Table of Contents

[1.0 Introduction 3](#_Toc171015479)

[2.0 User Manual 5](#_Toc171015480)

[2.1. Patient User Interface 9](#_Toc171015481)

[2.2. Doctor User Interface 14](#_Toc171015482)

[2.3. Admin User Interface 19](#_Toc171015483)

[3.0 Sample Input Output and Explanation. 27](#_Toc171015484)

[3.1. File Handler (Create Walk in Appointment) 27](#_Toc171015485)

[3.2. File Handler (Import User Data from Text File) 28](#_Toc171015486)

[3.3. File Handler (Update User Details) 30](#_Toc171015487)

[3.4. File Handler (Remove Data from Text File) 32](#_Toc171015488)

[3.5. ArrayList and ArrayDeque 34](#_Toc171015489)

[4.0 Object Oriented Programming Concept with Sample Code 36](#_Toc171015490)

[4.1. Encapsulation 36](#_Toc171015491)

[4.2. Inheritance, Polymorphism and Interface Implementation 38](#_Toc171015492)

[4.3. Abstraction 39](#_Toc171015493)

[4.4. Singleton Pattern 40](#_Toc171015494)

[4.5. Setter 41](#_Toc171015495)

[4.6. Getter 41](#_Toc171015496)

[4.7. Single Responsibility Principle (SRP) 42](#_Toc171015497)

[5.0 Additional Functions 44](#_Toc171015498)

[5.1. Quick Access Table 44](#_Toc171015499)

[5.2. Report Function 45](#_Toc171015500)

[6.0 Conclusion 47](#_Toc171015501)

[7.0 References 49](#_Toc171015502)

# Introduction

The project below details a comprehensive and robust clinic management system built for Saint Denis Clinic, which was built using Java, more specifically using jdk 13 and java Swing, which results in a lightweight program that can be run on most Windows systems. The preferred IDE was Visual Studio code for its robust UI as well as its many IntelliSense packages, while GitHub was used as the primary version control platform. The developers have also ensured to use clean coding and object-oriented programming (OOP) concepts so that the source code is easy-to-read, well-documented and scalable.

The primary purpose of this system is to streamline the clinic’s day-to-day operations, which was previously done manually either by writing or by word-of-mouth. Once the main purpose of making the overall operations of the clinic has been achieved, it inadvertently allows for better care and attention towards patients since it allows for better utilization of the clinic’s staffs’ time. Moving on, another pertinent goal of a better system would be to allow for more security and integrity when it come to data-keeping, since all patient records are stored digitally and if possible, with frequent backups to the cloud, thus reducing the probability of data loss or data manipulation by outsiders. Finally, we understand that patients of a general practicing clinic would be coming from all walks of life, which is why we emphasize on having a simple and uniform UI which is easy to navigate and learn.

The main functionalities of our system to achieve the goals listed above revolve around the booking of doctor’s appointments, which is why we have made it easy for users to view available slots and make bookings. Doctors can also easily see how many they have for the day and also set their available booking sessions accordingly. There is also a receptionist/admin role in our system who will be in charge of the general affairs of the clinic, from assigning spots of walk-in customers based on the severity of their case to collecting the payment of patients who have already received treatment.

# User Manual

A screenshot of a login page

Description automatically generated

Figure 1: Login Form

A screenshot of a computer error

Description automatically generated

Figure 2: Login Failed Error Message

This is the first page which users are greeted with when they first open our program. The user will have to input their Username and Password, then click the “Login” button to enter. If the user’s credentials are incorrect, a pop will appear to inform them that the login has failed. To register, a user will have to walk into the clinic and give their details to the receptionist. The user will then be redirected based on their roles (Patient, Doctor, Admin).

A screenshot of a computer program

Description automatically generated

Figure 3: Logout and Setting Buttons

A screenshot of a computer

Description automatically generated

Figure 4: Successfully Logout Message

The Logout button is available to all users in the top right corner to end their session. Once clicked, they will be redirected to the Login Form with a message.

A screenshot of a computer

Description automatically generated

Figure 5: Setting Page

The Setting Page is also available for users of every role. They can change their password by entering their old password and then entering the or delete their account.

