Collaging Project README

Overview

This project allows the user to create collages with different images and filters. The user can import different images that they would like in their collage and can then apply filters based on the images that they give or based on a preset condition (such as the red filter). The user can then save and load their current projects from their computer directly and will have access to a GUI so they can easily add more features to their collage design. It is important to note that the user can save their project at any layer and below that they choose in the GUI mode, but the Text UI mode will only allow the user to save their project with all of the layers given.

Currently, the user has access to an extensive list of features such as creating a project, adding a layer, setting the filter of a layer, opening a project, saving a project, and saving an image. They can apply these filters to the images on layers. This is only for the GUI version of the program. The text version allows the user to do pretty much the same operations as the GUI version but it does not allow the user to select layers to edit and save. Instead, the text version allows saving the image, but only the final image of all layers combined.

Dependencies

- Java 11 or higher JRE
- JUnit 4 for running the tests

Usage

The usage for this project is detailed in the file USEME.pdf, which can be found in the root directory of this project.

Implementation Details

Main

- CollageCreator: Runs the main class and allows for the collager program to run with the specified controller, view, and model. The collage program can run in GUI Mode where the user can interact with graphical components to use the program, Text Mode where the user can type out commands to the console to use the program, and it can be used to run a script file.
    - main(String[] args): runs the program.
        - GUI Mode is used when there are no args given to main
        - Text Mode is used when the first arg given to main is “-text”
        - A script file can be run when the first arg given to main is “-file” and the second arg is the filepath to the script file
        - This information can be displayed when the first arg is “--help”

Model

- Collager: main model interface that creates collages with images and layers.
    - createProject(int height, int width): creates a new project given the height and width of the canvas.
    - addLayer(String name): adds a layer to the project with the specified name. The layer starts with an image by default.
    - loadProject(String collagerContents): loads a collager project given a String that represents the contents of a collager file.
    - setFilter(String layerName, String filterName): sets the filter of a given layer.
    - addImage(String layerName, String fileContents, int startY, int startX): adds an image to the current layer given the layer name, the file contents of the image, and the starting coordinates to add the image at.
    - addImage(String layerName, Image img, int startY, int startX): adds an image to a layer of the collage.
    - getLayerNames(): gets a list of the layer names in the collager.
    - getFinalImage(): gets the final image or the combination of all the filters and layers.
    - getImageAtLayer(String layername): gets the image at the specified layer, the combination of all the filters and layers up to that point.
    - getHeight(): gets the height of the project.
    - getWidth(): gets the width of the project.
    - getLayers(): gets the list of layers in the collager.

    - createJavaImage(Image img, int bufferedImageType): converts an image of the type in the collage to a Java BufferedImage type.
    - getFilterNames(): Gets the filter names available in this Collager as an array.
- RGBACollager: an implementation of Collager that uses RGBALayers to create the collage program.
    - makeImageAt(int layerIndex): a helper method that allows code reuse between the getFinalImage() and getImageAtLayer(String layername) functions.
    - createJavaImage(Image img, int bufferedImageType): converts an image of the type in the collage to a Java BufferedImage type. This method only works with the “BufferedImage.TYPE_INT_ARGB” and “BufferedImage.TYPE_INT_RGB” types of BufferedImages.
    - The remaining methods are implemented as specified

- Image: an interface that represents a collection of pixels to make an image.
    - getPixels(): provides pixels from the image as a 2D array
    - combine(Image that, int startY, int startX): combines two images and returns the resulting image
    - applyFilter(Filter filter): applies filter to image and returns the resulting image.
    - applyTwoFilter(TwoFilter filter, Image other): applies filter with both images to get the resulting image.
    - getHeight(): returns height of image
    - getWidth(): returns width of image
- RGBALayerImage: an implementation of Image that uses RGBA pixels
    - All methods are implemented as specified

- Layer: Represents a combination of images under a single layer.
    - combine(Image that, int startY, int startX): combines a layer and image.
    - combine(Layer that, int startY, int startX): combines two layers.
    - updateFilter(String filterName): updates the filter for the layer given a String that matches a filter in the model. If the filter exists, the layer's filter is updated to that filter.
    - applyFilter(): applies the layer’s filter on its image.
    - applyTwoFilter(Image that): applies the filter to the layer with the image.
    - applyTwoFilter(Layer that): applies the filter to the layer with another layer.
    - getLayerName(): returns the layer’s name.
    - getFilterName(): returns the layer’s filter name that will be applied.
    - getImage(): returns a copy of the layer’s image.

