CPSC 131: Introduction to Computer Programming II

Program 4: GUI for Course Display

1 Description of the Program

In this assignment, you are asked to implement a graphical application that displays a list of courses. Briefly, this GUI will take the inputs (i.e., course information), store them in an array list, and display course information in the output area. Figure 1 shows the layout of the CourseDisplay GUI.

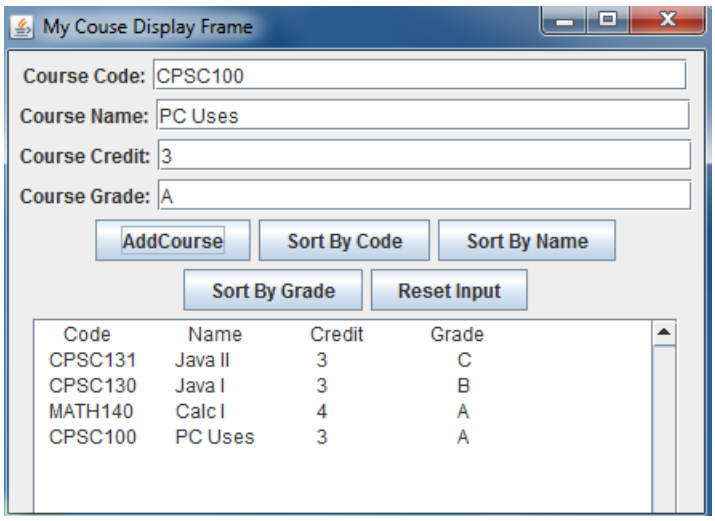

Figure 1: A screenshot of the CourseDisplay GUI.

2 Details of the program

You should have three Java files:

1. `CourseDisplayFrame` class that extends `JFrame` class (The main part of your program).
2. `CourseDisplayViewer` class, which contains the main method where a CourseDisplay object will be created, and the GUI will pop up (Suggested frame width of 450, and frame height of 400).
3. `Course` class, which is the class that models the `Course` objects (This file will be given to you).

In your `CourseDisplayFrame` class file, you should have the followings:

- Input area:
  - A text label and a text field (suggested width of 30) for course code;
  - A text label and a text field (suggested width of 30) for course name;
  - A text label and a text field (suggested width of 30) for course credit;
  - A text label and a text field (suggested width of 30) for course grade;
- Buttons:
  - One button (AddCourse) used for adding a course into your course list, and append this course in the output area;
  - One button (SortByCode) used for sorting the course list by course code. The soThe output should be displayed as in Figure 2.

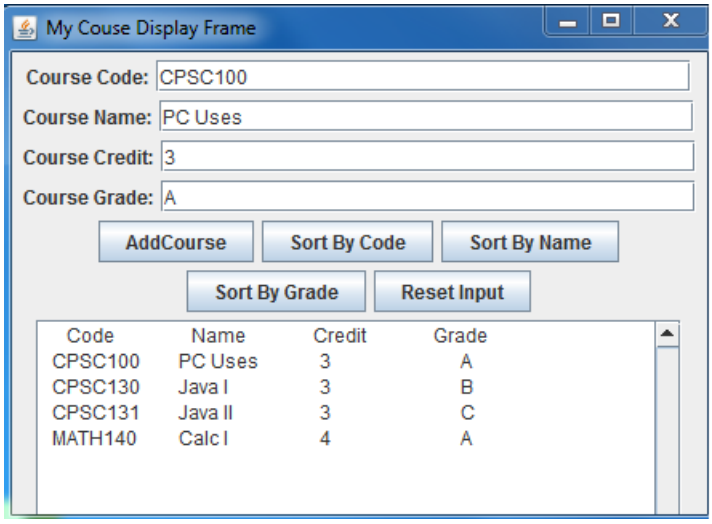

Figure 2: A screenshot of the sorted Courses by course code.

