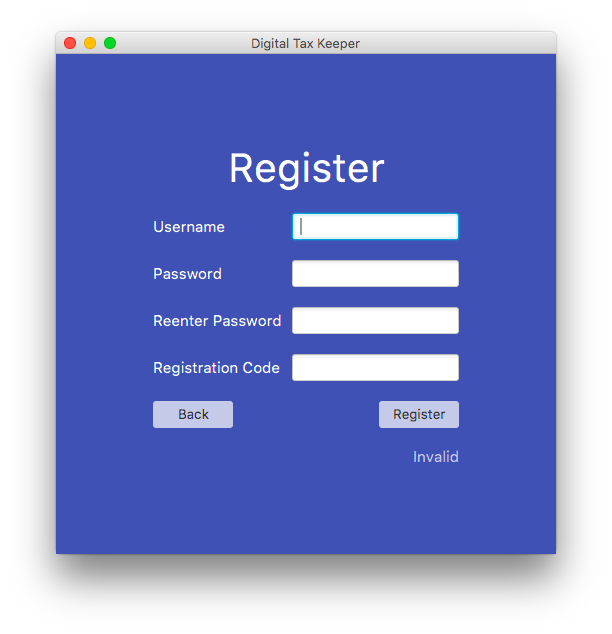
## **Criterion C: Development**

**Technique #1: Login and Register**

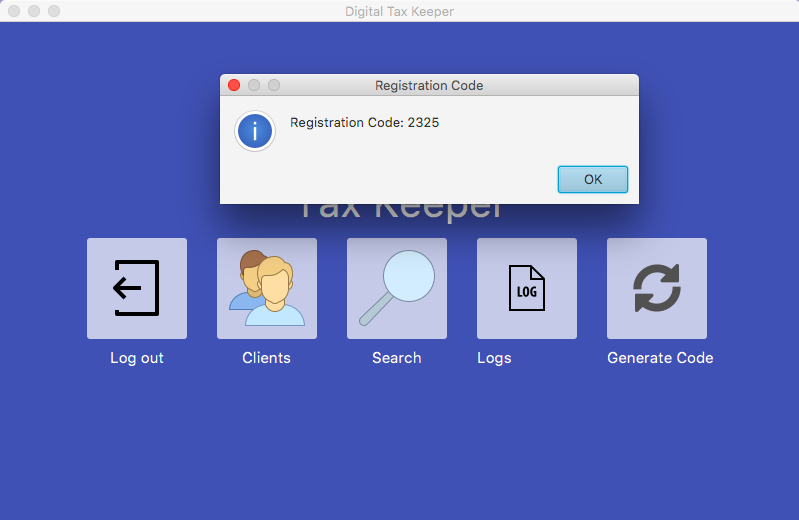
*Techniques:* Storage Mechanism, Thinking Ahead

*Evidence:* Use of Database, Randomly Generated Registration Code, Re Enter Password, Log

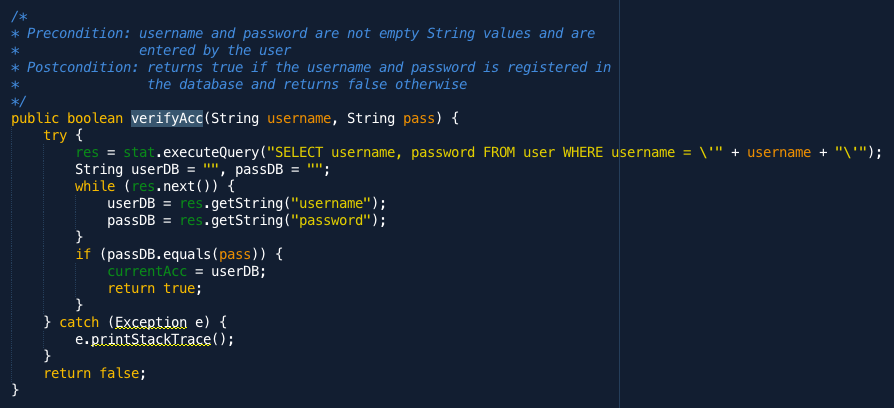
**GUI Snippet #1 (Login) and #2 (Register)**



**GUI Snippet #3 (Registration Code)**

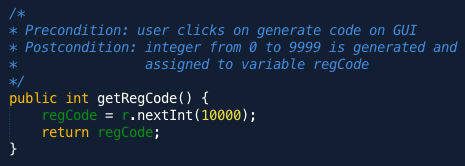


**Code Snippet #1** *(Refer to the Appendix F for full details of code)*



The second success criteria states that users can only login if they enter an existing username or password in the database *(Refer to Criterion A)*. This verifyAcc method satisfies this success criteria as it checks if the account exists in the “user” table. This feature is important as Mrs. Erlina deals with sensitive documents and restricting access by making sure that only authorized people can login provides a layer of security needed by the client.

