*********************************** COVER PAGE ***********************************

Class: CV
Name: Loyd Flores
Project: Project 2
Project Name: Bi-Means auto threshold selection
Language: Java
Due Date: 09/15/2024
Submit Date: 09/15/2024

==

Top-level algorithm steps

==

```
Step 0: Check if argc count is correct
    inFile ← open input file use args[0]
    histFile, GaussFile, logFile ← open from args[1], args[2], args[3]
    Check if all files can be opened.

Step 1: numRows, numCols, minVal, maxVal ← read from inFile
Step 2: Initialize the BiMeans class using image header values

Step 3: Load histogram data using loadHist()
Step 4: Print the histogram data in numeric format using printHist()
Step 5: Display the histogram data in 2-D format using dispHist()
Step 6: Perform Bi-Gaussian thresholding using biGaussian()
Step 7: Write the selected threshold value to GaussFile
Step 8: Output the best-fit Gaussian array in numerical format using printBestFitGauss()
Step 9: Plot and display the best-fit Gaussians using plotGaussGraph() and dispGaussGraph()
Step 10: Plot and display the gaps between the histogram and the best-fitted Gaussians using plotGapGraph() and dispGapGraph()
Step 11: Clean up and close all files
```

===

loadHist(inFile, logFileName)

===

```
Step 1: Log to log file upon function call
Step 2: Open the input file and read image header values
Step 3: Initialize histAry and other arrays based on maxVal
Step 4: Read histogram data from the input file and store in histAry
Step 5: Track the maximum value in the histogram to set histHeight
Step 6: Initialize GaussGraph and gapGraph with blank spaces
Step 7: Log to log file upon exit
```

```
==================================================
printHist(histCountFileName, logFileName)
==================================================
Step 1: Log to log file upon function call
Step 2: Write image header values to histCountFile
Step 3: Iterate through histAry and write each index and value to histCountFile
Step 4: Log to log file upon exit
==================================================
dispHist(histGraphFileName, logFileName)
==================================================
Step 1: Log to log file upon function call
Step 2: Write image header values to histGraphFile
Step 3: Iterate through histAry
Step 4: For each index, write the count and represent occurrences using '+' symbols
Step 5: Log to log file upon exit
==================================================
copyArys(ary1, ary2, logFileName)
==================================================
Step 1: Log to log file upon function call
Step 2: Check if ary1 and ary2 have the same length; throw an error if not
Step 3: Copy values from ary1 to ary2
Step 4: Log to log file upon exit
==================================================
setZero(ary, logFileName)
==================================================
Step 1: Log to log file upon function call
Step 2: Iterate through ary and set each element to 0
Step 3: Log to log file upon exit
==================================================
setBlanks(graph, logFileName)
==================================================
Step 1: Log to log file upon function call
Step 2: Iterate through graph, setting each element to a blank space ' '
Step 3: Log to log file upon exit
```

****

biGaussian(histAry, GaussAry, maxHeight, minVal, maxVal, Graph, logFileName)

****

```
Step 0: logFile ← output “Entering biGaussian method”
(double) sum1
(double) sum2
(double) total
(double) minSumDiff
offSet ← (int) (maxVal - minVal) / 10
dividePt ←  offSet
bestThr ← dividePt
minSumDiff ← 99999.0 // a large value

Step 1: setZero (GaussAry) // reset in each iteration
Step 2: sum1 ← fitGauss (0, dividePt, histAry, GaussAry, maxHeight, Graph, logFile) // first Gaussian curve
Step 3: sum2 ← fitGauss (dividePt, maxVal, histAry, GaussAry, maxHeight, Graph, logFile) //second Gaussian curve
Step 4: total ← sum1 + sum2
Step 5: if total < minSumDiff
minSumDiff ← total
bestThr ← dividePt
copyArys (GaussAry, bestFitGaussAry)

Step 6: logFile ← “In biGaussian (): dividePt = , sum1= , sum2= , total= , minSumDiff = and bestThr=”
//print those values.
Step 7: dividePt ++
Step 8: repeat step 1 to step 7 while dividePt < (maxVal – offSet)

Step 9:  logFile ← “leaving biGaussian method, minSumDiff =   bestThr is ” print minSumDiff and bestThr
step 10: return bestThr
```

****

fitGauss(leftIndex, rightIndex, histAry, GaussAry, maxHeight, Graph, logFileName)

****

```
Step 0: logFile ← “Entering fitGauss method”
(double) mean
(double) var
(double) sum ← 0.0
(double) Gval
Step 1: mean ←  computeMean (leftIndex, rightIndex, maxHeight, histAry, logFile)
var ← computeVar (leftIndex, rightIndex, mean, histAry, logFile)
```

```
Step 2: index ← leftIndex
Step 3: Gval ← modifiedGauss (index, mean, var, maxHeight) // see equation below.
Step 4: sum += abs (Gval – (double)histAry[index])
Step 5: GaussAry[index] ← (int) Gval
Step 6: index ++
Step 7: repeat step 3 – step 6 while index <= rightIndex
Step 8: logFile ← “leaving fitGauss method, sum is;” print sum // debug print
Step 9: return sum
************************************
```

computeMean(leftIndex, rightIndex, maxHeight, histAry, logFileName)

```
************************************
Step 0: logFile ← output “Entering computeMean method”
maxHeight ← 0 // maxHeight came via parameter, it is a reference variable, NOT local variable!
// If you like, maxHeight need NOT passes in the parameter, just use it as global variable.
sum ← 0
numPixels ← 0
Step 1: index ← leftIndex
Step 2: sum += (hist[index] * index)
numPixels += hist[index]
Step 3: if hist[index] > maxHeight
maxHeight ← hist[index]
Step 4: index++
Step 5: repeat Step 2 to step 4 while index < rightIndex

Step 6: (double) result ← (double) sum / (double) numPixels
Step 7: logFile ← output “Leaving computeMean method maxHeight =  result = ” print
maxHeight and result
Step 8: return result
************************************
```

computeVar(leftIndex, rightIndex, mean, histAry, logFileName)

```
************************************
Step 0: logFile ← output “Entering computeVar() method”
sum ← 0.0
numPixels ← 0
Step 1: index ← leftIndex
Step 2: sum += (double) hist [index] * ((double) index – mean)^2)
numPixels += hist[index]
Step 3: index++
Step 4: repeat Step 2 to step 3 while index < rightIndex
Step 5: (double) result ← sum / (double) numPixels
Step 6: logFile ← output “Leaving computeVar method returning result ” print result
Step 7: return result
```

```
************************************
modifiedGauss(x, mean, var, maxHeight, logFileName)
************************************
Step 0: Log to log file upon function call
Step 1: Compute Gaussian value using the formula: maxHeight * exp(-((x - mean)^2 / (2 * var)))
Step 2: Store the result in GaussAry
Step 3: Log to log file the computed value and input parameters
Step 4: Log to log file upon exit
Step 5: Return the computed Gaussian value
==============================================
printBestFitGauss(bestFitGaussAry, GaussFileName, logFileName)
==============================================
Step 1: Log to log file upon function call
Step 2: Open GaussFile for writing, appending if necessary
Step 3: Iterate through bestFitGaussAry
Step 4: Write each index and value to GaussFile
Step 5: Log to log file upon exit
Step 6: Close GaussFile
==============================================
plotGaussGraph(bestFitGaussAry, GaussGraph, logFileName)
==============================================
Step 1: Log to log file upon function call
Step 2: Clear the GaussGraph by setting all elements to blanks using setBlanks()
Step 3: Iterate through bestFitGaussAry
Step 4: For each index, plot the graph using '*' characters in reverse order
Step 5: Log to log file upon exit
==============================================
dispGaussGraph(GaussGraph, GaussFileName, logFileName)
==============================================
Step 1: Log to log file upon function call
Step 2: Open GaussFile for writing, appending if necessary
Step 3: Write the image header to GaussFile
Step 4: Iterate through GaussGraph
Step 5: For each element in the 2-D array, write to GaussFile to display the graph
Step 6: Log to log file upon exit
Step 7: Close GaussFile
==============================================
plotGapGraph(histAry, bestFitGaussAry, gapGraph, logFileName)
==============================================
Step 1: Log to log file upon function call
Step 2: Clear the gapGraph by setting all elements to blanks using setBlanks()
Step 3: Iterate through the histAry and bestFitGaussAry
Step 4: Determine the gap between the histogram and Gaussian values
Step 5: Plot the gaps using '@' characters in reverse order
Step 6: Log to log file upon exit
```

==

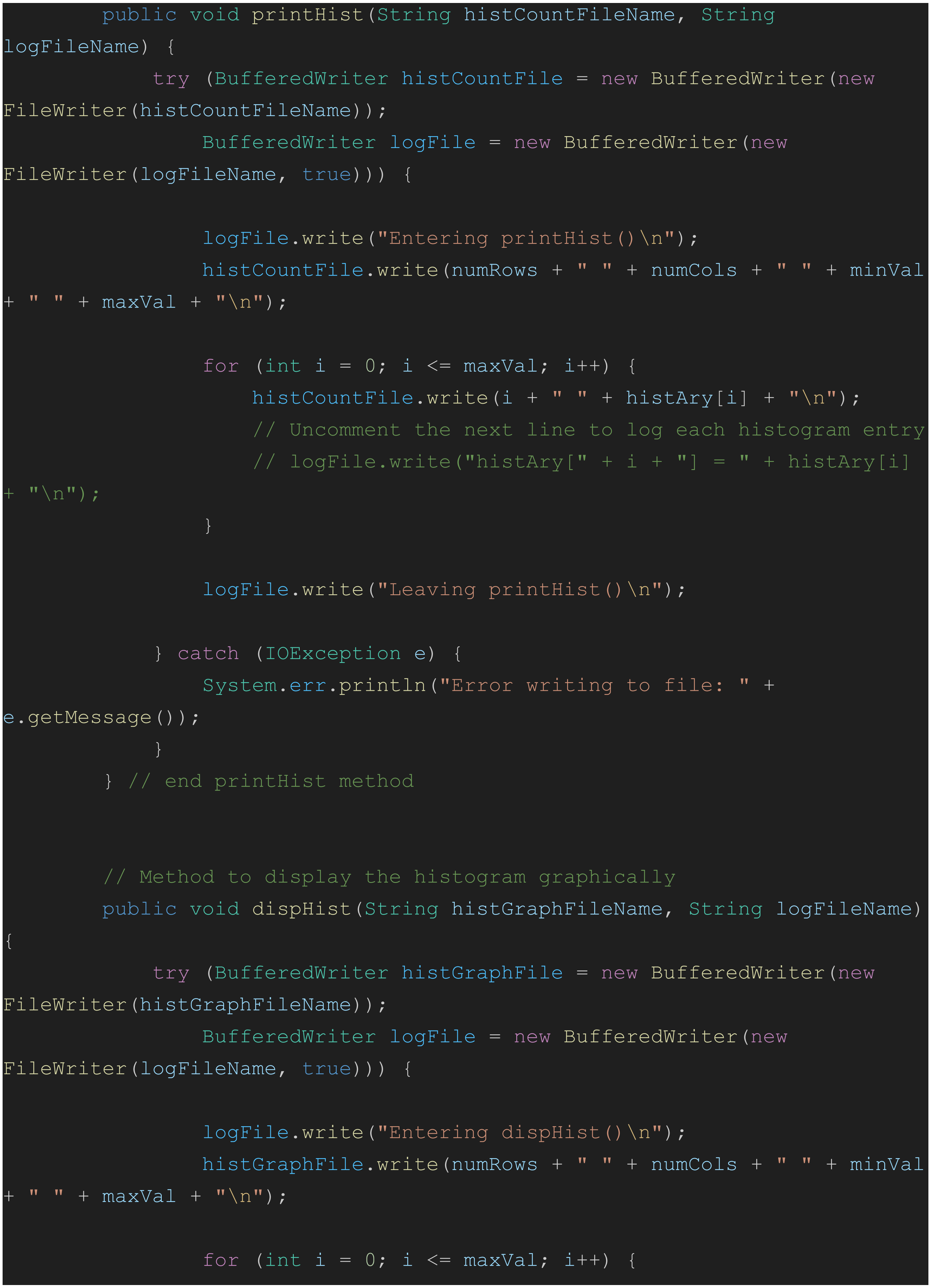
dispGapGraph(gapGraph, GaussFileName, logFileName)

