Oliver Baldwin Edwards and Isaac Caruso
COSC 112
Intellage Program Specification

Overview:

Our program is a piece of image software wherein the user can input a folder containing numerous images and our software will create an intelligent collage of the images by mapping them onto either a separate image (which the user designates) or one of several image presets. In short, the final output of our program is a large image whose pixels are made up of numerous smaller inputted images.

This program includes a graphical user interface which easily allows the user to interact with the program when choosing which images they would like the software to use. The user uses their mouse to select / designate photo folders and template photos in an easily navigable way. Once our program receives the selected folder of photos and template photo, it splits the template photo into small chunks to figure out the HSV values for each pixel (the number of chunks will be equivalent to the number of photos in the inputted folder). It will then search through each inputted photo from the folder to find similar HSV values for each chunk of the template image. The result is a collage estimation of the template photo from the inputted folder of many photos.

Our program also gives the user the option to save their created collage photo once created by creating a JPG file of the product image and placing it in the user inputted folder once created. Our program is a piece of photo editing software which allows the user to tell a larger story by stitching a number of their photos into one larger image. Its purpose is one of creative expression.

Detailed class specifications:

Classes:

1. **FolderOfPhotos**
    a. This class contains an array of IndividualImages from the user-inputted folder and gets the average HSV values for each image and stores them in a 2D int array
    b. Member Variables:
        i. An IndividualImage array ‘photoArray’ that contains instances of IndividualImages that will have length numPhotos (the number of photos in the user-inputted folder)
        ii. A File ‘directory’ that contains the path to the user-inputted folder
        iii. A static int ‘numPhotos’ that contains the number of photos in the selected folder
            1. It’s static because we wanted to be able to refer to it using ‘FolderOfPhotos.numPhotos’ in other classes
        iv. A boolean array ‘alreadyMatchedSections’ that is used to make sure we only check each photo once
        v. A 2D int array ‘avgValues’ to hold average H (hue), S (saturation), and V (value) values for each photo
            1. It will have the dimensions [numPhotos][3] so that for each photo in the folder we hold each of the three HSV values that we want

vi. A boolean 'perfectSquare' that checks that the numPhotos is a perfect square

c. Constructor:
   i. This will contain one constructor with nothing to pass in because we will always have the same type of information for a user-inputted folder of photos
   ii. It will read the 'selectedFolder' from our GraphicalUserInterface class and set that to the 'directory' variable
   iii. It will then get the 'numPhotos', fill the 'photoArray', resize our 'photoArray', and fill in the average HSV values to 'avgValues'
   iv. One instance of it will be created in our World class

d. Methods:
   i. A method 'initializeNumPhotos()' that will return the number of photos in the selected folder. We used https://www.daniweb.com/programming/software-development/threads/341929/h ow-to-read-images-from-a-folder# to learn how to list all files in a directory within a for loop.
      1. We used EboMike's answer from here: https://stackoverflow.com/questions/3571223/how-do-i-get-the-file-extensi on-of-a-file-in-java to check to make sure the only files we count are ones that are jpgs or jpegs within this method
   ii. A method 'fillPhotoArray()' that goes back through all of the images in the inputted folder and fills our 'photoArray' with instances them
      1. Returns nothing
   iii. A method 'resizePhotoArray()' that resizes each photo in 'photoArray'
      1. Returns nothing
   iv. A method 'fillAvgHSVValues()' that fills 'avgValues' with the average HSV values for each photo in the user-inputted folder in our avgValues 2D array
      1. Returns nothing
   v. A static method 'isSquare(int n)' to check if an integer is a perfect square that returns a boolean True if the integer is a perfect square and False if it isn't
      1. This whole method was obtained from https://algorithms.tutorialhorizon.com/check-whether-the-given-number-isa-perfect-square/
   vi. A method 'orderImageReturnNewLocation(int photoInGrid)' that reorders the folder of photos into the order that will match up with the selected template image based on matching H, S, and V values.
      1. Will return the integer location in the template image where the current photo should go
   vii. A method 'compareImages(int locationInTemplatePhoto, int newPhoto)' that compares the H, S, and V values from the template image and inputted folder of photos.
      1. Computes and returns how far away the H, S, and V values at the passed in location are from each other (an integer value)