**Code Snippet #2** *(Refer to the Appendix F for full details of code)*

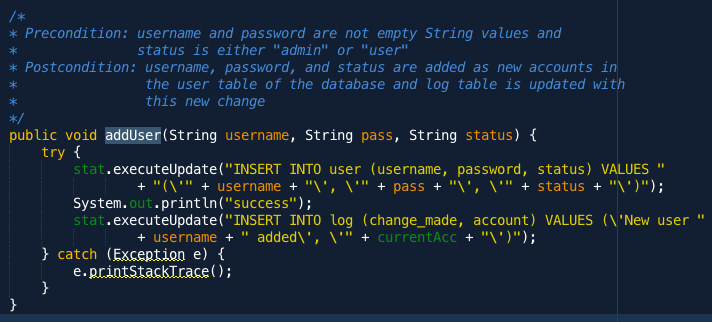
**

**Code Snippet #3** *(Refer to the Appendix F for full details of code)*

**

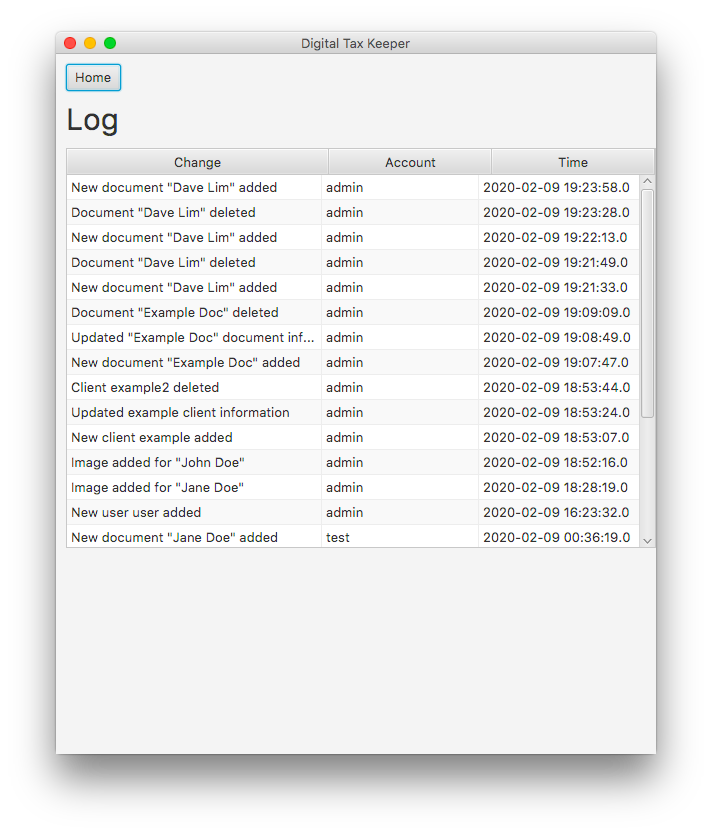
The first criteria states that new accounts can only be made using a valid registration code *(Refer to Criterion A)*. The getRegCode method satisfies this criteria as it allows an admin to randomly generate a new registration code and the checkRegCode method checks if new accounts are registered with the correct registration code. Security is a primary concern for Mrs. Erlina and ensuring that new accounts will need to be approved by an admin who is the only one who can generate a registration code makes access to the app secure.

**Code Snippet #4** *(Refer to the Appendix F for full details of code)*

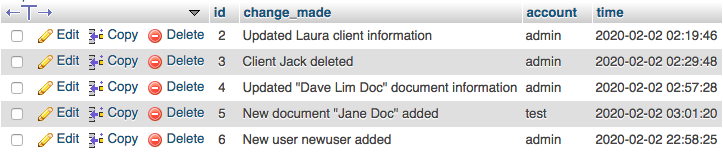


This addUser method also satisfies the first success criteria as after it checks that the registration code is valid, this method will be called to add the new account into the “user” table so that Mrs. Erlina’s new employees can use their new account to login to the app. This method also updates the “log” table to show Mrs. Erlina that a new user has been added to the database. This satisfies the eleventh success criteria that states that there should be a “log” page in the app that is constantly updated and later displayed in a table as shown in the GUI below.

**GUI Snippet #4 (Log Page)**

****

**Database Screen #1**

****

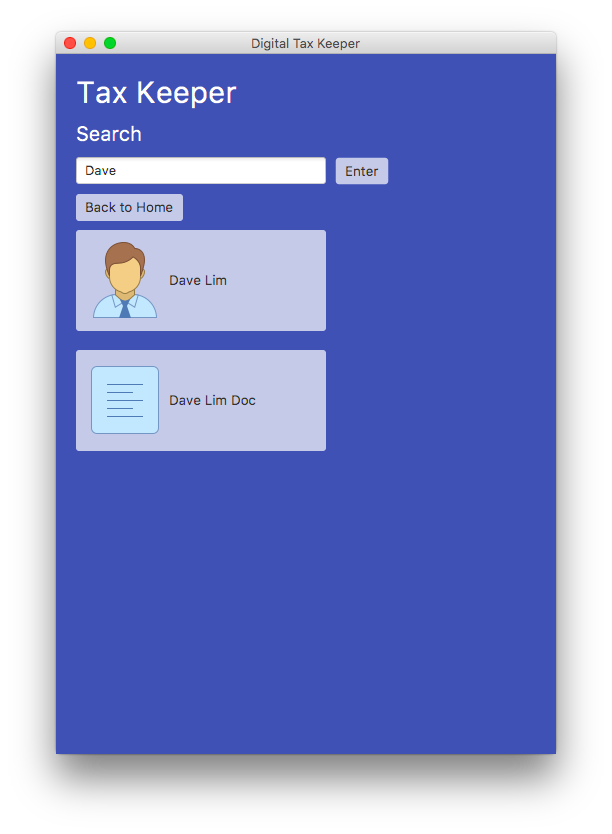
Above is taken from the ‘log’ table of the database that contains 4 columns. Each row represents each change made on the app. This satisfies the eleventh success criteria this table feeds the app with the information needed to be displayed in the table of the ‘log’ page.

