Manual for running quadtree comparison

Classes used:

Colour: This class is specifically used to store and get colours of each pixel present in image. The colour format used here is alpha, red, green and blue.

Coordinate: Coordinate class represents a point in image which is represented as x and y.

MathematicalOperations: This specific class is used to compute log calculations to check if the image is in correct format for constructing quadtree.

ImageScaling: Image scaling is used for preprocessing the images into specific format of 1024x1024 resolution to form more accurate quadtree.

QuadTreeNode: QuadTreeNode is the basic building block of quadtree which stores the information of image in form of quadtree with parent and children nodes.

QuadTree: This class is responsible for constructing quadtree from image by using recursion. Some other functionalities of this class are to compare two quadtrees and see if they are same to the given level.

CompareImages: This class has a naïve implementation of how two images are compared to each other. The worst case of this comparison will be tested by passing same images for comparison.

QuadTreeMain: This is the driver class which has public static void main in it. This class get the image inputs, levels of quadtrees to be compared and calls in appropriate functions for classes to compare the images.

How to run the project:

Importing Project:

1. Unzip QuadTree.zip into desired location.
2. Open eclipse.
3. Import project QuadTree by following the below steps,
   i) **File -> Import**.
   ii) In the import wizard choose **General -> Existing projects into workspace**.
   iii) Click **next**.
4. In the Import wizard browse and select the QuadTree project which is being extracted in step 1.
5. Click **finish** after choosing the project.

Inputs:

- The project takes two files as input from file path mentioned in line 26 and 27 of file QuadTreeMain. The inputs images path can be changed in these two lines to compare two different images. Sample images are present in **images folder,** you can add new images to the folder and use it for comparison.
- In addition to that, the project also takes to which level the comparison should be made in two constructed quadtrees.

Output:

There two outputs for this particular project. The main motivation is to compare two outputs so that we can prove quadtree comparison after constructing quadtrees is more optimal and faster when compared to naïve implementation of image comparison.

- Outputs if the quadtree of first image is same as quadtree of second image or not. We clock this comparison in nano seconds and print it with result.
- Outputs if first image is equivalent to second image or not. Again, we clock this image comparison in nano seconds and print it with result.

Compiling and running the project:

- Once the project is imported, open QuadTreeMain to change the images paths if needed.
- Right click on any empty space of QuadTreeMain.
- From the pop-up menu choose **Run As -> Java Application.**
- The application will run on console and will ask for levels to be checked in quadtree.
- Enter the levels and continue.

Limitations:

- The quadtree construction is optimal when the input image resolution is $2^n * 2^n$. Therefore, scaling of image is done to match default 1024 * 1024 image resolution. This image resolution can be changed in code to any $2^n * 2^n$ combination.
- This evaluation between quadtree and naïve implementation does not take in quadtree construction time into account.
- For too large images the recursive call stack may exceed. We have not encountered any such behavior while developing and testing this project. Since we use recursion there is a possibility that the call stack may exhaust and cause runtime exception.