CS2110 Homework 5
Graphical User Interface

Due via Collab on March 29, 2019 at 11:30pm

Collaboration/Plagiarism Policy: This assignment is **individual work**. Sending, receiving, posting, reading, viewing, comparing, or otherwise copying any part of course assignments or solutions is not allowed except when explicitly permitted in assignment instructions. This includes solutions from other students in the course, past or present. See the course syllabus for penalties for collaboration policy violations.

Learning Objectives

- Practice your Java programming.
- Experiment with Event-Driven Programming.
- Build a Graphical User Interface.

Instructions: This assignment mimics a possible internship assignment. As a software engineer, part of your job is to deduce requirements from general instructions and design an elegant solution. You will typically not be given starter code. Your client or manager may say "I want an iOS app that does X, Y, and Z" and you will be expected to ask the appropriate design questions so you can deliver what the customer had in mind effectively and efficiently.

Some of this assignment is intentionally ambiguous. In this case, instead of a client to pose questions to, some creative license is required, meaning that you will need to make some assumptions about how this software will best work. **Throughought this assignment, comment your code with any assumptions you make.** There are intentionally features in this assignment beyond the in-class material – it is expected you will use the Java documentation to look up APIs to allow you to execute these requirements. **Cite sources in the code comments.**

Photograph Viewer

The project owner wants a Graphical User Interface (GUI) front-end to the `PhotoLibrary` application that we've been building. To start development of this application, you have been assigned to develop a prototype of the album viewer GUI.

- Use the Classes you have already created in the previous homework assignments to start this new project.

- To work with image files, add a new field to the `Photograph` class named `imageFile`.
  - Instantiate this field in any constructors that accept the `filename` field. *e.g.*

    ```
    imageFile = new File(filename);
    ```

  - Add a getter and a setter for this new field.
- Add a new class, `PhotoViewer`, to define the GUI interface.
  - `PhotoViewer` should have one instance variable, `imageAlbum`, that is a `PhotoContainer`. *e.g.* `private PhotoContainer imageAlbum;`
  - Any form components needed to implement the requirements should also be created as instance variables.
  - Create a folder named `images` in the root of your Eclipse project and put 5 images in it. At least some of these images should be 1280 x 1024 or larger – typical sizes for photos taken by a modern camera. By locating your images folder here, Java will be able to find the images using the relative path below.
  - In the main method of the `PhotoViewer` class, write a test that creates a `PhotoContainer` containing 5 images and loads your prototype, like in the following example:

    ```
    public static void main(String[] args) {

        PhotoViewer myViewer = new PhotoViewer();

        // relative path for Macs/Linux:
        String imageDirectory =
           "images/";

        // relative path for PCs:
        String imageDirectory =
          "images\\";

        Photograph p1 =
          new Photograph("Caption", imageDirectory + "img.jpg", "2015-06-30", 5);
        // four more photographs like the line above

        myViewer.imageLibrary = new PhotoLibrary("Test Library", 1);

        myViewer.imageLibrary.addPhoto(p1);
        // four more photographs added like the line above

        Collections.sort(myViewer.imageAlbum.photos);

        javax.swing.SwingUtilities.invokeLater(() ->  myViewer.createAndShowGUI() );
    }
    ```

  - The window will require four main areas:
    - An area for displaying the current image
    - An area for a thumbnail images - one thumbnail image for each of the five photos in the album

    * An area for form controls
    * An area for rating the current image

  Layout of these spaces is up to you, the developer, but most of the window should display the current image, which should default to the first image in the collection.

- Thumbnail-sized images (having a width of 100 or 200 pixels) should display down the left or right side of the window or across the top or bottom of the window, one thumbnail for each image in the collection.
- In addition to the image, thumbnails should provide a caption that displays the image caption, as well as the date and rating.
- Clicking on a thumbnail image should cause that image to be displayed in the current image area.
- **Note** Using a `JLabel` component will allow you to add a thumbnail image as an `ImageIcon`. The `JLabel` component will also allow you to set a caption to display caption, date and rating information, as well as add a `MouseListener` object. View the Java docs on how to write a MouseListener:
  https://docs.oracle.com/javase/tutorial/uiswing/events/mouselistener.html.
- By default, order the thumbnails by date.
- Provide a control by which the thumbnails can be ordered by Date, Caption or Rating. Selecting a new sort order should update the list of displayed thumbnail images and reset the current image to the first image in the album.
- Provide **Previous** and **Next** buttons that, when clicked, advance the image in the main display to the previous or next image in the container, respectively.
- When the end of the container is reached, clicking the **Next** button should display the first image.
- When the first image is selected, clicking the **Previous** button should display the last image.
- Provide a rating feature with radio buttons labeled 1 through 5, respectively. When an image is displayed in the current image area, the corresponding rating radio button should be selected matching the rating for that `Photograph`. If another rating radio button is selected, the rating value for that `Photograph` should be updated and reflected in the list of thumbnail images.

Hint 1: The appearance of GUI applications are dependent on screen resolution. Laptops come in a wide variety of screen resolutions, and it is likely your grader may not have the same resolution. For this reason, it is wise to avoid fixed-size windows. Take advantage of the built-in Swing features (Layouts) to size your overall GUI window. Size images consistently to keep the image presentation uniform and avoid skewing or stretching the image. Don't forget to include a screenshot with your submission.