A screenshot of a computer screen

Description automatically generated

Figure 6: Change Password Page

A screenshot of a computer error

Description automatically generated

Figure 7: Incorrect Password Error Message

A screenshot of a computer screen

Description automatically generated

Figure 8: New and Old Pass cannot be the same Error Message

A screenshot of a computer

Description automatically generated

Figure 9: Password Successfully Changed Message

Users will have to enter the correct old password, otherwise an error message will pop up. An error message will also pop up if the user keys in the same password as the old one. They can then either choose to cancel or press the submit button to confirm the process, where they will be notified if the process succeeded.

A screenshot of a computer

Description automatically generated

Figure 10: Delete Account Confirmation

A screenshot of a computer

Description automatically generated

Figure 11: Account Deleted Successfully Confirmation Message

If users click on the delete account button, there will be a pop up to confirm the action. Once the user clicks “Yes”, a final confirmation will pop up and then the user will be redirected to the Home Page.

## Patient User Interface

A screenshot of a computer

Description automatically generated

Figure 12: Patient Home Page

For patients, their user interface is extremely simple. There are only 2 mains functions which the user can make use of, which is to make an appointment and also to track their history. For users to know that they have logged into the correct account, their name will be displayed at the top of the page along with a welcome message.

A screenshot of a computer

Description automatically generated

Figure: Cancel Appointment Page

A screenshot of a computer screen

Description automatically generated

Figure 13: Cancel Booking Confirmation Message

A screenshot of a computer error

Description automatically generated

Figure 14: Appointment Cancelled Message

The first button which the user will see is the “Cancel Appointment” button, which gives the user access to cancel their previously booked appointment, such as if they booked it wrongly, or if they want to make any changes. The quick access table on the right allows users to see all their appointments and fill in the text fields quickly by clicking on the row of the record. A pop up will ask the users for a final confirmation before cancelling it giving them a confirmation message.

A screenshot of a computer

Description automatically generated

Figure 15: Make Appointment Page

A screen shot of a computer screen

Description automatically generated

Figure 16: Time Slot Conflict Error Message

A screenshot of a computer error

Description automatically generated

Figure 17: Appointment Added Successfully Message

The main functionality of the User UI is the Make Appointment Page. Here, users can select a doctor from the dropdown list and also search for their available timeslots for that doctor. Click on the row, then just fill in the details and click on the “Make Appointment” button. If the user clicks on a session which has already been booked, an error message will pop up. Otherwise, they will be greeted by a pop up confirming the appointment.

A screenshot of a computer

Description automatically generated

Figure 18: History Page

The History Page allows the user to view two separate reports, which is the user’s general medical record and also the user’s appointment record.

A screenshot of a computer

Description automatically generated

Figure 19: Patient's Personal Record

They are both similar, just that the value they store differs. A medical record stores every medicine which the patient has received from the clinic, while the appointment record only stores the completed appointment sessions which the patient has received from the clinic. When the row is clicked, the values will be inputted to the text fields.

## Doctor User Interface

A screenshot of a computer

Description automatically generated

Figure 20: Doctor Homepage

The doctor’s interface is comprised of 4 main functions, which is to set their own schedule, view his upcoming appointments, view a patient’s record and finally add a record for one of their patients. There are 2 tables on the right, one is to view the upcoming patient queue who did not make any bookings and their own appointment lists.

A screenshot of a schedule

Description automatically generated

Figure 21: New Schedule Page

A screenshot of a computer

Description automatically generated

Figure 22: Schedule Saved Successfully Message

A screenshot of a computer

Description automatically generated

Figure 23: Select at least 1 Time Slot Message

This the scheduling page which lists out all available slots a doctor can choose to set as open, to ensure that users only choose certain timeslots and not book appointments at odd times. When the doctor clicks the submit button, a message confirming the schedule will pop up. The doctor will have to select at least 1 time slot otherwise an error message will pop up.

A screen shot of a computer screen

Description automatically generated

Figure 24: Schedule Cleared Message

The doctors should also click on clear appointment if they would like to change their schedule to ensure that there are no unsatisfied users who book an appointment, but the doctor is not available.

A screenshot of a computer

Description automatically generated

Figure 25: Edit Appointment Page

The edit appointment page is a page for the doctor to update the details of their appointment once they are done treating their patient. There is a table on the right side which displays all pending appointments for the day, and the doctor has to click on the row with the record to edit it.

A screenshot of a computer

Description automatically generated

Figure 26: Empty Text Fields Error Message

If the doctor clicks on the “Update Appointment” button but did not fill in the remarks section, they will be reminded to do so.

