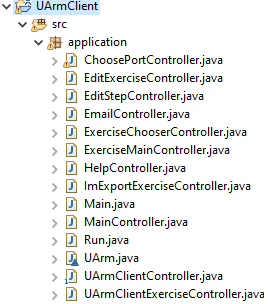
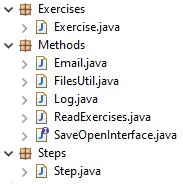
# **Criterion C: Product Development**

Figure 1- All Classes



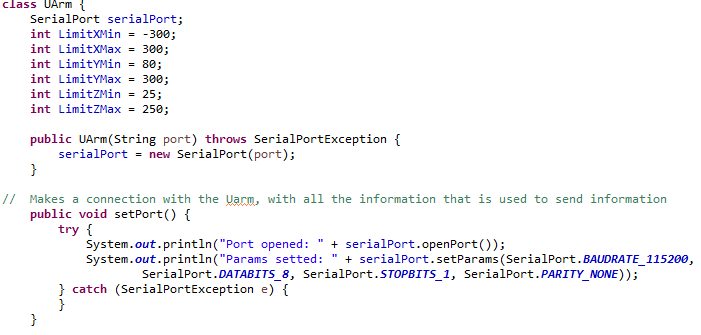


## Some Techniques used:

* Development of Methods
* Manipulation of sequential files
* Use of Graphic User Interface
* Establishing connection between user and uArm
* Static methods and variables
* Modular Programming

## Methods

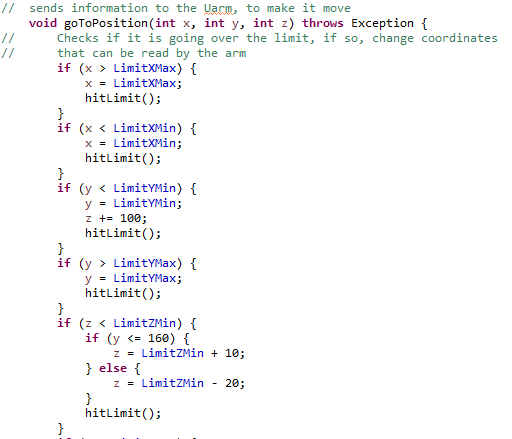
The following method is essential to establish a connection between the user and the **uArm**.

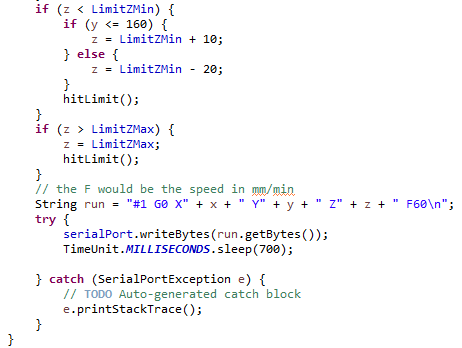


This opens the port and sets the parameter of the **uArm** for the speed of transmission and for reading data using a simple third party library called *JSSC* (Java Simple Serial Controller) because it can easily establish a connection without using complex algorithms.[[1]](#footnote-2)



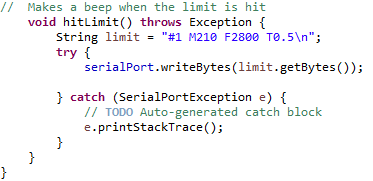
The following method makes the arm move and checks the limits set.



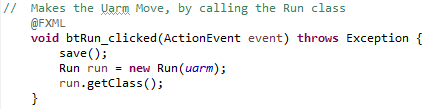


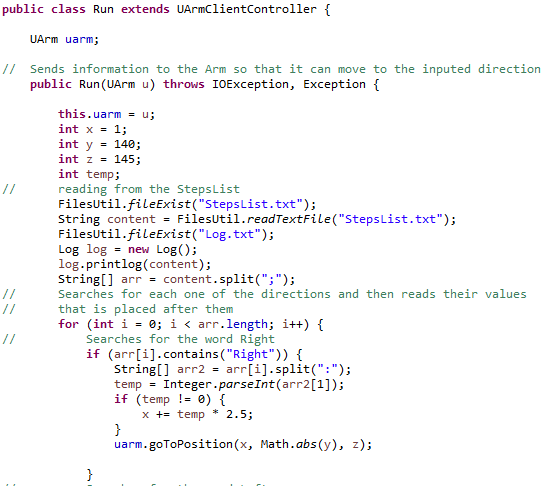
The string is converted into bytes and read by the arm, to go to a certain coordinate. Additionally, the Enumerated type **TimeUnit** creates an interval before the next step for the user to watch the steps taken.[[2]](#footnote-3) These limits were developed through trial and error, so no unnecessary movements are executed (shown in the video) as it calls the next method to make the arm beep, when a limit’s hit to inform the user, for user-friendliness reasons.

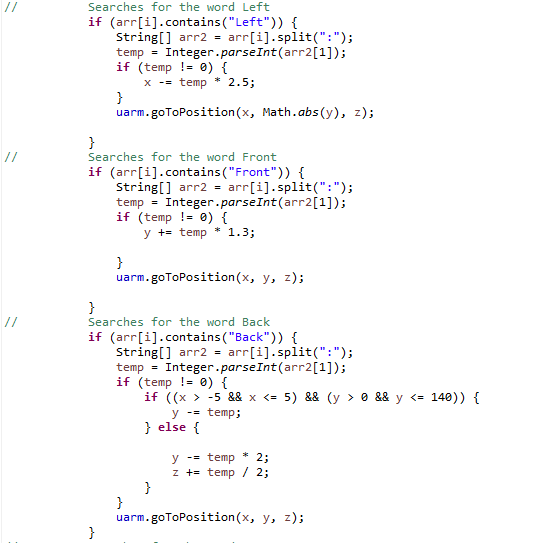
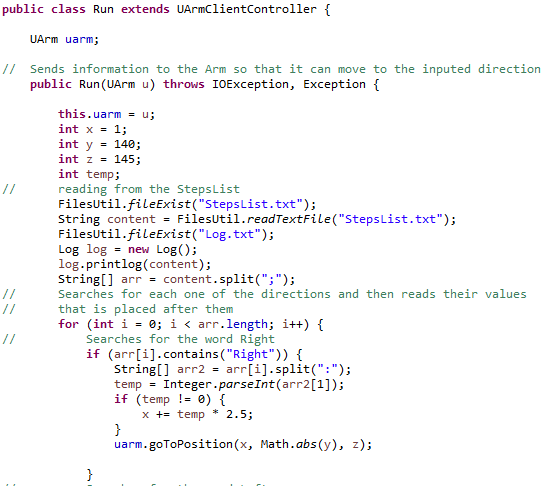