Hint 2: You will need to use multiple panels and multiple layouts to achieve the desired display.

Below is an example of one possible layout of the `PhotoViewer` GUI window. You do not need to match this layout exactly. Feel free to be creative, as long as you meet all of the requirements.

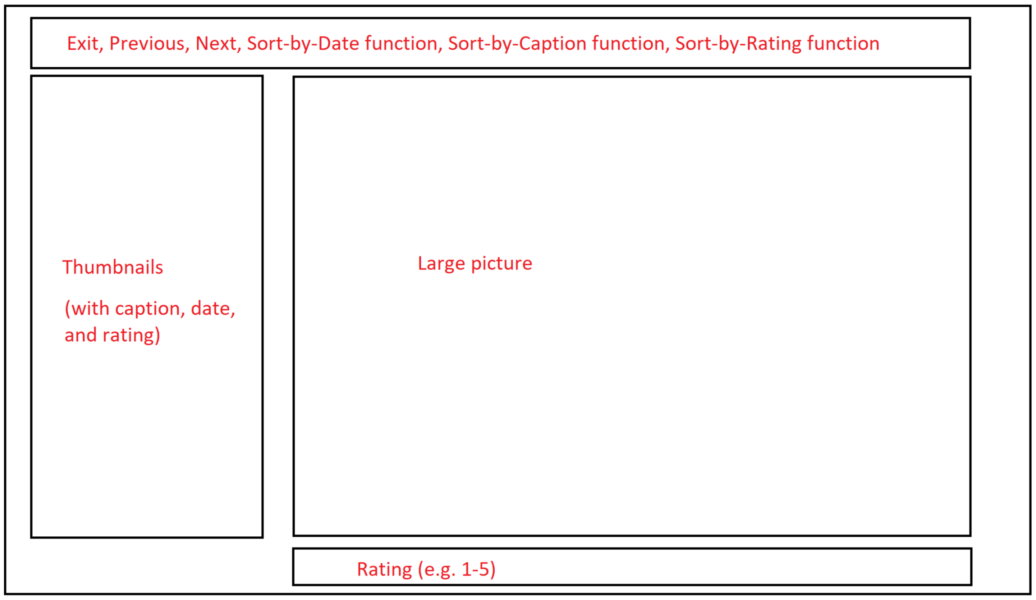
Exit, Previous, Next, Sort-by-Date function, Sort-by-Caption function, Sort-by-Rating function
Thumbnails
(with caption, date, and rating)
Large picture
Rating (e.g. 1-5)

Figure 1: Example layout of PhotoViewer GUI

Grading Rubric

- 55% Functionality
- 15% User interface
  - 5% User interface is intuitive
  - 5% User interface is aesthetically pleasing
  - 5% Overall interface structure is logical and natural
- 30% Design and readability (See CS2110 Coding Style Guide on Collab under Resources)
  - 10% Correct indentation
  - 10% Naming Conventions
  - 10% Well-commented
    - Comment at the head of each file (see "Style" section below)
    - Comment at the head of every major (and new) method
    - General in-line commenting as needed to highlight interesting logic
    - Comment at the head of each JUnit test (brief 1-line is fine)
    - Comment main-method testing code within your `main()` method
- Up to -20% late penalty as defined in the syllabus. The latest submission or resubmission will count as the official submission time.

Style: Follow the CS2110 Coding Style Guide, posted under Resources on Collab. This includes:

- Correct naming conventions, including appropriate camelCasing and TitleCasing.
- Comment each file, with a block at the top of the file denoting assignment information and comments for each field and method of your classes. You should also comment portions of your code that may be difficult to follow. We would like you to get into the habit of commenting your code. This adds to the readability of your code which contributes to "good quality" code.
- Use correct indentation. Eclipse makes this easy: select all code and choose "Correct Indentation" or Control-I (Windows/Linux) or Command-I (Mac).
- Do not put your classes into a package. (If you don't know what this means, don't worry about it.)
- If two methods share identical logic, you should factor that out into a separate method (a "helper" method).

Submission: You must submit on **Collab**. Submit `Photograph.java`, `PhotoLibrary.java`, `Album.java`, `PhotographContainer.java`, `CompareByRating.java`, `CompareByCaption.java`, and `PhotoViewer.java`. They must have those names exactly (including capitalization). Make sure you do not submit the `.class` files. Also include a screenshot of the window as it appears on your machine. This file should be named Screenshot.jpg, or Screenshot.png. In Windows, you can capture the active window by pressing alt+prtscreen buttons, which copies the active window to your clipboard. Then, open MS Paint and use ctrl+v to paste the image and save it. On a Mac, use Command+Shift+4 to draw around the active screen. This will immediately save an image to your desktop, which you can appropriately rename and include in the zip file.

To submit on Collab you need to zip up your files. If you are not sure how to do this you can follow the instructions below:

1. Right-click your `src` folder in Eclipse
2. Select `Export...`, `General`, `Archive File`
3. Check the `.java` files you want to submit
4. Browse to a save location you can find again
5. Finish
6. In a browser, go to `collab.its.virginia.edu`
7. Log in
8. Click the Assignments tab
9. Select Homework 5 - Graphical User Interface
10. Browse and select the `.zip` file you just created
11. Upload
12. Confirm