Week 11: Homework 2: Project: Pattern Recognition

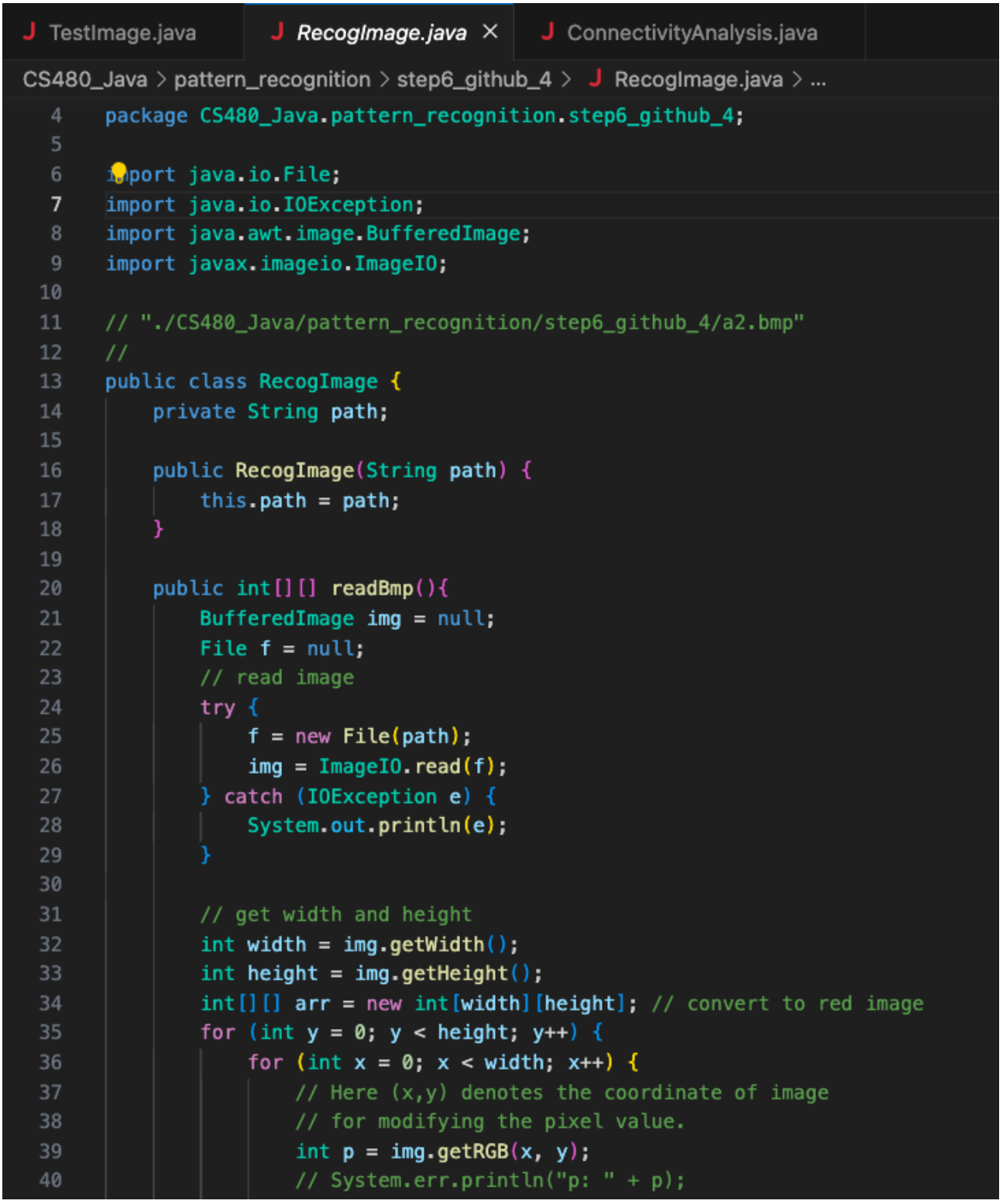
1. Code Snapshot:

TestImage.java

CS480_Java > pattern_recognition > step6_github_4 > J TestImage.java > ...

```java
package CS480_Java.pattern_recognition.step6_github_4;

// import CS480_Java.pattern_recognition.step6_github_4.RecogImage;

public class TestImage {
    Run | Debug
    public static void main(String[] args) {
        String filepath = "./CS480_Java/pattern_recognition/step6_github_4/a2.bmp";
        RecogImage reco =  new RecogImage(filepath);

        int[][] pixels =  reco.readBmp();

        ProcessImage process = new ProcessImage(pixels);
        int[][] labeledImage = process.connected_component();


        for (int i = 0; i < labeledImage.length; i++) {
            for (int j = 0; j < labeledImage[i].length; j++) {
                System.out.print(String.format("%d ", labeledImage[i][j]));
            }
            System.out.println();
        }

        double R1 = ObjectRecognition.computeRoundness(labeledImage, 1);
        double R2 = ObjectRecognition.computeRoundness(labeledImage, 2);

        System.out.println("Object 1 is " + ObjectRecognition.classifyObject(R1));
        System.out.println("Object 2 is " + ObjectRecognition.classifyObject(R2));
    }
}

```

RecogImage.java



```java
                int a = (p >> 24) & 0xff;
                int r = (p >> 16) & 0xff;
                int g = (p >> 8) & 0xff;
                int b = p & 0xff;
                // calculate average
                int avg = (r + g + b) / 3;
                arr[x][y] = avg;
                // replace RGB value with avg
                p = (a << 24) | (avg << 16) | (avg << 8) | avg;
                img.setRGB(x, y, p);
            }
        }
        // System.out.println("height" + height);
        // System.out.println("width" + width);

        // for (int i = 0; i < width; i++) {
        //     for (int j = 0; j < height; j++) {
        //         // System.out.print(String.format("%1$3s", arr[i][j]));
        //         System.out.print(String.format("%d ", arr[i][j]));
        //     }
        //     System.out.println();
        // }
        return arr;
    }
    Run | Debug
    public static void main(String args[]) throws IOException {

        // try {
        //     f = new File("./out3.bmp");
        //     ImageIO.write(img, "jpg", f);
        // } catch (IOException e) {
        //     System.out.println(e);
        // }
    }
}
```

ProcessImage.java

TestImage.java | ProcessImage.java | RecogImage.java | ConnectivityAnalysis.java

CS480_Java > pattern_recognition > step6_github_4 > ProcessImage.java > ProcessImage > displa

```java
package CS480_Java.pattern_recognition.step6_github_4;
class ProcessImage {
    private int[][] image;

    public ProcessImage(int[][] inputImage) {
        image = inputImage;
    }

    // step 1 gaussian filter
    public void gaussian_filter() {
        int rows = image.length;
        int cols = image[0].length;
        int[][] result = new int[rows][cols];

        double[][] kernel = {
            {1.0/16, 1.0/8, 1.0/16},
            {1.0/8, 1.0/4, 1.0/8},
            {1.0/16, 1.0/8, 1.0/16}
        };

        for (int i = 1; i < rows - 1; i++) {
            for (int j = 1; j < cols - 1; j++) {
                double sum = 0.0;
                for (int x = -1; x <= 1; x++) {
                    for (int y = -1; y <= 1; y++) {
                        sum += kernel[x + 1][y + 1] * image[i + x][j + y];
                    }
                }
                result[i][j] = (int) Math.round(sum);
            }
        }

        // Update the original image with the filtered values
        for (int i = 1; i < rows - 1; i++) {
            for (int j = 1; j < cols - 1; j++) {
                image[i][j] = result[i][j];
            }
```

TestImage.java | ProcessImage.java | RecogImage.java | ConnectivityAnalysis.java

CS480_Java > pattern_recognition > step6_github_4 > ProcessImage.java > ProcessImage > displayImage()

```
37            }
38        }
39    }
40
41
42    // step 2 histogram
43    public int[] histogram() {
44        int[] hist = new int[256];
45        for (int i = 0; i < image.length; i++) {
46            for (int j = 0; j < image[i].length; j++) {
47                hist[image[i][j]]++;
48            }
49        }
50        return hist;
51    }
52
53    // step 3 threshold
54    public int[][] threshold() {
55        int[][] threshold = OtsuThresholdFilter.filter(image);
56        return threshold;
57    }
58
59    // step 4 connected component
60    public int[][] connected_component() {
61        int[][] threshold = threshold();
62        ConnectivityAnalysis ca = new ConnectivityAnalysis(threshold);
63        int[][] labeledImage = ca.computeConnectivity(threshold);
64        return labeledImage;
65    }
66
67
68    public void displayImage() {
69        int rows = image.length;
70        int cols = image[0].length;
71
72        for (int i = 0; i < rows; i++) {
73            for (int j = 0; j < cols; j++) {
```