==

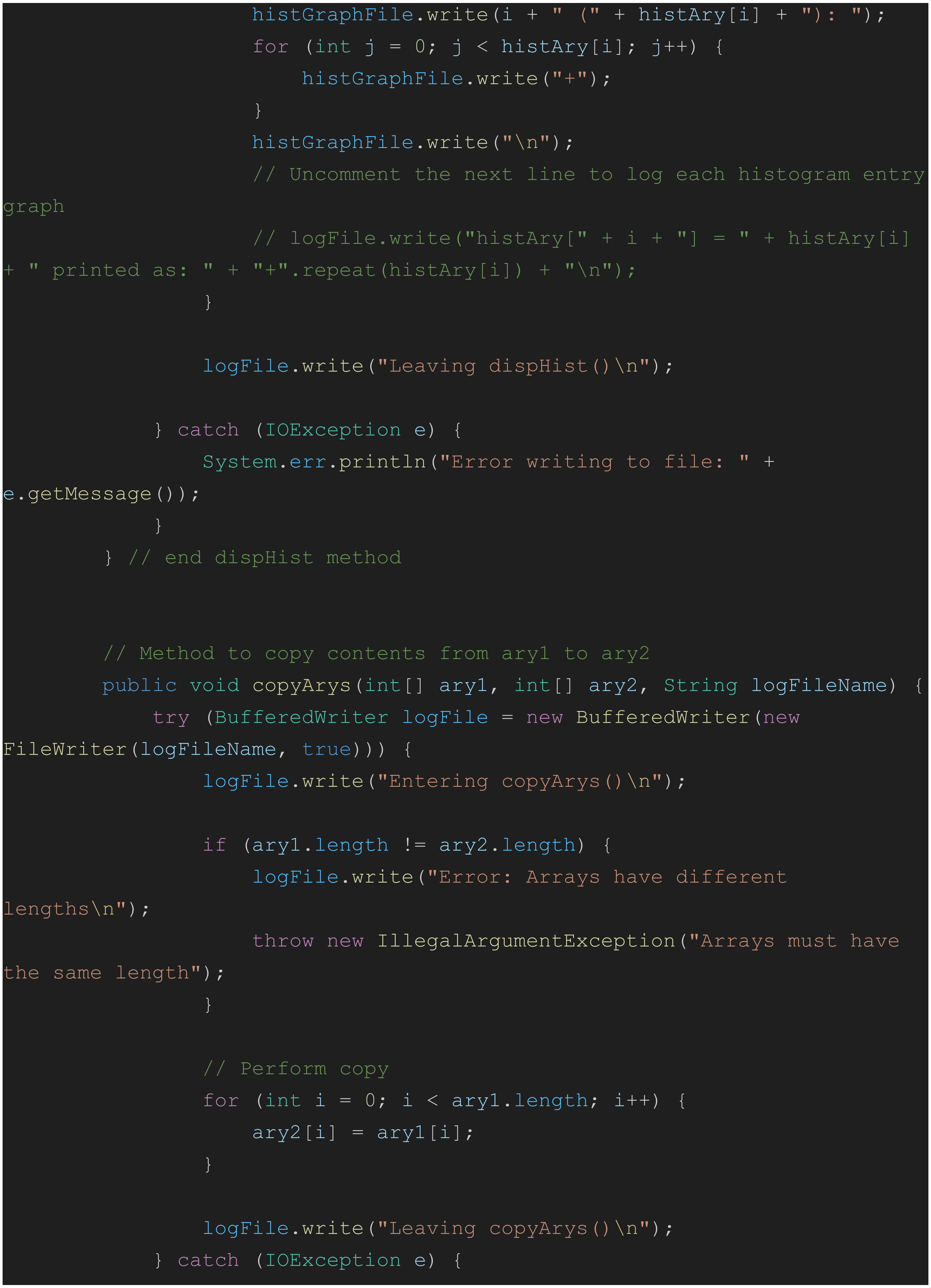
```
Step 1: Log to log file upon function call
Step 2: Open GaussFile for writing, appending if necessary
Step 3: Write the image header to GaussFile
Step 4: Iterate through gapGraph
Step 5: For each element in the 2-D array, write to GaussFile to display the graph
Step 6: Log to log file upon exit
Step 7: Close GaussFile
```

************************************ SOURCE CODE ************************************

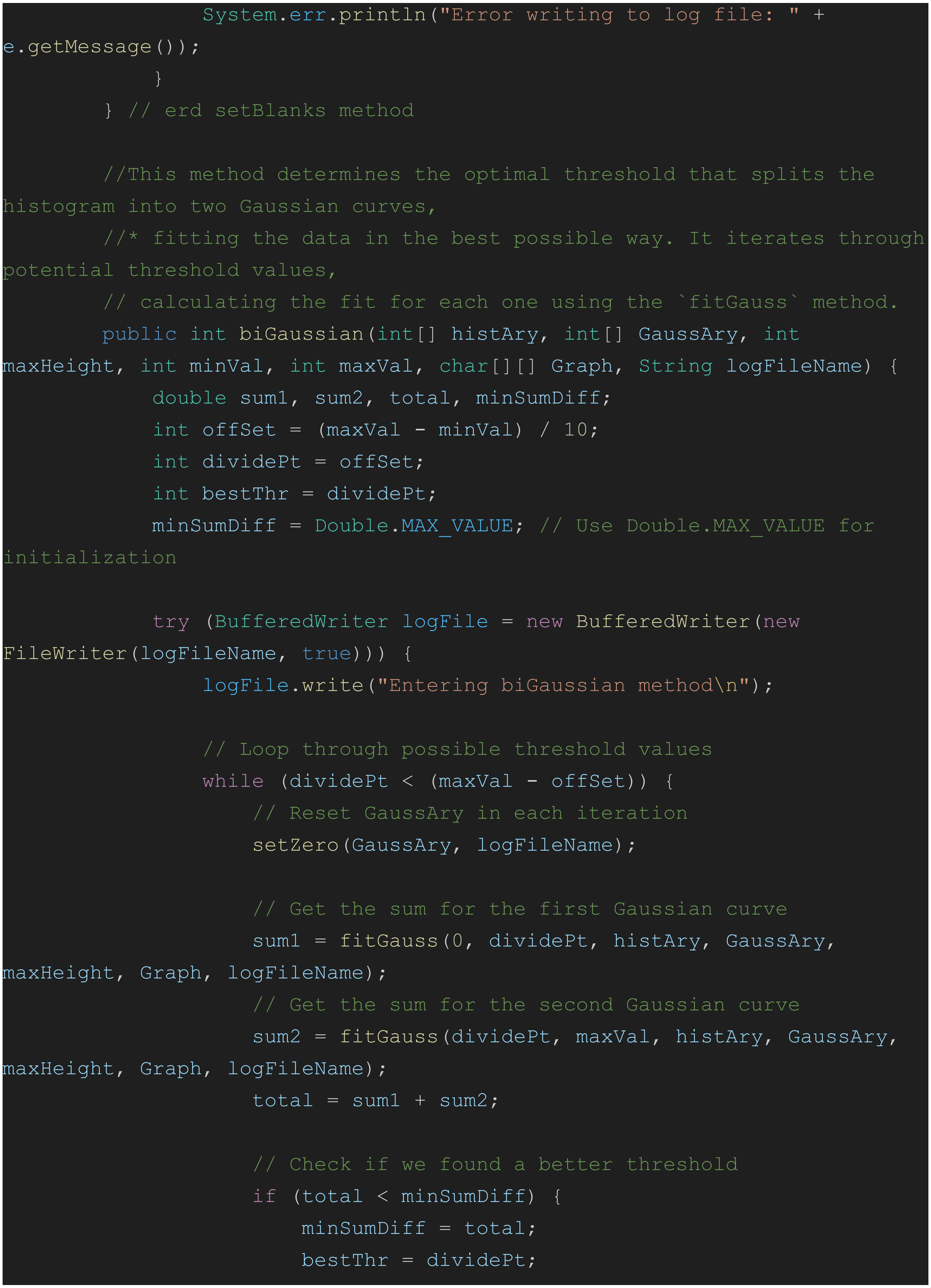
```java
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;

public class FloresL_Project2_Main {

    class BiMeans {
        // Image Header
        private int numRows;
        private int numCols;
        private int minVal;
        private int maxVal;

        // Bi-Gaussian method threshold value
        private int BiGaussThrVal;      //  Auto-selected threshold value
by the Bi-Gaussian method

        // Histogram attributes
        private int histHeight;         // The largest hist[i] in the
input histogram
        private int maxHeight = 0;      // The largest hist[i] within a
given range of the histogram

        // Arrays
        private int[] histAry;          // Stores histogram
        private int[] GaussAry;         // Modified Gaussian curve Values
        private int[] bestFitGaussAry;  // Best biGaussian Curves
        private char[][] GaussGraph;    // 2-D array for visualizing the
best-fit Gaussian curves
        private char[][] gapGraph;      // 2-D array for visualizing gaps
between histogram and Gaussians

        // Constructor to initialize and allocate arrays
        public BiMeans(int numRows, int numCols, int minVal, int maxVal) {
            this.numRows = numRows;
            this.numCols = numCols;
            this.minVal = minVal;
            this.maxVal = maxVal;

            // Initialize arrays with the appropriate sizes
            histAry = new int[maxVal + 1];
            GaussAry = new int[maxVal + 1];
            bestFitGaussAry = new int[maxVal + 1];
        } // end BiMeans Constructor


        // loads histogram from inputfile
        public int loadHist(String inFile, String logFileName) {
            int maxHistVal = 0;
            try (BufferedReader br = new BufferedReader(new
FileReader(inFile));
                BufferedWriter logFile = new BufferedWriter(new
FileWriter(logFileName, true))) {

                logFile.write("Entering loadHist()\n");

                // Read the first line to get the image header information
                String[] header = br.readLine().trim().split("\\s+");

                // Store header values
                numRows = Integer.parseInt(header[0]);
                numCols = Integer.parseInt(header[1]);
                minVal = Integer.parseInt(header[2]);
                maxVal = Integer.parseInt(header[3]);

                // Re-initialize the arrays if necessary based on updated
maxVal
                histAry = new int[maxVal + 1];
                GaussAry = new int[maxVal + 1];
                bestFitGaussAry = new int[maxVal + 1];

                // Read and load histogram data into histAry
                String line;
                while ((line = br.readLine()) != null) {
                    String[] parts = line.trim().split("\\s+");
                    int index = Integer.parseInt(parts[0]);
                    int value = Integer.parseInt(parts[1]);
                    // Check if index is within bounds
                    if (index >= 0 && index <= maxVal) {
                        histAry[index] = value;

                        // Keep track of the maximum histogram value
                        if (value > maxHistVal) {
                            maxHistVal = value;
                        }
                    } else {
                        logFile.write("Error: Index " + index + " out of
bounds\n");
                        System.err.println("Error: Index " + index + " out
of bounds");
                    }
                } // end-while loop

                histHeight = maxHistVal; // Update histHeight based on
loaded data

                // Initialize the 2D arrays with blank spaces based on
histHeight
                GaussGraph = new char[maxVal + 1][histHeight + 1];
                gapGraph = new char[maxVal + 1][histHeight + 1];
                setBlanks(GaussGraph, logFileName);
                setBlanks(gapGraph, logFileName);

                logFile.write("Leaving loadHist()\n");

            } catch (IOException e) {
                System.err.println("Error reading the file: " +
e.getMessage());
            } // end try-catch

            return maxHistVal;
        } // end loadHist Method


         // Method to print the histogram counts to a file
```