2. **TemplateImage**
    a. This class gets and splits the user-inputted template image into chunks and then reads the average HSV values for those chunks and stores them in a 2D int array
    b. Member Variables:
        i. Static ints 'width', 'height', and 'imagePieces' that deal with the width and height of the template image and the number of images in the user inputted folder respectively.
            1. 'imagePieces' is synonymous with 'numPhotos' from the FolderOfPhotos class
        ii. A static IndividualImage 'backgroundImage' that contains an instance of IndividualImage that holds the user-selected template / background image iii. A 2D static int array 'avgTemplateValues' that holds the average H, S, and V values for each chunk in the template image.
            1. This array is of the same format as 'avgValues' in the FolderOfPhotos class so as to easily compare the H, S, and V values between the two
            2. It will have the dimensions [imagePieces][3] so that for each photo in the folder we hold each of the three HSV values that we want
    c. Constructor:
        i. This will contain one constructor with nothing to pass in because we will always have the same type of information for a user-inputted template image
        ii. Will get 'numPhotos' from the FolderOfPhotos class and pass it into 'imagePieces'
        iii. Will use the GraphicalUserInterface class to get the 'selectedBackgroundImage' and initialize 'backgroundImage'
        iv. One instance of it will be created in our World class
    d. Methods:
        i. A method 'fillAvgTemplateValues()' that fills in the average H, S, and V values for each chunk of 'backgroundImage' 1. Returns nothing

3. **ProductImage**
    a. This class builds our product image which is shown to the user at the end of running our program.
    b. Member Variables:
        i. An IndividualImage 'builder' which will be used for drawing the images to the template image
        ii. An int 'numPhotos' for the number of photos in the user-inputted folder
        iii. A File 'directory' that holds the location
    c. Constructor:
        i. This will contain one constructor with nothing to pass in because we will always have the same type of information for a user-inputted template image
        ii. Will get the 'selectedFolder' path directory from the GraphicalUserInterface class
        iii. Will get 'numPhotos' from the FolderOfPhotos class
        iv. Will build the final image and then delete the temporarily used images
    d. Methods:
        i. A method 'buildImageUpdated' that builds the final image using the renamed and resized images

            1. (Built using information about BufferedImage and Graphics2D from https://stackoverflow.com/questions/6575578/convert-a-graphics2d-to-an-i mage-or-bufferedimage)
            2. Returns nothing
        ii. A method 'deleteTemporaryImages' that deletes the images that were temporarily created for mapping to the template image
            1. Returns nothing

4. **Individual Image**
    a. This class contains functionality that will take individual images from the user inputted folder and contains functions to get the HSV values of an IndiviualImage. There will be an instance of this class for every image in the inputted folder and for the TemplateImage.
    b. Member Variables:
        i. Ints 'xPos', 'yPos', 'height', 'width', 'numpPixels'
        ii. A File 'image' which is the IndividualImage itself
        iii. A ColorModel 'getValues' which allows us to get RGB values for the image which we can then convert to HSV
    c. Constructor:
        i. This class will have two constructors to allow for an instance of individual image to be created either with a string or file inputted containing the image's location
        ii. Will set the image size in each constructor using 'setImageSize()'
    d. Methods:
        i. Two methods 'getAverageHSV' which will either be given no input parameters or will be given an initial x and y location and a desired width and height of a region will return the average H, S, and V values for the region of this individual image instance in an int array
            1. The former is used for FolderOfPhotos, the latter for TemplateImage
        ii. Two methods 'getHSVArray' which will either be given no input parameters or will be given an initial x and y location and a desired width and height of a region and then create and return a 2D int array of HSV values
        iii. A method 'getHSV' which takes in an integer rgbValue and returns an integer array of length 3 containing one H, S, and V value. Adapted from a Python script on https://www.geeksforgeeks.org/program-change-rgb-color-model-hsv-color-model/
        iv. A method 'setImageSize()' which gets the height and width of a file
            1. Adapted from https://stackoverflow.com/questions/672916/how-to-get-image-height-andwidth-using-java
        v. A method 'resizeImage' which resizes an image given the File, the desired width and height after being resized, and the format of the image. Adapted from: https://www.youtube.com/watch?v=YM02Q3s9pbY

