CISC 3120 Fall 2012 Homework 7

Instructions

- This assignment **is due by midnight on Wednesday November 14th** and you will submit this online — instructions below.
- **Follow these submission instructions:**
  1. Create a ZIP archive from the `.java` files (and any image files if you include any) that make up your homework.
  2. Go to:

     `http://agents.sci.brooklyn.cuny.edu/parsons/cisc3120/`
  3. Login and submit your ZIP archive.
  4. You can submit as many times as you like before the deadline—only the final version of the homework will be graded.
  5. Failure to follow these instructions will result in points being taken away from your grade.

     The number of points will be in proportion to the extent to which you did not follow instructions.

1 Building an interface

The last homework had you design a user interface. Now you have to build it. The idea here is to be creative and explore the various interface components available in Java.

What I want to see here is the user interface only — not the underlying tasks. You don't have to implement zooming, sorting, annotating, and so on. All you have to do is to implement the interface components that will facilitate the user requesting that the 3 tasks be performed, i.e., moving from a main screen to each of the task screens and back to the main screen.

So, for example, you could decide to implement zoom, sort by date and annotate. The main screen of your interface could simply have three buttons on it (boring!), labeled zoom, sort and annotate. If the user clicks on zoom, then all you have to do for this part of the assignment is to generate a new screen that says something like the user will select an image to zoom here and then a third screen that says something like the zoomed image will be displayed here.

Your implementation should include:

1. A main screen

   Note that this screen should provide a method for going to each of the task screens (see below), a method for accessing the help screen (see below), and an album view of all 20 images in your sample database of photographs
2. A screen for each of the three tasks

   Note that this could be a variation of the main screen and/or it could involve a number of sub-screens (windows?) required to request information from the user (e.g., select image to zoom)
3. A help screen

   This should be easy to find and easy to follow, and should include a text description of how the user can (will be able to) execute each of the 3 tasks.

Overall, your implementation should include an example of at least three of the interface components that we covered in class:

1. Buttons
2. Textfields
3. Menus
4. Mouse buttons (ie taking some action that involves listening for `mouseMoved` or `mouseDragged`)
5. Radio buttons
6. Check boxes

2 Challenge problems

You will get additional credit for:

- The use of any of the interface items listed above beyond the three required items. For example, if your program uses buttons, menus, check boxes and radio buttons it has used four of teh interface items we covered in class, and so will get some additional credit for the fourth item.
- Additional screens beyond the five listed above. For example, an "About" screen that provides information about the author of the code would gain some extra credit.
- *Any* interfaces features that we dd not cover in class. For example, we didn't cover dialog boxes, so if you program includes a dialog box, then you will get additional credit.

As usual, your program should document any of the features that is includes beyond the basic requirements.

However, good interface design is not a matter of throwing in every conceivable kind of widget. As a result you can lose some of the (additional) credit if the additional elements you include end up detracting from the usability of the interface.

3 How I will mark your work

1. You will get credit for each of the things listed above that you have implemented in your program and which work as I have described them.
2. You will also get credit for the **way** in which you do this. That is programs that do not adhere to what I consider to be good programming style will not gain as much credit.

   You can get an idea of what I consider to be good style by looking at the code I have given you.
3. However, your program **must compile and run** for you to get any credit at all.

   (At this point in your computer science career it is unacceptable to submit programs that don't compile and run).

   Bear in mind that I will be using Java version 1.6 to assess whether your code compiles and runs.
4. Thus it is better for you to submit a program that works and contains less functionality than it is to submit a program with more functionality that does not work.
5. You can get full credit for this homework **without** solving the challenge parts.