CS105 Exemption Project 2013

Name: Andrew Gough

Registration Number: 201237950

Course: Physics

GitHub Repository: https://github.com/andrew-gough/1stYearExemptionProject

Except where explicitly marked to the contrary by quotation or reference, what follows is all my own work. I have not knowingly allowed others to copy my work.

Andrew Gough Signature: Date: 22/04/2013

A Brief explanation of the original imageViewer System:

DarkerFilter – This is a subclass of Filter and iterates through all pixels in the input OFImage and calls the Color.darken() method on of them before replacing them with the new variant – which results in an overall darker picture.

Edge Filter – This is a subclass of Filter and iterates through all pixels and it highlights each pixel in accordance to their surrounding pixels – if the surrounding pixel is a much different colour it will highlight it a large amount and therefore highlighting the edges between different elements in a picture

Filter – This is an abstract class which is used for all filters to be subclasses of – Filter has a constructor which allows the user to supply the name of the Filter and a getName() method which allows for getting the constructed name. Filter also has an abstract Method apply(OFImage) which is implemented in all subclasses of Filter and is called to apply the filter to an image.

FishEyeFilter – This is a subclass of Filter and though use of the Math class (specifically the trig functions) the FishEyeFilter serves to increase the density of the original pixels around the edges of the picture and to decrease the density of the original pixels in the center – creating the Fish Eye Effect

GrayScaleFilter – This is a subclass of Filter and iterates through all pixels – It first finds the average of the RGB values of each pixel and then replaces the current pixel with a new Color object which is constructed using the average value for that pixel.

ImageDriver –This is the Driver class of the program – it contains the Public static void main(String[]) method and so execution of the program begins here – it creates an instance of ImageViewer which forms the main framework to the program

ImageFileManager – This is a class which has static methods to serve two purposes – to convert File Objects into OFImage Objects in the method loadImage(File) and to save an image to backing storage in the method saveImage(OFImage,File)

ImagePanel – This is a subclass of JComponent and serves as an implemented JComponent which is intended specifically to provide a canvas to display an image on – The setImage(OFImage) method allows for the ImagePanel to display an image while clearImage() provides a way for the ImagePanel to clear its current viewing area with gray-space

Image Viewer –This class is where most of the logic and the GUI is performed – all parts of the original GUI originate from this class and so is the “control” class as it decides which other classes to use in accordance to the user choices. The GUI in ImageViewer is built by the makeFrame() method and then the makeMenu(Frame) method – for the JFrame and JMenu container components respectively.

InvertFilter –This is a subclass of Filter which iterates through all pixels in the image and gets their Red,Green and Blue values, it then replaces the currentPixel with a new Color which is made using 255-Red,Green and Blue values respectively, which results in an inverted Image

Lighter Filter –This is a subclass of Filter and iterates through all pixels in the input OFImage and calls the Color.brighter() method on of them before replacing them with the new variant – which results in an overall lighter picture.

Mirror Filter – This is a subclass of Filter and iterates through the left half of a OFImage and stores the current pixel colour in a field and then sets the current pixel to the corresponding pixel from the right half and then sets the corresponding pixel from the right half to the stored pixel colour – this results in a Filter which mirrors the OFImage horizontally

OFImage – Extends BufferedImage and provides a two methods – setPixel(int, int, Color) and getPixel(int,int) which wrap around the two BufferedImage Methods setRGB(int,int,Color) and getRGB(int,int) respectably

PixelizeFilter – This is a subclass of Filter and iterates through a OFImage in steps of 5pixels at a time for each and then stores the colour value of the pixel at the upper left hand side of that 5\*5 pixel square. It then iterates through this 5\*5 pixel square and if the current pixel that is selected is within the image, copies the stored colour to the current pixel.

SmoothFilter – This is a subclass of Filter and iterates through an OFImage and calculates the average of the 8 surrounding pixels and then sets the current pixel value to this found average – having the effect of “smoothing” the image out.

SolarizeFilter – This is a subclass of Filter and iterates through an OFImage and stores the current Pixel colour in a variable, it then checks to see if the red value is less than 127 and if it is – takes the red value away from 255 and saves it in a variable – it then repeats this for green and blue and then sets the current Pixel to the colour corresponding to the stored red, green and blue values.

