Project X: E-Reader GUI

CSc 335 Fall/Spring 20XX
Due: XX/XX at X pm

For this project, you will create a Java program with a graphical user interface that mimics the basic functionality of an e-reader. This will provide practice using `JFrame`, various types of event listeners, and the Observer Design Pattern.

Overview

Your e-reader will provide a page-by-page view of the book *Alice's Adventures in Wonderland*. A file of the book's text will be provided (Alice.txt). The filename can be hard-coded in your program.

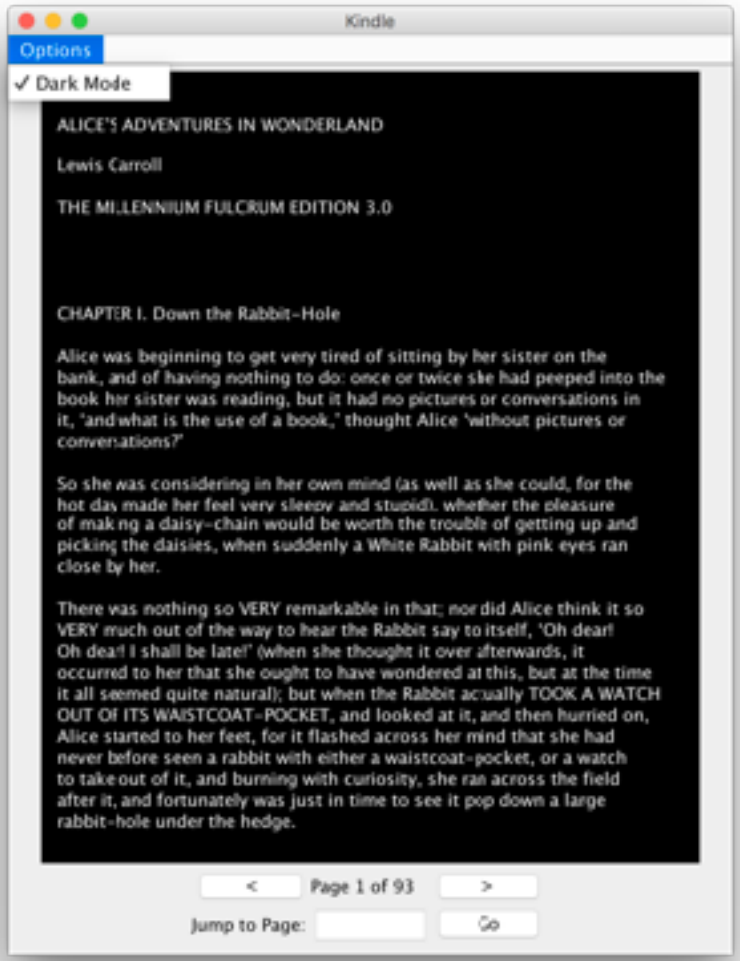
You will create a `JFrame` window to arrange the graphical elements. The book's text should appear in a large, centered `JTextArea` and should display 36 lines at a time. The user should not be able to edit the text.

By default, the text should be black on a white background. You must also provide an option to change to a 'Dark Mode', where the text appears white on a black background. The user should be able to toggle 'Dark Mode' on and off from the menu bar.

Note: You are not required to darken the other elements of the `JFrame`, only the main `JTextArea`.

Here are two screenshots of an example GUI:

Default



Dark Mode

The arrangement of the other components is up to you. You can arrange them the way they are laid out in the example, or you can find other creative ways to lay out the elements. However, all buttons, labels, text areas, and text fields must be functional and unobscured by the other elements.

The following two classes are required:
(Your class headings must match these exactly, though you may have any number of additional classes, if you wish.)

```
public class EReader extends Observable {
    . . .
```

`EReader` is the main model class that opens the file and processes the data. It should also include public methods for page turning, page jumping, current page, and total pages. It extends Observable, so make sure to use `setChanged()` and `notifyObservers()` when appropriate.

```
public class EReaderGUI extends JFrame implements Observer {
    . . .
```

`EReaderGUI` will have a main method that constructs a new instance of itself and launches the `JFrame` graphical user interface. The constructor should instantiate a new instance of `EReader` and should add itself as an observer of that instance. All updates to the text area or current page label should be handled solely by the `update()` method from `Observer`.

`EReaderGUI` must have the following elements:

- Two `JButtons` for turning pages (user can also use arrow keys on the keyboard to turn pages)
- A `JTextField` for jumping directly to a page
- A `JButton` for initializing the jump
- A `JLabel` for displaying the current page number, starting at page 1 (not zero)
- A `JMenuBar` with an 'Options' `JMenuItem` and a 'Dark Mode' `JCheckBoxMenuItem`

`EReaderGUI` must have the following four event listeners as private inner classes:
(Your class headings must match these exactly.)

```
private class ArrowKeyListener implements KeyListener {
    . . .
```

`ArrowKeyListener` will allow the user to use the left and right arrow keys on their keyboard to turn pages.

```
private class ButtonListener implements ActionListener {
    . . .
```

`ButtonListener` will allow the user to turn pages via the onscreen buttons and also jump to the page number typed in the `JTextField` when that button is pressed.

```
private class MenuItemListener implements ActionListener {
    . . .
```

`MenuItemListener` will allow the suer to toggle 'Dark Mode' on and off via the menu bar.

```
private class TextListener implements ActionListener {
    . . .
```

`TextListener` will allow the user to press enter after typing a page number in the `JTextField` to jump to a page. (After page jump, the `JTextField` should be cleared.

Error Handling

If the user tries to turn the page backward from the first page or forward from the last page, no error message is required. The program should just ignore the command and continue to display the current page.

When the user attempts to jump to a page, the program must display error messages under the following two conditions (error message must be unique to the error in question):

- The input text is not an integer
- The number entered is outside of the page range

The error messages should pop up in `JOptionPane` windows. Use the following code:

```
JOptionPane.showMessageDialog(null, "Your error message");
```

Turn In

Export the project from Eclipse as a .zip file and turn in on D2L with the following name format:
FirstnameLastname_EReader.zip (with your own first and last name)

Grading Criteria

(Grading categories below are all-or-nothing, unless otherwise noted)

___/ 5 Well-tested. For full credit, 90% or better code coverage on the non-GUI class(es) using JUnit tests and EclEmma

___/ 5 Handles errors as stated above

___/ 10 Employs the Observer Design Pattern using java.util.Observable and java.util.Observer
-5 if page or page count do not always update correctly
-5 if updates are handled somewhere other than in the Observer `update()` method

___/ 10 All required event listeners are utilized

___/ 10 Book text displays correctly
-5 for missing/extra pages
-5 for missing/extra lines or parts of lines on a page

___/ 10 'Dark Mode' is functional and accessible from the menu bar
-5 if `JCheckBoxMenuItem` check mark does not update
-5 if Dark Mode does not toggle on and off correctly

___/ 40 All other JComponents are functional
-10 for each missing JComponent (back/forward buttons count as one JComponent)
-5 for each JComponent that is present but non-functional or behaves differently than stated in the spec
-5 if any JComponent is obscured in any way (one-time deduction)

___/ 5 Consistent indentation, comments for each class and method, and a header comment at the top with your name and section

___/ 5 Project turned in on D2L with the correct naming format (FirstnameLastname_EReader.zip)