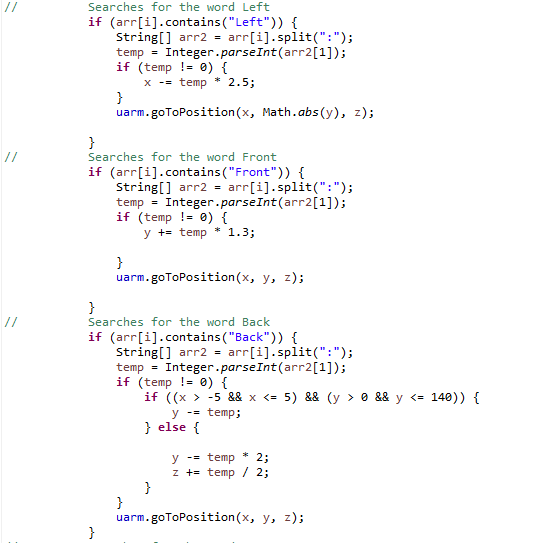


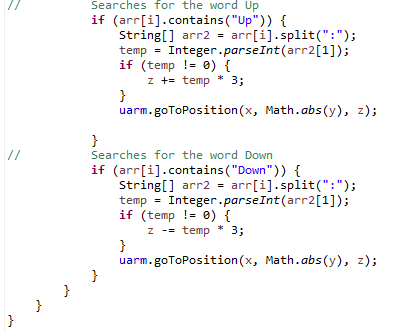
The following method located in the **UArmClientController** executes the list of Steps.



The **Run** class inherits this controller, for organizational reasons, as it makes a *linear search* for each direction and corresponding distance in the order the user selected. 







Each *if statement* was developed through trial and error to ensure the arm goes to the desired location without error.

The **Log** method shows evidence of completion and a form of backup as it utilizes the **Date[[3]](#footnote-4)** and **SimpleDateFormat[[4]](#footnote-5)** class to inform the user the time of the steps executed.

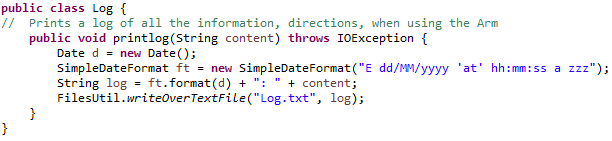
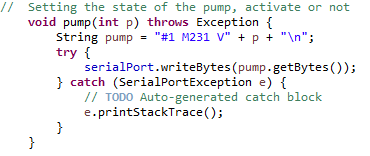


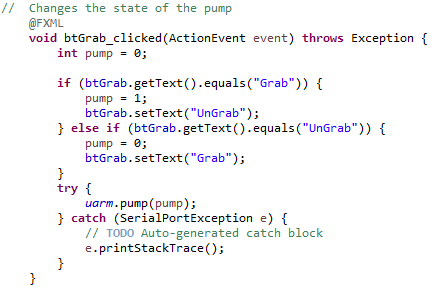
Figure 2- Log.txt



The **pump** can be activated or disactivated.



For usability reasons, one click from the **Grab** button changes the pump’s status.



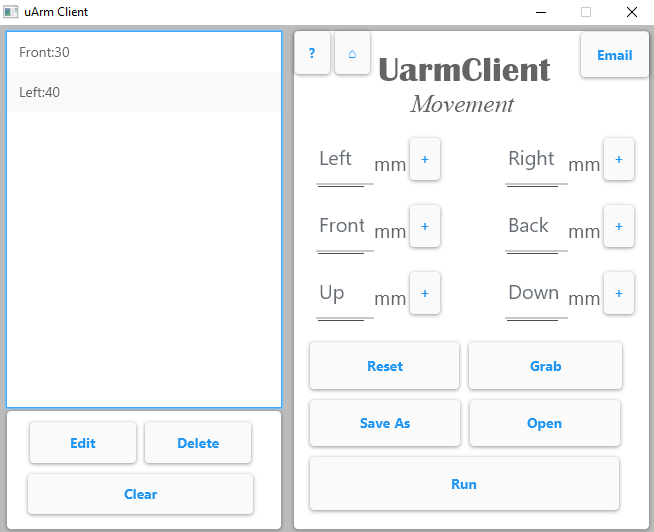
*Exception handles* are used to maintain the flow of the application, so it doesn’t crash or gets interrupted, making the application work more easily.

## Graphic User Interface (GUI)

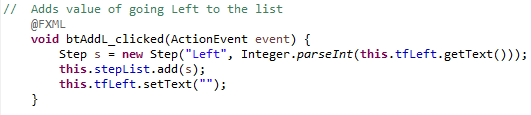
The *GUIs* and *JavaFX* makes the application aesthetically pleasant and facilitates the use of the uArm as it improves the efficiency, functionality, and usability.[[5]](#footnote-6)

The following *GUI* allows simple communication between the user and uArm.

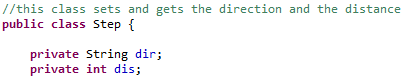
Figure 3- UArmClientController

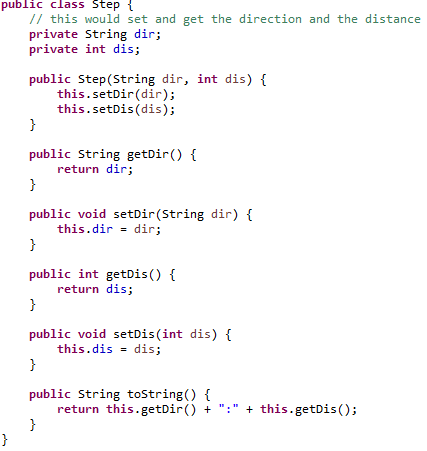


I used the “+” sign when adding the direction and distance to the list (one of six).



