README

This is an image processing application developed in the Java programming language using the MVC architecture. The application currently only supports terminal as its view container to display the messages of the image operations while running the code. The application can also parse a file input containing the list of commands and perform the operation asynchronous.

Model Package:

Our model represents an image. The model package of our MVC architecture contains the one main model class named *RGBImage.java* which implements the *RGBImageInterface.java* interface. This class contains all the commands as public methods which can be called from the controller class as per the required user input.The model contains several utility classes for each individual operation on the image.

The model contains the following packages:

1. enums: This stores all the enums data that are used in the model implementation of the model. It stores the name and value mapping along with some enums consisting of lambda methods which are used dynamically as when required by the model utility classes.
2. imageoperations: It contains multiple packages for handling different commands of the image operations like single input single output, single input multiple output, and multiple input single output. All the files in the packages deal with the image operations that are handled for the present image processing application.
3. RGBImage.java: It is the main model file which interacts with the controller, gets the command from controller and redirects the given command to above image operation utility classes.
4. RGBImageInterface.java: Interface which has the overview of all the methods that are expected to work for the given MVC architecture and user requirements.

Controller Package

The controller is made based on the command builder design pattern.
The controller contains the following packages:

1. imagecommads: This is similar to model class. It has all the helper utility commands divided into hierarchical packages and files. The main controller redirects the commands to these utility classes for passing the input to the model.
2. filehandling: The package deals with read and write operations on the files stored on the hard disk of the local machine containing the images. The package contains the cases for both imageIO and ppm file handling functionality.
3. RGBImageController.java: The main controller class which contains the command parsing and builder design. All the commands are mapped to a lookup table which

creates the objects of the controller command utility classes during runtime when received an input from the user.

Test Package

The test package has following structure-

Controller Package

It contains the test related to the controller and its integration test with the model.

1. ControllerModelIntegration.java: This file tests the controller and model combined as a whale. The input is passed from the user perspective and the output is checked against assertions.
2. filehandling: The package tests if the read and write operation works correctly for ppm, jpg, png, jpeg etc extensions. The ppm testing is done in the PPMUtilest.java and rest of the ImageIO extensions are checked in the ImaeIOTest.java.
3. ControllerModelMock.java: This file checks if the input passed to the model from the controller is correct or not by mocking the model class and creating another model class for keeping the log of data.

Model Package

It contains the test related to the main model class and its image operation classes in the respective individual packages.

The model test has following packages-

a. multiin: This checks for test cases where the model's multiple input methods work correctly. For now we only have Combine rgb which is checked by this package.
b. multiout: This checks for test cases where the model outputs multiple images after an operation is performed. For now we only have the split rgb method which splits the single image to r,g and b images.
c. singlein: This package checks for the rest of the functionalities of the image processing application like brightness, flip operation, greyscale, single component(Monochrome.java), sepia and sharpness, level-adjustment, color-correction, compression, histogram operations.
d. RGBImageTest.java: This file checks for the testing cases for the main model class calling all the image operations one after the another.

Assumptions

1. For compression, the values have been rounded off to two decimal places since no specific rounding off value has been provided..
2. For histogram, since the grid is optional, it has been ignored.

3. For color correction, if the input image belongs to a single channel(so the other two channels are zero), after color correcting the other two channels become non-zero since it has not been specified that they remain zero.
4. For split, the operation works successfully only with blur, sharpen, sepia, greyscale, color correction and level adjustment, as per the assignment requirement. However, the program will take split command input for other operations like brightness, flip, RGB filter but will not perform the split operation.

Design Changes

Controller Component-

- Refactoring existing command classes to support the split operation command-
    - The AbstractCommandIP class has been extended to support the input commands for both with and without split operation. The abstract class is inherited by other classes like brightness, flip, RGB Filter, sharpness but the split command operation is supported only in the command operation mentioned in the assignments. If the split is not passed as an input to the controller commands then the default split value is taken to be 100%. We have implemented the constraints in the abstract class to preserve the code reusability principle.
    - In the main controller class we have added new commands object in the existing hash map for command builder design pattern.
    - Made minor changes in the ColorTransformation command class to support the input of split operation arguments to preserve the code reusability principle.. If the split value is not provided then a default value of 100 is assumed for the same.
    - As per the requirement, RGBImageController.java now shows the error message whenever an exception or an unexpected event occurs which allows the program to behave gracefully which earlier just used to show the exception and close the application.

- Addition of more classes to support new features-
    - Added new command classes for Histogram plotting, Compression, Level Adjustment and Color Correction to support the new functionality that needs to be added as per the requirements of the assignment. All of these classes contain the command calls and command parsing for each of the individual operations in their respective classes.

- Migration of I/O operation package from model to controller package- In an MVC architecture controller is the one which interacts with the external environment which includes file handling operations. So, the implementation of read and write operations

of an image has been shifted to the controller package where it now has a file reader and writer which calls methods from different static methods as per the file path provided to it.

Model Component

- Refactoring of Model interface-
    - As the I/O related operations have been moved to the controller because it is the one that needs to take the input and interact with all of the other components outside the application so the save method in the interface is no longer required in the main image model interface and hence was removed for the same reason.
    - Further as per the requirements of the assignment, few additional features need to be supported by the application so new methods for histogram plotting, level-adjustment, color-correction and compression were added to be able to call it from the command classes.
    - Also as the split operation needs to be handled by the blur, sharpen, sepia and grayscale so a new method parameter was added to take the split percentage as input.
- Addition of new classes- In order to support the new functionality image operation classes were added like histogram, color correction, level adjustment and compression. This addition is with respect to the SOLID principles where single class represents single responsibility.
- Refactoring in Color Mapping enum: Added a new color component in the enum to use it during the histogram plotting operation. This helps to preserve the color data and use it to plot it on an image.
- Addition of Split Preview operation as a buffer class between main image model and operation classes: The split operation class acts as a buffer class which takes the input from the main model and as per the split percentage sends the subset of data to required image operation. The reason for opting this design principle is-
    - The buffer now acts as a middle person which makes sure that no changes would be required either in the main model class and rest of the image operation classes.So, this class allows us to keep the changes isolated, and follows ‘S’ part of SOLID principle strongly.
    - If required it can now support functionality for any other image operation as well without changing any line of code in the buffer class.

ImageProcessingApplication.java

- The main file now successfully parses the ‘-file’ flag and takes the script file input as command line arguments to support the requirements mentioned in the assignment.

Resources Package

The resources package contains the script, readme and useme files which can be used to run to generate the images as required for all the functionalities. The resources folder contains the source image along with the other images generated as a result upon the source image.

Citations

[1] Image used: Pixelation to represent endangered species counts | FlowingData