- RGBALayer: an implementation of Layer that stores images as RGBALayerImages
    - updateFilter(String filterName): updates the filter for the layer given a String that matches a filter in the model. If the filter exists, the layer's filter is updated to that filter. This implementation uses the filters in the FilterName enum.
    - The remaining methods are implemented as specified

- Pixel: Represents different pixels in an image.
    - asRGBA: returns a list containing the pixel's RGBA values.
    - asRGB(): returns a list containing the pixel's RGB values (alpha-adjusted).
    - combine(Pixel that): combines two pixels together.
    - getMax(): returns the Max value of a Pixel.
- RGBAPixel: an implementation of Pixel that defines pixels using their red, green, blue, and alpha components. RGBA values are converted to have a max of 255.
    - All methods are implemented as specified

- Filter: defines how a layer's image should be changed when the filter is applied
    - apply(Image img): applies a filter to an image.
- NormalFilter: The filter that returns the image as it was in the beginning.
- RGBAColorFilter: an abstract class that implements Filter, which defines generally how a filter should be applied to an image.
    - apply(Image img): applies a filter to an image.
    - protected abstract createResultPixel(Pixel pixel): defines how the resulting pixels from the filter application should be created.
- RGBABlueFilter: a subclass of RGBAColorFilter that isolates the blue channel of pixels.
- RGBARedFilter: a subclass of RGBAColorFilter that isolates the red channel of pixels.
- RGBAGreenFilter: a subclass of RGBAColorFilter that isolates the green channel of pixels.
- RGBABrightenDarken: an abstract class that extends RGBAColorFilter to define how general brightening and darkening filters should create pixels.
    - protected createResultPixel(Pixel pixel): defines how the brightening and darkening filters create pixels by adding or subtracting a value to each pixel.
    - protected abstract createDifference(int red, int green, int blue): defines how the value to add or subtract from a pixel is computed
- RGBABrightenLuma: a subclass of RGBABrightenDarken that adds the combined luma value to brighten the image.

    - protected createDifference(int red, int green, int blue): calculates the luma value to add
- RGBADarkenLuma: a subclass of RGBABrightenDarken that subtracts the combined luma value to darken the image.
    - protected createDifference(int red, int green, int blue): calculates the luma value to subtract
- RGBADarkenIntensity: a subclass of RGBABrightenDarken that subtracts the intensity value to darken the image.
    - protected createDifference(int red, int green, int blue): calculates the intensity value to subtract
- RGBABrightenIntensity: a subclass of RGBABrightenDarken that adds the intensity value to brighten the image.
    - protected createDifference(int red, int green, int blue): calculates the intensity value to add
- RGBADarkenMax: a subclass of RGBABrightenDarken that subtracts the maximum RGB value to darken the image.
    - protected createDifference(int red, int green, int blue): finds the maximum value to subtract
- RGBALightenMax: a subclass of RGBABrightenDarken that adds the maximum RGB value to brighten the image.
    - protected createDifference(int red, int green, int blue): finds the maximum value to add
- TwoFilter: an interface that extends Filter was made to allow applying filters on the current image and the image below it. If there is no second image provided, it is assumed to be the default image of RGBA (255, 255, 255, 0).
    - apply(Image img1, Image img2): applies the filter on img1 (top image) and img2 (bottom image)
- RGBAColorTwoFilter: an abstract class that implements TwoFilter and extends RGBAColorFilter to allow code reuse for TwoFilters.
    - protected abstract createResultPixel(Pixel pixel1, Pixel pixel2): creates the resultant pixel for a TwoFilter
    - protected createResultPixel(Pixel pixel): calls the createResultPixel for two pixels, using the default pixel
- RGBAScreen: a subclass of RGBAColorTwoFilter that brightens an image by taking the complement of the product of the complements of the HSL lightness values of the pixels.
- RGBAMultiply: a subclass of RGBAColorTwoFilter that darkens an image by multiplying the HSL lightness value of the pixels.
- RGBADifference: a subclass of RGBAColorTwoFilter that inverts an image by taking the absolute value of the difference of the RGB values of the two pixels.