**Technique #2: Getting Search Results**

*Techniques:* Storage Mechanism, Thinking Abstractly, Thinking Procedurally

*Evidence:* Use of Database, ArrayList, Displaying the Search Results, Client Objects

**GUI Snippet #5 (Search Page)**

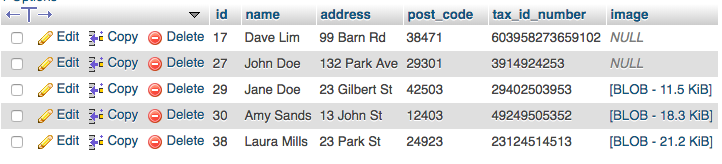


**Code Snippet #5** *(Refer to the Appendix F for full details of code)*



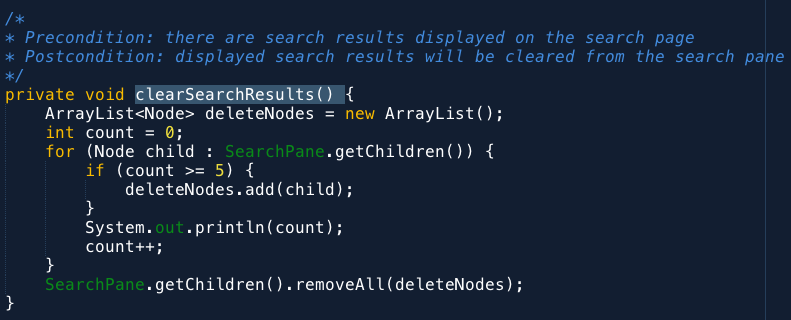
This getSearchResultsClients method returns an ArrayList of Client objects from the database whose name matches the searched keyword. A loop structure will continuously fetch data that matches the conditions laid out in the SQL query. This feature satisfies the third criteria that states that users should be able to search clients in the search page and this method satisfies this by providing the list of clients to be displayed. This feature helps the client quickly navigate her archive and access the specific information/data attached to the client she was specifically looking for.

**Database Screen #2**



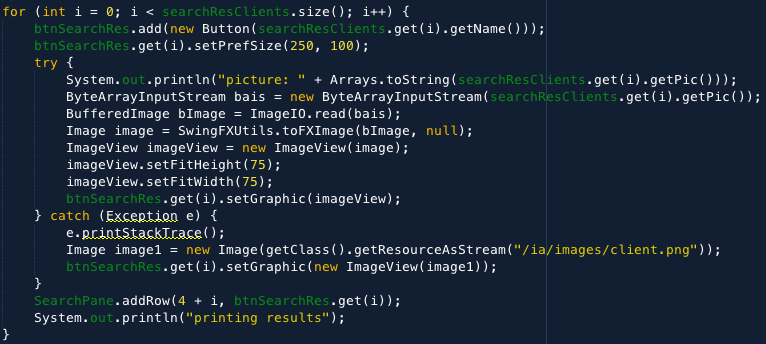
Above is a screenshot of the “clients” table in the database that has 6 columns and each row represents each client’s information. This is the table where **Code Snippet #5** retrieves its data from.

**Code Snippet #6** *(Refer to the Appendix F for full details of code)*



This clearSearchResults method clears the previous search results from the GridPane so that it can be replaced with the new search results. This code snippet demonstrates my ability to Think Procedurally as I realize that before being able to display the new search results, I will need to clear the GridPane first so that only relevant search results will be displayed to the client.

**Code Snippet #7** *(Refer to the Appendix F for full details of code)*



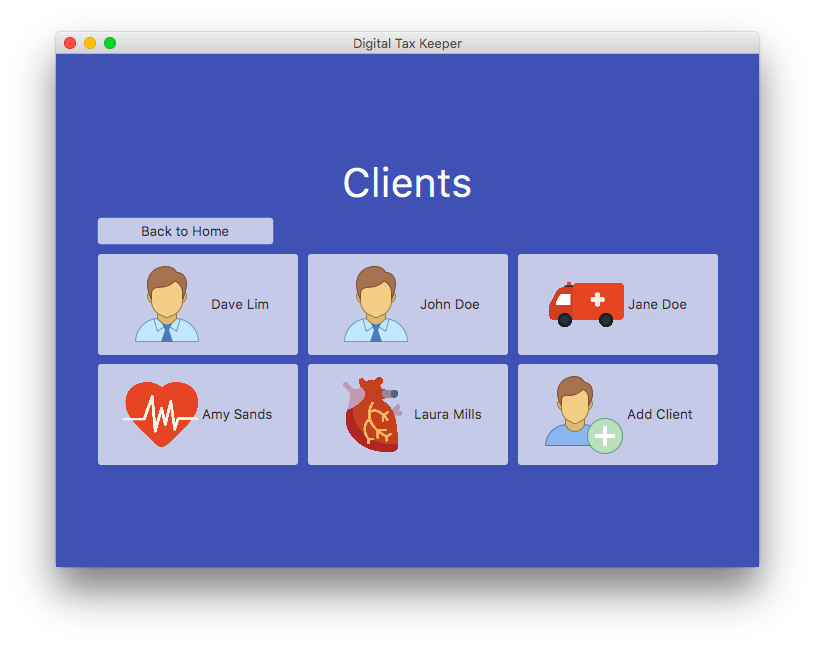
This is from the getSearchResults method where the ArrayList of Clients retrieved from **Code Snippet #4** is displayed. The ArrayList of searched Client results is traversed using a loop structure and added to an ArrayList of Buttons. The client's image is also displayed on the button which demonstrates complexity as the byte array is converted to ImageView. If clients don’t have an image, a default “client.png” picture is used as a placeholder, evidence of Thinking Ahead. This satisfies the third success criteria as it displays the user’s search results on the search page and also the tenth success that states that each client can have a picture identifying them which is achieved with how their images are displayed on the buttons.

