**Project 2 - Phase \_\_1\_\_\_\_**

**Name: \_\_\_Jane Doe\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Email: \_\_\_janedoe@horizon.csueastbay.edu\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Section 1 Execution Instructions:**

*Download the Zip file. Unzip to a directory. Go to Eclipse->File->import. Choose “Existing Project into Workspace”. Choose the directory you unzipped. Run ImageApplication.*

***EXAMPLE \_\_\_YOU REPLACE***
**screenshot 1.2 = directory view of "temp" directory you unzipped file to showing the unziped files and directory structures**.

***EXAMPLE \_\_\_YOU REPLACE* **
**screenshot 1.3 = Eclipse running where you have opened up project file in "temp" directory.**

***EXAMPLE \_\_\_YOU REPLACE***
**screenshot 1.4 = Eclipse running the application - show screenshot of it running. If I must do something beyond simply hitting the "run" button, you need to give screenshots and step by step instructions.**

**Section 2 Code Description**

The code has several files- the main one is ImageApplication. I chose not to use the ImageFrame class but rather have one image shown at one time through ImageApplication. Pictures is a class I found on the net (credit given in the beginning of the class definition) that converts an image to a bufferedimage to use for image IO operations.. QuitDialog is included in the project, but is not used- I decided to go with simple close of file and termination of application. AboutDialog is slightly modified version of the original code. ImageData is an essential class that was modified to create ImageData objects in full color and not grayscale.

**Section 3 Testing**:

**section 3.1: loading image** -

***EXAMPLE \_\_\_YOU REPLACE*** 
screenshot 3.1a= pop-up file dialog box where you have selected the image you are going to load in your program.

***EXAMPLE \_\_\_YOU REPLACE*** 
screenshot 3.1b= the image being viewed in your application that was just loaded.

**section 3.2: saving image**
***MISSING\_\_\_YOU REPLACE-CAN BE SAME AS screenshot 3.1b***

screenshot 3.2a = screen shot of active image in your application you are going to save


screenshot 3.2b = screen shot of file save as dialog box where you have specified a DIFFERENT name not used yet for this image.

***EXAMPLE \_\_\_YOU REPLACE*** 

screenshot 3.2c = screen shot after you do an open on the image you just saved in 3.2b

**section 3.3: exit application**

screenshot 3.3a = screen shot showing application gone (really this is nothing) BUT, if the application has errors on closing I want to see this!!!



**section 3.4 and ON ---- you got the idea from above**

**Section 4 Comments**

Couple of things I need to mention: The way ROI works is that the user selects a rectangular area and then goes to Edit->ROI. If a rectangle is not selected, ROI is greyed out. This is achieved by SelectionArea()

ROI is modular. Ie, the user can select a ROI of ROI and undo the last such modification. See screen:



It is possible that the ROI is so small there is an empty area on the label left; however, SelectedArea was modified In such way that the user cannot draw a rectangle on that empty area but only on the actual ActiveImage (or, more precisely, an ImageIcon overlaying the ActiveImage).

One issue that the project spec did not mention was grayscale conversion, so both roi and image saving save the image exactly as it was before- in full color.