Project #5

Computer Science 2334
Spring 2010

User Request:

"Expand on an Olympics Information System by Adding New Graphical Displays."

Milestones:

1	Create appropriate classes, complete with variables and methods, for the new views described below.	*20 points*
2	Create appropriate classes, complete with variables and methods, for additional controllers for the additional views as described below.	*20 points*
3	Revise as necessary the classes, including variables and methods, to handle the application data on Olympics as described in Project #4 under model.	*10 points*
4	Revise as necessary the classes, including variables and methods, of the model to allow the model to correctly interact with the controllers and the views.	*5 points*
5	Revise as necessary the classes, including variables and methods, for the views described in Project #4.	*10 points*
6	Revise as necessary the classes, including variables and methods, for the controllers as described in Project #4.	*10 points*

- ► Develop and use a proper design. *15 points*
- ► Use proper documentation and formatting. *10 points*

Description:

An important skill in software design is extending the work you have done previously. For this project you will build on Project #4 in order to add two new types of views to allow users to view Olympic information graphically. You will also have the opportunity to revise your code from Project #4 to ensure that it is implemented correctly. As with Project #4, this project will be organized around the model, view, controller (MVC) design pattern which gives us a way to organize code involving graphical user interfaces (GUIs). In particular, you will create one or more models to hold the application data and extend them to act as sources for communicating with views, several views to display and manipulate different aspects of the data, and one or more controllers to moderate between user gestures and the model(s) and views. For this program you should reuse some or all of the classes that you developed for your Project #4, although you are not required to do so. If your Project #4 was designed and implemented well, you should be able to use those classes with little or no modification. If your project #4 had significant design or implementation problems, you will need to significantly change that code as well as added new code for the new views.

Models, Views, and Controllers:

The models, views, and controllers described in Project #4 all need to be present and functioning in Project #5. See Project #4 for a description of those elements. In addition, two new view types will be added and the Home View will be modified by adding a second menu to its menu bar. These new and changed views are described below.

Views:

Home View:

To display the new views, the *Home View* will be modified to include a second menu – the display menu – in the menu bar. All functionality of this menu will be grayed out and inactive until appropriate data has been entered using the views from Project #4 or loaded using Open. Once this data is available, the options "Medal Count" and "Event Outcome" will become active. Under "Medal Count" there will be sub-options for "Gold," "Silver," "Bronze," and "Total."

Medal Count Views:

If the user chooses Medal Count→Gold, a Medal Count View will query the model(s) for all gold medals for all teams in the current Olympics. Note that your model(s) will not have a single method that can fulfill this query. Instead, this Medal Count View will need to build up the data it needs through a series of queries about the National Teams in the Olympics, the Athletes in the National Teams, the Participation Records of the Athletes, etc. This Medal Count View will then become visible and display a histogram that shows the number of gold medals won by each National Team, ordered from most to least. The histogram may be oriented either horizontally or vertically. The length of each bar will be proportional to the number of medals won by a given team and the team name (or an abbreviation of it) will appear along the baseline where that bar originates. The other end of the bar will be labeled with the number of gold medals. The entire view should have the title "Gold Medal Count by National Team." Note that if a gold medal is won by a National Team in a team event, that will count as one gold medal toward the medal count of that National Team, *not* one medal per participating team member. For example, if the USA takes the gold medal in the Four Man Bobsled competition, that counts as one gold medal toward the USA medal count, *not* four gold medals.

If the user chooses Medal Count→Silver, Medal Count→Bronze, or Medal Count→Total, a Medal Count View will behave in a similar way, except of course that the queries and histograms will be for silver, bronze, and total medals, respectively.

Event Outcome View:

If the user chooses Event Outcome, an *Event Outcome View* will present the user with a list of events in the current Olympics. When the user selects one of these events, this Event Outcome View will query the model(s) for data relevant to this event. In particular, it will determine which National Team won which medals for this event. It will then become visible and display a winner's podium showing which team won gold, silver, and bronze in that event. The flag of each corresponding National Team will be displayed above the appropriate riser on the podium and the National Team's name (or an abbreviation of the name) will be displayed below the appropriate riser. The entire window will have the title "<Event> Results" where <Event> is replaced by the name of the event (for example, "Women's Hockey Results").

Persistence of Display Views:

Note that both Medal Count Views and Event Outcome Views will persist until closed by the user. While they are open, they will not interfere with accessing other views, whether data entry views or other display views. So, it should be possible for a user to have open up to four different Medal Count View windows (one each for gold, silver, bronze, and total) and Event Outcome View windows up to the number of events in the present Olympics. If the user modifies data in the model(s) while a display view is open, that display view should update itself if the data relevant to it has been changed.