**Technique #3:** **Showing Clients on Clients Page**

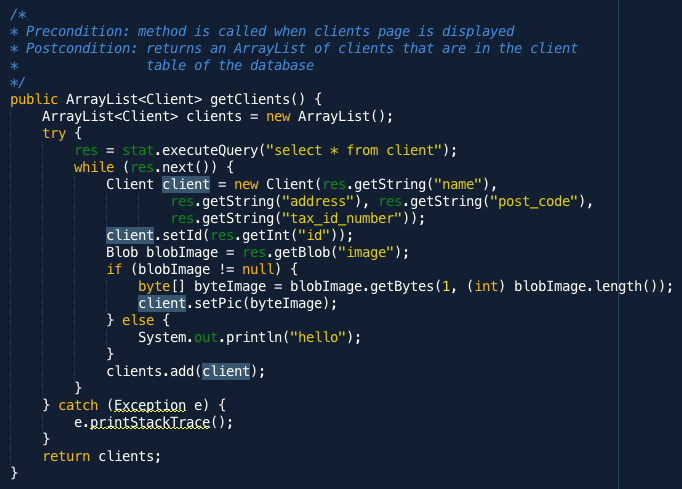
*Techniques:* Storage Mechanism, Thinking Logically

*Evidence:* Use of Database, Displaying clients in rows of three

**GUI Snippet #6 (Clients Page)**



**Code Snippet #8** *(Refer to the Appendix F for full details of code)*



This getClients method returns an ArrayList of clients in the database. A Client Object is used to represent each client’s information is saved in an ArrayList because of its dynamic size, indicating my ability of Thinking Abstractly. This code represents complexity as a BLOB is converted to byte array so it is easier to manipulate, satisfying the seventh success criteria that states that documents should be saved in the database and presented in a PDF format, at which was made possible through the mentioned processes. Having the documents in PDF format is convenient for Mrs. Erlina as she uses this document format for all her legal matters.

**Code Snippet #9** *(Refer to the Appendix F for full details of code)*



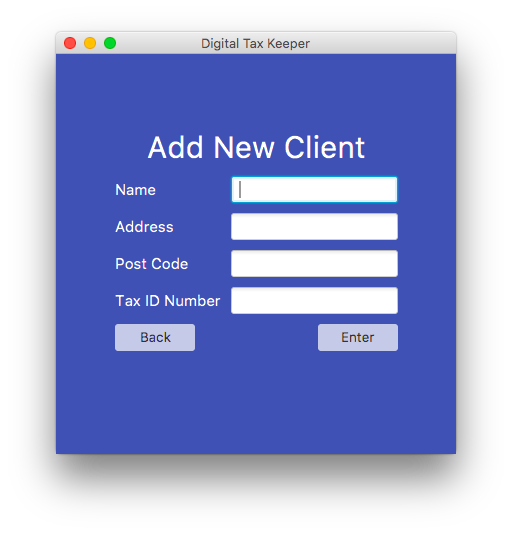
This snippet is taken from the showClients method that displays the ArrayList of Clients retrieved from the database shown in **Code Snippet #8** onto the Clients page. This code displays the clients into rows of three to have a concise list displayed which allows the client to conveniently go through the list of clients as it is not structured as a single long list. This demonstrates my ability to Think Logically to determine how to organize the clients and also complexity for converting a byte array to an ImageView.

**Technique #4: Adding, Updating, Deleting New Clients**

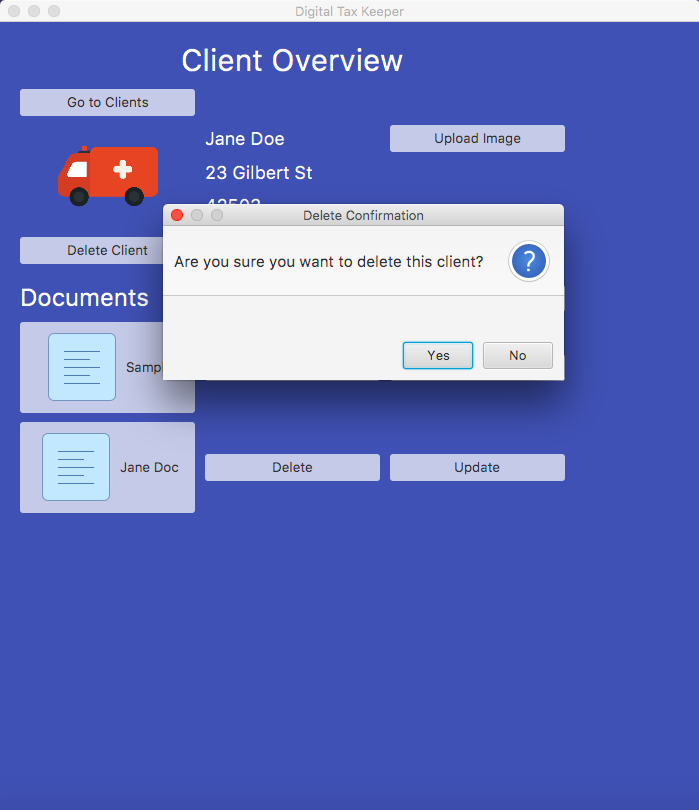
*Techniques:* Storage Mechanism, Thinking Abstractly, Thinking Ahead