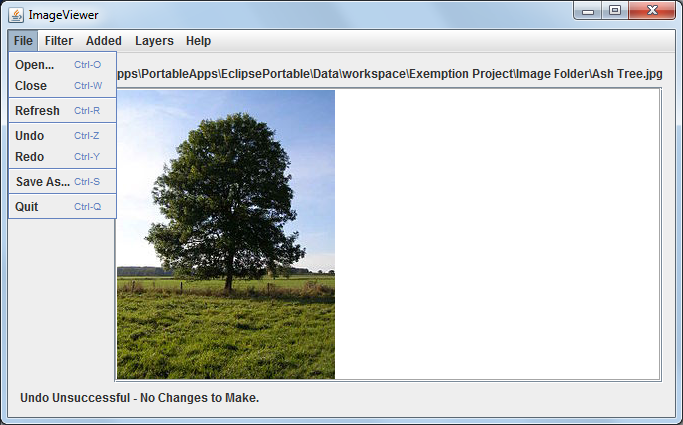
ThresholdFilter – This is a subclass of Filter and iterates through an OFImage and stores the current pixel colour in a variable – It then calculates the average value of the Red, green and blue values of the pixel and then sets the current pixel colour to black if this value if less than or equal to 85, to grey if less or equal to 170 and to white otherwise.

High Level Summary of the work that you have done:

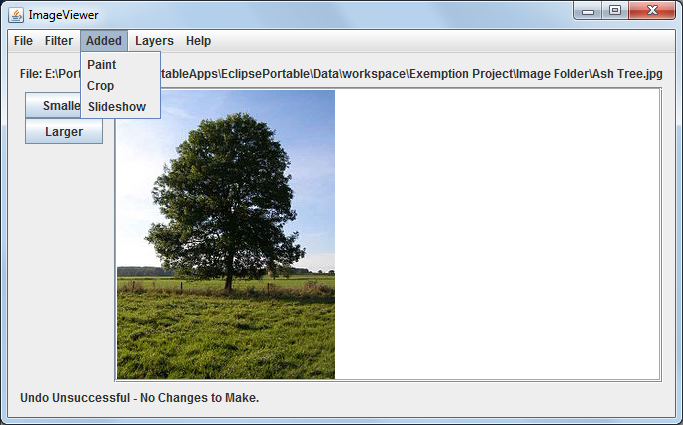
I have achieved 7 Main functions in my additions to the ImageViewer program.

Undo/Redo:

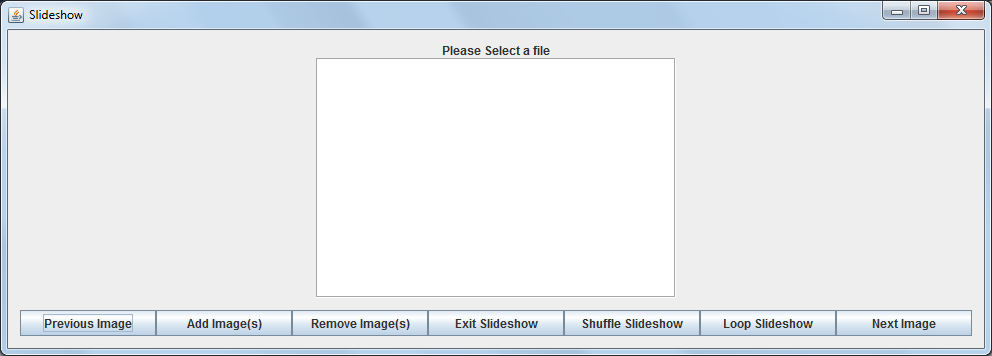
The Undo/Redo Functions do exactly as they are named – they undo changes to the original Image and redo changes to the original Image respectively:

The GUI for the Undo/Redo Functions is as follows: 

Slideshow Function:

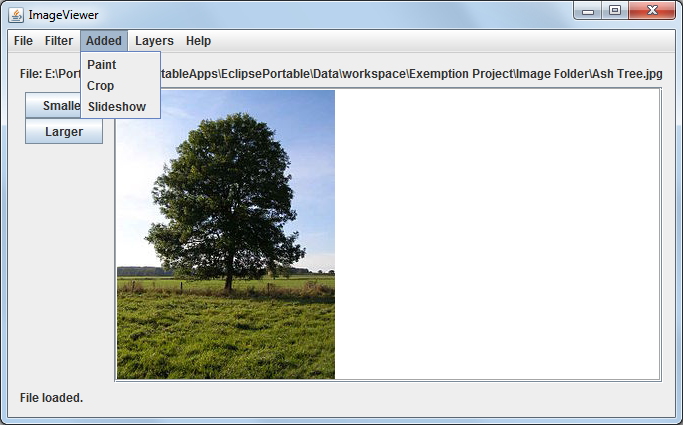
The Slideshow Function implemented in the ImageViewer program is mainly contained within the SlideshowMain class This class is accessed through the “Added” menu in the ImageViewer class:

The main body of the Slideshow Function looks as follows:

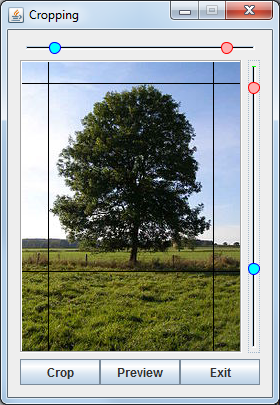


Where the “Previous Image” button selects the previous Image in the slideshow, the “Add Image(s)” button allows you to add image(s) to the slideshow, the “Remove Image(s)” button allows you to remove Images from the slideshow, the “Exit Slideshow” button allows you to exit the slideshow, the “Shuffle Slideshow” button allows you to run the Slideshow automatically in a random manner, the “Loop Slideshow” button allows you to run the Slideshow automatically in the order of which you added the images to the Slideshow and the “Next Image” button allows you to progress to the next Image

Crop Function:

The Crop Function implemented in the ImageViewer program is mainly contained within the CropFilter class This class is accessed through the “Added” menu in the ImageViewer class:

The Crop Function itself looks like:

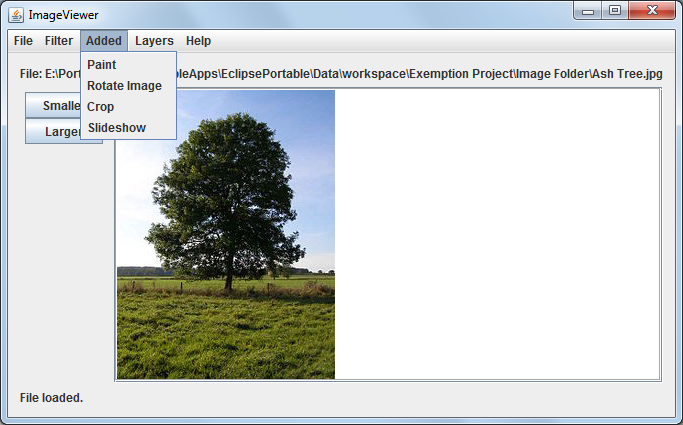


Where the black Lines are the bounds which are about to be cropped by the cropFilter, the “Crop” button crops the Image, the “Preview” button allows for the displaying of the Cropped image in the ImageViewer ImagePanel without saving and the Exit button exits the Crop Filter and returns to the ImageViewer class.

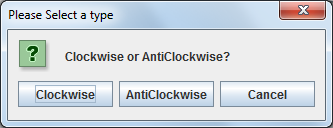
The JSliders with two “Sliders” are an instance of a “Range Slider” which is a custom imported package which I will go into more detail on in the detailed explaination of all the functionality.

Rotate Function:

The Rotate Function implemented in the ImageViewer program is mainly contained within the RotationalFilter class. This class is accessed through the “Added” menu in the ImageViewer class:



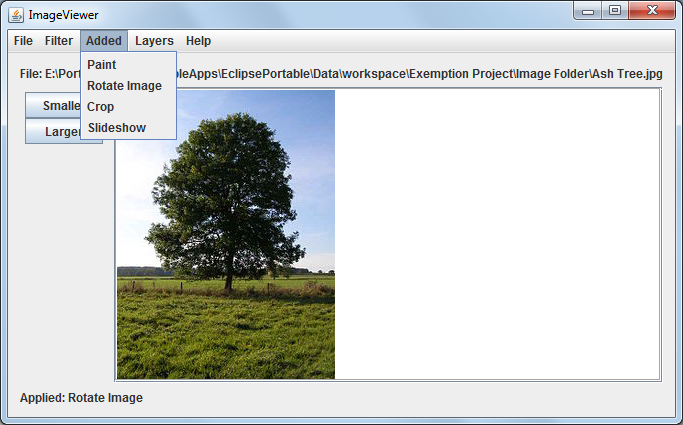
The Rotate Function itself has the GUI of:



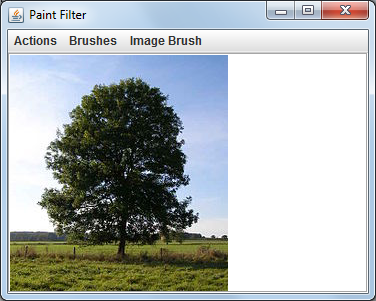
Where “Clockwise” rotates the OFImage clockwise, “AntiClockwise” rotates the OFImage AntiClockwise and cancel exits the Filter without modifying the Image.