For each step created, the direction and distance is set.





Using *encapsulation* and *private variables* protects variables from being easily changed. However, clicking the **Edit** button allows editing.

Figure 4- Editing Distance

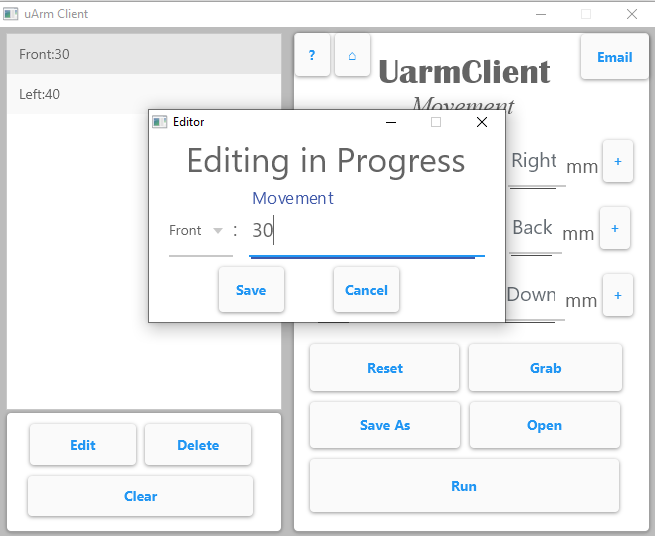
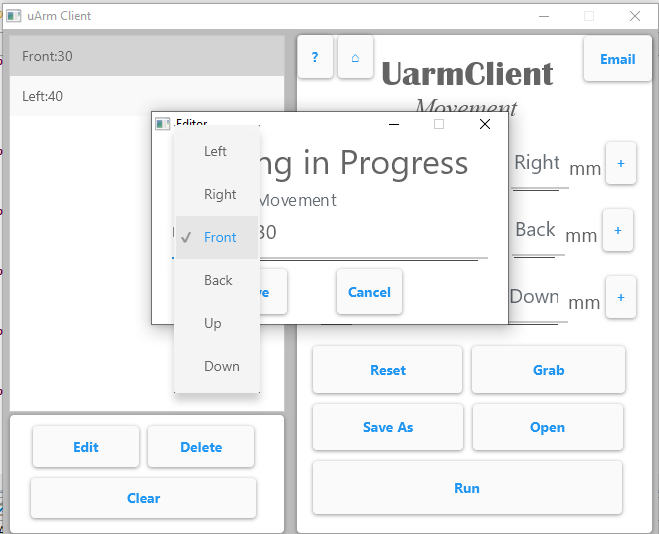
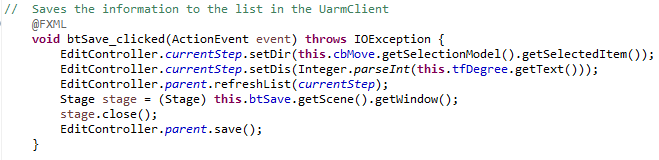


Figure 5-Editing Direction

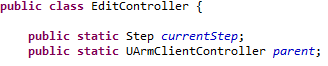


The **Save** button overwrites the selected step from the list to increase functionality, so the user wouldn’t need to delete and add another step.

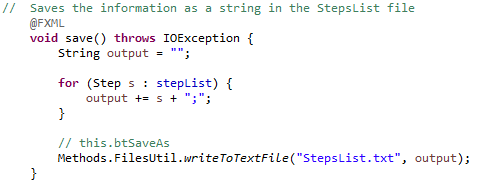


The parent class is the **UArmClientController**.





Using *inheritance* enables me to use the **save** method to save each step into the *current directory* as a .txt called “StepsList” right after editing.



The **Save As** and **Open** button creates a File Explorer by using the **FileChooser** class to facilitate user navigation when working with sequential files.[[6]](#footnote-7)

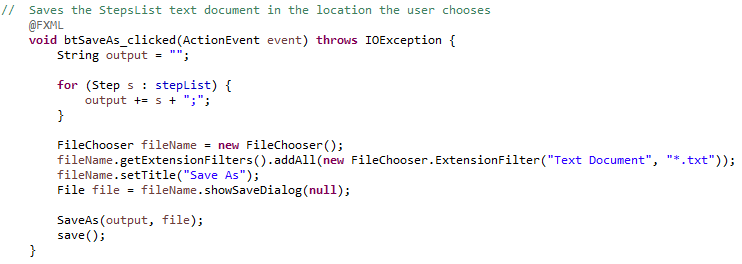


Figure 6- Save As button

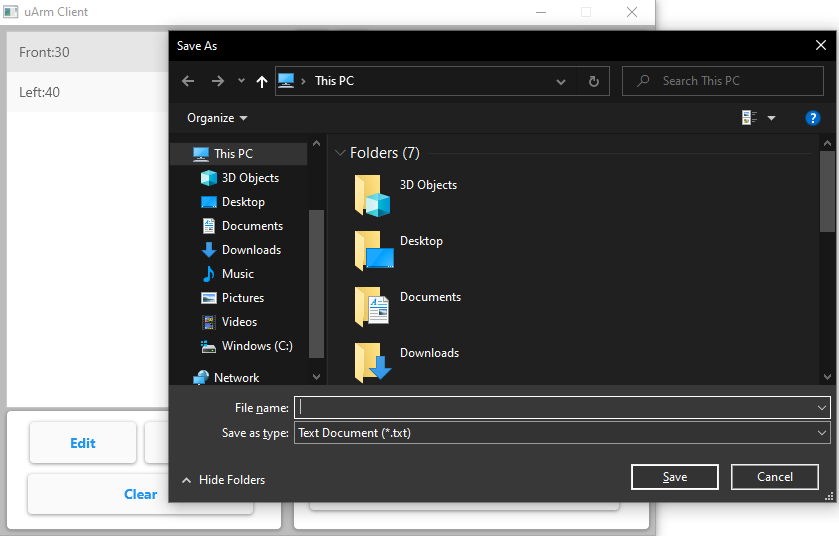
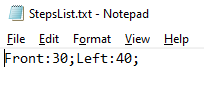
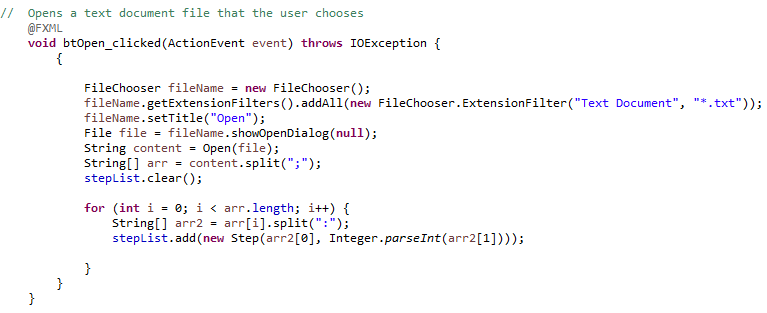