5. **World**
    a. This class creates the ProductImage using many other classes.
    b. Member Variables:
        i. A 'ProductImage' product which will be our final product image

ii. The user-inputted FolderOfPhotos ‘folder’
iii. The user-inputted TemplateImage ‘template’
iv. An int array ‘newPhotoLocations’ to store the new photo locations for the photos in
‘folder’ so that they match up to the HSV values in the TemplateImage

c. Constructor:
   i. One instance of it will be created when the program is run in our Main class.
   ii. First an instance of FolderOfPhotos is called followed by an instance of TemplateImage iii. The ‘newPhotoLocations’ array is then filled using the ‘orderImageReturnNewLocation’ method from FolderOfPhotos
   iv. The ProductImage is then created (an instance of it is called)

d. Methods:
   i. None

6. **GraphicalUserInterface**
   a. This class creates our user interface where a user can select both the FolderOfPhotos and TemplateImage that they want and allows our program access to both respective file paths / files
   b. Member variables:
      i. A JFileChooser ‘chooser’ that allows the user to navigate their file system.
      ii. A String ‘choosertitle’ created by WindowBuilder automatically.
      iii. Static Files ‘selectedFolder’ and ‘selectedBackgroundImage’ which store the user’s choice for a folder and background image to be accessed later in the program.
      iv. JButtons ‘selectFolderButton’, ‘selectFileButton’, and ‘btnGo’ which create mutable buttons which the user can press to perform a corresponding event in actionPerformed.
      v. A private final Action ‘action’ created by WindowBuilder automatically.
      vi. Static booleans ‘goPressed’, ‘backgroundChosen’, and ‘folderChosen’ which record if the user has accurately performed each of these actions.
      vii. JLabels ‘lblNewLabel’ and ‘lblChooseASource’ print to the GUI strings to point the user to the correct button.
      viii. A JFrame ‘frame’ which contains a contentPane to hold all of the various components of the GUI.
   c. Constructor:
      i. Much of the code here was automatically created by WindowBuilder
      ii. One constructor was used to create the GUI which includes actions such as clicking on buttons to select the FolderOfPhotos and TemplatePhoto
   d. Methods:
      i. A method ‘actionPerformed’ which is where button clicking becomes an action
         1. Contains code for what happens if either of the “select a background image”,
            “choose a source folder”, or “Go” buttons are pressed
         2. This is where the file path of both the selected FolderOfPhotos and TemplateImage is recieved and stored by GUI for later use by other classes in our program

  ii. A method ‘openGUI’ that causes the GUI window to open and begins the initial GUI side of the program
  iii. A method ‘displayImage’ that displays the image "finalImage.jpg", located in the source folder, in the GUI window
    1. Code for resizing and image adapted from: https://coderanch.com/t/331731/java/Resize-ImageIcon
  iv. A method ‘getFinalDisplaySize’ that gets the height and width of a file
    1. Code here adapted from https://stackoverflow.com/questions/672916/how-to-get-image-height-andwidth-using-java
  v. A method ‘getPreferredSize’ which returns the PreferredDimensions (automatically added by Window Builder)

e. Additional classes:
  i. A private class SwingAction extends AbstractAction
    1. This was something automatically added by Window Builder

7. **Main**
  a. This class runs our program and ties together the GraphicalUserInterface and World classes
  b. Member variables:
    i. None
  c. Constructor:
    i. None
  d. Methods:
    i. The main method runs our program
    ii. An instance of GraphicalUserInterface ‘gui’ is first created
    iii. The ‘gui’ is then opened
    iv. An instance of World is then created which creates our ProductImage
    v. The ‘gui’ then displays the image using the ‘displayImage’ method from the GraphicalUserInterface

How the program runs:

The Main class is the overall class that is used to run our project. It first creates an instance of GraphicalUserInterface and opens the GUI which allows the user to interact with the program and select which folder of photos and template image they would like to use. An instance of World is then called. World creates an instance of FolderOfPhotos and TemplateImage and then compares the average HSV values of each in order to rearrange the FolderOfPhotos in a way that closely matches / represents the TemplateImage. This image is then created by creating an instance of ProductImage. The GUI then displays the image and saves it as a file called “finalImage.jpg” to the same folder as the one that the user initially inputted. This allows the user to save the final product image if they desire.