A screenshot of a computer

Description automatically generated

Figure 27: View Patient Record

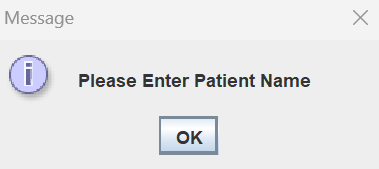


Figure 28: Input Patient Name Error Message

The doctor can also click on the “Patient Record” button to view all records for the current patient which is entered in the text field. The doctor has to input the patient’s name, or they can click on a record with the patient’s name. Otherwise, an error message will pop up. A table on the left displays all the records, and when clicked will display it in a more readable format.

A screenshot of a computer

Description automatically generated

Figure 29: New Record

A screenshot of a computer

Description automatically generated

Figure 30: Fill in All Fields Message

The doctor can add a new record for the patient, where he has to include the issue, medication and fees. The doctor can then click on the “Finish Consultation” button to complete it and allow the admin to collect payment. There is validation to check whether all fields are empty or not as well.

## Admin User Interface

A screenshot of a computer

Description automatically generated

Figure 31: Admin Homepage

The role with the most impact in this system is the admin which also serves as the clinic’s receptionist. The Admin Home Page consists of an Edit User button, an Appointment button to view today’s appointments, a Medical Record button with a text field beside it to search for a user’s medical record and a collect payment feature, which allows the admin to collect payment for any medications. The Queue button allows the admin to view the current appointments by severity. The last Report Button allows the admin to view the expenses for the month.

A screenshot of a computer

Description automatically generated

Figure 32: Edit User Page

The Edit user UI has 3 major functions which are to delete a user, update a user’s details and add a new user. Beside the 3 main buttons there is a convenient “Clear” button which clears all the text fields above. Beside text fields, there is a table which lists out all current users in the system, and the admin can select to either view all the patients or all the doctors by clicking on the “Patient List” or “Doctor List”. The selected row will be highlighted blue, while the rest will be red to help the admins view the records mor easily. To go back, the admin has to click the “Return to Homepage” button.

A screenshot of a computer

Description automatically generated

Figure 33: Delete Account Confirmation

A screenshot of a computer error

Description automatically generated

Figure 34: Account Deleted Successfully Confirmation Message

A screenshot of a computer

Description automatically generated

Figure 35: Account Updated Successfully Message

A screenshot of a computer

Description automatically generated

Figure 36: Username Exists Message

A screenshot of a computer

Description automatically generated

Figure 37: Account Created Successfully Message

For each of those buttons, there is validation and feedback for the admin. If the admin wants to delete an account, the admin will be asked for confirmation first. If the admin wants to update a username, the system will check whether that username exists yet or not. If the admin creates a new account, a message confirming the account creation will appear.

A screenshot of a medical appointment

Description automatically generated

Figure 38: Today's Appointment Page

A list of all appointments for the day can be viewed by clicking the “Appointment” button, which will show a list of all the appointments in a table. By clicking on the rows, the admin can see the details in a more readable format. The data displayed here are the doctor’s name, PatientID, time and additional details.

A screenshot of a computer

Description automatically generated

Figure 39: Walk in Appointment Page

A screen shot of a computer

Description automatically generated

Figure 40: Fields not filled in Error

This is the Queue interface is a receptionist-like function where the admin can view all the current appointments for the day and also add to it. The admin has to also set the severity of the cases ranging from minor to extreme. An error will be thrown if the fields are not filled in.

A screenshot of a medical record

Description automatically generated

Figure 41: Patient's Medical Record Page

Another one of the more receptionist-like functions for the admin is to view the current medical record of the patients who require treatment, in order to understand more about the patient. A table will display the records in rows which will also display the information in a more readable format once clicked.

A screenshot of a computer

Description automatically generated

Figure 42: Collect Payment Page

The collect payment feature is another receptionist-like function which is just a simple page to verify that payment has been collected for any medication issued.

A screenshot of a computer screen

Description automatically generated

Figure 43: Income Report (Additional Function) Page

The income report shows the statistics for the month, including income and number of patients. The admin can then input the expenses for the month and then click on the “Calculate” button which then displays the estimated total profit.

