Sprint 4
Graphical User Interface

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

- Experience writing code using Event-Driven Programming.
- Design and develop a Graphical User Interface using Swing.
- Implement try/catch blocks to handle Exceptions.
- Develop features described by requirements rather than detailed specifications.

Grading Rubric

- 70% Functionality
- 30% Design and readability (See Coding Style Guide)
  - 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 main-method code within your `main()` method
- 10% User interface creativity (Extra Credit)

In this final sprint, you will be utilizing all the of the previously implemented logic in a graphical user interface. 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.**

Instructions: 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 viewer GUI using Swing. A graphical layout is provided, depicted in Figure 2, but you are encouraged to modify and extend the existing design for extra credit.

Photograph: You should modify your `Photograph` class to add the image data of the photograph.

- Fields:
  - `imageData` (**protected**)
    A `BufferedImage` object that contains the image's data.
- Methods:
  - `public boolean loadImageData(String filename)`
    Given a filename as a String, this method loads the Image data from the file and stores it into the `imageData` field. The `filename` is stored in the `filename` field. This method returns `true` on success and `false` on failure. Hint: reading a file from disk may throw an exception.
    An Image can be loaded from a String using the following code:
    `BufferedImage img = ImageIO.read(new File(filename));`
- Accessors (AKA getters): Provide a public method that returns a reference to the new field. You must use the standard naming convention for it.
- Mutators (AKA setters): Add a mutator for the new field.

PhotoViewer: Download and edit the `PhotoViewer` class, which defines and launches the GUI interface. It has an instance variable `imageLibrary` of type `PhotographContainer` that utilizes your code from Sprint 3. The following features must still be implemented:

- Event handlers for GUI elements:
  - `NavigationListener`
    An event handler for the "Next" and "Previous" buttons that skip between full-sized images displayed on the large `imageDisplayLabel`. When the "Next" button is clicked, this handler should display the next image in the `imageLibrary`'s photos. When the "Previous" button is clicked, this handler should display the previous image. If the current image is the first image, "Previous" should display the last image in the list; likewise, if the current image is the last image, "Next" should display the first image.
  - `RatingChangeListener`
    An event handler for the radio buttons associated with the ratings display. When a different rating radio button is selected by the user, the rating for the `Photograph` displayed in the `imageDisplayLabel` should be updated and the rating reflected in the list of thumbnail images.
  - `SortNavigationListener`
    An event handler for the sorting buttons, "Sort by Date," "Sort by Caption," and "Sort by Rating," which control the ordering of thumbnails. Selecting a new sort order should sort the `imageLibrary`'s list of `Photograph`s, update the list of displayed thumbnail images, and reset the currently displayed image to the first image in the `imageLibrary`'s photos.
- `private void drawThumbmails()`
  This method draws the thumbnail pane. It loops through all photographs in the `imageLibrary`'s photos list and displays the thumbnail and a label containing the photograph's `caption`, `dateTaken`, and `rating`. You must implement and add a `MouseListener` to the `thumbnailLabel` so that when the thumbnail is clicked by the user, `displayPhoto()` is invoked to display the full-size version of that thumbnail in the main viewer. View the Java docs on how to write a MouseListener: https://docs.oracle.com/javase/tutorial/uiswing/events/mouselistener.html.

- `private void displayPhoto(int position)`
  This method updates the main image being displayed. Specifically, it must call `imageDisplayLabel.setIcon()` to display the image at index `position` in `imageLibrary`'s photos. It also updates the ratings radio buttons to show that photograph's rating as selected.

- Testing the prototype using `main` method testing
  - 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, write a test invocation that creates a `PhotographContainer` containing 5 hard-coded images and loads your prototype GUI, like the example in Figure 1.

Hint 1: 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.

Hint 2: 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. Include a screenshot with your submission.