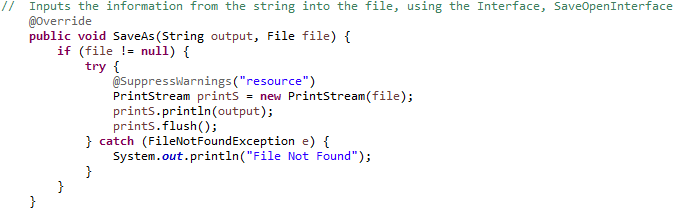
Figure 7- Text Document of Steps List



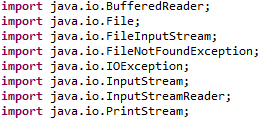
Formatting in *CSV* (“;”) is necessary to effectively read files as the *split method* can help read individual steps (direction and distance). Shown when **Open** is clicked.

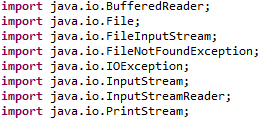


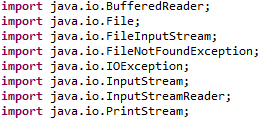
The **SaveAs** method uses the **PrintStream** class because it is suitable to print various types of data values without throwing *Exceptions*, which helps me better understand future errors.[[7]](#footnote-8)

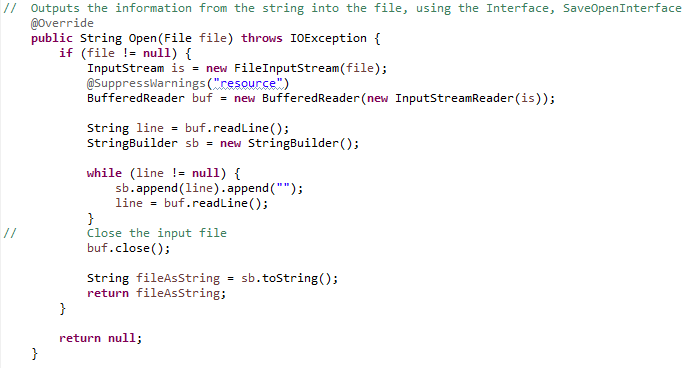


The **Open** method uses the **BufferReader** class to efficiently read strings as it reads a stream of characters from the file after it converts them into a string. An **InputStream** object first needs to be declared as its constructor accepts the **InputStream** object as a parameter to function.[[8]](#footnote-9)

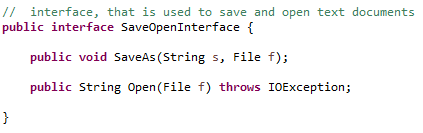




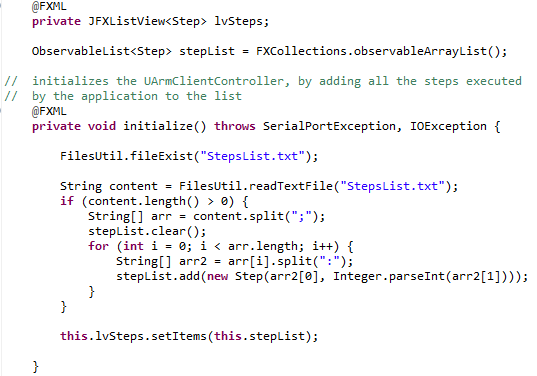


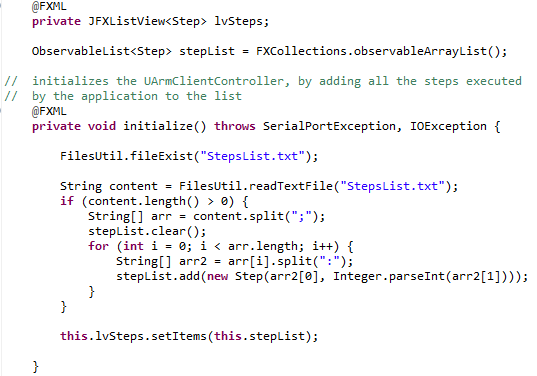


These methods implement the *interface*, **SaveOpenInterface** which defines the protocol used by **Exercises** to make the application run more smoothly since relationships between classes aren’t forced.[[9]](#footnote-10)

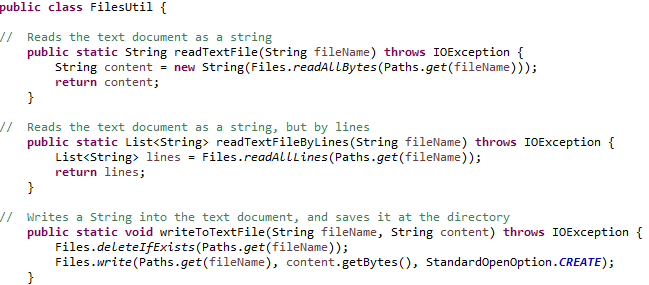


Similar to the **Open** method, the **UArmClientController** reads a file in the *current directory* to improve the functionality as an error wouldn’t occur. (explained later)

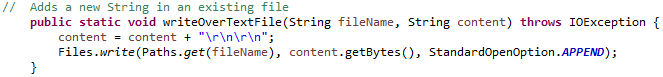




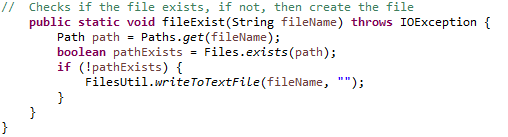
When working with files, the following methods are necessary to read and write files in the *current directory*. Developed using the **Files** classto provide a system-independent view of paths and simple manipulation of user accessible files.[[10]](#footnote-11)



The following method uses an *append*to logs all steps executed.