# Sample Input Output and Explanation.

## File Handler (Create Walk in Appointment)

A screenshot of a computer

Description automatically generated

Figure 44: Walk in Appointment GUI

A computer screen shot of colorful text

Description automatically generated

Figure 45: Add Patient to Queue Code

The addToQueue method handles the logic for adding patient’s information to a queue file hence we will use a private modifier to ensure the method will only be accessible by the same class or type of user which in this case is the Administrator. The method uses a try catch statement for easier maintenance in the future. Inside the block, the code starts with acquiring the text from username, name text field, and severity combo box. An If statement is also used here to ensure the text field is filled to avoid empty entry. Once it passed the if validation, filewriter will open and write into PatientQueue.txt file which includes the patient’s username, name, severity, current Time and Status. The current Time is a function from the Java.time package. It closes the file once it is finished writing to prevent any resource leak from occurring. A message dialog will pop up to inform the user the process is completed.

## File Handler (Import User Data from Text File)

A screenshot of a computer

Description automatically generated

Figure 46: Doctor Homepage



Figure 47: Queue List Method

This method will be called alongside with the other Jframe components to ensure the table is displayed to user as soon as possible. It creates a table after reading through the patient queue file for the doctor to press on and minimize the need of typing to ensure the workflow goes as smooth as possible.

## File Handler (Update User Details)

A screenshot of a computer

Description automatically generated

Figure 48: Doctor Edit Appointment

A screenshot of a computer

Description automatically generated

Figure 49: Patient's Appointment History Text File after Update

A screen shot of a computer program

Description automatically generated

Figure 50: Update Appointment Information Code

This method starts with fetching text from a variety of text fields and storing them into a variable. Validation is also applied to ensure all the necessary fields are filled up before running the method. After that, it checks for the file directory and creates an Appointment folder if it doesn’t exist as well as the file for the corresponding user. Once it has located the file, it writes the information into the user’s appointment file and deletes it from the doctor’s appointment file. This is being done to ensure there are no redundancies.

## File Handler (Remove Data from Text File)

A screenshot of a computer

Description automatically generated

Figure 51: Administrator Edit User Page

A screen shot of a computer screen

Description automatically generated

Figure 52: Delete User Method Source Code

DeleteUser method has a private modifier to prevent any malicious party from tampering with the database which can lead to irreversible damage. It starts with a validation to ensure a valid row is chosen on the table. Then it will prompt the user for a confirmation before running the rest of the method. Once the user has confirmed, the reader will look for the information of the selected user in credentials.txt. It uses the username to identify the user because it is a primary key in the text file. If the reader can find the username it will return a true and allow the writer to overwrite the selected information with empty string which is equivalent to deleting the user.

## ArrayList and ArrayDeque

A screen shot of a computer program

Description automatically generated

Figure 53: ImportData method from EditUser.java

A screenshot of a phone

Description automatically generated

Figure 54: Credentials.txt Data Structure

Due to our unique way of storing our information in the figure above, we will need to use ArrayDeque to store our information before populating our table. In our code we start by searching for the roles which then append the 4 lines before the line which includes information of the user into the ArrayList we defined. A while loop is used to ensure every line is checked and no users are left out. By using the ArrayDeque we can ensure all the old user will be at the top and the new users will be at the bottom. This allows us to have a table that correctly represents our database.

# Object Oriented Programming Concept with Sample Code

## Encapsulation

A screen shot of a computer

Description automatically generated

Figure 55: Store Data Method Source Code

A screenshot of a computer program

Description automatically generated

Figure 56: Clear Payment Method Source Code

Encapsulation in Java is defined as compiling all functions and variables which operate on certain data into a single unit or class, thus restricting access to the object’s components. For our system, one of the implementations of encapsulation is Clear Payment method which is a method used by the StoreData method to clear out the pending payment once the administrator accepts payment from the patients. It is crucial to us a “private” modifier for the method as the program will be handed out to users and they can make modifications without the developers ever knowing. This can prevent users from altering the payment logic and ultimately consulting without paying. By making the method private, this ensures the method will only be accessible through the correct interfaces.