TestImage.java | ProcessImage.java | RecogImage.java | ConnectivityAnalysis.java

CS480_Java > pattern_recognition > step6_github_4 > ProcessImage.java > ProcessImage > displayImage()

```
73            for (int j = 0; j < cols; j++) {
74                System.out.print(image[i][j] + " ");
75            }
76            System.out.println();
77        }
78    }
79}
```

ObjectRecognition.java

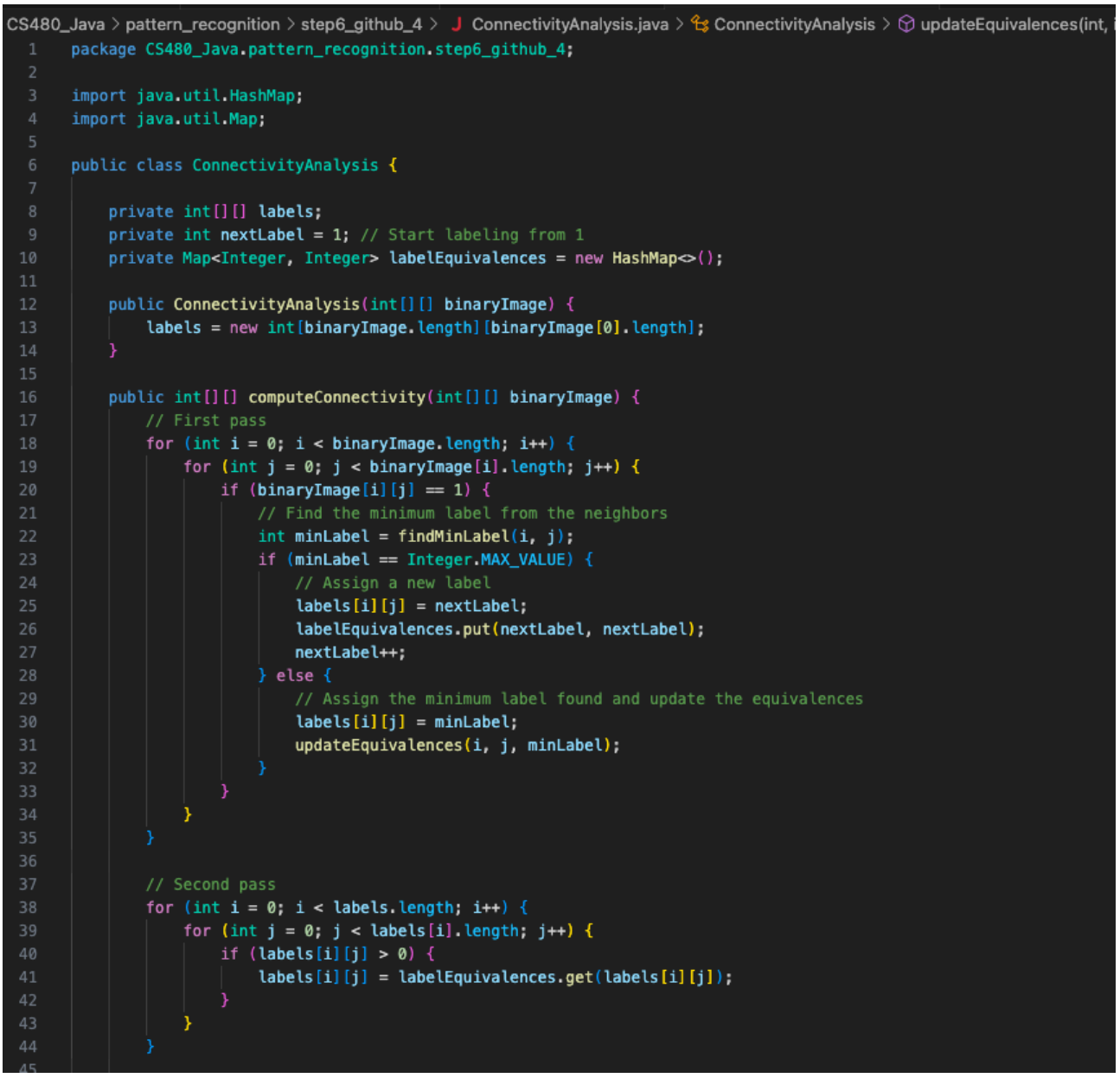
TestImage.java | ObjectRecognition.java | ProcessImage.java | RecogImage.java | ConnectivityA