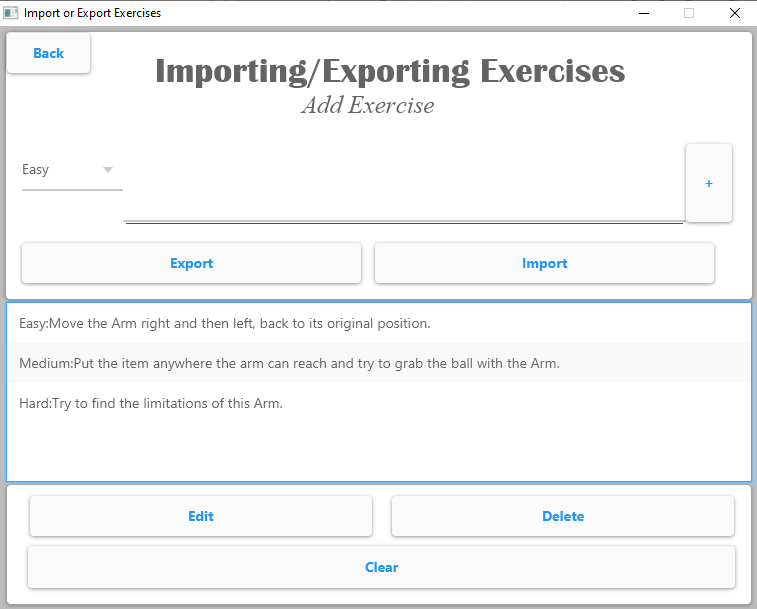


The following method checks if the file being read is in the *current directory*, if not, it’s created to ensure there aren’t errors when reading files. This improves flexibility as it creates a relative path for the application, making it work in any location installed.

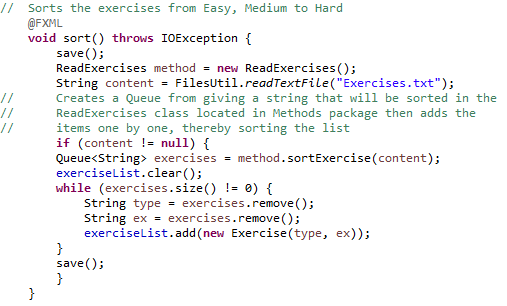


The **ImExportExerciseController** contains alist of exercises in three levels (easy, medium, hard).

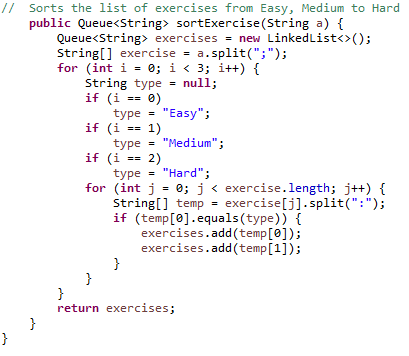
Figure 8- ImExportExerciseController



To improve organization, the following method automatically sorts the list of exercise when exercises are edited or added.



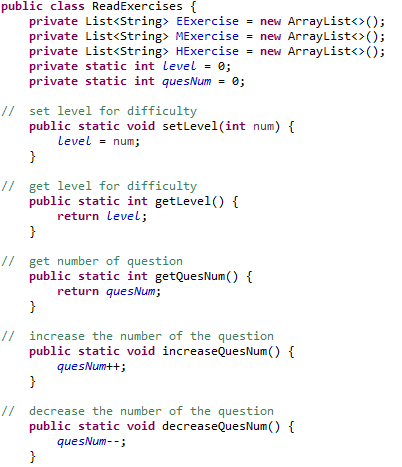
This invokes a method called “sortExercise” located in the **ReadExercises** class, which utilizes a *Queue to sort exercises by level*. The method uses the *enqueue* and *dequeue* concepts (**add()** and **remove()** in Java) *and* it outmatches arrays and stacks.

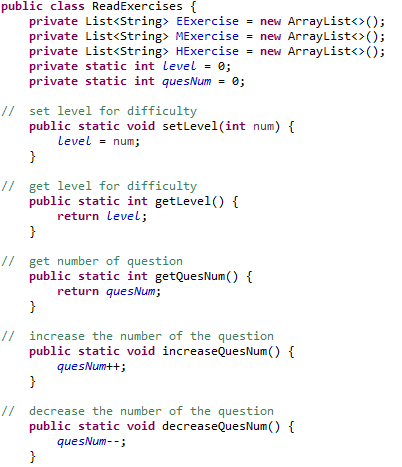


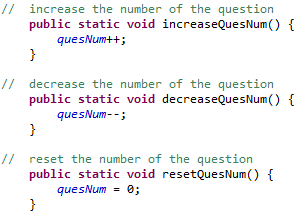
The two loops iterate three times for each level and *enqueues* them to the list.

The **ReadExercises** class contains three *private Array Lists* for each level since there isn’t a set size, so there isn’t a limit to the number of exercises to being added. Moreover, searching isn’t necessary since its already organized in levels.

