CS 152 Final Project Report
Release: The Journaling App

Alan Viollier

December 1st, 2022
CS 152 Section 5
Programming Paradigms

Contents

Introduction

For this project, I was to create anything I wanted in my CS 152 class as long as it followed a programming paradigm. Programming paradigms are the topic because this class is about learning and exploring the different programming paradigms. I will go more into depth on what a programming paradigm is and which one I decided to use later in the report.

With the possibility of “anything I want,” I had a broad spectrum of what I could do. However, it was challenging to think of something specific in a wide range. In the end, I decided to choose journaling and symptom tracking, which I’ve been exploring recently.

Mental health has been considerably declining due to the surprise of Covid-19 that happened almost three years ago. This decline has forced people, myself included, to find ways to care for their mental health in tough times.

This struggle is where journaling comes in. Journaling is just writing down or typing anything that goes through your mind, whether it’s just your thoughts as is, symptoms you may be worried about, things you have to do, something you want to improve, or things that are stressing you out. Journaling allows you to release all the thoughts bearing your mind down, be aware of them, process them, and take a step away from them. It will enable you to take the weight of your thoughts out of your mind. Since I started journaling, I have felt much better about myself, my thoughts, my emotions, and the things I must do. Journaling is a simple, low-cost way of improving mental health.

Through this semester, I designed and developed a GUI-based application that I named “Release” that allows you to journal, view & modify your journal, track symptoms, and view a history of your symptoms.

Programming Paradigms

First, a paradigm is a strategy for solving a problem or doing a specific task. Therefore, a **programming paradigm solves a problem using specific programming languages, tools, and techniques**.

There are many programming languages out there, but all of them fall into the category of a particular paradigm. The main programming paradigms are **Imperative**, **Object Oriented**, **Functional**, and **Logic**.

The **Imperative** paradigm follows the von Neumann-Eckert model, where program and data are indistinguishable in memory. A program is a sequence of commands; the state is the value of all variables when the program runs. More extensive programs use procedural abstraction—popular programming languages such as Cobol, Fortran, C, Ada, and Perl.

The **Object-Oriented** paradigm uses a collection of objects interacting with each other by passing messages that transform the state. Object Oriented paradigms can make programming easier as inheritance, polymorphism, and encapsulation make things clear and straightforward. Popular programming languages that use this paradigm are Smalltalk, Java, C#, and Python.

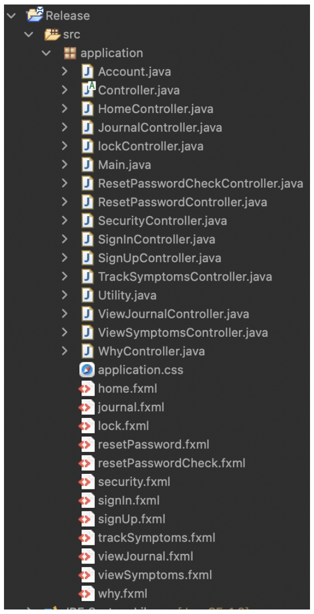
The **Functional** paradigm models a computation as a group of mathematical functions; functional composition, recursion, no variable assignments, and no state changes characterize it. Popular programming languages that use this paradigm are Lisp, Scheme, ML, and Haskell.

Finally, the **Logic** paradigm declares what an outcome of a program should be rather than how the program should accomplish it. As a result, we usually see programs that achieve all possible solutions, that are nondeterministic, and that are a set of constraints on a problem. Popular programming languages that use this paradigm are Prolog and CLP.

I chose **Java** to develop my GUI-based journaling application. Java is a multi-paradigm language that uses Object Oriented, Functional, and Imperative. However, Java is primarily Object-Oriented, making designing and developing classes for a GUI program easier.