Paint Function:

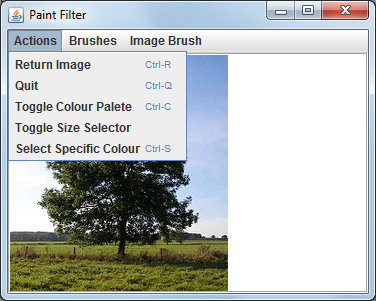
The Paint Function implemented in the ImageViewer program is mainly contained within the PaintFilter class. This class is accessed through the “Added” menu in the ImageViewer class



The Paint Function GUI itself looks like:

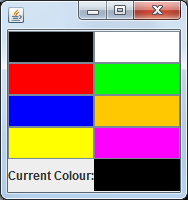


Where the Actions menu contains:



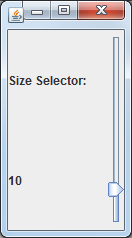
Where “Return Image” returns the edited image to the main body of the ImageViewer, “Quit” returns to the main body of the ImageViewer without editing the image, “Toggle Colour Palete” Toggles the Colour Palete on and off, “Toggle Size Selector” toggles the Size Selector on and off and “Select Specific Colour” toggles the Select Specific colour menu on and off.

The Colour Palete GUI looks like:



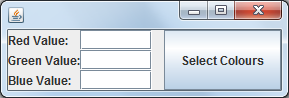
Where current Colour changes as the current selected colour changes and clicking the 8 JButtons above changes the current Colour to the clicked Colour.

The Size Selector GUI looks like:



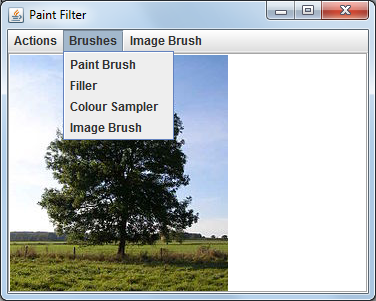
Where the JLabel displaying “10” displays the current size of the brush, and the JSlider to the right-handside dictates the size of brush.

The Select Specific colour Menu looks like:



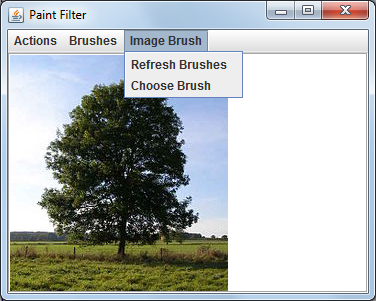
Where the 3 Text fields are inputs for the Bitmapped Red, Green and Blue value and the “Select Colours” JButton creates a colour with these values and sets the current brush colour to it.

The Brushes Menu contains the different types of brushes for the Paint Function:



Where “Paint Brush” sets the brush to the default paint brush, “Filler” sets the brush to the “Filler” brush, “Colour Sampler” sets the brush to the “Sampler” brush and Image Brush sets the Brush to the “Image Brush” brush.

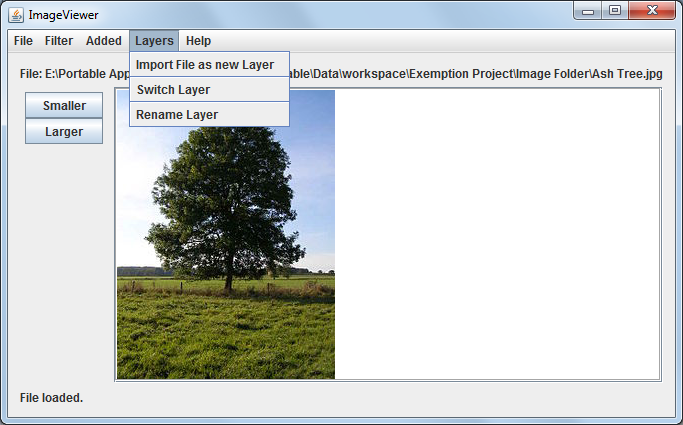
And the Image Brush contains:



Where “Refresh Brushes” calls the BrushManger.loadBrushFiles() method and “Choose Brush” allows the user to choose what brush they would like to use.