How to Complete this Project:

1. Create figures, on engineering paper or using drawing software, to show approximately what each view will contain. These figures do not need to exactly match the appearance of the final windows but should contain all major components and show their basic layout.

2. Revise your UML design from the lab session. Be sure to clearly write your name and the names of your group members and "Project #5" on the cover sheet. **Remember to not include any personally identifying information on your project other than on your cover sheet.** Make sure to keep a copy of your UML when you turn it in.

3. Create the classes and methods specified in your design, but do not put code in the methods. Add the required documentation to your classes and methods as specified in the documentation requirements posted on the class website. This is called "stubbing" your classes and methods.

4. Run your stubbed Java files through Javadoc as described in the Lab #2 slides. This will create a set of HTML files in a directory named "docs" under your project directory.

5. Submit your **view figures**, UML design, stub code, and Javadoc as your initial design. (See below for due dates and requirements regarding submission of paper and electronic copies of project components.)

6. Implement the design you have developed by coding each method you have defined as well as any others you have left out of your design. As you do this, make sure to modify and annotate the changes to your design on your UML and properly document all new code. A good approach to the implementation of your project is to follow the project's milestones in the order they have been supplied.

7. Test your program and fix any bugs.

8. Once you have completed the project and are ready to submit it for grading, create a new set of Javadoc files using Eclipse and inspect them to make sure your detailed design is properly documented in your source code. (Actually, it is a good practice to keep your Javadocs up to date as you develop your software so that they can be of use to you and your team members and to the instructor and/or TA if you come to office hours for help.)

9. Submit all parts of your completed project. (See below for due dates and requirements regarding submission of paper and electronic copies of project components.)

Extra Credit Features:

You may extend this project with more search features for an extra 5 points of credit. For example, you could think of ways to present the user with helpful information about the program, such as a context-dependent help system.

To receive the full five points of extra credit, your extended features must be novel (unique) and must involve effort in the design of the extra features and their integration into the project and the actual coding of the features. Also, you must indicate on your final UML design which portions of the design support the extra feature(s); and you must include a write-up of the feature(s) in your milestones.txt file. The write-up must indicate what each feature is, how it works, how it is unique, and the write-up must cite any outside resources used.

Due Dates and Notes:

An electronic copy of your revised design including **view figures** (optional in electronic version), UML (optional in electronic version), stub code, and detailed Javadoc are due on **Wednesday, April 29th**. Submit the project archive following the steps given in the submission instructions **by midnight**. Submit your **view figures on *engineering paper* or hardcopies made using drawing software**, revised UML design on *engineering paper* or a hardcopy made using UML layout software, and your cover page at the **beginning of lab on Thursday, April 30th**.

An electronic copy of the final version of the project is due on **Thursday, May 6th**. Submit the project archive following the steps given in the submission instructions **by midnight**. Submit your **final view figures on *engineering paper* or hardcopies made using drawing software**, final UML design on *engineering paper* or a hardcopy made using UML layout software, a hardcopy of the cover page for your project, and a hardcopy of the milestones.txt file at the **beginning of lecture on Friday, May 7th**.

You may write your program from scratch or may start from programs for which the source code is freely available on the web or through other sources (such as friends or student organizations). If you do not start from scratch, you must give a complete and accurate accounting of where all of your code came from and indicate which parts are original or changed, and which you got from which other source. Failure to give credit where credit is due is academic fraud and will be dealt with accordingly.

As noted in the syllabus, you are required to work on this programming assignment in a group of at least two people. It is your responsibility to find other group members and work with them. The group should turn in only one (1) hard copy and one (1) electronic copy of the assignment. Both the electronic and hard copies should contain the names and student ID numbers of all group members on the cover sheet. If your group composition changes during the course of working on this assignment (for example, a group of five splits into a group of two and a separate group of three), this must be clearly indicated in your cover sheet, including the names and student ID numbers of everyone involved and details of when the change occurred and who accomplished what before and after the change.

Each group member is required to contribute equally to each project, as far as is possible. You must thoroughly document which group members were involved in each part of the project. For example, if you have three functions in your program and one function was written by group member one, the second was written by group member two, and the third was written jointly and equally by group members three and four, your cover sheet must clearly indicate this division of labor. Giving improper credit to group members is academic misconduct and grounds for penalties in accordance with school policies.