*Evidence:* Use of Database, Client Objects, Confirmation Dialogs

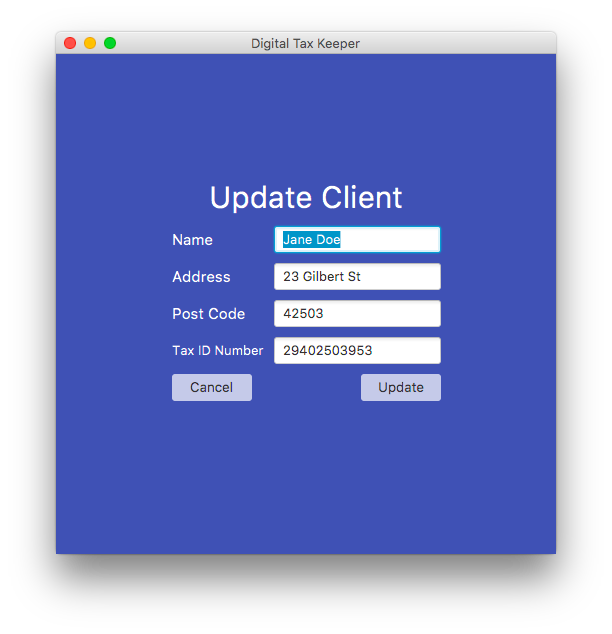
**GUI Snippet #7 (Add Client Page)**

****

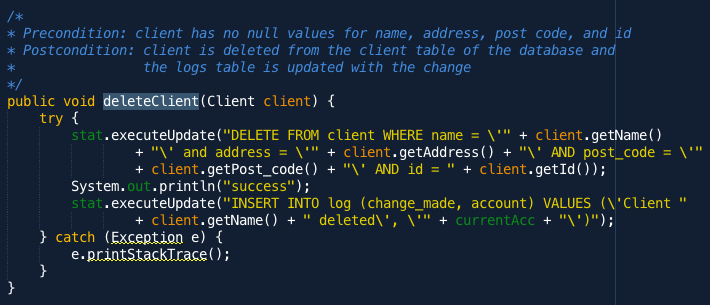
**GUI Snippet #8 (Delete Client Dialog)**

****

**GUI Snippet #9 (Update Client Page)**



**Code Snippet #10** *(Refer to the Appendix F for full details of code)*



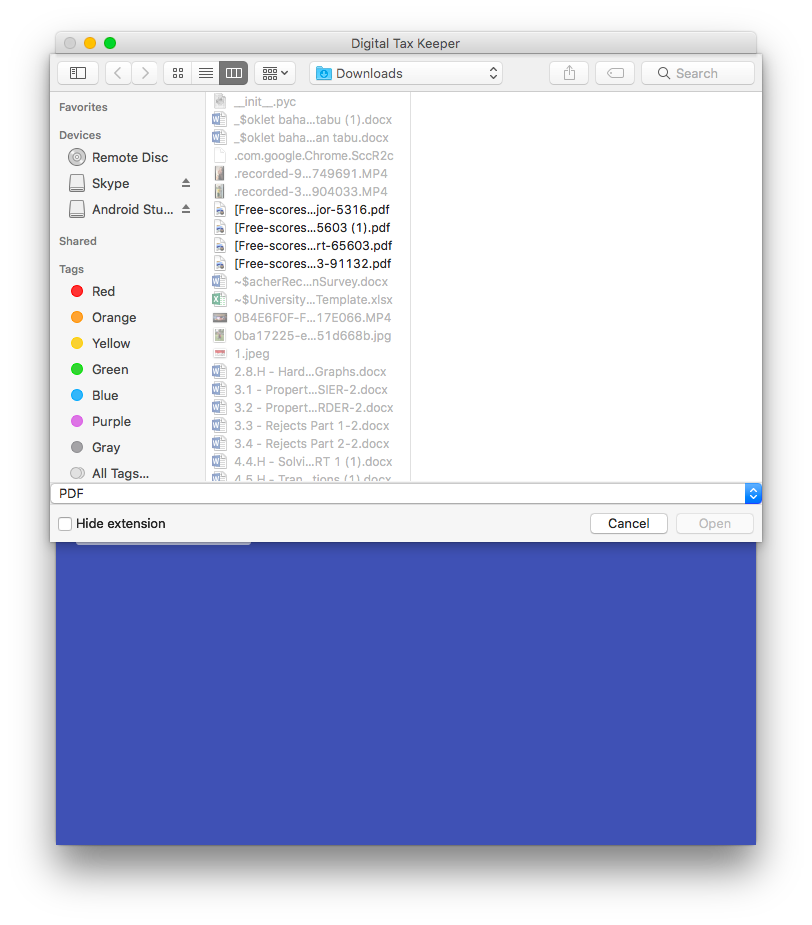
This deleteClient method will delete a specific client but only called when the user confirms as seen in **GUI Snippet #8**. This method will receive a Client object, an example of Thinking Abstractly, and deletes the client from the database and the update/add client features work similarly. This satisfies the fifth success criteria users will be able to delete clients when Mrs. Erlina wishes to delete all of the documents and information associated with a client she no longer works with from her archives. This also satisfies eleventh success criteria as the after the change is made, this method will update the “log” table in the database with the newest change so Mrs. Erlina can keep track of which clients have been deleted in the future.