Layers Function:

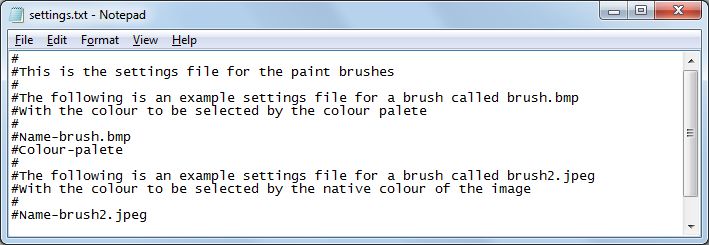
The Layers Function implemented in the ImageViewer program is mainly contained within the LayerManger class. This class is accessed through the “Layers” menu in the ImageViewer class.



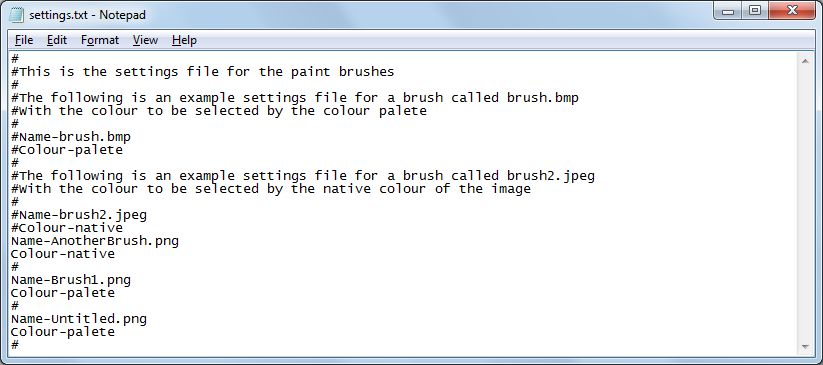
Where “Import File as new Layer” allows the user to create a new Layer using an external File, “Switch Layer” allows the user to change layers and “Rename Layer” allows the user to rename a selected layer.

User-Definable Brushes:

The User-Definable Brushes function has no GUI but allows the user to use their own images as “Brushes” akin to something like photoshop – on running the program it creates a “Brush Folder” at the default dictionary and then populates this folder with a “settings.txt” with the contents of:



It also generates information automatically from the brushes in the brush folder, for example:



The formatting and types of information will be explained within my detailed explanation of the code.

A Detailed explanation of my code:

Undo/Redo Functions:

First of all two ArrayLists<OFImage> are defined within the ImageViewer Class; undoFunction and redoFunction.

I have then implemented 78 methods – addUndo(OFImage),addUndo(),doUndo(),resetUndo(),addRedo(OFImage),doRedo(),resetRedo()

addUndo(OFImage):

This method adds the input OFImage to the undoFunction ArrayList.

addUndo();

This method calls the addUndo(OFImage) function using the currentImage variable.

This method is called before any method which will change the currentImage

doUndo()

This method first of all calls addRedo(OFImage) with the currentImage variable as the parameter to store a redo and then gets the last-added currentImage(like popping from a stack) from the undoFunction ArrayList and then removes that currentImage from the undoFunction ArrayList. It then sets up the GUI to show the new currentImage.

resetUndo()

This method calls undoFunction.clear() to clear the undoFunction completely – this is called when the program either closes or opens a file.

addRedo(OFImage)

This method adds the parameter to the redoFunction ArrayList using redoFunction.add(image)

doRedo()

This method first of all calls addUndo() to store a undo and then gets the last-added currentImage(like popping from a stack) from the redoFunction ArrayList and then removes that currentImage from the redoFunction ArrayList. It then sets up the GUI to show the new currentImage.

resetRedo()

This method calls the redoFunction.clear() function to clear the ArrayList. This method is called whenever a method which will change the currentImage is called.

Slideshow Function:

This function required a whole new class to be built – SlideshowMain.

Within the Slideshow class there are several methods:

addFiles()

This method declares and initializes a JFileChooser called chooser and then sets it to be able to select multiple files at the same time – it then checks that the “Ok” button was clicked in the JFileChooser and if not- returns to the main body of the program. It then adds all files which were returned to the slideshowList of OFImages and then selects the first image to be shown.

selectImage(int)

This method takes in the index of the image to be selected and opens a try bracket, within this it sets the currentImage to be equal to the OFImage at the index of slideshowList and then sets the label to be the name of that OF Image. The try bracket then closes and ends with a catch statement to catch a Index out of bounds Exception – if this occurs it sets the current image to the OFImage at the index of slideshowList and the corresponding file name.

