ITEC 2150
Spring 2015
IC – 6

In this program we are going to create an application that will allow you to view images and do some basic manipulation of these images. When you have completed this program you will have an application that can load an image, invert the colors and allow the user to modify the colors across the image.

This application can be done as single or multiple class program. The directions below will allow you to build a single class version of this application.

Directions:

1. Create a class file that extends Application. You will have to import javafx.application.Application.
2. Create a main method with the command Application.launch(args) as the only line. This is only required to run the program from Eclipse.
3. In the start method, takes an argument of type Stage, create an ObservableList<String> . This will be used to hold the choices for the combo box that we will add to our stage. The ObservableList is a generic collection of type FXCollections.observableArrayList. You should add your text at the time you declare the ObservableList.
4. Once you have the ObservableList created, create a ComboBox which will hold ObservableList. ComboBox also takes a type argument as a generic. Set the default value to "Load an Image". Disable the ComboBox until an image is loaded. This will be controlled through the OK button later.
5. Create a BorderPane to hold the entire application. Create another BorderPane to be used as the top container in the main BorderPane. Create a third BorderPane to be used to hold the buttons on the bottom of the screen.
6. Create a Label to hold a prompt for the user. Create a TextField to allow the user to enter the integer value for the color offset.
7. Add the ComboBox to the left of the top BorderPane. Add the Label to the center and the TextField to the right of this top BorderPane.
8. Create an ImageView control and add it to the center of the main BorderPane.
9. Next, create two buttons. An OK button which will be used to start all actions and an Exit button to close the program.
    a. Button okBtn = new Button("OK");
10. Once we create the buttons, we need to handle the button events. Button events can be handled in one of two ways. The first is to use an anonymous inner class. The next is to create a lambda expression
    a. okBtn.setOnAction(new EventHandler<ActionEvent>(){}); This must implement the method handle with a parameter of ActionEvent.
    b. okBtn.setOnAction((event) ->{});
11. Add the buttons to the bottom pane, I added mine to Center and Right

12. Create a Scene, set it to a default size. Scene scene = new Scene(border, 800, 600); The first argument in the scene is the Parent. I declared my central BorderPane as border.
13. Set the scene in the stage and then show the stage. You should have the opening screen with no image.
14. Now we need to create the methods to actually perform the work in this program.
    a. getFile() returns a String. This String will be the file name returned from this method.
        i. Create a FileChooser, this is the javafx and not the swing version. Next call the ShowOpenDialog with your stage as the argument. You will need to process the return value, a File, using the .toURI ().toString() methods. Later when we create the Image, we need to have a String URI.
        ii. Return the file name
    b. invertColors(Image) returns an Image.
        i. Create a WritableImage. Set its size in the constructor to the width and height of the image passed as an argument. You will have to cast the return of getWidth() and getHeight() to ints.
        ii. Create a PixelWriter to write the pixels to the new image. You will need to get this from the writable image. You cannot directly instantiate a PixelWriter.
        iii. Create a PixelReader. Again, you cannot instantiate it but will need to use the image passed in this method.
        iv. Create a nested loop to go from the width to the height of the image. In the loop use your PixelWriter to set the color. The color will come from the PixelReader at the outer loop and inner loop coordinates. This returns a Color on which you can call invert().
        v. Return the WritableImage created.
    c. adjustColors(Image, int). This method returns an image. The int is the offset parameter that we are going to use to adjust the image color.
        i. Again create and initialize a WritableImage, PixelWriter and PixelReader.
        ii. Create the nested loop.
        iii. In the nested loop, use the setArgb and getArgb methods. Argb is a 32 bit integer that represents the 24 bits of the colors and 8 bits for the Alpha channel. The Alpha channel controls transparency.
15. Finally, we need to implement the event handlers for the buttons.
    a. In the OK handler, you will need to handle the action for loading an image, inverting an image and adjusting the color.
        i. Load
            1. Get the file name, call getFile() and assign the return String to a variable

            2. Create a new Image, this can be an instance variable. Pass the file name as the argument to the constructor.
            3. Call setImage on the ImageView created above with the image as the argument.
            4. Set the updated ImageView in the center.
            5. Adjust the height of the main stage, image.getHeight() + the height of the top and bottom panes.
            6. Adjust the width of the main stage to handle the image width
            7. Enable the ComboBox.
        ii. Invert Colors
            1. Assign the return of invertColors(image) to an image, recommend a new Image variable so you will have the original image for further image manipulation
            2. Set the returned image in the ImageView control and reload it into center.
        iii. adjustColors
            1. Read the offset amount from the TextField, getValue(). This will return an Object which you can cast to a String.
            2. Convert it to an int
            3. Call adjustColor and do 1 and 2 from invertColors, at least part of them.
    b. In the Exit handler
        i. Close the stage
        ii. Platform.exit();

16. Recommended enhancements
    a. Save your image, a png is easiest
    b. Color, gray.
    c. Feel free to experiment.