CS480_Java > pattern_recognition > step6_github_4 > ObjectRecognition.java > ObjectRecognition > classifyObject(do

```java
package CS480_Java.pattern_recognition.step6_github_4;

public class ObjectRecognition {

    // Constants for the area and perimeter calculation
    private static final double PI = Math.PI;
    // private int[][] labeledImage;

    // public ObjectRecognition(int[][] labeledImage1) {
    //     labeledImage = labeledImage1;
    // }

    public static double computeRoundness(int[][] labeledImage, int label) {
        int area = 0;
        int perimeter = 0;

        for (int i = 0; i < labeledImage.length; i++) {
            for (int j = 0; j < labeledImage[0].length; j++) {
                if (labeledImage[i][j] == label) {
                    area++;
                    // Check if the pixel is on the border of the object for perimeter
                    if (isBorderPixel(labeledImage, i, j, label)) {
                        perimeter++;
                    }
                }
            }
        }

        // Compute roundness ratio R
        double R = (4 * PI * area) / (Math.pow(perimeter, 2));
        return R;
    }

    private static boolean isBorderPixel(int[][] labeledImage, int i, int j, int label) {
        for (int di = -1; di <= 1; di++) {
            for (int dj = -1; dj <= 1; dj++) {
                if (di == 0 && dj == 0) continue; // Skip the pixel itself
```

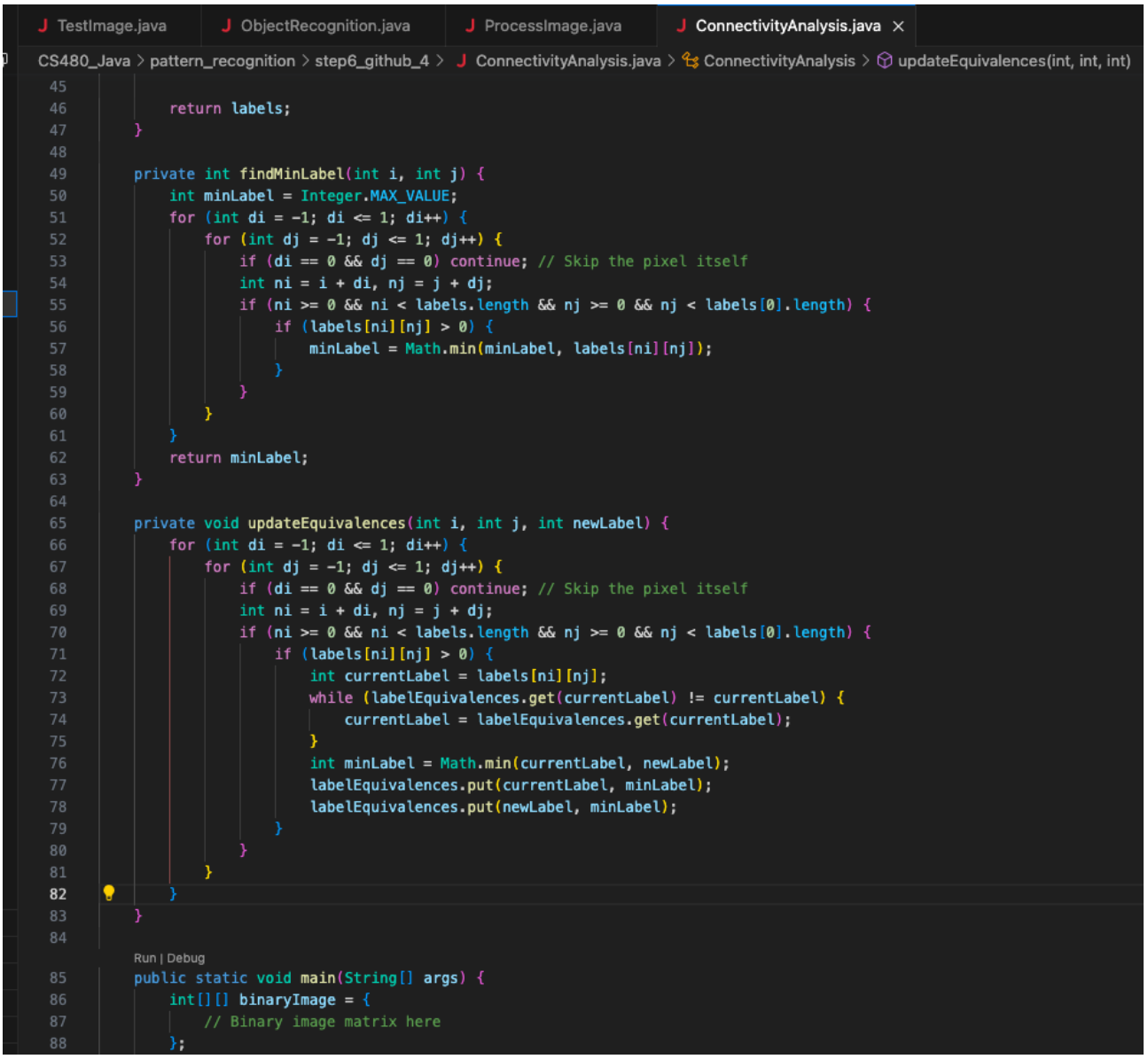
```java
                if (di == 0 && dj == 0) continue; // Skip the pixel itself
                int ni = i + di, nj = j + dj;
                if (ni >= 0 && ni < labeledImage.length && nj >= 0 && nj < labeledImage[0].length) {
                    if (labeledImage[ni][nj] != label) {
                        return true;
                    }
                } else {
                    return true;
                }
            }
        }
        return false;
    }

    public static String classifyObject(double R) {
        if (R >= 0.85) {
            return "circle";
        } else if (R >= 0.75) {
            return "square";
        } else {
            return "triangle";
        }
    }

    Run | Debug
    public static void main(String[] args) {
        int[][] labeledImage = {
            {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
            {0, 1, 1, 1, 0, 0, 0, 0, 0, 0},
            // ... the rest of the labeled image
        };

        // Assuming there are only two objects and labels used are 1 and 2
        double R1 = computeRoundness(labeledImage, 1);
        double R2 = computeRoundness(labeledImage, 2);

        System.out.println("Object 1 is " + classifyObject(R1));
        System.out.println("Object 2 is " + classifyObject(R2));
    }

}
```