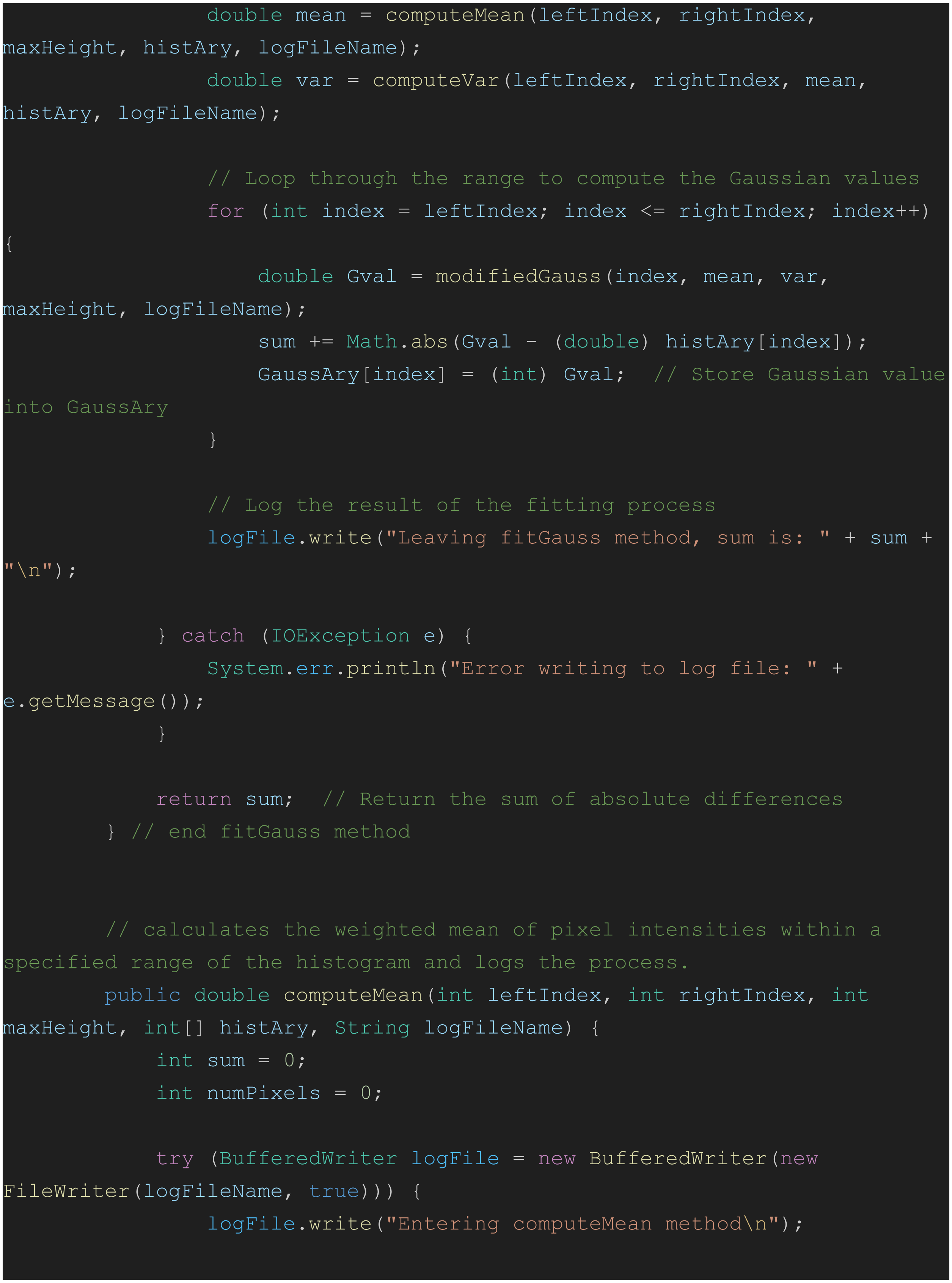
```
        public void printHist(String histCountFileName, String
logFileName) {
            try (BufferedWriter histCountFile = new BufferedWriter(new
FileWriter(histCountFileName));
                BufferedWriter logFile = new BufferedWriter(new
FileWriter(logFileName, true))) {

                logFile.write("Entering printHist()\n");
                histCountFile.write(numRows + " " + numCols + " " + minVal
+ " " + maxVal + "\n");

                for (int i = 0; i <= maxVal; i++) {
                    histCountFile.write(i + " " + histAry[i] + "\n");
                    // Uncomment the next line to log each histogram entry
                    // logFile.write("histAry[" + i + "] = " + histAry[i]
+ "\n");
                }

                logFile.write("Leaving printHist()\n");

            } catch (IOException e) {
                System.err.println("Error writing to file: " +
e.getMessage());
            }
        } // end printHist method


        // Method to display the histogram graphically
        public void dispHist(String histGraphFileName, String logFileName)
{
            try (BufferedWriter histGraphFile = new BufferedWriter(new
FileWriter(histGraphFileName));
                BufferedWriter logFile = new BufferedWriter(new
FileWriter(logFileName, true))) {

                logFile.write("Entering dispHist()\n");
                histGraphFile.write(numRows + " " + numCols + " " + minVal
+ " " + maxVal + "\n");

                for (int i = 0; i <= maxVal; i++) {
```