After the catch statement it then sets the imagePanel image to that image and then packs the frame.

makeFrame()

This method creates the frame to be used in the Slideshow Class.

It first creates the JFrame frame and then creates the corresponding JPanel contentPane from this JFrame. It then selects a BorderLayout(6,6) to be used as the layout for the contentPane. An ImagePanel called imagePanel is then created. Then a new JLabel called fileLabel is added. A JPanel called topPanel with GridBagLayout is then creates to which content fileLabel and imagePanel are added to. The contentPane then adds the topPanel JPanel.

The Buttons are then created and added to the list of actions to be listened to and are then added to the contentPane via a third JPanel called P1.

The frame is then packed and then set to visible.

setLabel(String)

This method changes the Text and ToolTip of the fileLabel to the String parameter provided.

shuffleFunction()

This method creates two ArrayLists – one for Files and one for the Indexes of the Slideshow Images. These are then both randomized by the two different randomGens with the same seed – allowing for both ArrayLists to still be in sync. After this a new timer is started to pulse every two seconds and to send it’s events to the current instance of the SlideshowMain class.

loopFunction()

This method initializes a new timer to repeat every two seconds and to send its events to the current instance of the SlideShowMain class

loopInput()

This method is called every time the timer sends an action event which corresponds to the loop function and allows the timer to progress to the next image in the slideshow.

shuffleInput()

This method is called every time the timer sends an action event which corresponds to the shuffle function and allows the timer to progress to the next image in the randomized ArrayList of indexes which allows for the seemingly randomly generated progression of images.

removeFunction()

The purpose of this function is to remove selected images from the slideshow –

It first of all sends out a JOptionPane to the user asking what image they would like to remove. It then finds the index of that image and removes that index from both the fileList and the slideshowList arrayLists which removes that image from the slideshow.

actionPerformed(ActionEvent)

In this class the ActionEvent source is found using ActionEvent.getSource() and then this source is used to decide the action to take on receiving this ActionEvent.

Crop Function:

This function required a whole new class to be built – CropFilter.

Within the CropFilter class there are several methods:

In this class an external package was used which was not programmed by me:

The RangeSlider package is a subclass of JSlider which allows for two sliders instead of one.

All code within the rangeSlider package is from an external source and all credit goes to ernieyu with the GitHub Repository of: <https://github.com/ernieyu/Swing-range-slider>

This package is also licenced under “The MIT License” which allows for use without explicit permission.

apply(OFImage)

This is called whenever the Filter menu Item is clicked, it sets the inputImage to be equal to the parameter and then makes the frame.

makeFrame(OFImage)

This class is called in the apply(OFImage) method only and sets out the frame for this method, it uses a JFrame to set up the window and then a JPanel to define the contentPane inside. In this contentPane a ImagePanel is added to the center, a Horizontal RangeSlider is added to the North, a vertical RangeSlider is added to the EAST and three Buttons (“Crop”, “Preview” and “Exit”) are added to the south.

getOutput()

This public method returns the newImage which has been cropped.

crop()

This method sets newImage to a OFImage which has been initialized with the width of the cropped image and the height of the cropped image, the method then proceeds to iterate around the pixels contained within the cropped region and then transpose them onto the newImage in the relative co-ordinates. After this is done the object is flipped around the horizontal axis as this was an artifact from the way I handled cropping the image.

updateWindow()

This method updates the imagePanel with the new image to be displayed, this is normally called at the end of the drawLines() method.

drawLines()

This method first of all gets a copy of the input image called displayImageand then draws on the horizontal lines across the display Image according to the positions of the sliders, it then does the same for the vertical sliders and then updates the window.

actionPerformed(ActionEvent)

In this class the ActionEvent source is found using ActionEvent.getSource() and then this source is used to decide the action to take on receiving this ActionEvent.

Within this method there is some manual feedback to the main instance of ImageViewer via ActionEvents – this was done to provide the user with time to choose the area to crop in the GUI and to provide feedback to the ImageViewer Instance

stateChanged(ChangeEvent)

In this class both rangeSliders are assigned to the stateChanged implementation and when the stateChanged method activates it assigns either the horizontal or the vertical range values according to which slider was moved and then calls drawLines() to draw the lines on the displaying Image

**Rotate Function:**

This function required a whole new class to be built – RotationalFilter.