  - One button (SortByName) used for sorting the course list by course name. The output should be displayed as in Figure 3.
  - One button (SortByGrade) used for sorting the course list by course grade. The output should be displayed as in Figure 4.
  - One button (ResetInput) used for resetting all input fields;
- Output area:
  - One text area (suggested row of 10 and column of 35) that displays course information; You may initialize the text area by including the headings (Code, Name, Credit and Grade).

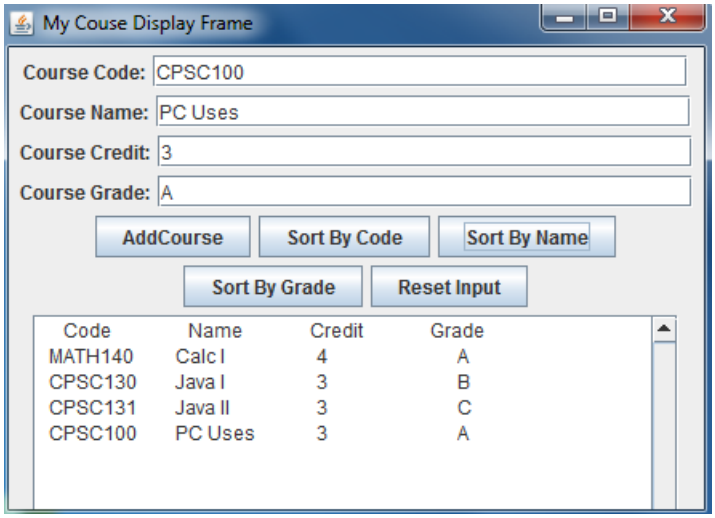

Figure 3: A screenshot of the sorted Courses by course name.

  - A scrollpane that includes the output text area;

- One `ArrayList<Course>` instance variable (e.g., `courseList`) to store the courses you add;
- Five inner `ActionListener` classes, with each of them implements its method of `actionPerformed`. Specifically,
  - `AddCourseListener`: read the inputs and create a course object, add the course object into the `courseList`, and append this into the output area;
  - `SortByCodeListener`: sort all courses in the list by course code, and display the sorted list in the output area.
  - `SortByNameListener`: sort all courses in the list by course name, and display the sorted list in the output area.
  - `SortByGradeListener`: sort all courses in the list by course grade, and display the sorted list in the output area.
  - `ResetInputListener`: reset all input fields;

  Important Note: The attached sample program "`PersonTester`" contains an examples of how to sort a list of people by `ID`, and `Date`. This should be useful for you to sorting courses by Code, Name and Grade in this program.

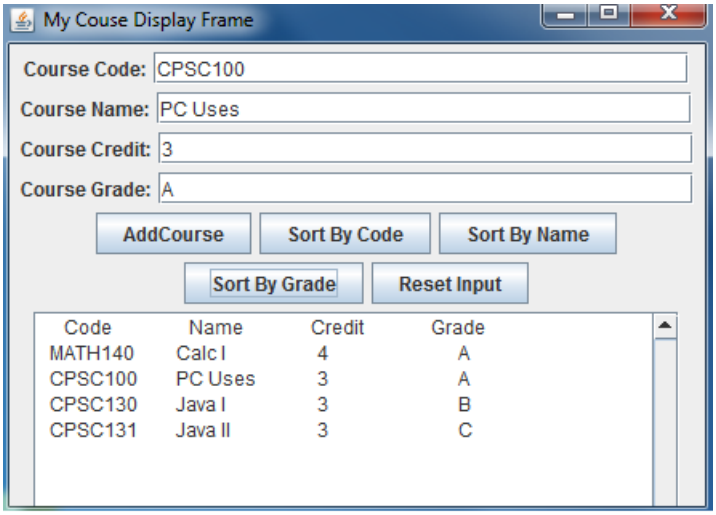

Figure 4: A screenshot of the sorted Courses by course grade.

3 Submission

Upload the following items on D2L dropbox, including:

1. The source code (`CourseDisplayFrame.java` and `CourseDisplayViewer.java`).
2. Screenshots of **four** program outputs (Similar to sample outputs shown in Figures 1-4). **Important**: Copy all screenshots in one word file (or save as pdf file)