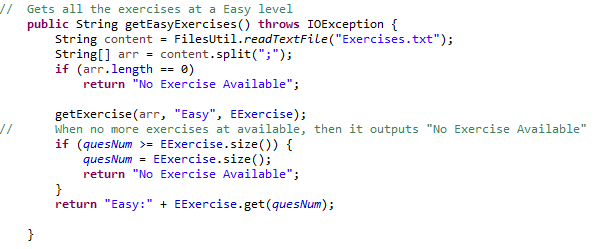






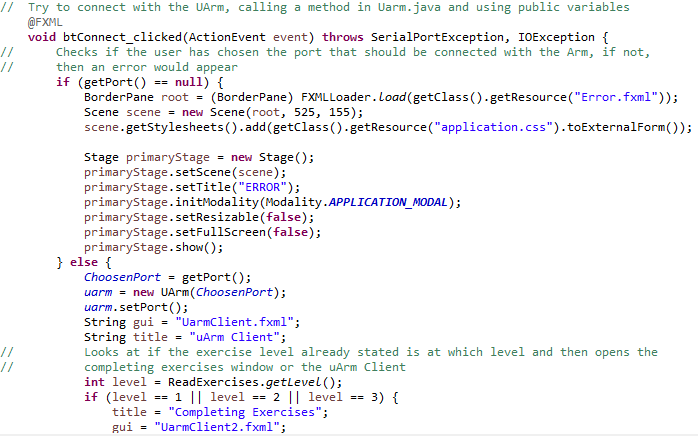
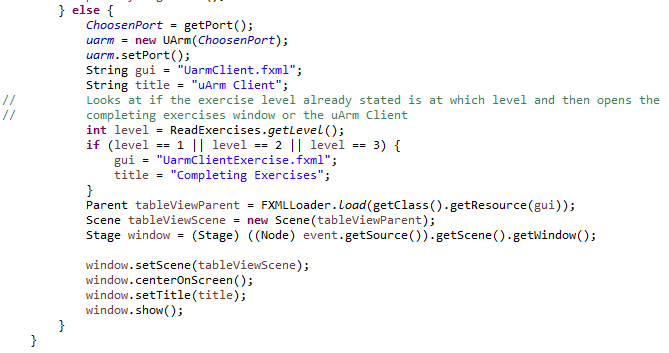


The following method (one out of three) returns the exercise at the chosen level.



For user-friendliness, a *Breakpoint* is used, to inform when there aren’t any more exercises.

To get rid of redundancies, instead of creating different connection controllers, an *if statement* is used. Moreover, if the **uArm** isn’t connected, an error appears since it’s not possible to use it.

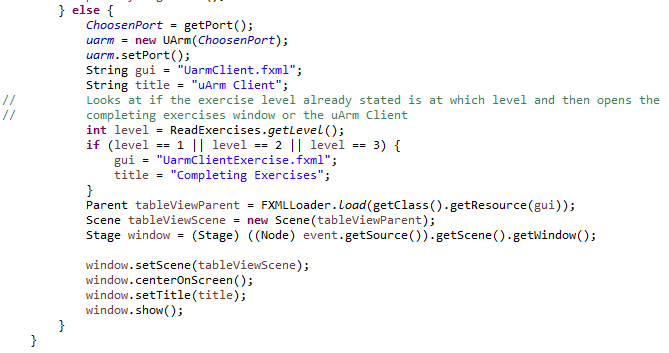
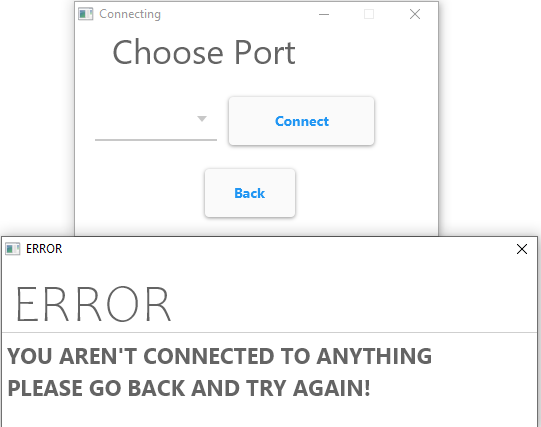
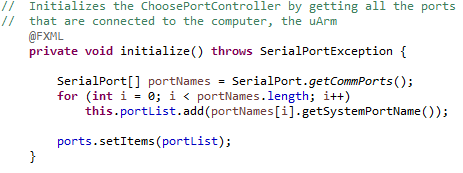


Figure 9- Connecting to uArm



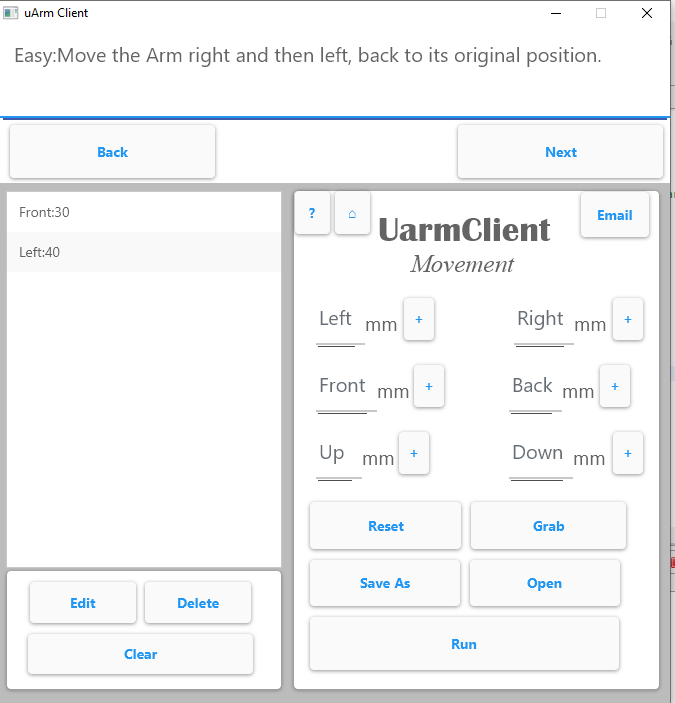


This shows the connections by using the *jSerialComm* library as its superior when getting the name of connected ports as it doesn’t require external libraries.[[11]](#footnote-12)



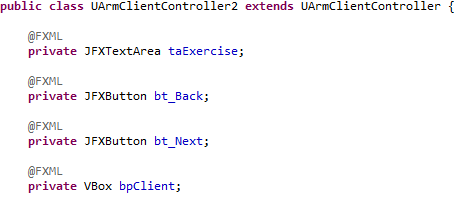
The following GUI is to complete exercises.