Within the RotationalFilter class there are two methods:

apply(OFImage)

This method is not used for this Filter and is only implemented to provide compatability with the Filter abstract method.

applyReturn(OFImage)

This method is used when the rotate function is to be performed

First of all, this method prompts the user for input on how to rotate the image via the JOptionPane class and if the user presses cancel, it returns the input image without any alteration. Otherwise – if the user presses clockwise, the clockwise rotation is activated, and if the anticlockwise button is pressed, the anticlockwise rotation is activated.

To rotate the image the method iterates through the whole OFImage and assigns the bitmapped values to the opposing side of the array resulting in a rotation.

**Layer Function:**

This function required a whole new class to be built – LayerManager.

Within the LayerManager class there are several methods:

addNewLayer(String)

This method is used to add a new layer to the ArrayLists involved in storing the Undos,Redos, currentImages, and layer Names.

addNewLayer(String)

This method checks for another layer existing with this name, and if it does exist then it returns false.

Otherwise , it prompts the user to select an image to import as a new layer and after this it adds the new import to the image ArrayList, it then adds the name to the name arrayList and adds null to the undo and the redo ArrayList<OFImage> Arraylists.

addFirstLayer(OFImage)

This method takes in the opened Image and adds the “Opened Image” to layerNames, null to undoFunctionLayer and redoFunctionLayer and then finally the OFImage parameter to currentImageLayer.add(image)

setUndoFunctionLayer(ArrayList<OFImage>,String)

This method takes in the ArrayList<OFImage> which is the undoFunction from the currentLayer from ImageViewer and then uses findIndex(String) to find out the index for the input layer name and then sets the element at that point to the input ArrayList<OFImage>

setRedoFunctionLayer(ArrayList<OFImage>,String)

This method takes in the ArrayList<OFImage> which is the redoFunction from the currentLayer from ImageViewer and then uses findIndex(String) to find out the index for the input layer name and then sets the element at that point to the input ArrayList<OFImage>

setCurrentImageLayer(OFImage,String)

This method takes in the OFImage which is the currentImage from the currentLayer from ImageViewer and then uses findIndex(String) to find out the index for the input layer name and then sets the element at that point to the input OFImage.

setUndoFunctionLayerIndex(ArrayList<OFImage>,int)

This method takes in the undoFunction and then replaces the element at the int parameter index.

setRedoFunctionLayerIndex(ArrayList<OFImage>,int)

This method takes in the redoFunction and then replaces the element at the int parameter index.

setUndoFunctionLayerIndex(OFImage,int)

This method takes in the currentImage and then replaces the element at the int parameter index.

renameLayerIndex(String, String)

This is used to rename a Layer, by providing the method with the old name and then with the new name for the layer, it will then replace the name in that layer to the new name.

findIndex(String)

This method cycles through the program and returns the index of the Layer with name provided by the String parameter

getUndoFunction(String)

This method returns the undoFunction ArrayList<OFImage> associated with the String parameter of the layerName

getRedoFunction(String)

This method returns the redoFunction ArrayList<OFImage> associated with the String parameter of the layerName

getCurrentImage(String)

This method returns the currentImage OFImage associated with the String parameter of the layerName

getLayerNames()

This method returns the ArrayList<String> which contains all of the different names of the layers

layerExist(String)

This method iterates through all of the Layer names and if the String parameter matches with any of them it returns true, otherwise it returns false

clearAllLayers()

This method clears all of the stored layers

**Paint Filter:**

This function required a whole new class to be built – PaintFilter.

Within the Paint Filter class there are several methods:

setImage(OFImage)

This public method sets the image to be used in the PaintFilter

getOutput()

This method returns an OFImage which is the output from the PaintFilter

setCurrentRGB(Color)

This method sets the current brushColour

setBrush(int)

This method sets the image brush to be used with the image brush

makeSizeAdjustor()

This method builds the size Adjustor.This consists of a JSlider on the east side of the JFrame’s contentPane , and a JPanel containing two JLabels illustrating what the Size Adjustor is.

makeSpecificColourPalete()

This method builds the Specific Colour Palete. This consists of 3 JLabels and 3 JTextBoxes and a JButton which allows the Colour to be chosen.

makeColourPalete()

This method builds the Colour Palete. This consists of 8 JButton and a JLabel and a JPanel, as the JButtons are pressed, they change the currentColour corresponding to which JButton was pressed. The JPanel also changes colour as the JButtons are pressed.

makeFrame()

This method builds the main frame of the Paint filter, it introduces a ImagePanel and assigns a MouseListener and a MoustMotionListener to the ImagePanel.

updateIndicator()

