Sabrina Bergsten

Dr. Rivas

Software Development 1

7 May, 2017

# Programming a Photo Editor

## Abstract

In this paper I'll cover the ins and outs of my final project. From the drawing board and initial project idea to the gradual development and eventual completion of what would be my final project. Building my photo editor took a lot of trial and error, but the final product was something incredibly rewarding! Utilizing the best of javafx allowed me to implement a pane to display my image on and buttons to give the program an interactive feel. The final outcome is a finished photo editor that allows users to increase and decrease the saturation, brightness, and hue of an image, and if need-be revert back to the image's original setting to start all over again.

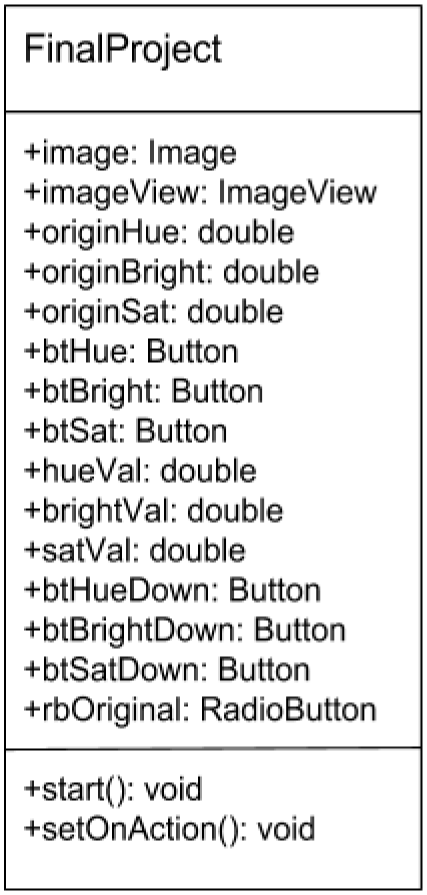
## Introduction

A famous quote once said, "If you want to know what someone fears losing, watch what they photograph." Photos and the way we present them are an integral way of how we, as people, express ourselves. Throughout history people have used iconic photographs and artistic renditions of them to symbolize entire time-periods, movements, and eras. For my final project I wanted to give my users the ability to create their own symbolic images to keep and share for a lifetime. Through this initial goal, I was inspired to create my final project, a java run Photo Editor. While this Photo Editor was not my primary idea for the project, throughout my time planning and coding, and with the help of many trials and errors, slowly it was developed and the final product is something I can really feel proud of.

## Detailed System Description

Throughout the development of my photo editor, the amount of imported java classes ended up being more than I ever thought would be necessary. The basic design of this project revolves around the thorough use of javafx. Using a number of prewritten methods provided by the javafx class and library, I was able to manipulate uploaded images and create an interactive program for users. The entire program is written within one primary class, named FinalProject which was an extension of the Application class. The first necessary step in the program was to override the start method in the Application class, so my public start method could run properly. Once the class and start method were correctly implemented, I now had to give the program and image somewhere to be displayed. Unlike most of our projects throughout the semester, this one would not be solely based in the terminal. To display my image and create a place for my multiple buttons to be placed, I had to introduce myself to the idea of a Pane. This pane acts a frame to hold the image uploaded by the program. The pane gave me, the programmer, flexibility to increase or decrease the padding and decide the dimensions, in pixels. To display anything onto the pane, within the Scene, I utilized the prewritten method, getChildren().add(). Whatever I decided should be displayed within the pane, I simply put in the parameters of the add() method attached to the end of getChildren(). Once I figured my way around the pane settings, it was time to upload a picture. To do that I created a new Image object named, image, and set it equal to the image or photograph I wanted to edit. Getting the image to display across the pane involved another step as well, one involving ImageView. ImageView is a node used for painting, editing, and displaying images within the Image class. I also used ImageView to set the width and height of my images. Next I began utilizing the ColorAdjust class. To do this I first initialized the class (line 54)  and began working with all it had to offer. This included setting variables like originHue, originBright, and originSat equal to the original hue, brightness, and saturation levels of the uploaded image. I next added six javafx supported buttons and one radio button to the pane. I went on to program the six buttons, with the help of the colorAdjust class, to slowly increase or decrease the brightness, hue, and saturation of the current image by .01. 3 of the 6 buttons increased each of those elements by .01 for each click, and the other 3 decreased those elements by .01. This gave the user complete control of how much they wanted to edit or unedit their photo. The final button given action through javafx was my one radio button. This radio button, with label “Original Photo” is set to be initially unselected, and upon selection it reverts the photo’s hue, saturation, and brightness to their original settings, as saved earlier in the “origin” variables. After the image is reverted back to its original settings, the radio button automatically returns to a state of unselected, so users can go back to their original pictures over and over again. After each button changes its element’s setting by .01 in either direction, I utilized imageView.setEffect(colorAdjust) to set the effect and upload it to the pane within my scene. This gives users a current view of their work as they go.

## UML Diagram



## Requirements

The physical requirements of this program involve being able to increase and decrease the brightness, saturation, and hue levels of an image. Another requirement is being able to utilize the radio button to revert back to the original image.

## Literary Survey

When it comes to my Photo Editor program, as much as I wanted to give users as much freedom as possible, I knew that my code still created limitations. Specifically with the image being used. My program takes in one image that I have saved in my prj2 folder as “image.JPG”, while this technically works and compiles fine, if I ever want users to be able to upload and edit their *own* images, the program will definitely need to be developed further. One goal I hope to achieve in the future regarding this program is the implementation of a java.util.Scanner import, to help users edit their own pictures. Using the Scanner import, I can prompt user for the title of their image, and as long as this image is saved in the correct folder, I will be able to save the image file name as a String variable, and use this variable with ImageView to upload and utilize the user's' photos throughout the rest of the program.

## User Manual

To use this photo editing program, all one must do is first clearly compile and run the java program within the terminal. Promptly after this initial step, a java pop-up window will occupy the screen. Within this window (or pane as it is referred to within the program) , there should be an image on the far left, along with 6 buttons and one radio button on the right. To begin editing the picture to one's desire all a user must do is click on each button. The labels will direct a user whether they are increasing or decreasing brightness, hue, and saturation, and the radio button allows users to revert back to the original unedited picture.

## Conclusion

Through the ups and downs of this project, I learned not to be completely (and even irrationally) devoted to one idea. Initially, as one can see in my milestone write up, this photo editor was not supposed to be anything close to what it ended up being like. I was set on the idea of manually collecting data from every pixel and all at once changing them from one color to another. This created so many complications, and as soon as I opened up to the idea that maybe another strategy might work better to edit photos using java, I came across javafx and almost all of my earlier problems had solutions in no time. Finishing this project also gave me incredibly valuable practice implementing prewritten methods from imported classes, and using java outside of the terminal. I’m very proud of my final project and cannot wait to look back on it as my programming skills develop throughout the next 3 years.

References/Bibliography

Liang, Y. Daniel. Introduction to Java programming with JBuilder 3. Upper Saddle River, NJ: Prentice Hall, 2000. Print.