| **Group** | **Module** | **Description** |
| --- | --- | --- |
| 5 | WordListParser | Has a method that reads the input file (an ascii file), parses it into words, and strips out punctuation. The method returns a WordList object. |
| 4 | StemMaker | Holds a stopList (a WordList object). Has a method that initializes this stopList by reading the stop list file. Has another method that converts the list of all words (another WordList object) to a StemmedList object. It does this by eliminating any words from the stopList and stemming each word. |
| 1 | FreqMaker | Has a method that converts a StemmedList object to a DocList object. The first word following the stem is the one that should be chosen for the DocList entry, along with the total number of occurrences for that stem. |
| 2 | WordMapMaker | Has a method that converts a frequency list (a DocList object) to a WordMap object. The range of frequencies should be mapped linearly to the ten available font sizes. |
| 5 | ICloudLayout | Specifies a method that will be implemented by the various layout classes. This method should update a WordMap by setting the locations (Point) for each word in the map. |
| 4 | RandomLayout | Implements the method in ICloudLayout by setting with locations for each word. Locations are determined randomly. Overlapping words are ok. |
| 5 | StackedLayout | Implements the method in ICloudLayout by setting with locations for each word. Words are stacked from the bottom of the page upwards, with font size going from biggest (at the bottom) to smallest (at the top). |
| 3 | LinesLayout | Implements the method in ICloudLayout by setting with locations for each word. Words are chosen randomly, then laid out left to right, line by line. Make sure words don’t overlap. |
| 1 | ColumnLayout | Implements the method in ICloudLayout by setting with locations for each word. Words are chosen randomly, then laid out in columns. Make sure words don’t overlap. |
| 2 | SquaresLayout | Implements the method in ICloudLayout by setting with locations for each word. Words are chosen randomly, then laid out first along the edge of the page (a row at the top, a column along the right edge, a row at the bottom, and a column along the left edge). The next set of words is laid out also in a square, but nested inside the first square. Make sure words don’t overlap. |
| 6 | FishEyeLayout | Implements the method in ICloudLayout by setting with locations for each word. The words in the biggest font are placed in one or more rows at the middle of the page. The next-largest font words are placed in rows above and below the first set, and so forth. Make sure words don’t overlap. |
| 3 | LatexMaker | Has a method that, given a WordMap, produces a latex document (with a .tex file extension). For the format of the latex document, see the sample document Comm\_Princ.tex. |
| 6 | WordList | Holds a list of strings (words). Has read() and write() methods for reading the words from a file and vice versa. |
| 1 | StemmedList | Holds a list of FreqList objects. Has read() and write() methods for reading the list of lists from a file and vice versa. (These methods defer most of that work to FreqList.) |
| 2 | DocList | Holds a FreqList and its associated file name. Has read() and write() methods for reading the FreqList from a file and vice versa. (These methods of course defer the work to the FreqList.) |
| 6 | FreqList | Holds a list of WordCount objects. Has read() and write() methods for reading the map from a file and vice versa. Provides a method that sorts the list by count, and a method that reduces the list by keeping only the most frequently occurring words. |
| 6 | WordCount | Holds a word and its associated count (which for our purposes is the number of times the word appears in the document). In addition to appropriate getters and setters, it must provide the asString() method that puts the object’s state (attribute values) into a string that it returns. It must also provide fromString() (the reverse of asString()). |
| 3 | WordMap | Holds a list of MapItem objects. Has read() and write() methods for reading the map from a file and vice versa. Provides a method that sorts the map by fontsize, and another method that randomizes the order of elements in the map. |
| 3 | MapItem | Holds a word and its associated fontsize, location (Point), and size (Size). In addition to appropriate getters and setters, it must provide the asString() method that puts the object’s state (attribute values) into a string that it returns. It must also provide fromString() (the reverse of asString()). |
| 3 | Point | Holds x and y coordinates. Needs appropriate getters and setters. |
| 3 | Size | Holds a height and width. Needs appropriate getters and setters. |
| 6 | LatexCloudMaker | End-to-end executable: reads an ascii input file and produces a .tex file that will be processed by latex. |
| 1 | DocToStems | Executable that reads an ascii input file and produces a file containing the stemmed list. |
| 1 | StemsToLatexCloud | Executable that reads the stemmed list from a file and produces a .tex file. |
| 2 | FreqListToLatexCloud | Executable that reads a frequency list from a file and produces a .tex file. |

THIS IS PSEUDO-CODE!!!

LatexCloudMaker.cpp (command line params: inputFileName, stopFileName, tracing flag)

WordListParser parser;

WordList\* allWords = parser(inputFileName);

StemMaker stemMaker;

stemMaker.setStopList(stopFileName);

StemmedList\* stems = StemMaker.getStems(allWords);

if (tracing is on)

stems->writeToFile(inputFileName where “.txt” is replaced with “\_stems.txt”);

FreqMaker freqMaker;

DocList\* docList = freqMaker.getDocList(stems);

If (tracing is on)

docList->writeToFile(inputFileName where “.txt” is replaced with “\_freq.txt”);

docList->sortByFreq();

docList->keepTopN(n); // n can be controlled by a compiler directive

WordMapMaker wordMapMaker;

WordMap\* wordMap = wordMapMaker.getWordMap(docList);

If (tracing is on)

wordMap->writeToFile(inputFileName where “.txt” is replaced with “\_map.txt”);

ICloudLayout\* cloud; // replace with a call to the factory method

Cloud->setLayout(wordMap);

LatexMaker latex;

latex.generateDoc(wordMap, inputFileName where “.txt” is replaced with “.tex”);

DocToStems.cpp (command line params: inputFileName, stopFileName)

StemsToLatexCloud.cpp (command line param: freqFileName, tracing flag)

FreqListToLatexCloud.cpp (command line params: freqListFileName, latexOutputFileName)