**Technique #5: Upload Document**

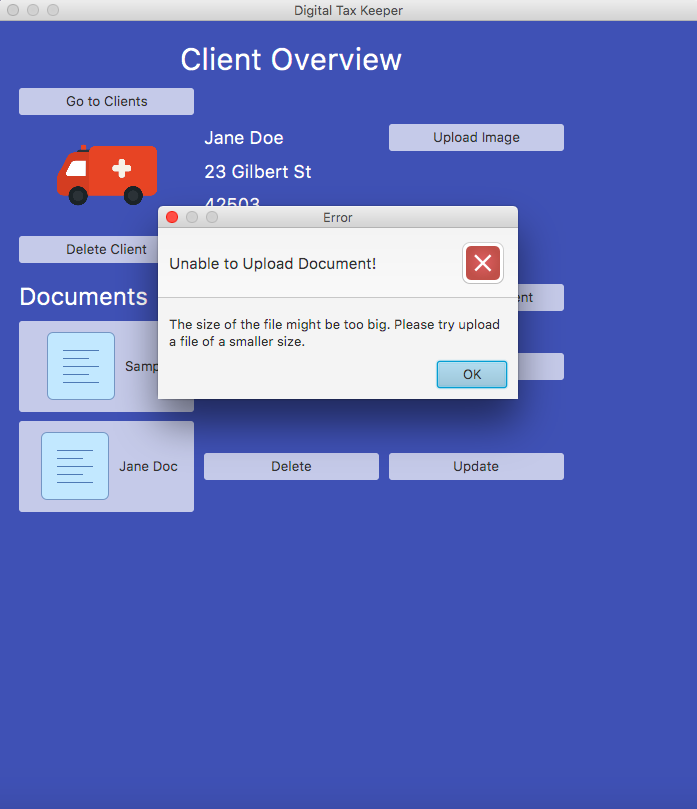
*Techniques:* Storage Mechanism, Thinking Ahead

*Evidence:* Use of Database, Error Dialogs, Document Upload

**GUI Snippet #10 (FileChooser Dialog)**

****

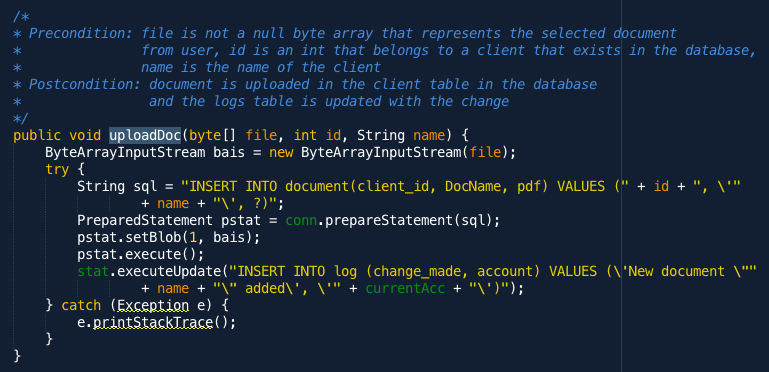
**GUI Snippet #11 (Client Overview)**

****

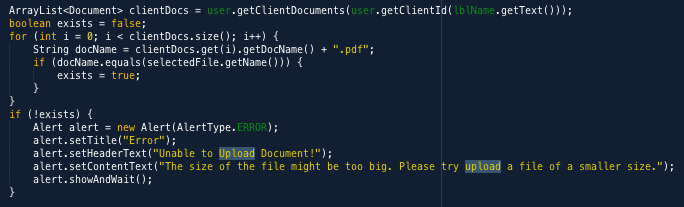
**Code Snippet #11** *(Refer to the Appendix F for full details of code)*