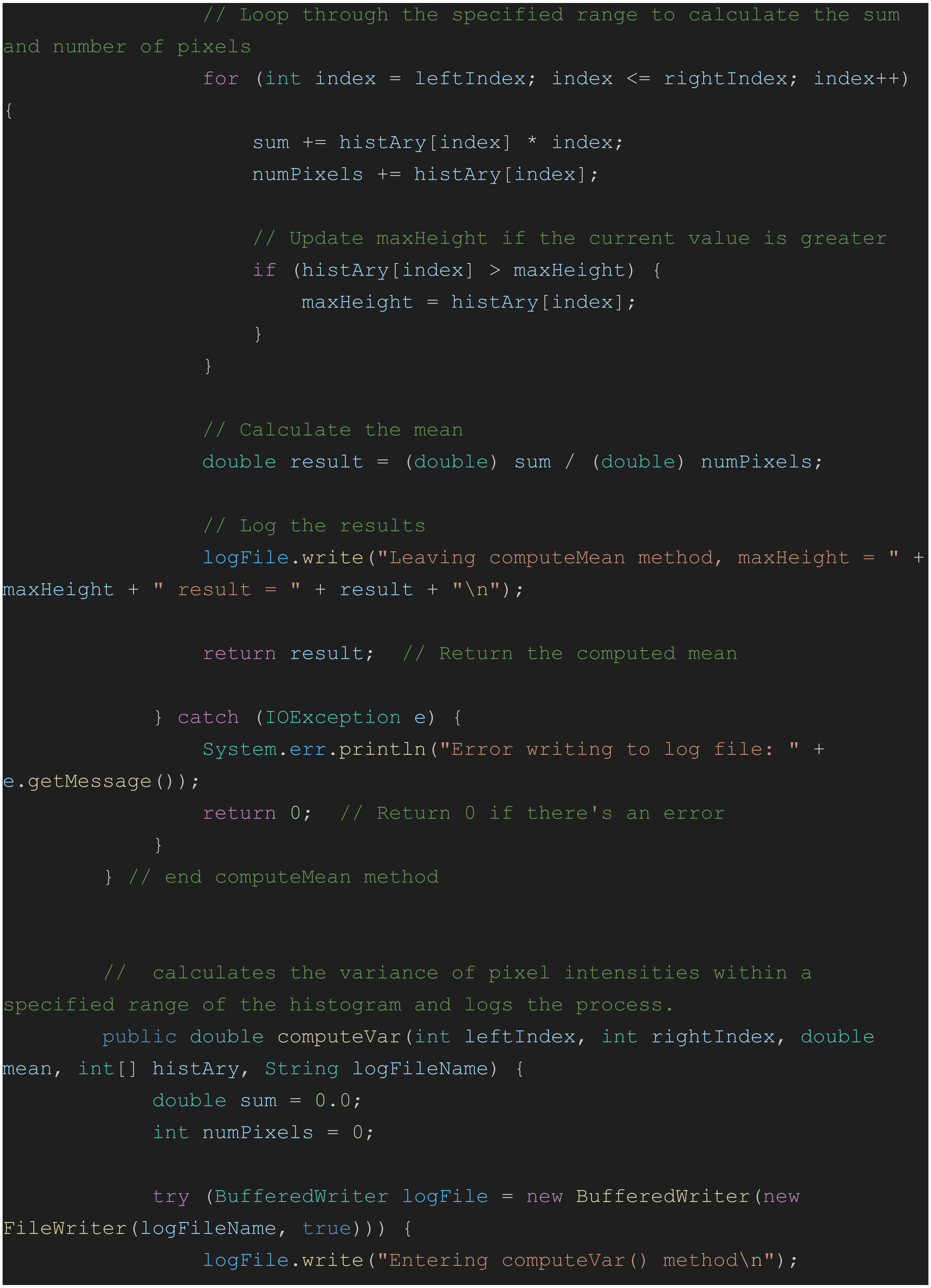
```
                    histGraphFile.write(i + " (" + histAry[i] + "): ");
                    for (int j = 0; j < histAry[i]; j++) {
                        histGraphFile.write("+");
                    }
                    histGraphFile.write("\n");
                    // Uncomment the next line to log each histogram entry
graph
                    // logFile.write("histAry[" + i + "] = " + histAry[i]
+ " printed as: " + "+".repeat(histAry[i]) + "\n");
                }

                logFile.write("Leaving dispHist()\n");

            } catch (IOException e) {
                System.err.println("Error writing to file: " +
e.getMessage());
            }
        } // end dispHist method


        // Method to copy contents from ary1 to ary2
        public void copyArys(int[] ary1, int[] ary2, String logFileName) {
            try (BufferedWriter logFile = new BufferedWriter(new
FileWriter(logFileName, true))) {
                logFile.write("Entering copyArys()\n");

                if (ary1.length != ary2.length) {
                    logFile.write("Error: Arrays have different
lengths\n");
                    throw new IllegalArgumentException("Arrays must have
the same length");
                }

                // Perform copy
                for (int i = 0; i < ary1.length; i++) {
                    ary2[i] = ary1[i];
                }

                logFile.write("Leaving copyArys()\n");
            } catch (IOException e) {
                System.err.println("Error writing to log file: " +
e.getMessage());
            }
        } // end copyArys method


        // Method to set all elements of a 1D array to zero
        public void setZero(int[] ary, String logFileName) {
            try (BufferedWriter logFile = new BufferedWriter(new
FileWriter(logFileName, true))) {
                logFile.write("Entering setZero()\n");

                // Perform zero assignment to all positions
                for (int i = 0; i < ary.length; i++) {
                    ary[i] = 0;
                }

                logFile.write("Leaving setZero()\n");
            } catch (IOException e) {
                System.err.println("Error writing to log file: " +
e.getMessage());
            }
        } // end setZero method


        // Method to set all elements of a 2D char array to blanks
        public void setBlanks(char[][] graph, String logFileName) {
            try (BufferedWriter logFile = new BufferedWriter(new
FileWriter(logFileName, true))) {
                logFile.write("Entering setBlanks()\n");

                // Method to set all positions to blanks
                for (int i = 0; i < graph.length; i++) {
                    for (int j = 0; j < graph[i].length; j++) {
                        graph[i][j] = ' ';
                    }
                }

                logFile.write("Leaving setBlanks()\n");
            } catch (IOException e) {
```



```
                System.err.println("Error writing to log file: " +
e.getMessage());
            }
        } // erd setBlanks method

        //This method determines the optimal threshold that splits the
histogram into two Gaussian curves,
        //* fitting the data in the best possible way. It iterates through
potential threshold values,
        // calculating the fit for each one using the `fitGauss` method.
        public int biGaussian(int[] histAry, int[] GaussAry, int
maxHeight, int minVal, int maxVal, char[][] Graph, String logFileName) {
            double sum1, sum2, total, minSumDiff;
            int offSet = (maxVal - minVal) / 10;
            int dividePt = offSet;
            int bestThr = dividePt;
            minSumDiff = Double.MAX_VALUE; // Use Double.MAX_VALUE for
initialization

            try (BufferedWriter logFile = new BufferedWriter(new
FileWriter(logFileName, true))) {
                logFile.write("Entering biGaussian method\n");

                // Loop through possible threshold values
                while (dividePt < (maxVal - offSet)) {
                    // Reset GaussAry in each iteration
                    setZero(GaussAry, logFileName);

                    // Get the sum for the first Gaussian curve
                    sum1 = fitGauss(0, dividePt, histAry, GaussAry,
maxHeight, Graph, logFileName);
                    // Get the sum for the second Gaussian curve
                    sum2 = fitGauss(dividePt, maxVal, histAry, GaussAry,
maxHeight, Graph, logFileName);
                    total = sum1 + sum2;

                    // Check if we found a better threshold
                    if (total < minSumDiff) {
                        minSumDiff = total;
                        bestThr = dividePt;
                        // Ensure this copies correctly
                        System.arraycopy(GaussAry, 0, bestFitGaussAry, 0, 
GaussAry.length);
                    }

                    // Logging current iteration details
                    logFile.write("In biGaussian (): dividePt = " + 
dividePt + ", sum1= " + sum1 + ", sum2= " + sum2 + ", total= " + total + 
", minSumDiff = " + minSumDiff + " and bestThr= " + bestThr + "\n");

                    dividePt++;
                } // end while-loop

                logFile.write("Leaving biGaussian method, minSumDiff = " + 
minSumDiff + " bestThr is " + bestThr + "\n");

            } catch (IOException e) {
                System.err.println("Error writing to log file: " + 
e.getMessage());
            } // end try-catch

            return bestThr;
        } // end biGaussian method





        //  This method computes the Gaussian curve fitting for a specific 
segment of the histogram.
        public double fitGauss(int leftIndex, int rightIndex, int[] 
histAry, int[] GaussAry, int maxHeight, char[][] Graph, String 
logFileName) {
            double sum = 0.0;
            try (BufferedWriter logFile = new BufferedWriter(new 
FileWriter(logFileName, true))) {
                logFile.write("Entering fitGauss method\n");

                // Calculate the mean and variance for the given segment
```

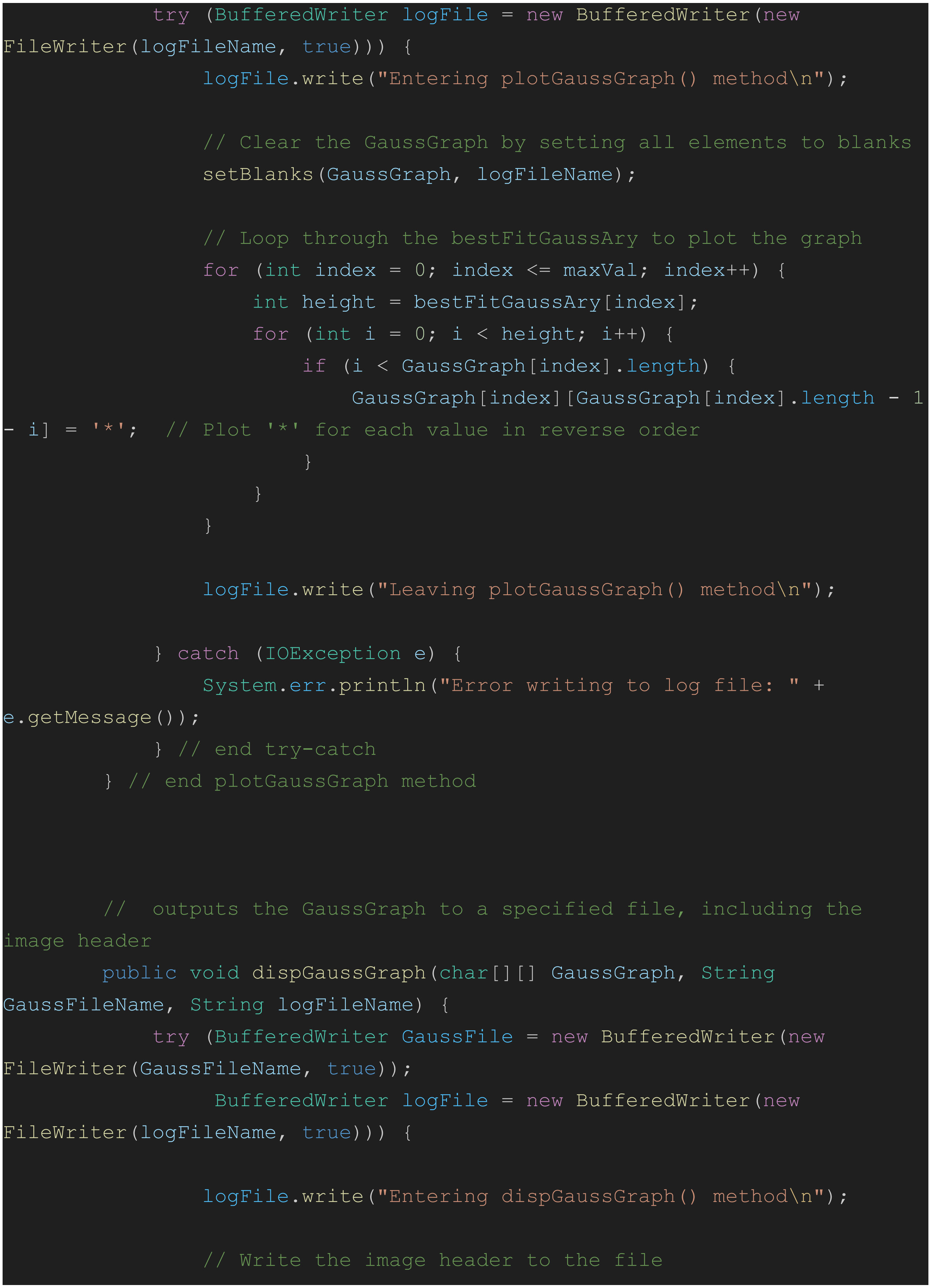


```
                double mean = computeMean(leftIndex, rightIndex,
maxHeight, histAry, logFileName);
                double var = computeVar(leftIndex, rightIndex, mean,
histAry, logFileName);

                // Loop through the range to compute the Gaussian values
                for (int index = leftIndex; index <= rightIndex; index++)
{
                    double Gval = modifiedGauss(index, mean, var,
maxHeight, logFileName);
                    sum += Math.abs(Gval - (double) histAry[index]);
                    GaussAry[index] = (int) Gval;  // Store Gaussian value
into GaussAry
                }

                // Log the result of the fitting process
                logFile.write("Leaving fitGauss method, sum is: " + sum +
"\n");

            } catch (IOException e) {
                System.err.println("Error writing to log file: " +
e.getMessage());
            }

            return sum;  // Return the sum of absolute differences
        } // end fitGauss method


        // calculates the weighted mean of pixel intensities within a
specified range of the histogram and logs the process.
        public double computeMean(int leftIndex, int rightIndex, int
maxHeight, int[] histAry, String logFileName) {
            int sum = 0;
            int numPixels = 0;

            try (BufferedWriter logFile = new BufferedWriter(new
FileWriter(logFileName, true))) {
                logFile.write("Entering computeMean method\n");
```