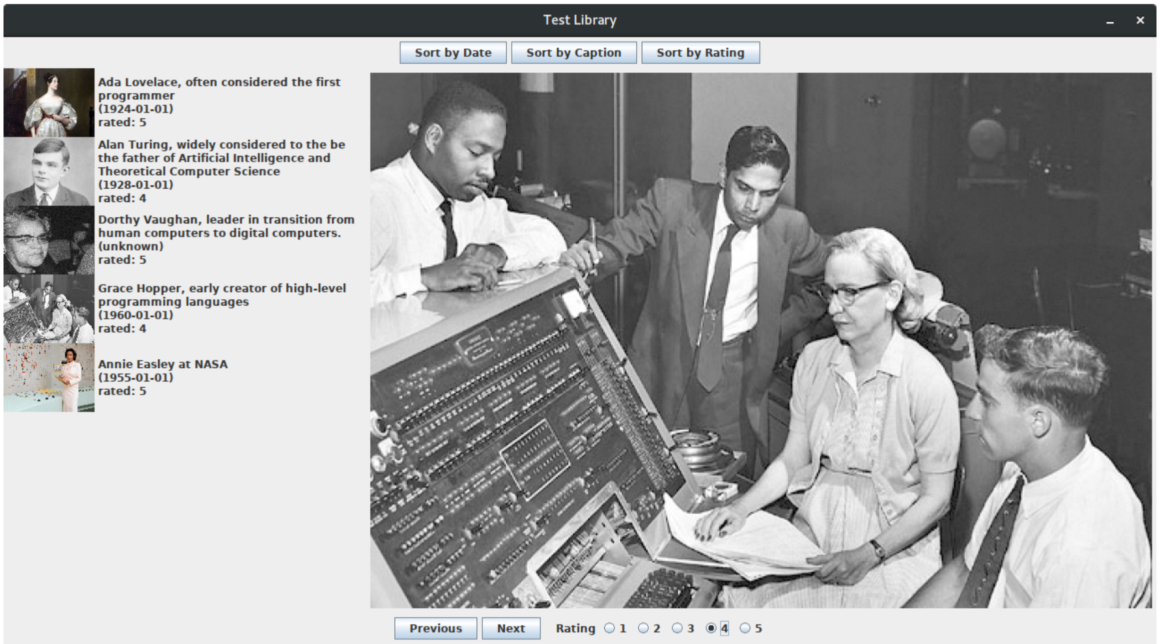
Full documentation and examples of Swing components are available at `https://docs.oracle.com/javase/tutorial/uiswing/components/componentlist.html`.

Style: Follow the Coding Style Guide. 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).

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

    // Instantiate the PhotoViewer Class
    PhotoViewer myViewer = new PhotoViewer();

    // the relative image directory (use one line from below)
    String imageDirectory = "images/";  // for Macs/Linux, or
    String imageDirectory = "images\\"; // for Windows

    Photograph p1 = new Photograph(imageDirectory + "img1.jpg",
       "caption", "2015-06-30", 5);
    // add four more photographs like the line above

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

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

    // Invoke and start the GUI
    javax.swing.SwingUtilities.invokeLater(()->myViewer.initialize());
    // or equivalently using an anonymous class
    javax.swing.SwingUtilities.invokeLater(new Runnable {
        public void run() {
            // This method is run to create and show the GUI
            myViewer.initialize();
        }
   });
}
```

Figure 1: An example main method to load images and invoke the GUI.



Ada Lovelace, often considered the first programmer
(1924-01-01)
rated: 5

Alan Turing, widely considered to the be the father of Artificial Intelligence and Theoretical Computer Science
(1928-01-01)
rated: 4

Dorthy Vaughan, leader in transition from human computers to digital computers.
(unknown)
rated: 5

Grace Hopper, early creator of high-level programming languages
(1960-01-01)
rated: 4

Annie Easley at NASA
(1955-01-01)
rated: 5

Rating ○ 1 ○ 2 ○ 3 ● 4 ○ 5

Figure 2: Example layout of PhotoViewer GUI, based on the provided code. Feel free to be creative in your design, as long as all components are still displayed.