ConnectivityAnalysis.java



CS480_Java > pattern_recognition > step6_github_4 > ConnectivityAnalysis.java > ConnectivityAnalysis > updateEquivalences(int,

```java
package CS480_Java.pattern_recognition.step6_github_4;

import java.util.HashMap;
import java.util.Map;

public class ConnectivityAnalysis {

    private int[][] labels;
    private int nextLabel = 1; // Start labeling from 1
    private Map<Integer, Integer> labelEquivalences = new HashMap<>();

    public ConnectivityAnalysis(int[][] binaryImage) {
        labels = new int[binaryImage.length][binaryImage[0].length];
    }

    public int[][] computeConnectivity(int[][] binaryImage) {
        // First pass
        for (int i = 0; i < binaryImage.length; i++) {
            for (int j = 0; j < binaryImage[i].length; j++) {
                if (binaryImage[i][j] == 1) {
                    // Find the minimum label from the neighbors
                    int minLabel = findMinLabel(i, j);
                    if (minLabel == Integer.MAX_VALUE) {
                        // Assign a new label
                        labels[i][j] = nextLabel;
                        labelEquivalences.put(nextLabel, nextLabel);
                        nextLabel++;
                    } else {
                        // Assign the minimum label found and update the equivalences
                        labels[i][j] = minLabel;
                        updateEquivalences(i, j, minLabel);
                    }
                }
            }
        }

        // Second pass
        for (int i = 0; i < labels.length; i++) {
            for (int j = 0; j < labels[i].length; j++) {
                if (labels[i][j] > 0) {
                    labels[i][j] = labelEquivalences.get(labels[i][j]);
                }
            }
        }
```