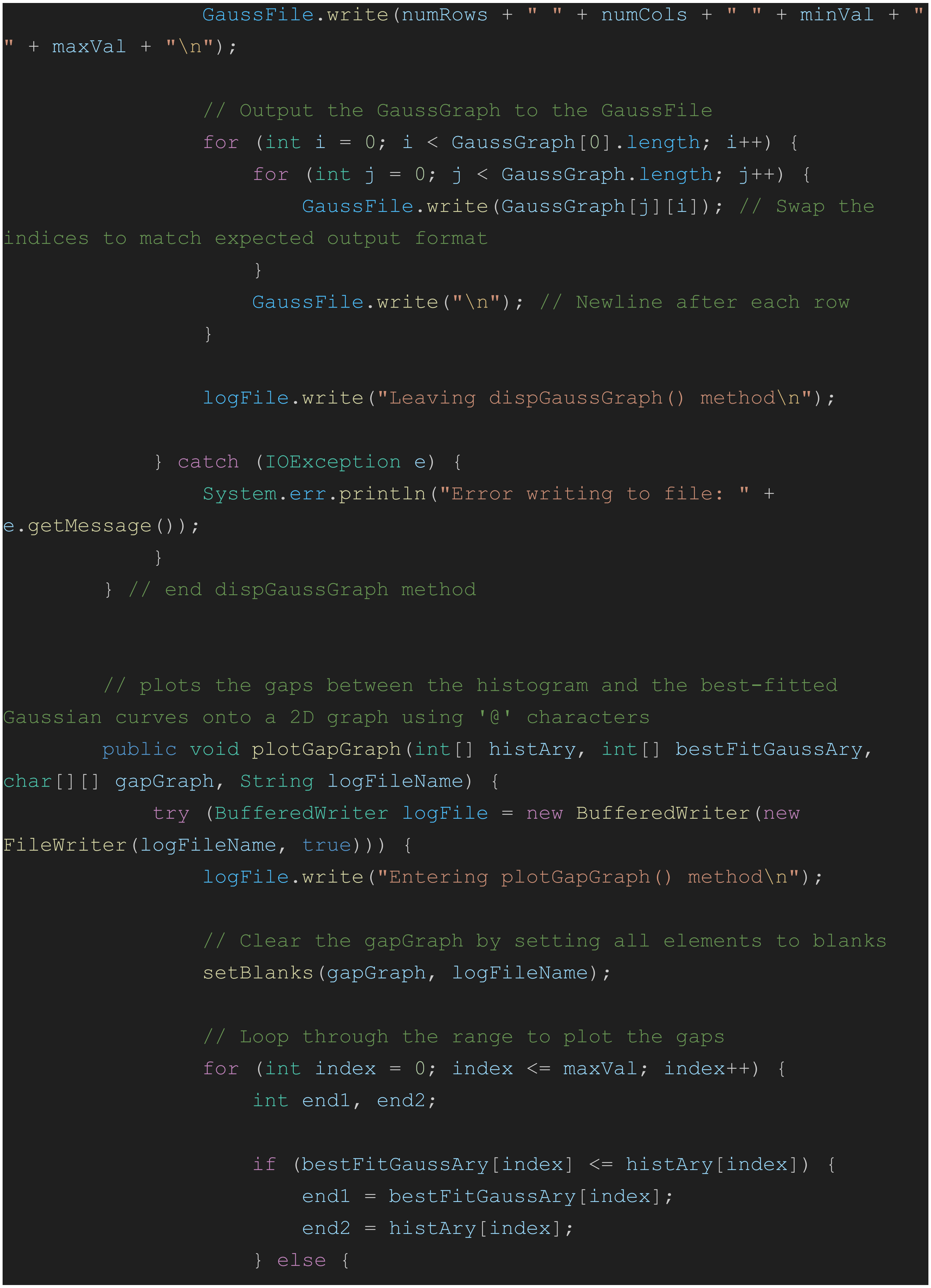


```
                // Loop through the specified range to calculate the sum
and number of pixels
                for (int index = leftIndex; index <= rightIndex; index++)
{
                    sum += histAry[index] * index;
                    numPixels += histAry[index];

                    // Update maxHeight if the current value is greater
                    if (histAry[index] > maxHeight) {
                        maxHeight = histAry[index];
                    }
                }

                // Calculate the mean
                double result = (double) sum / (double) numPixels;

                // Log the results
                logFile.write("Leaving computeMean method, maxHeight = " +
maxHeight + " result = " + result + "\n");

                return result;  // Return the computed mean

            } catch (IOException e) {
                System.err.println("Error writing to log file: " +
e.getMessage());
                return 0;  // Return 0 if there's an error
            }
        } // end computeMean method


        //  calculates the variance of pixel intensities within a
specified range of the histogram and logs the process.
        public double computeVar(int leftIndex, int rightIndex, double
mean, int[] histAry, String logFileName) {
            double sum = 0.0;
            int numPixels = 0;

            try (BufferedWriter logFile = new BufferedWriter(new
FileWriter(logFileName, true))) {
                logFile.write("Entering computeVar() method\n");
                // Loop through the specified range to calculate the sum
for variance
                for (int index = leftIndex; index <= rightIndex; index++)
{
                    sum += histAry[index] * Math.pow((index - mean), 2);
                    numPixels += histAry[index];
                }

                // Calculate the variance
                double result = sum / (double) numPixels;

                // Log the result
                logFile.write("Leaving computeVar method, returning result
= " + result + "\n");

                return result;  // Return the computed variance

            } catch (IOException e) {
                System.err.println("Error writing to log file: " +
e.getMessage());
                return 0;  // Return 0 if there's an error
            } // end try-catch
        } // end computeVar method


        //  computes the modified Gaussian value for a given x, using the
mean, variance, and maximum height of the histogram.
        public double modifiedGauss(int x, double mean, double var, int
maxHeight, String logFileName) {
            double result = maxHeight * Math.exp(-Math.pow((x - mean), 2)
/ (2 * var));

            try (BufferedWriter logFile = new BufferedWriter(new
FileWriter(logFileName, true))) {
                logFile.write("Entering modifiedGauss method\n");
                logFile.write("Computed modified Gaussian value for x = "
+ x + " with mean = " + mean + ", variance = " + var + ", maxHeight = " +
maxHeight + "\n");
```