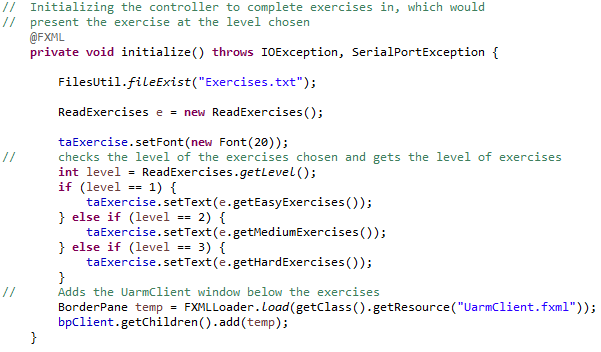
Figure 10- Completing Exercises



Initializing the controller to get the exercise corresponding to level.

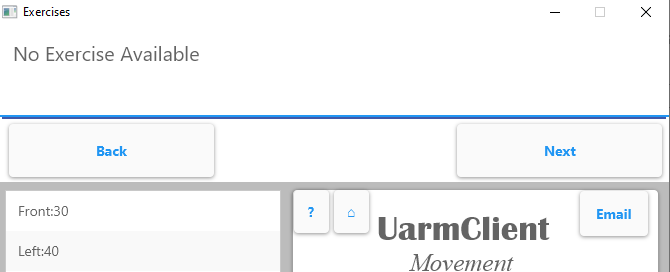




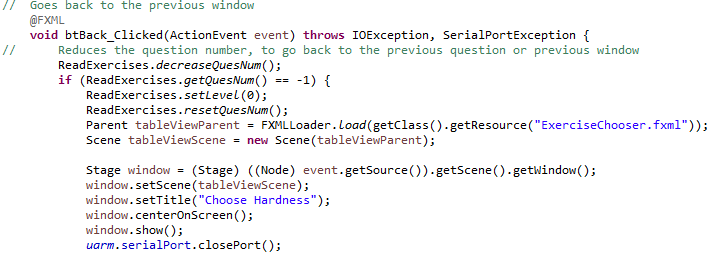


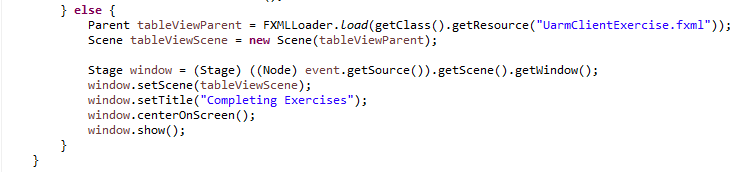
*Inheritance* is used to reduce code when duplicating the **UArmclientController’s** functionalities and GUI.

Figure 11- No Exercise Available

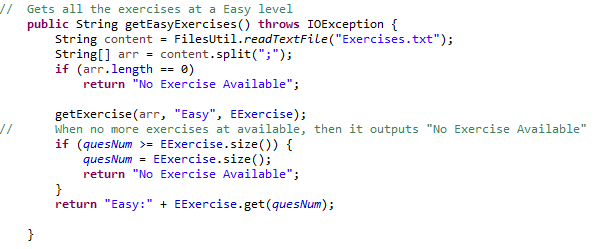


The **Back** button boosts navigation allowing the user to select the level again or see the previous exercise.

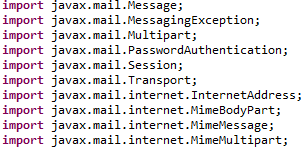


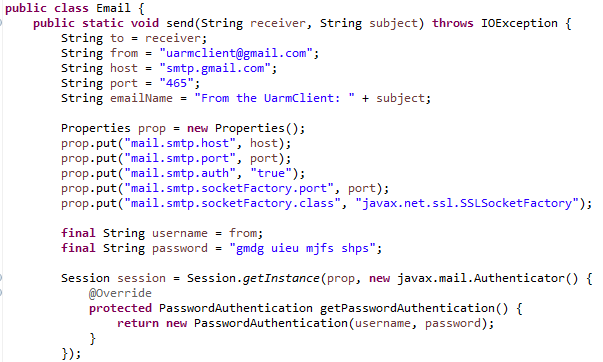


Furthermore, to improve efficiency, no matter how many times **Next** is pressed, pressing **Back** opens thelast exercise (located in the method to get exercises).



To enhance the application’s service, the files/progress can be sent through email. The **Email** class is in the **Methods** package, an example of *modular programming* used to make extensibility easier, improve the management and maintenance of the application.





The *outgoing mail server*, stmp.gmail.com is used because it’s free and easily sends emails with attachments as it uses a two-step verification, Gmail account, host, port and its properties, internet connection.[[12]](#footnote-13)

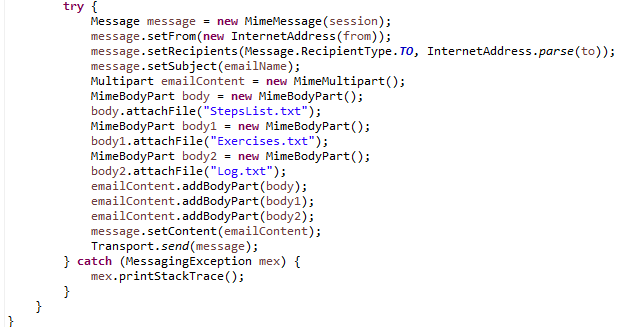
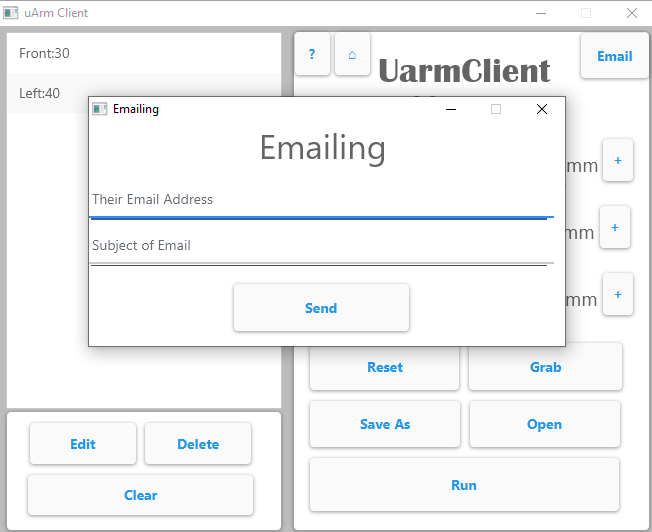


