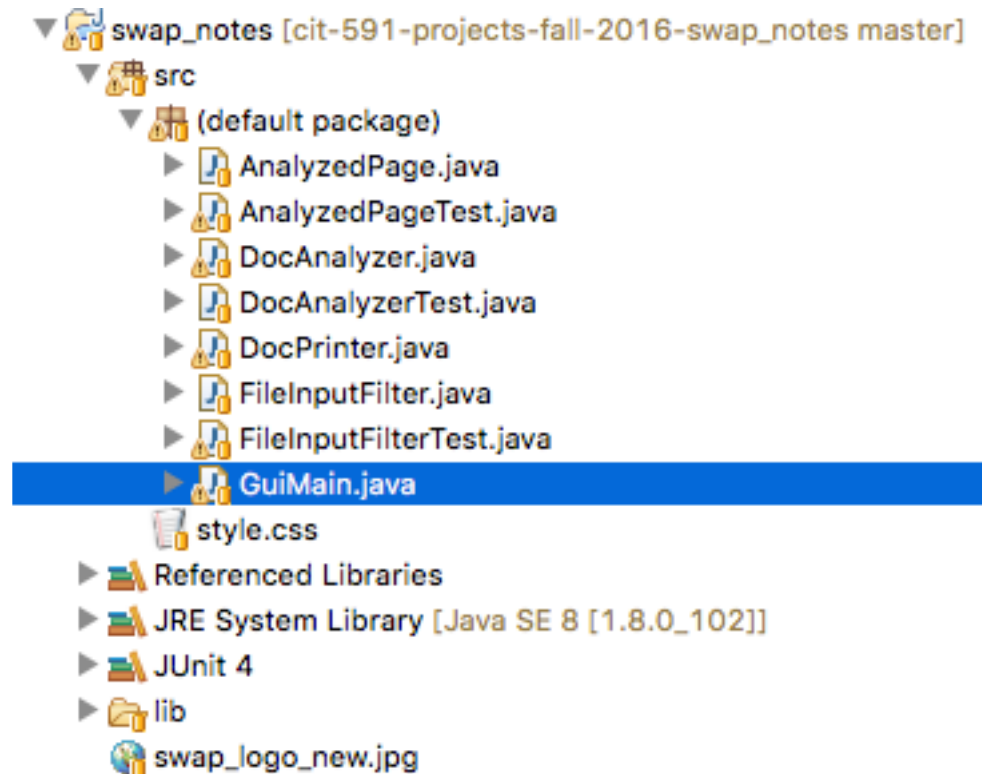
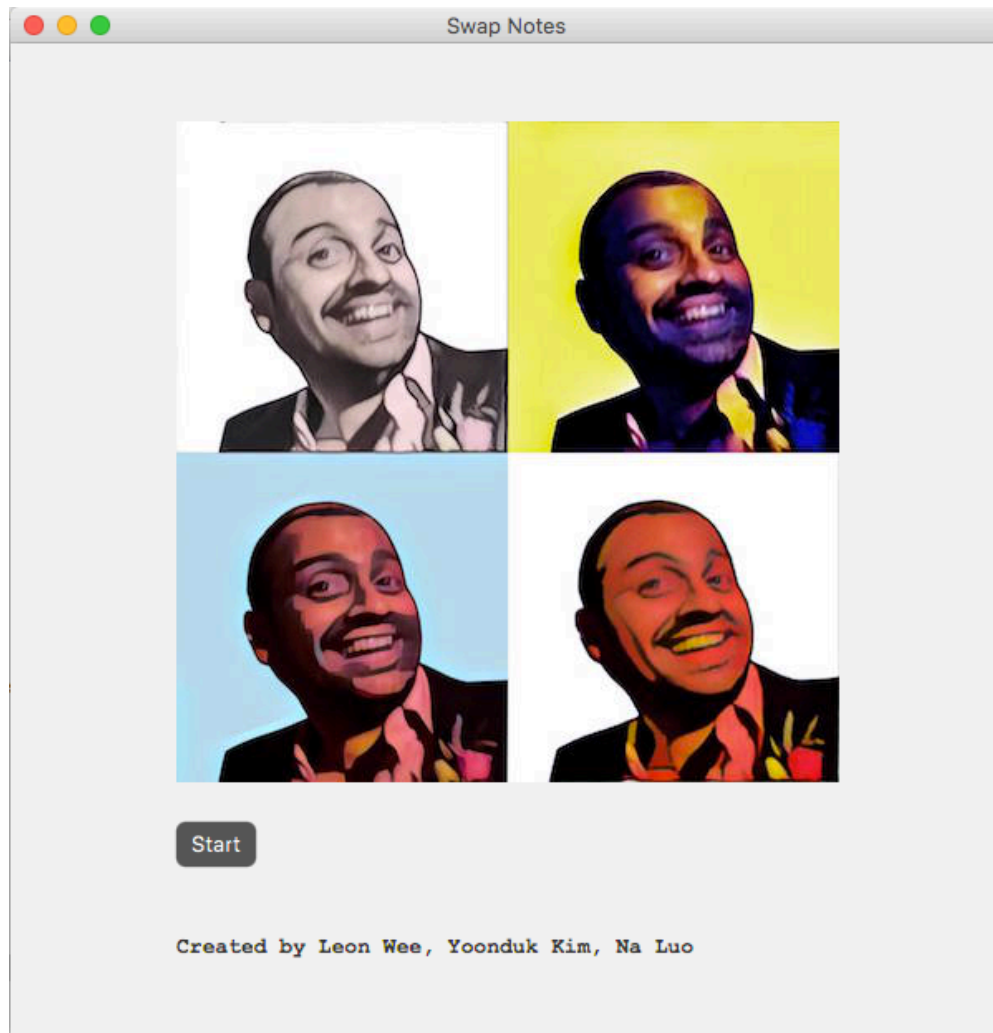