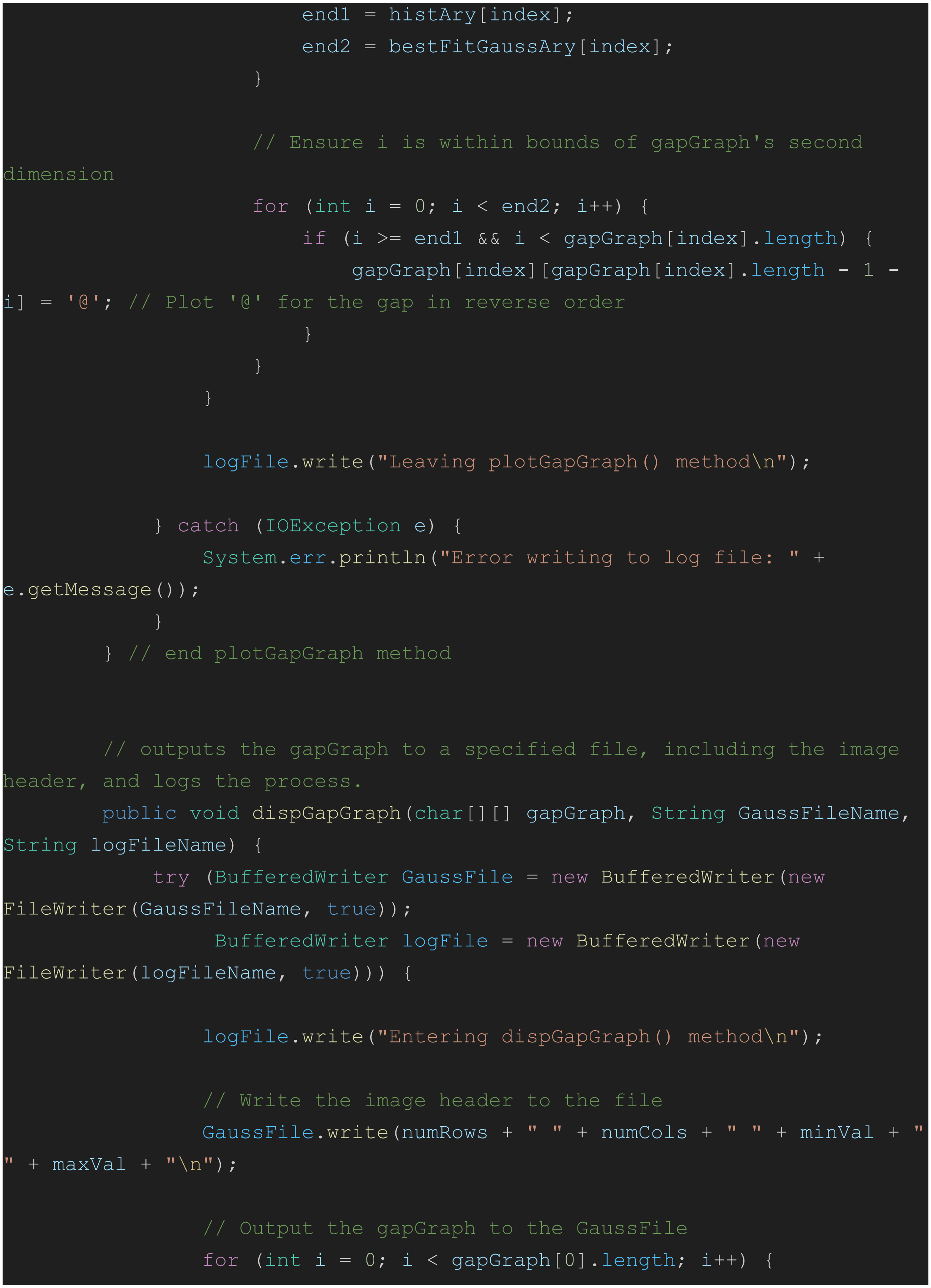
```
                logFile.write("Resulting Gaussian value = " + result +
"\n");
                logFile.write("Leaving modifiedGauss method\n");
            } catch (IOException e) {
                System.err.println("Error writing to log file: " +
e.getMessage());
            } // end try-catch

            return result;
        } // end modifiedGauss method


        public void printBestFitGauss(int[] bestFitGaussAry, String
GaussFileName, String logFileName) {
            try (BufferedWriter GaussFile = new BufferedWriter(new
FileWriter(GaussFileName, true));
                 BufferedWriter logFile = new BufferedWriter(new
FileWriter(logFileName, true))) {

                logFile.write("Entering printBestFitGauss method\n");

                // Output the bestFitGaussAry to the GaussFile
                for (int i = 0; i <= maxVal; i++) {
                    GaussFile.write(i + " " + bestFitGaussAry[i] + "\n");
                }

                logFile.write("Leaving printBestFitGauss method\n");

            } catch (IOException e) {
                System.err.println("Error writing to file: " +
e.getMessage());
            } // end try-catch
        } // end printBestFitGauss method


        // plots the best-fit Gaussian array onto a 2D graph using '*'
characters and logs the process.
        public void plotGaussGraph(int[] bestFitGaussAry, char[][]
GaussGraph, String logFileName) {
```



```
            try (BufferedWriter logFile = new BufferedWriter(new
FileWriter(logFileName, true))) {
                logFile.write("Entering plotGaussGraph() method\n");

                // Clear the GaussGraph by setting all elements to blanks
                setBlanks(GaussGraph, logFileName);

                // Loop through the bestFitGaussAry to plot the graph
                for (int index = 0; index <= maxVal; index++) {
                    int height = bestFitGaussAry[index];
                    for (int i = 0; i < height; i++) {
                        if (i < GaussGraph[index].length) {
                            GaussGraph[index][GaussGraph[index].length - 1
- i] = '*';  // Plot '*' for each value in reverse order
                        }
                    }
                }

                logFile.write("Leaving plotGaussGraph() method\n");

            } catch (IOException e) {
                System.err.println("Error writing to log file: " +
e.getMessage());
            } // end try-catch
        } // end plotGaussGraph method



        //  outputs the GaussGraph to a specified file, including the
image header
        public void dispGaussGraph(char[][] GaussGraph, String
GaussFileName, String logFileName) {
            try (BufferedWriter GaussFile = new BufferedWriter(new
FileWriter(GaussFileName, true));
                 BufferedWriter logFile = new BufferedWriter(new
FileWriter(logFileName, true))) {

                logFile.write("Entering dispGaussGraph() method\n");

                // Write the image header to the file
```



```
                GaussFile.write(numRows + " " + numCols + " " + minVal + "
" + maxVal + "\n");

                // Output the GaussGraph to the GaussFile
                for (int i = 0; i < GaussGraph[0].length; i++) {
                    for (int j = 0; j < GaussGraph.length; j++) {
                        GaussFile.write(GaussGraph[j][i]); // Swap the
indices to match expected output format
                    }
                    GaussFile.write("\n"); // Newline after each row
                }

                logFile.write("Leaving dispGaussGraph() method\n");

            } catch (IOException e) {
                System.err.println("Error writing to file: " +
e.getMessage());
            }
        } // end dispGaussGraph method


        // plots the gaps between the histogram and the best-fitted
Gaussian curves onto a 2D graph using '@' characters
        public void plotGapGraph(int[] histAry, int[] bestFitGaussAry,
char[][] gapGraph, String logFileName) {
            try (BufferedWriter logFile = new BufferedWriter(new
FileWriter(logFileName, true))) {
                logFile.write("Entering plotGapGraph() method\n");

                // Clear the gapGraph by setting all elements to blanks
                setBlanks(gapGraph, logFileName);

                // Loop through the range to plot the gaps
                for (int index = 0; index <= maxVal; index++) {
                    int end1, end2;

                    if (bestFitGaussAry[index] <= histAry[index]) {
                        end1 = bestFitGaussAry[index];
                        end2 = histAry[index];
                    } else {
```



```
                        end1 = histAry[index];
                        end2 = bestFitGaussAry[index];
                    }

                    // Ensure i is within bounds of gapGraph's second
dimension
                    for (int i = 0; i < end2; i++) {
                        if (i >= end1 && i < gapGraph[index].length) {
                            gapGraph[index][gapGraph[index].length - 1 -
i] = '@'; // Plot '@' for the gap in reverse order
                        }
                    }
                }

                logFile.write("Leaving plotGapGraph() method\n");

            } catch (IOException e) {
                System.err.println("Error writing to log file: " +
e.getMessage());
            }
        } // end plotGapGraph method


        // outputs the gapGraph to a specified file, including the image
header, and logs the process.
        public void dispGapGraph(char[][] gapGraph, String GaussFileName,
String logFileName) {
            try (BufferedWriter GaussFile = new BufferedWriter(new
FileWriter(GaussFileName, true));
                 BufferedWriter logFile = new BufferedWriter(new
FileWriter(logFileName, true))) {

                logFile.write("Entering dispGapGraph() method\n");

                // Write the image header to the file
                GaussFile.write(numRows + " " + numCols + " " + minVal + "
" + maxVal + "\n");

                // Output the gapGraph to the GaussFile
                for (int i = 0; i < gapGraph[0].length; i++) {
```

```java
                    for (int j = 0; j < gapGraph.length; j++) {
                        GaussFile.write(gapGraph[j][i]); // Swap the indices to match expected output format
                    }
                    GaussFile.write("\n"); // Newline after each row
                }

                logFile.write("Leaving dispGapGraph() method\n");

            } catch (IOException e) {
                System.err.println("Error writing to file: " + e.getMessage());
            }
        } // end dispGapGraph method
    } // end class BiMeans

    public static void main(String[] args) {
        // Step 0: Check if argc count is correct and each file can be opened.
        if (args.length != 4) {
            System.err.println("Usage: java FloresL_Project2_Main <inFile> <histFile> <GaussFile> <logFile>");
            return;
        }

        String inFile = args[0];
        String histFileName = args[1]; // This will be used now
        String GaussFileName = args[2];
        String logFileName = args[3];

        try {
            // Load histogram data to get image header values first
            BufferedReader br = new BufferedReader(new FileReader(inFile));
            String[] header = br.readLine().trim().split("\\s+");
            int numRows = Integer.parseInt(header[0]);
            int numCols = Integer.parseInt(header[1]);
            int minVal = Integer.parseInt(header[2]);
            int maxVal = Integer.parseInt(header[3]);
            br.close();
            // Step 1: Initialize and read input data
            BiMeans biMeans = new FloresL_Project2_Main().new
BiMeans(numRows, numCols, minVal, maxVal);

            // Load histogram data
            biMeans.histHeight = biMeans.loadHist(inFile, logFileName);

            // Output the histogram data in numeric format (i)
            biMeans.printHist(histFileName, logFileName);

            // Display the histogram data in 2-D format (ii)
            biMeans.dispHist(histFileName, logFileName);

            // Step 3: Perform Bi-Gaussian thresholding
            biMeans.BiGaussThrVal = biMeans.biGaussian(biMeans.histAry,
biMeans.GaussAry, biMeans.histHeight, biMeans.minVal, biMeans.maxVal,
biMeans.GaussGraph, logFileName);

            // Write the selected threshold value to the GaussFile
            try (BufferedWriter GaussFile = new BufferedWriter(new
FileWriter(GaussFileName, true))) {
                GaussFile.write("** The selected threshold value is " +
biMeans.BiGaussThrVal + "\n");
                GaussFile.write(numRows + " " + numCols + " " + minVal + "
" + maxVal + "\n");
            }

            // Step 4: Output the best-fit Gaussian array in numerical
format
            try (BufferedWriter GaussFile = new BufferedWriter(new
FileWriter(GaussFileName, true))) {
                GaussFile.write("** Below is the bestFitGaussAry array
**\n");
            }
            biMeans.printBestFitGauss(biMeans.bestFitGaussAry,
GaussFileName, logFileName);

            // Step 5: Plot and display the best-fit Gaussians
            biMeans.plotGaussGraph(biMeans.bestFitGaussAry,
biMeans.GaussGraph, logFileName);
            try (BufferedWriter GaussFile = new BufferedWriter(new
FileWriter(GaussFileName, true))) {
                GaussFile.write("** Above is the 2-D display of the best
fitted Gaussians (with *) **\n");
                biMeans.dispGaussGraph(biMeans.GaussGraph, GaussFileName,
logFileName);
            }

            // Step 6: Plot and display the gaps between the histogram and
the best-fitted Gaussians
            biMeans.plotGapGraph(biMeans.histAry, biMeans.bestFitGaussAry,
biMeans.gapGraph, logFileName);
            try (BufferedWriter GaussFile = new BufferedWriter(new
FileWriter(GaussFileName, true))) {
                GaussFile.write("** Above displays the gaps between the
histogram and the best fitted Gaussians (with @) **\n");
                biMeans.dispGapGraph(biMeans.gapGraph, GaussFileName,
logFileName);
            }

        } catch (IOException e) {
            System.err.println("Error: " + e.getMessage());
        }
    } // end main
} // End FloresL_Project2_Main.java class
```

```
************************************ Output histFile for hist1 ************************************
64 64 0 63
0 (10): ++++++++++
1 (14): ++++++++++++++
2 (17): +++++++++++++++++
3 (20): ++++++++++++++++++++
4 (22): ++++++++++++++++++++++
5 (31): +++++++++++++++++++++++++++++++
6 (28): ++++++++++++++++++++++++++++
7 (33): +++++++++++++++++++++++++++++++++
8 (45): +++++++++++++++++++++++++++++++++++++++++++++
9 (56): ++++++++++++++++++++++++++++++++++++++++++++++++++++++++
10 (70): ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
11 (90): ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
12 (120): ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
13 (150): ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
14 (192): ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
15 (210): ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
16 (192): ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
17 (172): ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
18 (132): ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
19 (100): ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
20 (89): +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
21 (78): ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
22 (42): ++++++++++++++++++++++++++++++++++++++++++
23 (20): ++++++++++++++++++++
24 (18): ++++++++++++++++++
25 (10): ++++++++++
26 (9): +++++++++
27 (8): ++++++++
28 (8): ++++++++
29 (7): +++++++
30 (6): ++++++
31 (5): +++++
32 (4): ++++
33 (4): ++++
34 (6): ++++++
35 (8): ++++++++
36 (10): ++++++++++
37 (12): ++++++++++++
38 (22): ++++++++++++++++++++++
39 (26): ++++++++++++++++++++++++++
40 (40): ++++++++++++++++++++++++++++++++++++++++
41 (45): +++++++++++++++++++++++++++++++++++++++++++++
42 (72): ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
43 (80): ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
44 (90): ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
45 (100): ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
46 (120): ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
47 (150): ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
48 (188): ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
49 (190): ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
50 (170): ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
51 (140): ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
52 (120): ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
53 (110): ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
54 (90): ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
55 (80): ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
56 (70): ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
57 (60): ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
58 (30): ++++++++++++++++++++++++++++++
59 (20): ++++++++++++++++++++
60 (12): ++++++++++++
61 (9): +++++++++
62 (8): ++++++++
63 (6): ++++++
```

************************************ Output GaussFilefor hist1 ************************************

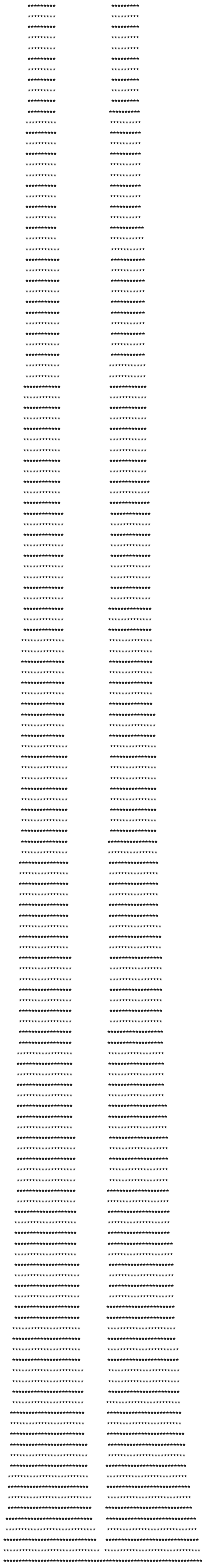
**** The selected threshold value is 32**