****

**Code Snippet #12** *(Refer to the Appendix F for full details of code)*



**Code Snippet #13** *(Refer to the Appendix F for full details of code)*

**

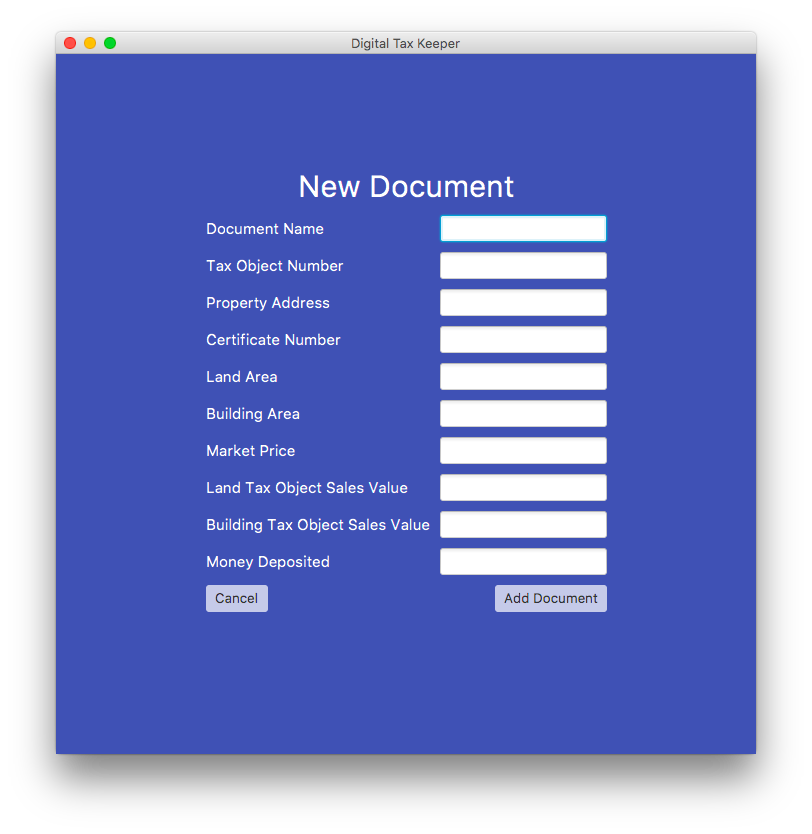
This uploadDocument method allows the client to select a PDF document from their device and upload it to the database. Filtering the FileChooser dialog so that users can only select PDF documents is an example of Thinking Ahead. If the file is too large, error dialog is displayed as seen in **Code Snippet #13,** an example of Thinking Ahead. This satisfies the ninth success criteria that states users should be able to upload documents to the application and this feature is important as Mrs. Erlina is hoping that she will be able to save existing documents that are already made onto the application without going through the manual labor of entering all of the information to make a new document.

**Technique #6: Creating Documents**

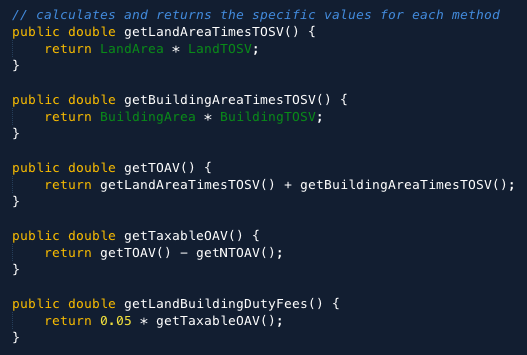
*Techniques:* Storage Mechanism, Thinking Abstractly, Computational Thinking, Thinking Logically

*Evidence:* Use of Database, Converting Numbers to Words, Document Calculations, Making Documents

**GUI Snippet #12 (Create Document Page)**

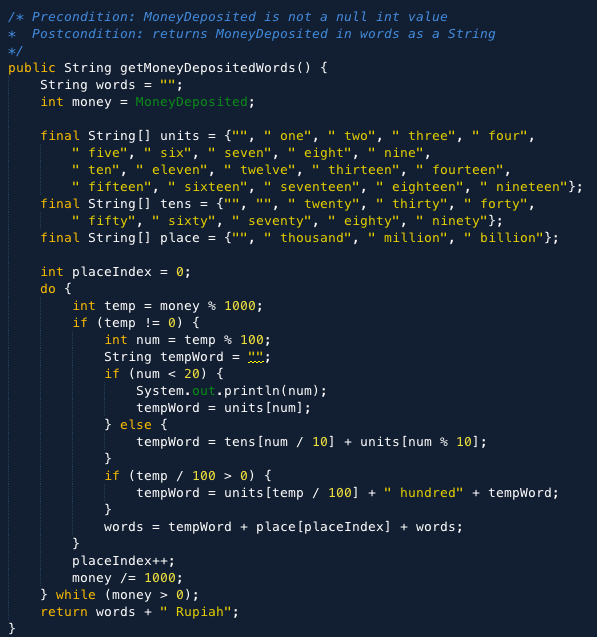
****

**Code Snippet #14** *(Refer to the Appendix F for full details of code)*



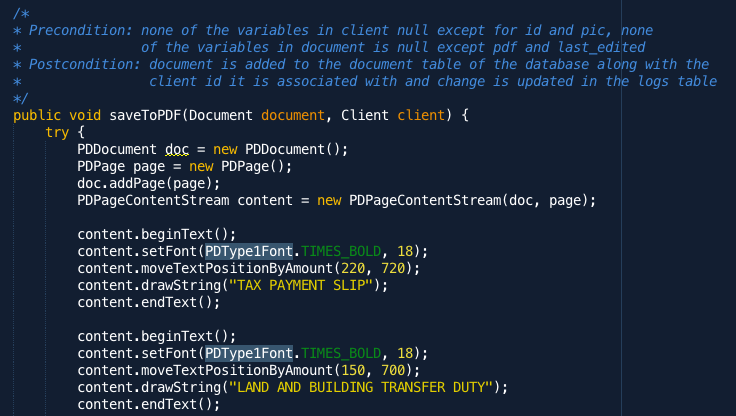
Above are calculations included in the PDF document based on the inputs provided. Automated calculations help users ensure that the numbers in the PDF are accurate and make this process more efficient, increasing productivity, which Mrs. Erlina wanted. This is an example of Computational Thinking and satisfies the sixth success criteria that mentions that the app will programmatically fill out parts of the document that can be calculated. This feature is important as this means that the user will not need to enter all of the fields as inputs and can help prevent the user from miscalculating anything, which Mrs. Erlina mentioned was a big concern especially when dealing with legal documents.