## Inheritance, Polymorphism and Interface Implementation

A screen shot of a computer program

Description automatically generated

Figure 57: AdminHomepage.java

Inheritance in Java is defined as allowing a class to inherit the properties and behaviour of an existing parent class. This allows for cleaner and more reusable code, since the number of lines will be cut down and the developer can re-call the required classes instead of defined an entirely new class. The AdminHomepage class is one of the examples in our system, since it inherits from BaseFrame, making AdminHomepage a subclass of BaseFrame. This inheritance is used to reduce redundancy and allow developers to set the size and position of the windows consistently. The super(false) call in the AdminHomepage constructor specifies whether the window should use DISPOSE\_ON\_CLOSE or EXIT\_ON\_CLOSE as its close operation, as shown in Figure 33.

In addition to inheritance, method overriding is also applied here. The AdminHomepage class overrides the initialize() method from its parent class BaseFrame. This allows the code to display unique content for the admin homepage by providing a specific implementation of the initialize() method in the AdminHomepage class.

## Abstraction

A screen shot of a computer screen

Description automatically generated

Figure 58: BaseFrame.java

Abstraction in Java is defined by the simplification of code by breaking code down into smaller, more manageable components. Similar to inheritance, it allows for more re-usable code thus allowing the system to be scalable. BaseFrame.java is an abstract class that takes a Boolean argument which defines the windows’ setDefaultCloseOperation. BaseFrame class also setBounds for the windows which help reduce redundancy and ensure all the windows are the same size and position. Initialize method is also declared to ensure every subclass provides its own implementation of initialize method.

## Singleton Pattern

A screen shot of a computer screen

Description automatically generated

Figure 59: UserSession.java

A Singleton Pattern is a concept in OOP which states that a class should only have one instance and provide a global access point to that instance. In our system, we decided to implement it in our public UserSession class. This pattern allows the class only to have 1 instance and provides a global access point to the content. In this context, it allows us to store username and call the username at any point of time which is much better than passing username as an argument in every method.

## Setter

A computer screen shot of a program code

Description automatically generated

Figure 60: Username Setter from LoginForm.java

A setter in Java is what its name is; it sets the value of a field in a controlled manner and with validation. This is a code snippet from LoginForm.java where once we find the user and its corresponding roles, we will set the username using the setUsername before redirecting user to their respective user interface. This avoided the need to pass “usernameInput” as an argument which helps keep the code neat.

## Getter

A screen shot of a computer

Description automatically generated

Figure 61: Username Getter from ChangePass.java

A in Java getter is also what its name is; it allows other parts of the program to update the value of a field without directly accessing it. This is a code snippet from ChangePass.java where we need to use the username to search for the record in the text file. We use the getUsername and getPassword to get our username and Password that is stored globally instantly. By using a getter, we can reduce redundancy and the need to pass username as argument into the ChangePass then use it in the method. We also avoided the need to use the username to find for old password in the text file which can be simplified by comparing directly with our UserSession.

## Single Responsibility Principle (SRP)

A computer screen shot of a program code

Description automatically generated

Figure 62: CheckUsername method from EditUser.java

The Single Responsibility Principle coined by Robert C. Martin and is one of the 5 SOLID principles of Object-Oriented Programming, and states that a class should only have one reason to changes, which means that it only has one job (Thorben, 2024).This means that if there are any changes to a class should happen for only one reason, thus reducing the workload to each class if future developers would like to make any major changes. An example implementation in our system is the private CheckUsername method that checks for a username existence when trying to add a new user. It uses a BufferedReader to read our text file which is “credentials.txt” and if it found any lines that contains the same username user trying to add, it will return true and vice versa. This method aligns with the Single Responsibility Principle of only handling a single task in this case, which is verifying the presence of a username within a file. If we were to change the text file name for example, we would have to change only one instance of it, and no other functions would be disturbed by this change.

# Additional Functions

## Quick Access Table

A screenshot of a computer

Description automatically generated

Figure 63: Quick Access Table from Doctor Home Page

A computer screen shot of a program code

Description automatically generated

Figure 64: Doctor Home Page Quick Access Table Source Code

In order to make the consultation process as smooth as possible, we implemented a quick access table for the doctor to press on instead of manually typing the patient’s username. This allows doctors to keep track of the sequence of the patients and reduce the burden by removing the need to type. This table is implemented through using the table model and action event. This is the main example, however, this concept of a table where you can click on a specific row and all required fields will be filled in based on the is implemented throughout the system. For example, in the admin Edit User Page to select a user and also the patient Make Appointment Page to select time slot.