64 64 0 63

**** Below is the bestFitGaussAry array ****

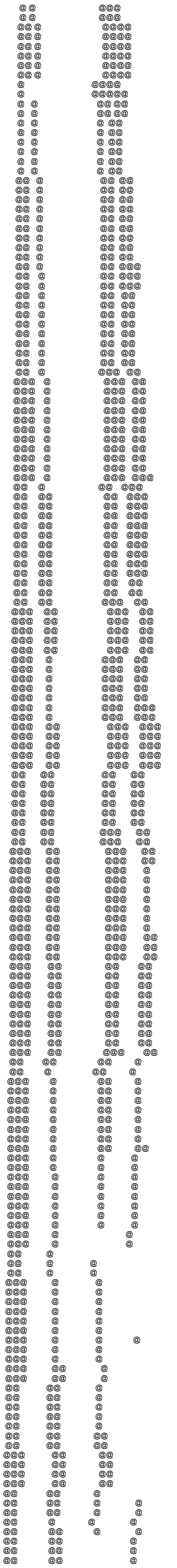
```
0 3
1 5
2 9
3 15
4 23
5 34
6 49
7 67
8 88
9 112
10 137
11 161
12 182
13 198
14 207
15 209
16 203
17 190
18 172
19 149
20 125
21 100
22 78
23 58
24 41
25 29
26 19
27 12
28 7
29 4
30 2
31 1
32 1
33 2
34 3
35 6
36 10
37 16
38 25
39 36
40 51
41 69
42 91
43 114
44 138
45 162
46 182
47 198
48 207
49 209
50 204
51 191
52 173
53 151
54 127
55 103
56 81
57 61
58 44
59 31
60 21
61 13
62 8
63 5
64 64 0 63
```

```
    *           *
      *           *
     **          **
     **          **
     **          **
     **          ***
     ***         ***
     ***         ***
     ***         ***
     ***         ***
     ***         ***
    ****        ****
    ****        ****
    ****        ****
    ****        ****
    ****        ****
    ****        ****
    ****        ****
    ****        *****
    *****       *****
    *****       *****
    *****       *****
    *****       *****
    *****       *****
    *****       *****
    *****       *****
    *****       *****
    ******      ******
    ******      ******
    ******      ******
    ******      ******
    ******      ******
    ******      ******
    ******      ******
    ******      ******
    ******      ******
    ******      *******
    *******     *******
    *******     *******
    *******     *******
    *******     *******
    *******     *******
    *******     *******
    *******     *******
    *******     *******
    *******     *******
    *******     *******
    *******     ********
    ********    ********
    ********    ********
    ********    ********
    ********    ********
    ********    ********
    ********    ********
    ********    ********
    ********    ********
    ********    ********
    ********    ********
    ********    *********
    ********    *********
    *********   *********
```



**** Above is the 2-D display of the best fitted Gaussians (with *) ****
64 64 0 63

```
   @
              @
              @
    @          @@
    @          @@
    @          @@
    @          @@@
    @ @         @@@
    @ @         @@@
    @ @         @@@
```



```
  @@      @@          @
  @@      @@    @      @
  @@      @@    @      @
 @@@       @@    @      @
 @@       @@    @      @
 @@       @@    @      @
  @      @@    @      @@
  @      @@    @       @
  @      @@@    @      @
      @@@      @       @
      @@@      @       @
      @@@           @
      @@@      @       @
      @@@      @       @
 @       @@@    @      @
      @@@           @
      @@@           @@
 @        @@          @
 @        @@@          @
 @        @@          @
 @@        @@          @
 @@        @@   @       @
 @       @@    @       @
@@         @@   @       @
@@         @@   @       @@
@@         @@@          @
@@         @@@          @
@@@         @@          @
@@       @
@@       @  @
@@        @ @
@@        @@ @        @
@        @@@ @
@        @@@@@
        @@@@
         @@
```

** Above displays the gaps between the histogram and the best fitted Gaussians (with @) **

```
*********************************** Output logFilefor hist1 ***********************************
Entering setBlanks()
Leaving setBlanks()
Entering setBlanks()
Leaving setBlanks()
Entering loadHist()
Leaving loadHist()
Entering printHist()
Leaving printHist()
Entering dispHist()
Leaving dispHist()
Entering setZero()
Leaving setZero()
Entering computeMean method
Leaving computeMean method, maxHeight = 210 result = 3.6549295774647885
Entering computeVar() method
Leaving computeVar method, returning result = 3.521771473913906
Entering modifiedGauss method
Computed modified Gaussian value for x = 0 with mean = 3.6549295774647885, variance =
3.521771473913906, maxHeight = 210
Resulting Gaussian value = 31.51760686257154
Leaving modifiedGauss method
Entering modifiedGauss method
Computed modified Gaussian value for x = 1 with mean = 3.6549295774647885, variance =
3.521771473913906, maxHeight = 210
Resulting Gaussian value = 77.19867636799329
Leaving modifiedGauss method
Entering modifiedGauss method
Computed modified Gaussian value for x = 2 with mean = 3.6549295774647885, variance =
3.521771473913906, maxHeight = 210
Resulting Gaussian value = 142.34736111780498
Leaving modifiedGauss method
Entering modifiedGauss method
Computed modified Gaussian value for x = 3 with mean = 3.6549295774647885, variance =
3.521771473913906, maxHeight = 210
Resulting Gaussian value = 197.59317203207368
Leaving modifiedGauss method
Entering modifiedGauss method
Computed modified Gaussian value for x = 4 with mean = 3.6549295774647885, variance =
3.521771473913906, maxHeight = 210
Resulting Gaussian value = 206.4797150473984
Leaving modifiedGauss method
```

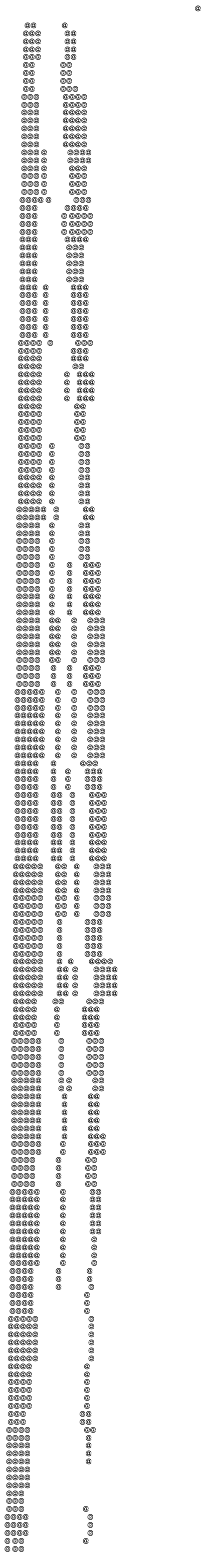
```
Entering modifiedGauss method
Computed modified Gaussian value for x = 5 with mean = 3.6549295774647885, variance =
3.521771473913906, maxHeight = 210
Resulting Gaussian value = 162.42983289959267
Leaving modifiedGauss method
Entering modifiedGauss method
Computed modified Gaussian value for x = 6 with mean = 3.6549295774647885, variance =
3.521771473913906, maxHeight = 210
Resulting Gaussian value = 96.19160012539179
Leaving modifiedGauss method
Entering fitGauss method
Leaving fitGauss method, sum is: 771.7579644528263
Entering computeMean method
Leaving computeMean method, maxHeight = 210 result = 32.94650929181316
Entering computeVar() method
Leaving computeVar method, returning result = 300.7894541635249
Entering modifiedGauss method
Computed modified Gaussian value for x = 6 with mean = 32.94650929181316, variance =
300.7894541635249, maxHeight = 210
Resulting Gaussian value = 62.80867498610171
Leaving modifiedGauss method
Entering modifiedGauss method
Computed modified Gaussian value for x = 7 with mean = 32.94650929181316, variance =
300.7894541635249, maxHeight = 210
Resulting Gaussian value = 68.58109127184964
Leaving modifiedGauss method
Entering modifiedGauss method
Computed modified Gaussian value for x = 8 with mean = 32.94650929181316, variance =
300.7894541635249, maxHeight = 210
Resulting Gaussian value = 74.6354752187613
Leaving modifiedGauss method
Entering modifiedGauss method
Computed modified Gaussian value for x = 9 with mean = 32.94650929181316, variance =
300.7894541635249, maxHeight = 210
Resulting Gaussian value = 80.95475534811987
Leaving modifiedGauss method
Entering modifiedGauss method
Computed modified Gaussian value for x = 10 with mean = 32.94650929181316, variance =
300.7894541635249, maxHeight = 210
Resulting Gaussian value = 87.51763600335359
Leaving modifiedGauss method
Entering modifiedGauss method
Computed modified Gaussian value for x = 11 with mean = 32.94650929181316, variance =
300.7894541635249, maxHeight = 210
```

```
Resulting Gaussian value = 94.2985343831342
Leaving modifiedGauss method
Entering modifiedGauss method
Computed modified Gaussian value for x = 12 with mean = 32.94650929181316, variance =
300.7894541635249, maxHeight = 210
Resulting Gaussian value = 101.26758631292026
Leaving modifiedGauss method
Entering modifiedGauss method
Computed modified Gaussian value for x = 13 with mean = 32.94650929181316, variance =
300.7894541635249, maxHeight = 210
Resulting Gaussian value = 108.39072619855133
Leaving modifiedGauss method
Entering modifiedGauss method
Computed modified Gaussian value for x = 14 with mean = 32.94650929181316, variance =
300.7894541635249, maxHeight = 210
Resulting Gaussian value = 115.62984524979709
Leaving modifiedGauss method
Entering modifiedGauss method
Computed modified Gaussian value for x = 15 with mean = 32.94650929181316, variance =
300.7894541635249, maxHeight = 210
Resulting Gaussian value = 122.94303048891956
Leaving modifiedGauss method
Entering modifiedGauss method
Computed modified Gaussian value for x = 16 with mean = 32.94650929181316, variance =
300.7894541635249, maxHeight = 210
Resulting Gaussian value = 130.28488530148203
Leaving modifiedGauss method
Entering modifiedGauss method
Computed modified Gaussian value for x = 17 with mean = 32.94650929181316, variance =
300.7894541635249, maxHeight = 210
Resulting Gaussian value = 137.6069303847137
Leaving modifiedGauss method
Entering modifiedGauss method
Computed modified Gaussian value for x = 18 with mean = 32.94650929181316, variance =
300.7894541635249, maxHeight = 210
Resulting Gaussian value = 144.8580819513789
Leaving modifiedGauss method
Entering modifiedGauss method
Computed modified Gaussian value for x = 19 with mean = 32.94650929181316, variance =
300.7894541635249, maxHeight = 210
Resulting Gaussian value = 151.98520200947664
Leaving modifiedGauss method
Entering modifiedGauss method
```

```
*********************************** Output histFile hist2 ***********************************
64 64 1 60
0 (0):
1 (1): +
2 (3): +++
3 (5): +++++
4 (4): ++++
5 (5): +++++
6 (7): +++++++
7 (4): ++++
8 (6): ++++++
9 (10): ++++++++++
10 (12): ++++++++++++
11 (15): +++++++++++++++
12 (10): ++++++++++
13 (14): ++++++++++++++
14 (15): +++++++++++++++
15 (22): ++++++++++++++++++++++
16 (20): ++++++++++++++++++++
17 (18): ++++++++++++++++++
18 (28): ++++++++++++++++++++++++++++
19 (38): ++++++++++++++++++++++++++++++++++++++
20 (44): ++++++++++++++++++++++++++++++++++++++++++++
21 (56): ++++++++++++++++++++++++++++++++++++++++++++++++++++++++
22 (70): ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
23 (90): ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
24 (120): ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
25 (150): ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
26 (190):
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++++++++++++++++
27 (214):
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++++++++++++++++++++++++++++++++++++++++
28 (190):
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++++++++++++++++
29 (172):
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++
30 (132): ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
31 (100): ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
32 (89): +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
33 (78): ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
34 (72): ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
35 (80): ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
36 (90): ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
37 (100): ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
38 (120): ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
39 (165):
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+++++
40 (186):
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++++++++++++
41 (195):
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+++++++++++++++++++++++++++++++++++
42 (185):
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+++++++++++++++++++++++++
43 (170):
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++
44 (165):
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+++++
45 (120): ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
46 (90): ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
47 (80): ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
48 (70): ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
49 (60): ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
50 (54): ++++++++++++++++++++++++++++++++++++++++++++++++++++++
51 (35): +++++++++++++++++++++++++++++++++++
52 (31): +++++++++++++++++++++++++++++++
53 (21): +++++++++++++++++++++
54 (19): +++++++++++++++++++
55 (12): ++++++++++++
56 (10): ++++++++++
57 (9): +++++++++
58 (11): +++++++++++
59 (8): ++++++++
60 (6): ++++++
*********************************** Output GaussFile hist2 ***********************************
** The selected threshold value is 34
64 64 1 60
** Below is the bestFitGaussAry array **
0 0
1 0
2 0
3 0
4 0
5 0
6 0
7 0
8 1
9 2
10 4
11 7
12 11
13 17
14 25
15 36
16 50
17 66
18 85
19 107
20 129
21 152
22 173
23 191
24 204
25 212
26 213
27 208
28 197
29 180
30 160
31 138
32 116
33 94
34 53
35 72
36 95
37 119
38 145
39 169
40 189
41 204
42 212
43 213
44 205
45 190
46 169
47 145
48 120
49 95
50 73
51 54
52 38
53 26
54 17
55 10
56 6
57 3
58 2
59 1
60 0
64 64 1 60
```

```
               *           *
              **           **
              **           **
              **           **
              **           **
              ***          **
              ***          **
              ***          **
              ***          ***
             ****         ****
             ****         ****
             ****         ****
             ****         ****
             ****         ****
             ****         ****
             ****         ****
             *****        ****
             *****        ****
             *****        ****
             *****        ****
             *****        ****
             *****        ****
            ******        ****
            ******        *****
            ******       ******
            ******       ******
            ******       ******
            ******       ******
            ******       ******
            ******       ******
            ******       ******
            ******       ******
            ******       ******
            *******       ******
            *******       ******
            *******       ******
            *******       ******
            *******       ******
            *******       ******
            *******       ******
           ********       ******
           ********       ******
           ********       ******
           ********       ******
           ********      ********
           ********      ********
           ********      ********
           ********      ********
           ********      ********
           ********      ********
           ********      ********
           ********      ********
           ********      ********
           *********      ********
           *********      ********
           *********      ********
           *********      ********
           *********      ********
           *********      ********
           *********      ********
           *********      ********
          **********     ********
          **********     ********
          **********     ********
          **********     ********
          **********     ********
          **********     ********
          **********     ********
          **********     **********
          **********     **********
          **********     **********
          **********     **********
          **********     **********
          **********     **********
          **********     **********
          ***********    **********
          ***********    **********
          ***********    **********
          ***********    **********
          ***********    **********
          ***********    **********
          ***********    **********
          ***********    **********
          ***********    **********
         ************    **********
         ************    **********
         ************    **********
         ************    **********
         ************    **********
         ************    **********
         ************    **********
         ************    **********
         ************    **********
         ************    ***********
         ************   ************
         ************   ************
         ************   ************
         *************   ************
         *************   ************
         *************   ************
         *************   ************
         *************   ************
         *************   ************
         *************   ************
         *************   ************
         *************   ************
        **************   ************
        **************   ************
        **************   ************
        **************   ************
        **************   ************
        **************   ************
        **************   ************
        **************   ************
        **************   ************
        **************   ************
        **************   ************
        **************   ************
        **************  **************
        *************** **************
        *************** **************
        *************** **************
        *************** **************
        *************** **************
        *************** **************
        *************** **************
        *************** **************
        *************** **************
       **************** **************
       **************** **************
       **************** **************
       **************** **************
       **************** **************
       **************** **************
       **************** **************
       **************** **************
       **************** **************
       **************** **************
       **************** **************
       **************** **************
       **************** ***************
       **************** ****************
       **************** ****************
       **************** ****************
       **************** ****************
       **************** ****************
       **************** ****************
      ***************** ****************
      ***************** ****************
      ***************** ****************
      ***************** ****************
      ***************** ****************
      ***************** ****************
      ***************** ****************
      ***************** ****************
      ***************** ****************
      ***************** ****************
      ***************** ****************
      ***************** ****************
      ***************** *****************
      ***********************************
      ***********************************
      ***********************************
     ************************************
     ************************************
     ************************************
     ************************************
     ************************************
     ************************************
     ************************************
     ************************************
     ************************************
     ************************************
     ************************************
     ************************************
     *************************************
     *************************************
    **************************************
    **************************************
    **************************************
    **************************************
    **************************************
    **************************************
    **************************************
    **************************************
    **************************************
    **************************************
    ***************************************
   ****************************************
   ****************************************
   ****************************************
   ****************************************
   ****************************************
   ****************************************
   ****************************************
   ****************************************
  ******************************************
  ******************************************
  ******************************************
  ******************************************
  ******************************************
  ******************************************
 *******************************************
 ********************************************
 ********************************************
 ********************************************
*********************************************
**********************************************
```

```
** Above is the 2-D display of the best fitted Gaussians (with *) **
64 64 1 60
```



```
    @ @
    @ @           @
    @            @
    @@
    @@
   @ @
   @
   @
   @@              @
   @@@              @ @
  @@@               @ @
  @@@               @@@
  @@@               @@@@
 @ @@               @@@@
 @ @@@               @@@@
@ @@ @@@               @@@@
@@@@@@@@                @@@@
@@@@@@@@@                @@@
@@@@@@@@                @@
@@@@@@@@                @
```