Figure 12- Email



As shown in this Criterion, most of the methods implemented were ideas thought of from the planning stage. Even though there were some changes, the concepts used were planned out as they helped me more efficiently create the application. Additionally, these ideas, preparations, developments, and implementations were effective and utilized thanks to learning computer science in the last two years of high school.

**Word Count:** 1065 (excluding Titles, Captions and Parentheses)

## List of References

"Class BufferedReader." *Oracle*, docs.oracle.com/javase/8/docs/api/java/io/BufferedReader.html. Accessed 10 Jan. 2021.

"Class Date." *Oracle*, docs.oracle.com/javase/7/docs/api/java/util/Date.html. Accessed 8 Jan. 2021.

"Class File." *Oracle*, docs.oracle.com/javase/7/docs/api/java/io/File.html. Accessed 13 Jan. 2021.

"Class FileChooser." *Oracle*, docs.oracle.com/javase/8/javafx/api/javafx/stage/FileChooser.html. Accessed 6 Jan. 2021.

"Class PrintStream." *Oracle*, docs.oracle.com/javase/7/docs/api/java/io/PrintStream.html. Accessed 15 Jan. 2021.

"Class SimpleDateFormat." *Oracle*, docs.oracle.com/javase/7/docs/api/java/text/SimpleDateFormat.html. Accessed 6 Jan. 2020.

"Enum TimeUnit." *Oracle*, docs.oracle.com/javase/7/docs/api/java/util/concurrent/TimeUnit.html. Accessed 8 Jan. 2021.

Fazecast, editor. "jSerialComm." *GitHub*, fazecast.github.io/jSerialComm/. Accessed 6 Jan. 2021.

The JavaTM Tutorial, editor. "What Is an Interface?" *IIT Kanpur*, www.iitk.ac.in/esc101/05Aug/tutorial/java/concepts/interface.html#:~:text=You%20use%20an%20interface%20to,artificially%20forcing%20a%20class%20relationship. Accessed 15 Jan. 2021.

"jfoenix documentation." *JFoeniX*, www.jfoenix.com/documentation.html. Accessed 10 Jan. 2021.

"jSSC (Java Simple Serial Connector)." *javalibs*, javalibs.com/artifact/org.scream3r/jssc. Accessed 8 Jan. 2021.

SiteGround, editor. "Google SMTP server - how to send emails for free?" *SiteGround*, www.siteground.com/kb/google\_free\_smtp\_server/#:~:text=Google's%20Gmail%20SMTP%20server%20is,can%20use%20to%20send%20emails.&text=Outgoing%20Mail%20(SMTP)%20Server%3A,mail%20client%2Fwebsite%20SMTP%20plugin). Accessed 17 Jan. 2021.

1. "jSSC (Java Simple Serial Connector)." *javalibs*, javalibs.com/artifact/org.scream3r/jssc. Accessed 8 Jan. 2021. [↑](#footnote-ref-2)
2. "Enum TimeUnit." *Oracle*, docs.oracle.com/javase/7/docs/api/java/util/concurrent/TimeUnit.html. Accessed 8 Jan. 2021. [↑](#footnote-ref-3)
3. "Class Date." *Oracle*, docs.oracle.com/javase/7/docs/api/java/util/Date.html. Accessed 8 Jan. 2021. [↑](#footnote-ref-4)
4. "Class SimpleDateFormat." *Oracle*, docs.oracle.com/javase/7/docs/api/java/text/SimpleDateFormat.html. Accessed 6 Jan. 2020. [↑](#footnote-ref-5)
5. "jfoenix documentation." *JFoeniX*, www.jfoenix.com/documentation.html. Accessed 10 Jan. 2021. [↑](#footnote-ref-6)
6. "Class FileChooser." *Oracle*, docs.oracle.com/javase/8/javafx/api/javafx/stage/FileChooser.html. Accessed 6 Jan. 2021. [↑](#footnote-ref-7)
7. "Class PrintStream." *Oracle*, docs.oracle.com/javase/7/docs/api/java/io/PrintStream.html. Accessed 15 Jan. 2021. [↑](#footnote-ref-8)
8. "Class BufferedReader." *Oracle*, docs.oracle.com/javase/8/docs/api/java/io/BufferedReader.html. Accessed 10 Jan. 2021. [↑](#footnote-ref-9)
9. The JavaTM Tutorial, editor. "What Is an Interface?" *IIT Kanpur*, www.iitk.ac.in/esc101/05Aug/tutorial/java/concepts/interface.html#:~:text=You%20use%20an%20interface%20to,artificially%20forcing%20a%20class%20relationship. Accessed 15 Jan. 2021. [↑](#footnote-ref-10)
10. "Class File." *Oracle*, docs.oracle.com/javase/7/docs/api/java/io/File.html. Accessed 13 Jan. 2021. [↑](#footnote-ref-11)
11. Fazecast, editor. "jSerialComm." *GitHub*, fazecast.github.io/jSerialComm/. Accessed 6 Jan. 2021. [↑](#footnote-ref-12)
12. SiteGround, editor. "Google SMTP server - how to send emails for free?" *SiteGround*, www.siteground.com/kb/google\_free\_smtp\_server/#:~:text=Google's%20Gmail%20SMTP%20server%20is,can%20use%20to%20send%20emails.&text=Outgoing%20Mail%20(SMTP)%20Server%3A,mail%20client%2Fwebsite%20SMTP%20plugin). Accessed 17 Jan. 2021. [↑](#footnote-ref-13)