## Report Function

A screenshot of a computer screen

Description automatically generated

Figure 65: View Report and Make Estimation

A computer screen shot of a program code

Description automatically generated

Figure 66: Calculate Profit Function

The current project requirement felt incomplete despite having the function of collecting payment from patients, but it does not allow administration to check the report. Hence, we implemented a simple income report for the administration, so that they can view the total income, number of patients as well input their monthly expense to be able to roughly estimate whether or not they are making profits.

# Conclusion

To sum it all up, St Denis Clinic will now be more efficient and effective in their day-to-day activities, thus allowing them to serve their community better and be more profitable to ensure their prosperity. This is something which cannot be solved by medical skill, if a clinic has the best doctor in the world, but is limited by a subpar clinic management system, that clinic will be unable to perform as well as a clinic with an all-encompassing and user-friendly system. This shows how important Software Engineering (SE) is in a world which is already in an IT dominated era and is still marching forward into more technological wonders, and how versatile it is since the skills in the SE field can be applied into any other industry.

The assumptions which can be made for this system are that the clinic owns a computer with Windows OS and can run our program. All dependencies or packages would also have to be set up correctly in order to avoid any errors or unwanted outputs. Similarly, all users would also need some degree of digital literacy so that they understand what values to input, and how to navigate a digital system. To ensure that the user data is secure, it would be assumed that the device which contains the Java program is encrypted securely, and never left unattended. All assumptions apply to doctors or users who may want to have the program on their own device.

Some limitations that we face are the difference in development environment which may cause errors whenever a new commit is pushed through GitHub. Even though each developer is using the exact same JDK and code editor, the program may not run as intended. This has caused the environment setup process to be extensive. Furthermore, since all data is currently only stored locally on each device in text files, users cannot use the program wherever and whenever they want. Finally, this application is limited to Windows devices only, which means that most mobile devices will not be supported, thus limiting the convenience.

With these limitations comes the need for improvements, which we predict will be easy due to the many object-oriented programming (OOP) concepts implemented. One of the first improvements we can make is to probably collect more information regarding the user, such as their occupation, age, and marital status, so that we can know more about our patients. Secondly, we have to make sure that an online database is implemented, so that data is updated in real-time. In line with this, we can make a mobile friendly version, and also make it easier for users to download and set up. With the availability of a database, we can also integrate a notification system to remind users about the upcoming appointments. Finally, if required, we can include a chat feature, or at least integrate an emergency hotline system, to allow for better communication between the clinic staff and any patients which might require their assistance urgently.

# References

Chuan, C. H. (2021). *Java Programming Tutorial Programming Graphical User Interface (GUI)*. Retrieved from NTU Singapore: https://www3.ntu.edu.sg/home/ehchua/programming/java/J4a\_GUI.html

Javatpoint. (2021). *Java JTable*. Retrieved from javatpoint: https://www.javatpoint.com/java-jtable

Jon. (2010). *How to close a jframe without closing the main program*. Retrieved from stackoverflow: https://stackoverflow.com/questions/4268749/how-to-close-a-jframe-without-closing-the-main-program

Oracle. (n.d). *Abstract Methods and Classes*. Retrieved from Oracle Java Documentation: https://docs.oracle.com/javase%2Ftutorial%2F/java/IandI/abstract.html

Pankaj. (2022). *Java read text file*. Retrieved from DigitalOcean: https://www.digitalocean.com/community/tutorials/java-read-text-file

Programiz. (n.d). *Java Abstract Class and Abstract Methods*. Retrieved from Programiz: https://www.programiz.com/java-programming/abstract-classes-methods

Refactoring Guru. (2024). *Singleton*. Retrieved from Refactoring Guru: https://refactoring.guru/design-patterns/singleton

Risha. (n.d). *Object Oriented Programming (OOPs) Concept in Java*. Retrieved from geeksforgeeks: https://www.geeksforgeeks.org/object-oriented-programming-oops-concept-in-java/

Thorben. (2024, February 28). *SOLID Design Principles Explained: The Single Responsibility Principle*. Retrieved from Stackify: https://stackify.com/solid-design-principles/

Visual Studio Code. (2022). *Working with GUI applications in VS Code*. Retrieved from Visual Studio Code: https://code.visualstudio.com/docs/java/java-gui

W3Schools. (n.d). *Java Read Files*. Retrieved from W3Schools: https://www.w3schools.com/java/java\_files\_read.asp