The Project

Note: I have submitted the project file system and a runnable Jar file. Although the runnable Jar file works and can read from text files, it cannot write to text files within the Jar file and will spew text files when saving something. However, writing and Reading to correct files from the file system works properly.



```
Release
  src
    application
      Account.java
      Controller.java
      HomeController.java
      JournalController.java
      lockController.java
      Main.java
      ResetPasswordCheckController.java
      ResetPasswordController.java
      SecurityController.java
      SignInController.java
      SignUpController.java
      TrackSymptomsController.java
      Utility.java
      ViewJournalController.java
      ViewSymptomsController.java
      WhyController.java
      application.css
      home.fxml
      journal.fxml
      lock.fxml
      resetPassword.fxml
      resetPasswordCheck.fxml
      security.fxml
      signIn.fxml
      signUp.fxml
      trackSymptoms.fxml
      viewJournal.fxml
      viewSymptoms.fxml
      why.fxml
```

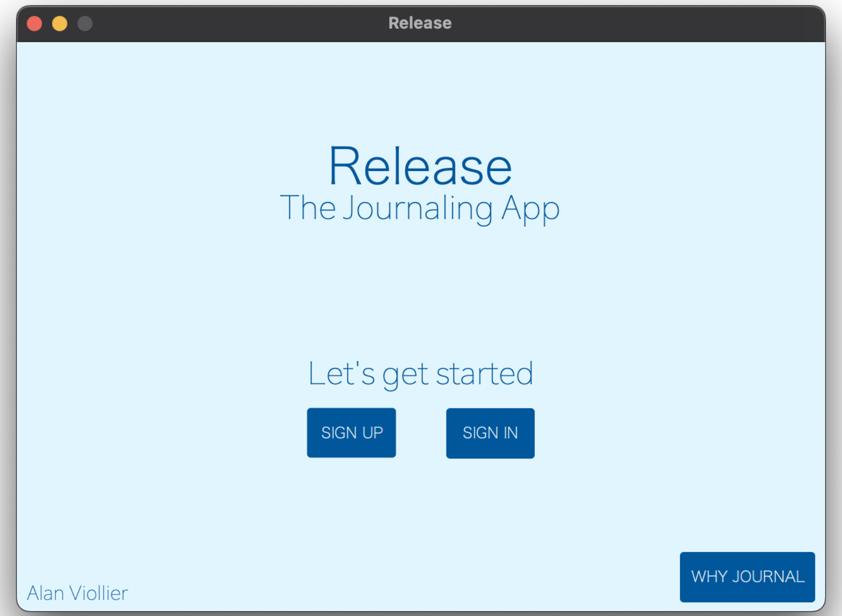
The project contains **twelve** pages in JavaFx format, each with its own controller class inherited from the parent Controller class. The utility class initializes in the Controller class that does all the data reading and storing. Next, the Account class stores account information and a logged-in user. Finally, the Main class starts the application, and four text files store different types of data.

There is a test account with the following:
Username: Mike
Password: password123

Let's take a look at all the pages:

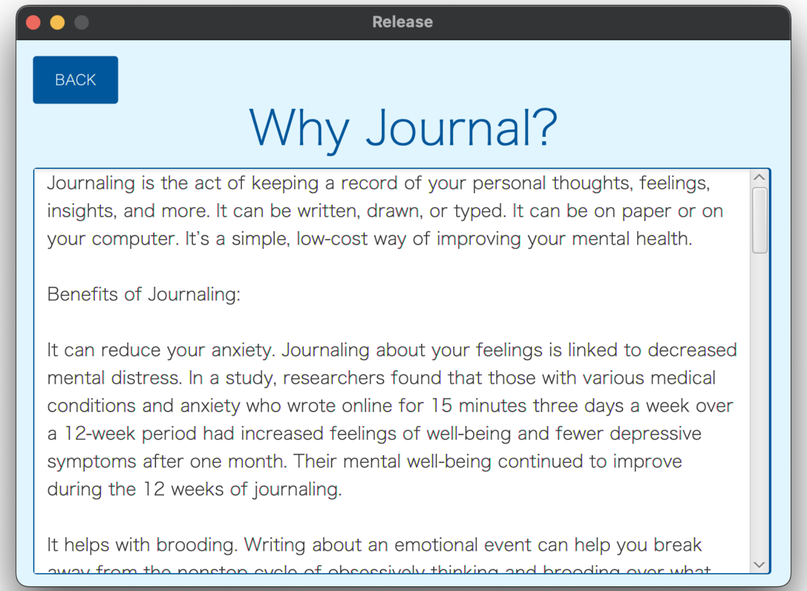
1. Lock Page

The Lock page presents the app's name, my name, and allows the user to sign-in, sign-up, or read about why they should journal.



2. Why Journal Page

The Why Journal page describes why and how the user can benefit from journaling.



Why Journal?

Journaling is the act of keeping a record of your personal thoughts, feelings, insights, and more. It can be written, drawn, or typed. It can be on paper or on your computer. It's a simple, low-cost way of improving your mental health.

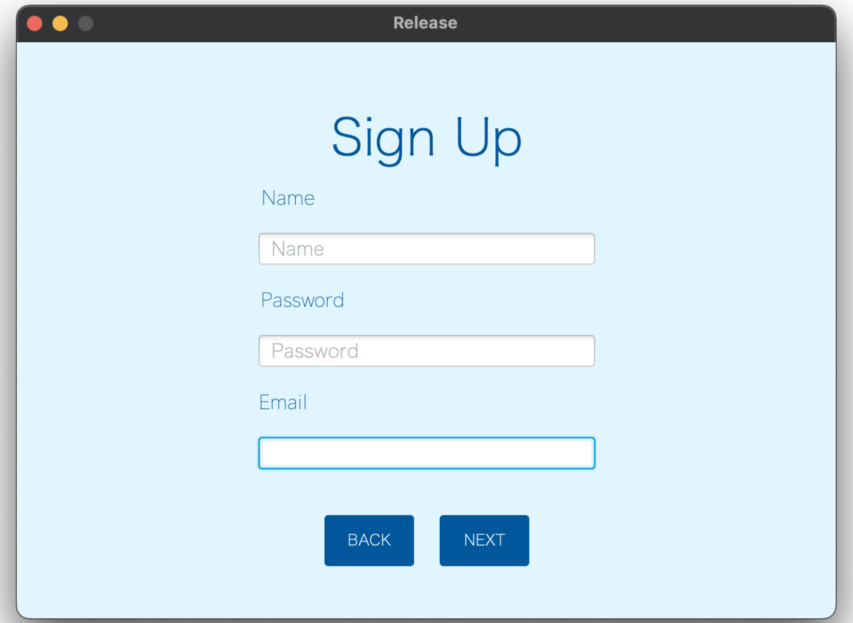
Benefits of Journaling:

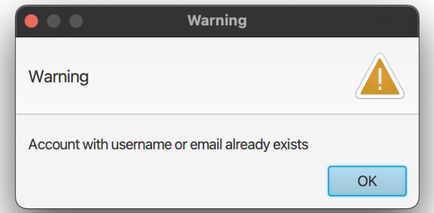
It can reduce your anxiety. Journaling about your feelings is linked to decreased mental distress. In a study, researchers found that those with various medical conditions and anxiety who wrote online for 15 minutes three days a week over a 12-week period had increased feelings of well-being and fewer depressive symptoms after one month. Their mental well-being continued to improve during the 12 weeks of journaling.

It helps with brooding. Writing about an emotional event can help you break

3. Sign Up Page

The Sign-Up page allows users to sign up for a new account with their name, password, and email. It will not allow empty fields, commas (for data storage purposes), or a name or email already used. Once the "Next" button is pressed, the sign-up controller will send the information to the Utility class, where it will be saved temporarily for the next page. The back button bring users back to the lock screen.