# Instruction Manual for Swap Notes



## Step 1: Run GuiMain.java as a Java Application

This class contains the main method and the GUI.



**Step 2: Press the Start button on the beautiful landing page**

So beautiful.

The screenshot shows a Java application window with the following elements highlighted by numbered red circles:

- 1**: Open File button
- 2**: Keywords input field
- 3**: Number of Pages in Output input field
- 4**: Page filtering condition dropdown menu (currently set to 'And')
- 5**: Sort By dropdown menu (currently set to 'Page Order')
- 6**: Save file location button and the output filename input field
- 7**: Summarize button

### Step 3: Enter inputs and begin summarization process

1. Open File: Opens a new File Selector window to select the pdf file.  
\*PDFBox has difficulty parsing PDF documents created through OCR. If possible, use a PDF document that are well-formatted.
2. Keywords: Space separated list of keywords (Will implement comma separation for increased diversity of inputs in the future). Case insensitive.
3. Number of Pages in Output: Limits the number of pages the user wants in output file.
4. Page Filtering Condition: AND or OR option available. Determines whether the program searches pages containing all the keywords or one of the keywords.
5. Sort By: Page Order outputs documents in their original page order. Relevance sorts the results as determined by their relevance scores from the vector space algorithm.
6. Save File: Automatically appends “.pdf” at the end of the name field. Save file location button opens a Folder Selector window.
7. Summarize: Runs application.



```
GuiMain [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_102.jdk/Contents/Home/bin/java (Dec
Page Number Limit: 30
Dec 18, 2016 10:56:55 PM org.apache.pdfbox.pdmodel.font.PDSimpleFont toUnicode
WARNING: No Unicode mapping for 49 (39) in font WEWCCR+Calibri
```

During the parsing process, this warning may pop up in the console. This just means that some of the fonts used in the PDF file are not contained in the API. As we are not using any functionality related to fonts, this message does not affect the accuracy of the application.

