CSE 464
DIGITAL IMAGE PROCESSING

HOMEWORK 03

REPORT

ÖMER ÇEVİK
161044004

1. Part 1

In that part, user gives the main argument as input image path which is salt and pepper noise filtered. In that programme, the *image/12.jpg* was used to show up all filter results.

First of all, the image is read and each bitmix, lexicographical, norm based and marginal class objects are created. The *Order* abstract class is implemented to override *filter()* method for each order styles. That *filter()* method gets the image which is already noised, filter's type name and kernel size of to apply filter mask.

Each filter creates its filter mask given kernel size of filter and applies on image using order method.

For *Marginal* order each RGB colors of filter sorted by distinctly. Then the median of RGB color set the center of filter and that filter works on image cumulatively like that.

For *Bitmix* order each pixels of filter compared as RGB colors are ordered bit by bit compressed. Then the median of RGB color set the center of filter and that filter works on image cumulatively like that.

For *Lexicographical* order each pixels of filter compared as alphabetical order. Then the median of RGB color set the center of filter and that filter works on image cumulatively like that.

For *Norm Based* order each pixels of filter compared as RGB colors are Euclidean order. Then the median of RGB color set the center of filter and that filter works on image cumulatively like that.

In the test files, there are three different kernel sized filters applied on image: 3, 5 and 7. The original image and salt and pepper noised image are given in result page. Salt and pepper noise probability is set to 0.04. The MSE results are printed to console for 3 kernel sized version of filter to see which of these filters cause error much and the Marginal filter gives the best result.

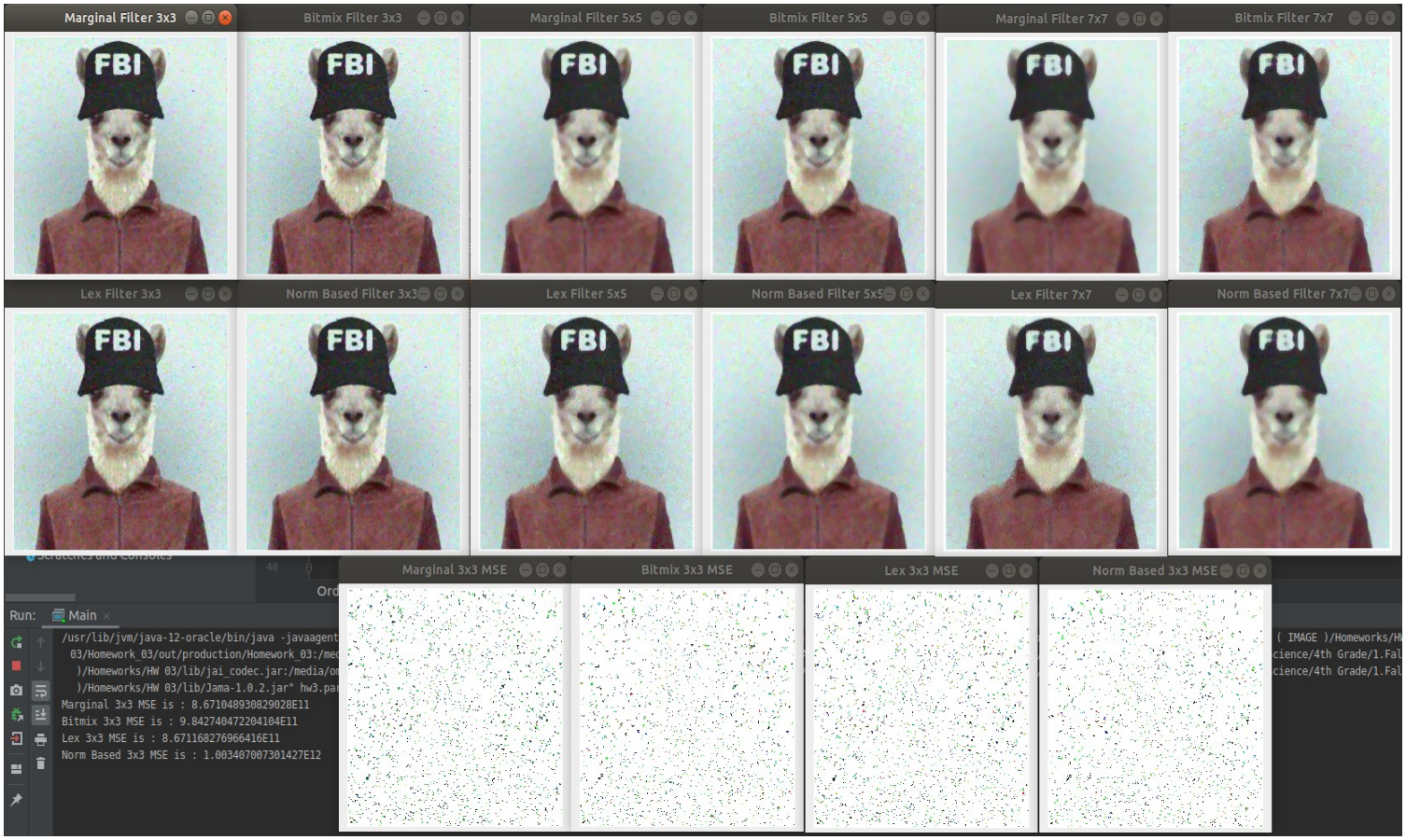
The results of filtered images are printed to screen to show up each filter effect. And also the result of MSE printed as image to screen to show up each difference imaginary.