**Code Snippet #15** *(Refer to the Appendix F for full details of code)*

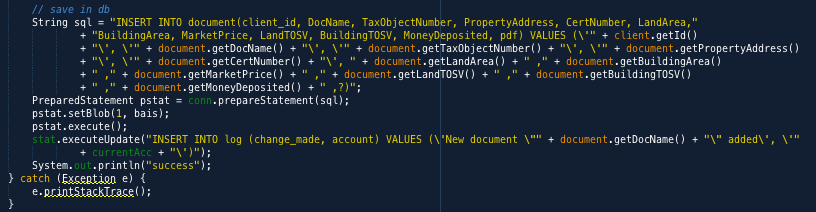


This method converts the MoneyDeposited from integer to words programmatically for the user. This method helps the user avoid typo errors and will continuously update itself if the value changes. The algorithm is an example of complexity and Thinking Logically. This satisfies the sixth success criteria as it programmatically translates the MoneyDeposited value entered by the user into words, which is convenient as the user will not need to enter it manually and it also reduces the risk of a human error/typos.

**Code Snippet #16** *(Refer to the Appendix F for full details of code)*

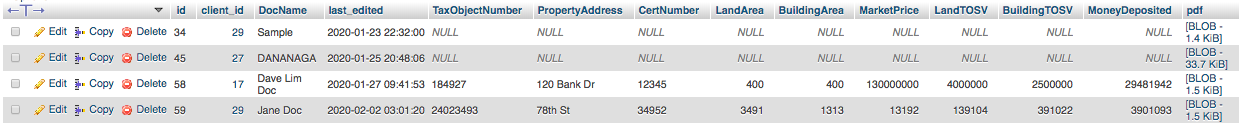


**Code Snippet #17** *(Refer to the Appendix F for full details of code)*



This saveToPDF method creates the PDF document based on input represented in the Document Object and Client Object, examples of Thinking Abstractly. In **Code Snippet #17,** the pdf document is uploaded to the “document” table in the database so it is retrievable in the future and “log” is updated with the change. This satisfies the sixth success criteria as users can easily make documents by entering the required input, which makes this app convenient for Mrs. Erlina as she will not have to make it separately on a different platform but do everything in this app. The seventh success criteria is also satisfied as the document is made as a PDF document and later on saved as BLOB, which makes it easy to convert back to a PDF document when the user views it.

**Database Screen #3**



This is the “Document” table of the database where it consists of 14 columns and each row represents each document. This is updated every time the user adds/updates/remove documents on the database.

**Words:** 1005

**Works Cited**

Anshudeep. “Converting Numbers to Words - Java Program.” *Blogspot.Com*, 14 Nov. 2018, netjs.blogspot.com/2018/11/converting-numbers-to-words-java-program.html. Accessed 12 Nov. 2019.

“FileNameExtensionFilter (Java Platform SE 7 ).” *Oracle.Com*, 6 Oct. 2018, docs.oracle.com/javase/7/docs/api/javax/swing/filechooser/FileNameExtensionFilter.html. Accessed 7 Dec. 2019.

“Flaticon, the Largest Database of Free Vector Icons.” *Flaticon*, 2013, www.flaticon.com/.

“Free Icons.” *Icons8.Com*, 2019, icons8.com/icons. Accessed 12 Nov. 2019.

“How to Convert from BufferedImage to JavaFX 2.2 Image.” *Java PDF Blog*, Nov. 2012, blog.idrsolutions.com/2012/11/convert-bufferedimage-to-javafx-image/. Accessed 7 Dec. 2019.

Jakob Jenkov. “JavaFX Button.” *Jenkov.Com*, 2016, tutorials.jenkov.com/javafx/button.html. Accessed 29 Oct. 2019.

Jakob, Marco. “JavaFX Dialogs (Official) | Code.Makery.Ch.” *Makery.Ch*, 28 Oct. 2014, code.makery.ch/blog/javafx-dialogs-official/. Accessed 2 Feb. 2020.

“Java: Convert a File to a Byte Array, Then Convert Byte Array to a File.” *Programcreek.Com*, 2020, www.programcreek.com/2009/02/java-convert-a-file-to-byte-array-then-convert-byte-array-to-a-file/. Accessed 2 Feb. 2020.

Prasad Saya. “JavaFX FileChooser Example | Examples Java Code Geeks - 2019.” *Examples Java Code Geeks*, 10 Apr. 2015, examples.javacodegeeks.com/desktop-java/javafx/javafx-filechooser-example/. Accessed 7 Dec. 2019.

Prasad Saya. “JavaFX TableView Example | Examples Java Code Geeks - 2020.” *Examples Java Code Geeks*, 25 Mar. 2015, examples.javacodegeeks.com/desktop-java/javafx/tableview/javafx-tableview-example/. Accessed 2 Feb. 2020.

“Skinning JavaFX Applications with CSS | JavaFX 2 Tutorials and Documentation.” *Oracle.Com*, 2013, docs.oracle.com/javafx/2/css\_tutorial/jfxpub-css\_tutorial.htm. Accessed 2 Feb. 2020.

“SQL Tutorial.” *W3schools.Com*, 2019, www.w3schools.com/sql/.

Tariq, Azeem. “PDFBox Example | Create PDF File With Text in Java - Radix Code.” *Radixcode.Com*, 2017, radixcode.com/pdfbox-example-create-pdf-file-with-text-in-java. Accessed 12 Nov. 2019.