## Result check

 <b>Page 1</b> 61 matches <b>java.util....LinkedList&lt;E&gt;</b> implements <b>List&lt;E&gt;</b> , <b>Queue&lt;E&gt;</b> , <b>Serializable</b> <b>java.util....java.util....java.util....Priority...</b>	 <b>Page 1</b> 10 matches <b>LinkedList&lt;E&gt;</b> implements <b>List&lt;E&gt;</b> , <b>Queue&lt;E&gt;</b> , <b>Serializable</b> <b>java.util....</b> <b>ArrayList&lt;E&gt;</b> implements <b>List&lt;E&gt;</b> , <b>Seri...</b>
 <b>Page 2</b> 65 matches <b>java.net</b> package, A-30–31 <b>java.net....</b> <b>URLConnection</b> class, A-31 <b>java.sql</b> package, A-31–34 <b>java.sql....Statement...</b>	 <b>Page 2</b> 4 matches <b>ArrayList&lt;E&gt;</b> class add method, 348... <b>LinkedList&lt;E&gt;</b> class, A-38–39... <b>List&lt;E&gt;</b> interface, 672, A-39... <b>ListIterator&lt;E&gt;</b> in...
 <b>Page 3</b> 58 matches <b>java.util....java.util....Calendar</b> class, A-36 <b>java.util....Element</b> interface, A-51 getAvailable method, <b>java.util....TimeZo...</b>	 <b>Page 3</b> 3 matches <b>ArrayList&lt;E&gt;</b> class, 348, A-35... <b>LinkedList&lt;E&gt;</b> class, 676, A-39...Linke...
 <b>Page 4</b> 18 matches <b>LinkedList.java</b> class, 726–729 <b>LinkedListStack.java</b> class, 736–737 <b>ListDemo.java</b> class, 679...Linear searc...	 <b>Page 4</b> 20 matches <b>LinkedList.java</b> class, 726–729 <b>LinkedListStack.java</b> class, 736–737 <b>ListDemo.java</b> class, 679... <b>List&lt;E&gt;</b> inte...
 <b>Page 5</b> 25 matches <b>SelectionSortDemo.java</b> class, 632 <b>SelectionSorter.java</b> class, 631–632 <b>SelectionSortTimer.java</b> class, 635 Se...	 <b>Page 5</b> 4 matches <b>ArrayList&lt;E&gt;</b> class, 349, A-35... <b>LinkedList&lt;E&gt;</b> class, 676, 720–721, A-39... <b>LinkedList&lt;E&gt;</b> class, A-39...Arra...
 <b>Page 6</b> 22 matches <b>Queen.java</b> class, 618–619 <b>QuestionDemo1.java</b> class, 425 <b>QuestionDemo2.java</b> class, 432–433 Q...	 <b>Page 6</b> 1 match as linked <b>lists</b> , 737–738 sample code, 739–740
 <b>Page 7</b> 9 matches The <b>Java</b> Library...Class <b>java.util....Class</b> <b>java.util....Class</b> <b>java.util....Class</b> <b>java.util....</b> ...Interface <b>java.util....Class</b> <b>java.util....C...</b>	 <b>Page 7</b> 6 matches This method checks whether the iterator is past the end of the <b>list</b> ...Returns: true if the iterator is not yet past the end of...
 <b>Page 8</b> 21 matches Font constructor, A-14 <b>FontFrame2.java</b> class, 865–868 <b>FontFrame.java</b> class, 855–858 <b>FontViewer2.java</b> class, 865...	 <b>Page 8</b> 2 matches array <b>lists</b> , 348...linked <b>lists</b> , 676

Above is an example of a resulting document created by running the following input.

Keywords:	<input type="text" value="java list"/>
Number of Pages in Output:	<input type="text" value="30"/>
Page filtering condition...	<input type="button" value="And"/> ▼
Sort By...	<input type="button" value="Relevance"/> ▼

We can see that the document correctly sorts the pages in the order of their relevance by the frequency of the two keywords “java” and “list”.

<b>Page 1</b>	1 match
Parsing <i>html</i> files and Informa'on Extrac'on	
<b>Page 2</b>	2 matches
Finding the Info You Need in <i>HTML</i> ...• Manually look through the <i>html</i> to find recurring keywords that occur around t...	
<b>Page 3</b>	1 match
GeVng <i>HTML</i> from a URL	
<b>Page 4</b>	2 matches
GeVng <i>HTML</i> from a URL...public void get <i>HTML</i> (String urlName) {	

However, in some cases we see that pages are not sorted by the absolute frequency of keyword matches. This is because our algorithm ranks the pages based on the keyword's importance relative to the length of the page. For the example above, we can see that in page1, "html" has a greater weight than page2, although the latter has more matches.

## Recitation 4

Parsing `html` files and Information  
Extraction

### Finding the Info You Need in `HTML`

- First we want to locate all the courses
- Manually look through the `html` to find recurring keywords that occur around the courses
- Find the lines that contain those keywords
- On the CS website all the classes have a "target" attribute
  - `<a target="_blank"`