**CSCI 1152, Summer 2020  
Lab Assignment 5**

# Objectives

1. Learn how to process two dimensional arrays.
2. Learn how to write methods to operate on two dimensional arrays.
3. Learn how to read data from a file.

# Description

Write a Java program that will process image data from a file. The file is given, imagedata.txt, is given in learn.cnm.edu. The program will read multiple images from the file into a two-dimensional data structure. The program will then process the data by applying two filters to the data and output the results.

### Step 1 - Reading Data

The file will contain 1 to n two dimensional arrays with each array value being an integer between 0 and 255. Each two-dimensional array is considered image data in which the number 0 represents pure black while the number 255 represents pure white. Any number between 0 and 255 is on the grayscale between black and white. Each array of image data is prefixed with a line before it that describes the size of the array. An example of the data file is below:

Image 1 Data

4 6

32 35 212 134 0 10

45 67 34 23 11 33

23 123 129 222 123 1

21 34 55 78 99 22

Image 1 Size: rows columns

Image 2 Data

4 4

45 67 23 12

12 200 212 129

45 67 89 34

111 129 34 99

Image 2 Size: rows columns

.

.

.

There can be 1 to N number of images. To read in the data properly the program will read the Image Size line to determine how big the array is for the following image. Then it will proceed to read the image data for that array. This will continue until the end of file is reached.

### Step 2 – Processing Data

Once the image data is read into a two-dimensional data structure the data can then be processed. The program should process the data using two methods by using the following filters.

#### Filter 1

For each cell in the array, the current cell, the value of the current cell should be replaced with an average of the current cell and all its neighboring cell values. A neighboring cell is any cell that is directly above, below, right, or left of the current cell. For example, given the data above for Image 1, cell at row 0 column 0, denoted as cell(0,0), value is 32. After filter 1 is applied to cell(0,0) the new value should be (32+35+45)/3. 35 and 45 are the neighboring cell values. Note: Not all cells have the same number of neighbors. This filter must be applied to all cells for each image.

#### Filter 2

For each cell in the array, the current cell, the value of the current cell should be replaced with the negated value of the current cell. The negated value is determined by following equation:

Negated value = 255 – current value

**However, the negation should only be applied if the current value is less than a THRESHOLD value.** The THRESHOLD value is equal to 128. If the THRESHOLD is not met than the cell value does not change. This filter must be applied to all cells for each image.

# Requirements

1. Your program should be able to work for any file with the data format given in Step 1 above. That is, do not hard code the number of 2D arrays you read in based on the given imagedata.txt file. I will test your code against multiple files with a different number of array data. For example, the imagedata.txt file has 3 2D arrays, but I will test your code with other .txt files with 1, 2, 3 or 5 2D arrays.
2. Your program should use 2D java arrays not Java ArrayList classes.
3. The java source file MUST abide by the google java style guide. It can be found at <https://google.github.io/styleguide/javaguide.html#s7.1-javadoc-formatting>

Insructions for automatically formatting your code using the google style guide can be found here: <https://learn.cnm.edu/bbcswebdav/pid-8979615-dt-content-rid-95578999_1/xid-95578999_1>

1. The class name is required to be Project\_<your email id>
   1. For example, my class would be:

public class Project\_mgonzales183 {

}

1. You must enter a javadoc comment before the class declaration. In the comment you must provide a small description of this program followed by your name.
   1. For example, my class would be:

/\*\*

\* Project is a program to…….put your description here…

\*

\* @author Mark Gonzales

\*/

public class Project\_mgonzales183 {

}

1. You must enter a javadoc comment before EACH method declaration. In the comment you must provide a small description of this method, specify the parameters and what the method returns. Follow the template below:

/\*\*

\* <write your description of method here>

\* @param <parameter name here> <write description of parameter is here>

\* @return <write description of what is returned here>

\*/

1. There must be at a minimum the following methods:
   1. (25 pts) Name: main
      1. The main method must perform Step 1 and Step 2 from above. That is, the main method must read the data from the file and process the image data by calling the applyFilter1 and applyFilter2 methods.
      2. The code must handle the ArrayIndexOutOfBoundsException when processing any array data. If the exception occurs the code must output an error to the user and exit.
      3. The code must handle all exeptions appropriatley when opening the file. The code should output a message to the user and exit the program.
   2. (25 pts) Name: applyFilter1
      1. The code must handle the ArrayIndexOutOfBoundsException when processing any array data. If the exception occurs the code must output an error to the user and exit.
      2. Parameter:
         1. int[][] - two dimensional array of integers that is the image data
      3. Return:
         1. int[][] - two-dimensional array of integers that have the updated values from applying filter 1.
      4. Description: Given the image array data apply filter1 and return the **new** filtered array. This method must call the getNeighbors method to help with processing.
   3. (25 pts) Name: getNeighbors
      1. The code must handle the ArrayIndexOutOfBoundsException when processing any array data. If the exception occurs the code must output an error to the user and exit.
      2. Parameters:
         1. Parameter one:
            1. int - Integer that represents the row of the current cell.
         2. Paramater two:
            1. int - Integer that represents the column of the current cell.
         3. Parameter three:
            1. int[][] - Two-dimensional Array of integers that represents the image data.
      3. Return:
         1. int[] - an array of integers that represents the neighbors of the current cell.
      4. Description: Given the row and column of the current cell return an array of the neighboring cells. Use this method in your applyFilter1 method.
   4. (25 pts) Name: applyFilter2
      1. The code must handle the ArrayIndexOutOfBoundsException when processing any array data. If the exception occurs the code must output an error to the user and exit.
      2. Parameters:
         1. Parameter:
            1. int[][] - Two-dimensional Array of integers that represents the image data.
      3. Return:
         1. int[][] – Two dimensional array of integers that have the updated values from applying filter 2.
      4. Description: Given the image array data apply filter2 and return the **new** filtered array.
2. Once the data is processed the new filtered data should be outputted in the same format as the sample output.