This method changes the colour of the colourIndicator in the ColourPalete to the currentColour when it is called.

applyBrush(Point,String,MouseEvent)

This method gets called whenever a MouseClick or a Mouse Drag event occurs, Depending on which brush is currently selected and which button is currently held down a different type of brush is called with the parameter of Point

applyImageBrush(Point)

This method allows the user to paint onto the imagePanel using an image brush, it seperates how it paints the image onto the ImagePanel due to if it is a “native” image or if it is a “palate” image, this allows for greater control of the image brush strokes.

colourClose(Color,Color,int)

This method calculates if the two colours which are passed in as parameters are equal, the integer value is used as a threshold to allow the Colors to be slightly different and also be treated as equal.

applyFiller(Point,Color)

This method sets the pixel at the point which is passed as parameter to the currentColour, it then checks if the pixels around it are the same as it’s original colour and if they are, it applys itself recursively until all space is filled.

applyPaintBrush(Point)

This method paints a circle with width = brushSize (Which is selected using the SizeSelector) and height = brushSize and colour = currentColour at the Point parameter which has been passed in.

applyPaintSelector(Point)

This method copies the colour of the point that it’s clicked on and uses it as the currentColour of the palate.

applyEraser(Point p)

This method erases a square area around point P with height = brushSize and width = brushSize, by erasing the area it reverts it to the original picture area.

refreshImage()

This method repaints the imagePanel.

makeMenu()

This method makes the menus at the top of the JFrame

These have the hierarchy of:

* Actions
  + - ReturnImage
    - Quit
    - Toggle Colour Palete
    - Toggle Size Selector
    - Select Specific Colour
* Brushes
  + - Paint Brush
    - Filler
    - Colour Sampler
* ImageBrush
  + - Refresh Brushes
    - Choose Brush

apply(OFImage)

This method sets the image to be used in the PaintFilter

**Brush Manager:**

This function required a whole new class to be built – BrushManager.

Within the BrushManager class there are several methods:

getNumberOfBrushes()

This is a public method which returns the number of brushes in the brushManager Instance

getBrushType(int)

This is a public method which returns the brushType which corresponds to the index which is passed in as a parameter.

getBrush(int)

This is a public method which returns the OFImage Brush with the Index which is passed in as an in parameter

findIndex(String)

This is a public method which returns the index of the input parameter of brushName

refresh()

This is a public method which refreshes the brushFiles and all of the brush ArrayLists

getBrushNames()

This public method returns an ArrayList<String> of all brushes in the Brush Manager

loadBrushFiles()

This method searches for the brush image files and the settings file and then loads the settings file.

loadSettingsFile()

This method loads the settings file and populates the brushTypes and brushNames ArrayLists

appendDataToSettingsFile()

This method adds the current brushes on to the end of the created settings file.

createSettingsFile(Boolean)

This method creates the settings file - If the input Boolean is true then it will rewrite the settingsFile.

List of the key lessons from the course that I have used:

Lecture1 -

Lecture 2- Objects and Classes – PaintFilter.java Class is an instance of both.

Lecture 3 – Fields and Constructors – Constructor is called for both CropFilter.java and PaintFilter.java

Lecture 4- If Statements – Used in throughout the program in the PaintFilter, CropFilter, SlideshowMain, BrushManager and Layer Manager classes.

Lecture 5- Boolean AND Operator and Boolean OR Operator – AND is used in the PaintFilter.colourClose(Color, Color, Int) method where as the OR operator is used in the PaintFilter.applyEraser(Point) method

Lecture 6-Internal and External Method Calls – Internal Method calls are used in the SlideshowMain. loopInput() method whereas external method calls are used throughout the program in the PaintFilter, CropFilter, SlideshowMain, BrushManager and Layer Manager classes but specifically in the SlideshowMain.selectImage(int) method

Lecture 7 – ArrayLists – undoFunction and redoFunction in the ImageViewer class

Lecture 8 – While Loop – A while loop is used in the BrushManager.loadSettingsFile() method

Lecture 9 – For Loop – A nested for loop is used in the PaintFilter.applyImageBrush(Point) method

Lecture 10 – Null Reference – A Null reference is used in the ImageViewer.showFilename(String) method to determine what to display in the filenameLabel.

Lecture 11 – Use of Eclipse – I have been using Eclipse to develop the project and will use it to demonstrate it

References:

(As the uni computers are using Java 6 I have been using the Java 6 APIs as reference)

Java 1.6 Docs

http://docs.oracle.com/javase/6/docs/api/