TestImage.java
ObjectRecognition.java
ProcessImage.java
ConnectivityAnalysis.java
CS480_Java > pattern_recognition > step6_github_4 > ConnectivityAnalysis.java > ConnectivityAnalysis > updateEquivalences(int, int, int)
45 return labels;
46 }
47
48 private int findMinLabel(int i, int j) {
49 int minLabel = Integer.MAX_VALUE;
50 for (int di = -1; di <= 1; di++) {
51 for (int dj = -1; dj <= 1; dj++) {
52 if (di == 0 && dj == 0) continue; // Skip the pixel itself
53 int ni = i + di, nj = j + dj;
54 if (ni >= 0 && ni < labels.length && nj >= 0 && nj < labels[0].length) {
55 if (labels[ni][nj] > 0) {
56 minLabel = Math.min(minLabel, labels[ni][nj]);
57 }
58 }
59 }
60 }
61 return minLabel;
62 }
63
64 private void updateEquivalences(int i, int j, int newLabel) {
65 for (int di = -1; di <= 1; di++) {
66 for (int dj = -1; dj <= 1; dj++) {
67 if (di == 0 && dj == 0) continue; // Skip the pixel itself
68 int ni = i + di, nj = j + dj;
69 if (ni >= 0 && ni < labels.length && nj >= 0 && nj < labels[0].length) {
70 if (labels[ni][nj] > 0) {
71 int currentLabel = labels[ni][nj];
72 while (labelEquivalences.get(currentLabel) != currentLabel) {
73 currentLabel = labelEquivalences.get(currentLabel);
74 }
75 int minLabel = Math.min(currentLabel, newLabel);
76 labelEquivalences.put(currentLabel, minLabel);
77 labelEquivalences.put(newLabel, minLabel);
78 }
79 }
80 }
81 }
82 }
83
84
Run | Debug
85 public static void main(String[] args) {
86 int[][] binaryImage = {
87 // Binary image matrix here
88 };

```
88          };
89
90          ConnectivityAnalysis ca = new ConnectivityAnalysis(binaryImage);
91          int[][] labeledImage = ca.computeConnectivity(binaryImage);
92
93          // Output the labeled image
94          for (int i = 0; i < labeledImage.length; i++) {
95              for (int j = 0; j < labeledImage[i].length; j++) {
96                  System.out.print(labeledImage[i][j] + " ");
97              }
98              System.out.println();
99          }
100     }
101 }
102
```

OtsuThresholdFilter.java

CS480_Java > pattern_recognition > step6_github_4 > OtsuThresholdFilter.java > OtsuThresholdFilter > varianceBetween(int, HashMap<Intege

```java
package CS480_Java.pattern_recognition.step6_github_4;
import java.util.HashMap;
import java.util.Map;

public class OtsuThresholdFilter {

    private static double varianceBetween(int threshold, HashMap<Integer, Integer> counts) {
        int countLower = 0;
        int countHigher = 0;
        int sumLower = 0;
        int sumHigher = 0;
        for (Map.Entry<Integer, Integer> e : counts.entrySet()) {
            if (e.getKey() < threshold) {
                countLower += e.getValue();
                sumLower += e.getKey() * e.getValue();
            } else {
                countHigher += e.getValue();
                sumHigher += e.getKey() * e.getValue();
            }
        }

        if (countHigher == 0 || countLower == 0) {
            return 0.0;
        }

        double uDiff = sumLower * 1.0 / countLower - sumHigher * 1.0 / countHigher;
        return (countLower * countHigher * 1.0 / ((countLower + countHigher)^2) * 1.0) * (uDiff * uDiff) ;
    }

    public static int[][] filter(int[][] matrix) {
        int[][] result = matrix.clone();

        HashMap<Integer, Integer> counts = new HashMap<>();

        for (int i = 0; i < matrix.length; i++) {
            for (int j = 0; j < matrix[0].length; j++) {
                counts.put(matrix[i][j], counts.getOrDefault(matrix[i][j], 0) + 1);
            }
        }

        int threshold = -1;
        double max = -1.0;
        for (Map.Entry<Integer, Integer> e : counts.entrySet()) {
            double vb = varianceBetween(e.getKey(), counts);
            // System.out.println("Threshold: " + e.getKey());
```

```
45              // System.out.println("Threshold: " + e.getKey());
46              // System.out.println("VB: " + vb);
47              if (vb > max || max == -1.0) {
48                  max = vb;
49                  threshold = e.getKey();
50              }
51          }
52          // System.out.println("Target Threshold: " + threshold);
53          // System.out.println("Max vb: " + max);
54
55          for (int i = 0; i < matrix.length; i++) {
56              for (int j = 0; j < matrix[0].length; j++) {
57                  if (matrix[i][j] < threshold) {
58                      result[i][j] = 0;
59                  } else {
60                      result[i][j] = 1;
61                  }
62              }
63          }
64
65          return result;
66      }
67
68  }
```

2. Execution Outcome:

input picture: a2.bmp:

Recognition Resulte:

Object 1 is circle
Object 2 is triangle

Execution Screenshot:

```
9794831/bin CS480_Java.pattern_recognition.step6_github_4.TestImage
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 2 2 2 2 2 0 0 0 0 0 0 0 0 0 0 0
0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 2 2 2 2 2 2 2 2 2 2 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 2 2 2 2 2 2 2 2 2 2 2 2 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 2 2 2 2 2 2 2 2 2 2 2 2 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 2 2 2 2 2 2 2 2 2 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 2 2 2 2 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Object 1 is circle
Object 2 is triangle
apple@appledeAir courses %
```