- FilterName: Enum that holds the filters and their corresponding names.
    - getFilter(): gets the filter object associated with the enum object
    - getName(): gets the name associated with the enum object
    - getTwoFilter(): returns a TwoFilter object if the filter is a TwoFilter, and otherwise throws an exception
- FilterType: Enum that holds the type of filters available

- RepresentationConverterUtil: Contains utility methods to read a PPM image from a file, read a project from a file, save an image to a file, and save a project to a file.
    - convertRGBtoHSL(double red, double green, double blue): converts an RGB representation in the range 0-255 into an HSL representation.
    - convertHSLtoRGB(double hue, double saturation, double lightness): converts an HSL representation into an RGB representation where each component is in the range 0-255.
    - convertFn(double hue, double saturation, double lightness, int n): helper method that performs the translation from the HSL polygonal model to the more familiar RGB model

Controller

- CollageController: an interface that defines how the Collager program should run, and manages the view and model.
    - runCollage(): runs the Collage program for the user to interact
    - saveProject(String filepath): saves the current project given the file path to save it at.
    - saveImage(String filename): saves the image at the specified file path.
    - saveImage(String filename, String layername): saves the image at the specified file path.
    - addImage(String layerName, String filepath, int startY, int startX): adds an image to the current layer given the layer name, the filepath of the image, and the starting coordinates to add the image at. If the image is not a PNG or JPEG image, it is assumed to be a PPM image.
    - loadProject(String filepath): loads a project from the given filepath.
    - getLayerNames(): Gets the layer names according to the model and returns the names as a list of strings.
    - setFilter(String layername, String filtername): sets the layer's filter in the model
    - addLayer(String name): adds a layer from the model

    - createProject(int height, int width): creates a project according to the model.
    - getFinalImage(): gets the final image according to the model.
    - getImageAt(String layername): gets the image at the given layer according to the model.
    - getFilterNames(): gets the names of the filters in the model as an array of Strings.
- RGBAController: implementation of CollagerController and an abstract class for a controller that uses the ImageUtil class to do image and project file I/O. This class includes all methods from the CollagerController except runCollage().
- RGBACollageController: extension of RGBAController that takes in a Readable to execute commands for the program
    - private renderExceptionMessage: helper method to limit code duplication when rendering messages caused by exceptions
    - runCollage(): runs the collager program and takes command inputs from the user/script.
- RGBAGUICollageController: extension of RGBAController that acts as a controller for the collager program for a GUI. This controller uses a GUIView and adds the controller (this object) to the view upon running the program.
    - runCollage(): runs the program with the GUI by adding the controller to the GUIView, which then displays the GUI.

- ImageUtil: A class that contains utility methods to read and write images and project files. It can read and write from PPM, JPEG, and PNG files.
    - readPPM(String filename): reads an image file in the PPM format and returns its contents as a string.
    - readCollage(String filename, Collager collager): reads a collager file given the filename of the collager file and the resulting Collager to open the project with.
    - saveCollage(String filepath, int height, int width, List<Layer> layers): saves a collager project to the given filepath, given the height and width of the images, and given the list of layers in the collager project.
    - saveImage(String filepath, Image img): saves the given image to the given file path in PPM format.
    - saveImage(String filepath, BufferedImage img): saves the given image to the given file path. Only works for PNG and JPEG file paths.
    - readJPEGPNG(String filename): reads a JPEG or PNG file to a buffered image.

View

- CollagerView: an interface that provides a general user view of the Collager program
    - renderMessage(String message): renders a given message to the data output in the implementation.
- CollagerTextView: implementation of CollagerView that displays text to the user through the console or an Appendable if provided upon object instantiation
- GUIView: an interface that extends CollagerView for use with a GUI.
    - addController(CollagerController controller): adds the controller to the GUI, which must happen before the GUI is displayed. Allows GUI components to interact with the model through the controller. Once this method is called and a valid controller is added, the GUI displays.
- CollagerGUIView: an implementation GUIView that also acts as an ActionListener, ListSelectionListener, and JFrame.
    - actionPerformed(ActionEvent e): listens for button pushes and calls the controller appropriately to execute commands given by the user
    - valueChanged(ListSelectionEvent e): listens for when a new layer is selected to be displayed and calls the controller to get the new image to display

Contact

Arnav Mishra: mishra.arn@northeastern.edu

Evan Lombardo: lombardo.e@northeastern.edu

Citations

The image used for this project is provided by: https://filesamples.com/formats/ppm