**** Above displays the gaps between the histogram and the best fitted Gaussians (with @) ****

```
************************************ Output logFilefor hist2 ************************************
 Entering setBlanks()
Leaving setBlanks()
Entering setBlanks()
Leaving setBlanks()
Entering loadHist()
Error: Index 61 out of bounds
Error: Index 62 out of bounds
Error: Index 63 out of bounds
Leaving loadHist()
Entering printHist()
Leaving printHist()
Entering dispHist()
Leaving dispHist()
Entering setZero()
Leaving setZero()
Entering computeMean method
Leaving computeMean method, maxHeight = 214 result = 3.5
Entering computeVar() method
Leaving computeVar method, returning result = 1.4722222222222223
Entering modifiedGauss method
Computed modified Gaussian value for x = 0 with mean = 3.5, variance =
1.4722222222222223, maxHeight = 214
Resulting Gaussian value = 3.3387572487321098
Leaving modifiedGauss method
Entering modifiedGauss method
Computed modified Gaussian value for x = 1 with mean = 3.5, variance =
1.4722222222222223, maxHeight = 214
Resulting Gaussian value = 25.619006196911876
Leaving modifiedGauss method
Entering modifiedGauss method
Computed modified Gaussian value for x = 2 with mean = 3.5, variance =
1.4722222222222223, maxHeight = 214
Resulting Gaussian value = 99.66604742992752
Leaving modifiedGauss method
Entering modifiedGauss method
Computed modified Gaussian value for x = 3 with mean = 3.5, variance =
1.4722222222222223, maxHeight = 214
Resulting Gaussian value = 196.58017328652775
Leaving modifiedGauss method
Entering modifiedGauss method
```

```
Computed modified Gaussian value for x = 4 with mean = 3.5, variance =
1.4722222222222223, maxHeight = 214
Resulting Gaussian value = 196.58017328652775
Leaving modifiedGauss method
Entering modifiedGauss method
Computed modified Gaussian value for x = 5 with mean = 3.5, variance =
1.4722222222222223, maxHeight = 214
Resulting Gaussian value = 99.66604742992752
Leaving modifiedGauss method
Entering fitGauss method
Leaving fitGauss method, sum is: 603.4502048785545
Entering computeMean method
Leaving computeMean method, maxHeight = 214 result = 34.53661523389665
Entering computeVar() method
Leaving computeVar method, returning result = 98.65228411034347
Entering modifiedGauss method
Computed modified Gaussian value for x = 5 with mean = 34.53661523389665, variance =
98.65228411034347, maxHeight = 214
Resulting Gaussian value = 2.571081288261703
Leaving modifiedGauss method
Entering modifiedGauss method
Computed modified Gaussian value for x = 6 with mean = 34.53661523389665, variance =
98.65228411034347, maxHeight = 214
Resulting Gaussian value = 3.4509842115819502
Leaving modifiedGauss method
Entering modifiedGauss method
Computed modified Gaussian value for x = 7 with mean = 34.53661523389665, variance =
98.65228411034347, maxHeight = 214
Resulting Gaussian value = 4.585301127711786
Leaving modifiedGauss method
Entering modifiedGauss method
Computed modified Gaussian value for x = 8 with mean = 34.53661523389665, variance =
98.65228411034347, maxHeight = 214
Resulting Gaussian value = 6.031015947397513
Leaving modifiedGauss method
Entering modifiedGauss method
Computed modified Gaussian value for x = 9 with mean = 34.53661523389665, variance =
98.65228411034347, maxHeight = 214
Resulting Gaussian value = 7.85255192069604
Leaving modifiedGauss method
Entering modifiedGauss method
Computed modified Gaussian value for x = 10 with mean = 34.53661523389665, variance =
98.65228411034347, maxHeight = 214
Resulting Gaussian value = 10.12112716264779
```

```
Leaving modifiedGauss method
Entering modifiedGauss method
Computed modified Gaussian value for x = 11 with mean = 34.53661523389665, variance =
98.65228411034347, maxHeight = 214
Resulting Gaussian value = 12.91352094315245
Leaving modifiedGauss method
Entering modifiedGauss method
Computed modified Gaussian value for x = 12 with mean = 34.53661523389665, variance =
98.65228411034347, maxHeight = 214
Resulting Gaussian value = 16.31015867150428
Leaving modifiedGauss method
Entering modifiedGauss method
Computed modified Gaussian value for x = 13 with mean = 34.53661523389665, variance =
98.65228411034347, maxHeight = 214
Resulting Gaussian value = 20.392450927788733
Leaving modifiedGauss method
Entering modifiedGauss method
Computed modified Gaussian value for x = 14 with mean = 34.53661523389665, variance =
98.65228411034347, maxHeight = 214
Resulting Gaussian value = 25.239363054677803
Leaving modifiedGauss method
Entering modifiedGauss method
Computed modified Gaussian value for x = 15 with mean = 34.53661523389665, variance =
98.65228411034347, maxHeight = 214
Resulting Gaussian value = 30.923246379018185
Leaving modifiedGauss method
Entering modifiedGauss method
Computed modified Gaussian value for x = 16 with mean = 34.53661523389665, variance =
98.65228411034347, maxHeight = 214
Resulting Gaussian value = 37.50502813463987
Leaving modifiedGauss method
Entering modifiedGauss method
Computed modified Gaussian value for x = 17 with mean = 34.53661523389665, variance =
98.65228411034347, maxHeight = 214
Resulting Gaussian value = 45.02893086727978
Leaving modifiedGauss method
Entering modifiedGauss method
Computed modified Gaussian value for x = 18 with mean = 34.53661523389665, variance =
98.65228411034347, maxHeight = 214
Resulting Gaussian value = 53.51696799439103
Leaving modifiedGauss method
Entering modifiedGauss method
Computed modified Gaussian value for x = 19 with mean = 34.53661523389665, variance =
98.65228411034347, maxHeight = 214
```