**Class Breaker**

***Features:***

***Members:***

***Functions:***

GetImages();  
WriteToFile();

**Class Noter**

***Features:***

***Members:***

***Functions:***

**Class Clusterer**

***Features:***

***Members:***

String code\_condition;

***Functions:***

+ void setsource();  
+ void setClusters();

+ void run();

void assign(String s1, String s2, String s3);

s1 : text file with the pixels to be assigned to clusters.   
s2 : text file with the cluster centers from the training.   
s3 : text file to write the assignments of pixels in file s1.

- int h\_close();  
- int h\_distance();  
- int h\_qualify();

It is important to have the results of Breaker intact. This file will be compared to the file generated finally to perform final statistical and visual analysis.

**Class Tabler**

***Features:***

***Members:***

***Functions:***

**Class Cuber**

***Features:***

***Members:***

***Functions:***

**Class Cohener**

***Features:***

***Members:***

***Functions:***

***How to Use:***

1. Name the source images, and labels as outlined in the ‘Naming Rules’ section of this document.
2. Place all labels from one rater inside a folder with the rater’s name. So, we now have as many folders as raters. Note, that this code runs only for two raters.
3. This root folder containing subfolders for each rater must be named ‘\_annotation’ and must reside one level above Cohener.java class.   
   For example: ‘\_annotation’ and ‘code’ folders on one level, and Cohener.java inside the folder ‘code’.
4. Make an instance of the Cohener class.
5. Use the setFolder() function to pass in the name of the root folder, ‘\_annotation’ in this case.
6. Use the setLabels()
7. Use the run() function to calculate the kappa agreement, and return kappa value.
8. Use the printLogs()function to display detailed information that displays class-wise agreement.

**Class Cleaner**

***Features:***

* Delete all files from a folder: Used to clear log files, predictions and class files at the beginning of a run.

***Members:***

***Functions:***

+ void deleteFilesFromFolder(String nameFolder);

nameFolder : name of the folder from which all files will be deleted. Also note, this will leave recursive folder structure untouched while deleting all files.

- ArrayList<String> h\_getFiles(final File folder);

**Class Analyzer**

***Features:***

***Members:***

***Functions:***

**General Rules**

1. ‘+’ infront of the function indicates public scope, ‘-‘ infront of the function indicates private scope.
2. It is important to have the results of Breaker intact. This file will be compared to the file generated finally to perform final statistical and visual analysis.
3. All folder parameters work at one level above the source file. For example, assume the code sits in a folder ‘source’. All folders in the same folder as ‘source’ (siblings of source) are accessible by default. Otherwise specific path variables have to be added. If a folder inside source has to be passed: source/folder.