The images (12 distinct) will be sent in directory to test and the original image of salt and peppered image is in the project too.

- **Output Results For Part 1**

Original Image
(256 x 256)

Salt and Peppered Image
(256 x 256) Prob: 0.04



2. Part 2

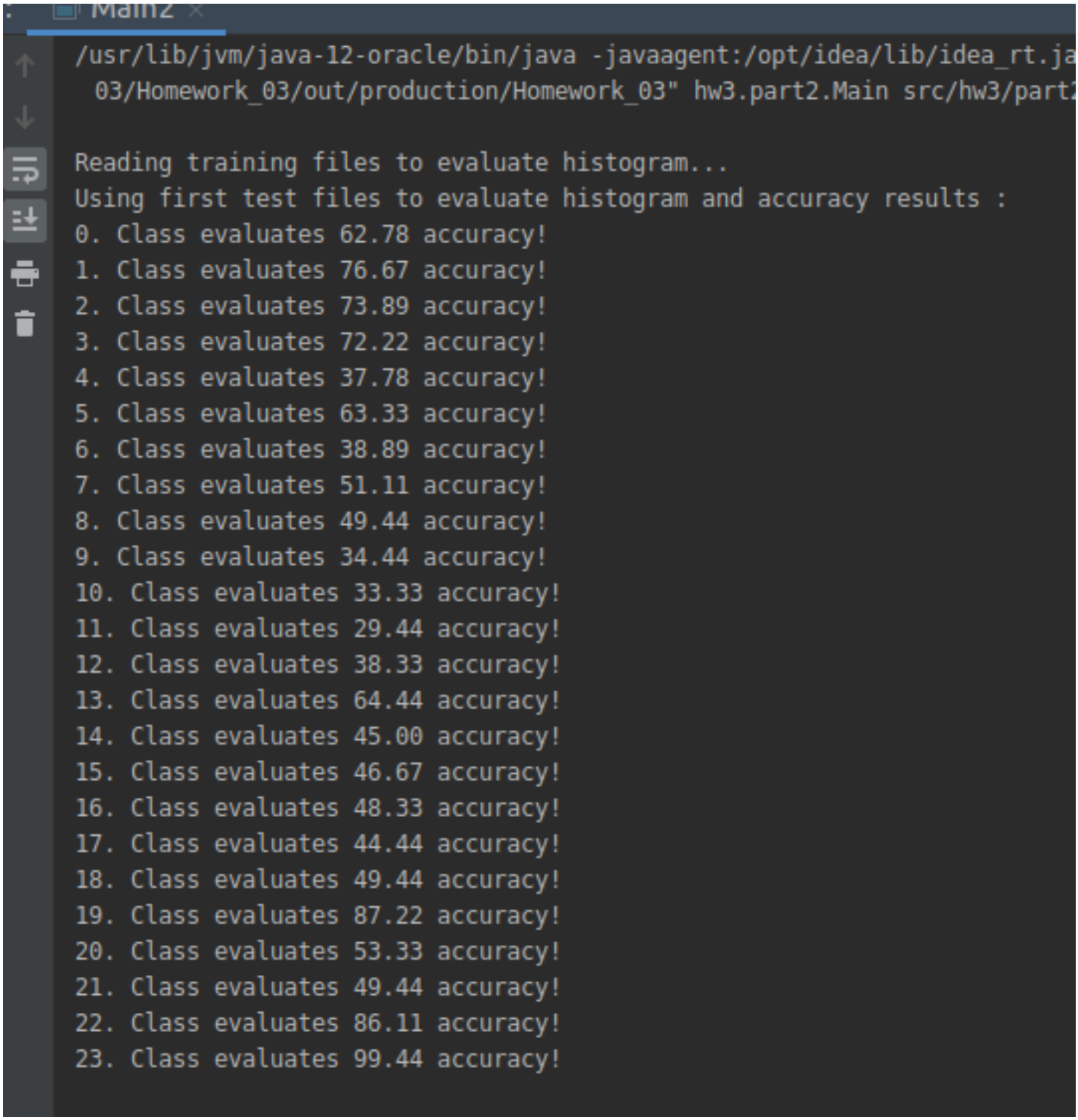
In that part, user gives the path of *Outex_TC_00012/images/* as main argument and the creates the *NN* object to apply nearest neighbor classifier calculations. There are two test files and that gives the two options. To test first test files is 0 and the other test files is 1. User must give the option while creating the *NN* object and also must give the main argument for path of images. Then the *evaluateNN()* method call will give the result.

