignment is due. Together with

your paper copy, you must attach a **coversheet**. (Please down load this coversheet from the BlackBoard course website, under the Homework Assignments link). This coversheet allows the grader to give comments and categorize the points for your homework.

Appendix

1. Heap Sort (a Java implementation can be found in your textbook, Chapter 11 or in the examples shown in class).
2. Average $\bar{x}$ of *n* values is defined as $\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n}$
3. Median: In a sorted data set, the median is the value of the data positioned at $i = n/2$ were i is rounded up if it is not an integer.
4. Standard deviation: $\sigma = \sqrt{\frac{\sum_{i=1}^{n} (x_i - \bar{x})^2}{n}}$, where x_i is the value of a particular instance, n is the number of instances.