# Sample Output

Image 1 Data:

91 114 75 145 149 250 31 103 88 123 27 120 187 140 52

108 23 43 126 51 9 107 29 20 221 18 41 178 245 159

187 53 7 107 96 82 70 2 171 2 36 84 135 187 92

176 30 199 230 178 61 48 211 208 143 178 207 79 196 217

26 6 117 170 4 245 63 93 108 44 163 19 6 220 131

7 210 228 154 213 96 102 254 63 34 25 215 168 23 207

131 254 198 215 164 6 141 150 147 26 242 199 237 131 240

102 22 248 74 26 3 138 145 132 120 97 126 8 86 166

178 68 73 127 56 180 182 152 3 149 184 42 204 9 172

104 203 99 132 204 137 33 69 140 19 131 38 239 56 180

163 249 88 222 98 150 58 155 115 188 70 189 162 33 1

113 141 52 196 139 241 68 37 89 215 166 7 200 179 26

21 68 167 53 226 167 193 232 133 79 212 163 101 157 233

122 17 114 224 129 247 35 84 120 51 91 149 127 150 19

217 253 145 45 205 179 107 76 214 48 230 218 8 135 25

Image 1 Filter 1 Data:

104 75 94 123 148 109 122 62 83 114 72 93 156 156 117

102 68 54 94 86 99 49 52 105 76 68 88 157 181 137

131 60 81 113 102 63 61 96 80 114 63 100 132 171 163

104 92 116 176 113 122 90 112 168 115 145 113 124 179 159

53 77 144 135 162 93 110 145 103 98 85 122 98 115 193

93 141 181 196 126 132 131 132 121 38 135 125 129 149 150

123 163 228 161 124 82 107 167 103 113 117 203 148 143 186

108 138 123 138 64 70 121 143 109 104 153 94 132 80 166

113 108 123 92 118 111 137 110 115 95 120 118 100 105 131

162 144 119 156 125 140 95 109 69 125 88 127 139 103 102

157 168 142 147 162 136 92 86 137 121 148 93 164 86 60

109 124 128 132 180 153 119 116 117 147 134 145 129 119 109

81 82 90 173 142 214 139 135 130 138 142 126 149 164 108

94 114 133 113 206 151 133 109 120 77 146 149 107 117 106

197 158 139 154 139 184 99 120 114 135 146 151 122 79 59

Image 1 Filter 2 Data:

164 141 180 145 149 250 224 152 167 132 228 135 187 140 203

147 232 212 129 204 246 148 226 235 221 237 214 178 245 159

187 202 248 148 159 173 185 253 171 253 219 171 135 187 163

176 225 199 230 178 194 207 211 208 143 178 207 176 196 217

229 249 138 170 251 245 192 162 147 211 163 236 249 220 131

248 210 228 154 213 159 153 254 192 221 230 215 168 232 207

131 254 198 215 164 249 141 150 147 229 242 199 237 131 240

153 233 248 181 229 252 138 145 132 135 158 129 247 169 166

178 187 182 128 199 180 182 152 252 149 184 213 204 246 172

151 203 156 132 204 137 222 186 140 236 131 217 239 199 180

163 249 167 222 157 150 197 155 140 188 185 189 162 222 254

142 141 203 196 139 241 187 218 166 215 166 248 200 179 229

234 187 167 202 226 167 193 232 133 176 212 163 154 157 233

133 238 141 224 129 247 220 171 135 204 164 149 128 150 236

217 253 145 210 205 179 148 179 214 207 230 218 247 135 230

Image 2 Data:

201 159 87 63 240 244

231 32 222 76 5 255

10 5 248 139 47 64

167 76 138 177 107 159

188 122 154 165 205 22

222 149 148 85 129 57

Image 2 Filter 1 Data:

197 119 132 116 138 246

118 129 133 101 124 142

103 74 150 137 72 131

110 101 158 145 139 88

174 137 145 157 125 110

186 160 134 131 119 69

Image 2 Filter 2 Data:

201 159 168 192 240 244

231 223 222 179 250 255

245 250 248 139 208 191

167 179 138 177 148 159

188 133 154 165 205 233

222 149 148 170 129 198

Image 3 Data:

201 159 87 63 240

231 32 222 76 5

10 5 248 139 47

167 76 138 177 107

188 122 154 165 205

Image 3 Filter 1 Data:

197 119 132 116 102

118 129 133 101 92

103 74 150 137 74

110 101 158 145 134

159 135 144 175 159

Image 3 Filter 2 Data:

201 159 168 192 240

231 223 222 179 250

245 250 248 139 208

167 179 138 177 148

188 133 154 165 205

# Testing

I will test your program as follows:

javac Lab4\_<your email id>.java

java Lab4\_<your email id>

# Psuedo Code

## Main Method

Open File

While data is in file

Read number of rows

Read number of columns

Create 2d array

Read image data and put it into 2d array

Print image data

Call applyFilter1 method pass image data

Print filter1 data

Call applyFilter2 method pass image data

Print filter2 data

## Apply Filter 1 Method

Create 2d array to put filtered data into

For every row of image data

For every column of image data

Call getNeighbors method

Find average of neighbors data and current cell

Populate filtered data int 2 array

Return 2d array that has filtered data

## Apply Filter 2 Method

Create 2d array to put filtered data into

For every row of image data

For every column of image data

If (current cell value is less than the threshold)

Filtered data[r,c] = 255 - current cell value

Else

Filtered data[r,c] = current cell value

Return 2d array that has the filtered data