The *NN* class creates the *LBP* object to evaluate histogram of images for training files and test files. The evaluated histogram values are stored in *Output/* directory as ".txt" files for each test and train images. Each histogram values lines' last element are set to class of image to calculate accuracy later. After the histogram equalization work then *LBP* object applies local binary pattern method to texture description of looking neighbor of each pixel. But *LBP* operations just works for radius 1. Other radius values are not implemented to test. After *LBP* operation the histogram of LBP is created. The rotation of each pattern is ignored. Because the shifting mechanism of filter didn't work on my programme. For each rotation of image is tried to solve the problem but to creating many image was costly (It gave *heap size error).* So the rotation is ignored.

The *LBP* histogram results are used to find the accuracy. Minimum values of histogram and maximum values of histogram are set. Using that minimum and maximum values the training and test image histograms are re-set. Then using the L_2 distance formula the minimum distance of test and train images is evaluated. Then the same values for train and test histogram for same classes are calculated to find accuracy and for each class the accuracy result printed to screen. The total time of programme gets approximately 7-9 minutes. The results of accuracy are in the output page as image.

* Java JDK 12 and Intellij IDEA platform is used.

- **Output Results For Part 2**



```
Reading training files to evaluate histogram...
Using second test files to evaluate histogram and accuracy results :
0. Class evaluates 65.00 accuracy!
1. Class evaluates 77.22 accuracy!
2. Class evaluates 73.89 accuracy!
3. Class evaluates 78.33 accuracy!
4. Class evaluates 37.22 accuracy!
5. Class evaluates 63.89 accuracy!
6. Class evaluates 42.78 accuracy!
7. Class evaluates 60.56 accuracy!
8. Class evaluates 50.00 accuracy!
9. Class evaluates 38.33 accuracy!
10. Class evaluates 32.78 accuracy!
11. Class evaluates 25.56 accuracy!
12. Class evaluates 36.11 accuracy!
13. Class evaluates 71.67 accuracy!
14. Class evaluates 48.89 accuracy!
15. Class evaluates 47.78 accuracy!
16. Class evaluates 43.89 accuracy!
17. Class evaluates 46.11 accuracy!
18. Class evaluates 51.67 accuracy!
19. Class evaluates 97.22 accuracy!
20. Class evaluates 55.56 accuracy!
21. Class evaluates 50.00 accuracy!
22. Class evaluates 93.89 accuracy!
23. Class evaluates 97.22 accuracy!

Process finished with exit code 0
```