README

For assignment 6 we have tried using various swing elements for designing our UI such as JFrame, JPanel, JButton, JRadioButton, JLabel, JTextField, JComboBox etc. We have tried managing error handling workflows smoothly, for example we display a message dialog box to user if he tries to perform any operation before loading the image or if he tries saving a file without giving the correct extension name. **The UI is also responsive to key press events such as after typing a value in the text field for entering the value of brightness if we press enter the brightness of the image is set accordingly.** We have also supported adding multiple images to the panel in case of RGB Split. Attaching a few images for reference:  
  
1. Performing any operation before image is loaded:  
  
Graphical user interface, text, email

Description automatically generated

2. RGB Split, adds Red, green and blue images to the panel.  
  
Graphical user interface, application

Description automatically generated

CHANGES TO DESIGN:

**We have made no changes to our existing interfaces. The earlier interfaces ImageManipulationsController, ExtendedImageManipulationsController, ImageManipulationsModel and NewImageManipulationsModel remain as is.**

**1. Implemented Command Design Pattern.**

As for assignment 5, first of all we have used command design pattern to eliminate the big switch case in ImageManipulationsControllerImpl class that implements the ImageManipulationsController interface.

**2. New Interface and Abstraction**

We have created a new interface i.e. NewImageManipulationsModel that has the new methods such as (blur, dither, sharpen, greyscale and sepia). Have used abstraction and have put all our common code that is supported for both PPM and Conventional Images such as (PNG, JPG, BMP) in this Abstract class which inherits from the NewImageManipulationsModel interface.

**3. Segregation of classes**

Next we have created two classes, PPMImageManipulationsModel and ConventionalImageManipulationsModel extending from AbstractImageManipulationsModel which have the unique functionalities such as load and save which differ for PPM and other conventional image types.

**4. Factory Class**

Now that we have two separate model classes PPMImageManipulationsModel and ConventionalImageManipulations model, the factory class ImageManipulationsModelFactory returns the model object based on the file extension.

**5. Avoided Changes to existing design and implementations**

We avoided changes to our existing implementation of controller and the commands by extending the controller to a new class ExtendedImageManipulationsControllerImpl and writing our new changes to the controller there. We avoided making changes to our command interface by having a new command interface (NewImageManipulationsCmd) and extending the new commands from this interface which also receives the NewImageManipulationsModel object. Just made changes to the earlier implementations of load and save by moving the file read/write operations to an Util class, called by the controller.

**Classes and Interfaces:**

**Package Model:**

1. ImageManipulationsModel: This interface represents the methods for various image manipulations.

2. NewImageManipulationsModel: The new interface that contains the new image manipulation methods.

3. AbstractImageManipulationsModel: The abstract class containing common code for both PPM ad conventional Image models.

4. PPMImageManipulationsModel: Contains the unique image Manipulation methods specific for PPM.

5. ConventionalImageManipulationsModel: Contains the unique image Mamipulation methods specific for PNG, BMP or JPEG.

6. IImageManipulationsModelfactory: Factory interface for returning model interface based on file extension.

7. ImageManipulationsModelfactory: Factory class for returning model interface based on file extension.

**Package Control:**

1. Command Classes

2. ImageManipulationsController: The controller interface called to receive user inputs or inputs from a script file.

3. ImageManipulationsControllerImpl: The controller class that has the old model object and calls the old model methods.

4. ExtendedImageManipulationsController: The controller class that has the new model object and calls the new model methods.

5. ImageManipulationsCmd: The command interface for the old commands.

6. NewImageManipulationsCmd: The command interface for the new commands.

7. GUICommandCallbackController: GUI Controller that extends from the text based controller and performs the GUI operations making use of the model and view objects.

**Package Util:**

ImageUtil: Util class used by both the model and the controller.

Pixels: This class is used to store the properties of an image.

As for Assignment 6 we implemented the view. The view classes and interface are as described here.

**View:**

Features: An interface that would contain methods that would implement the functionalities of an image used by the controller.

HistogramChart: A class that implements IHistogram interface to generate the line chart for a any given image.

IHistogram: An interface that would contain methods that would implement the functionalities of a histogram chart.

IView: An interface that would contain methods for the view.

JFrameView: Main view class used to implement the view functionalities.

CITATIONS:

BMP - <https://people.math.sc.edu/Burkardt/data/bmp/blackbuck.bmp>

JPG - <https://commons.wikimedia.org/wiki/File:HDRI_Sample_Scene_Balls_%28JPEG-HDR%29.jpg>

PPM: <https://purepng.com/photo/456/animals-cute-little-duckling>

PNG: <https://northeastern.instructure.com/courses/143020/assignments/1760192> (from canvas)