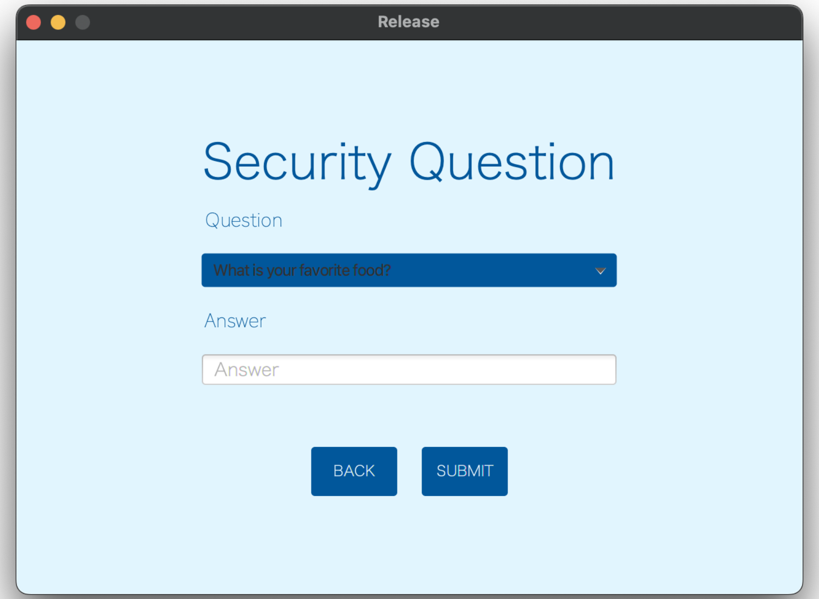


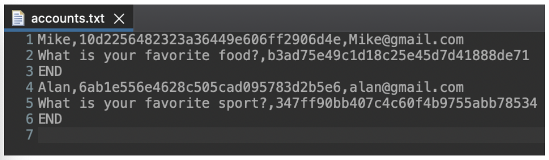


Account with username or email already exists

4. Security Page

The security page allows the user to select a prompt question and answer it for password resetting. Once the “Submit” button is pressed, the question and answer are added with temporary information to create a new account that is saved to the “accounts.txt” file. The password and answer to the security question are encrypted on the file. The user is also sent to the “Sign In” page. The “Back” button brings users back to the lock screen.

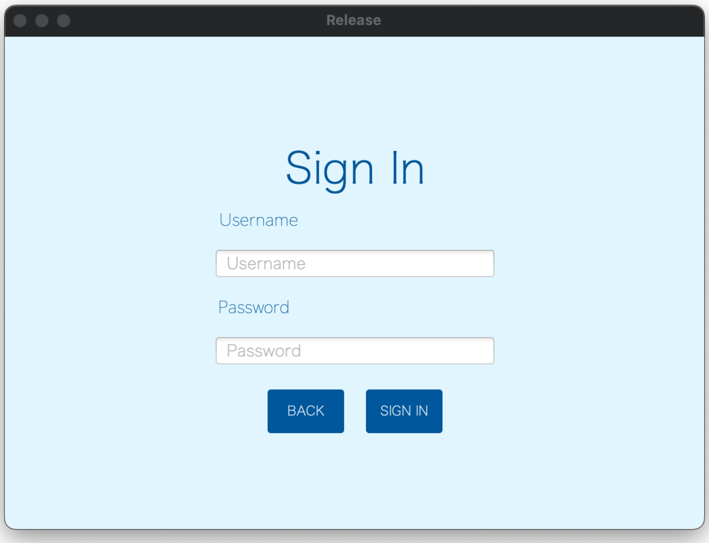




```
Mike,10d2256482323a36449e606ff2906d4e,Mike@gmail.com
What is your favorite food?,b3ad75e49c1d18c25e45d7d41888de71
END
Alan,6ab1e556e4628c505cad095783d2b5e6,alan@gmail.com
What is your favorite sport?,347ff90bb407c4c60f4b9755abb78534
END
```

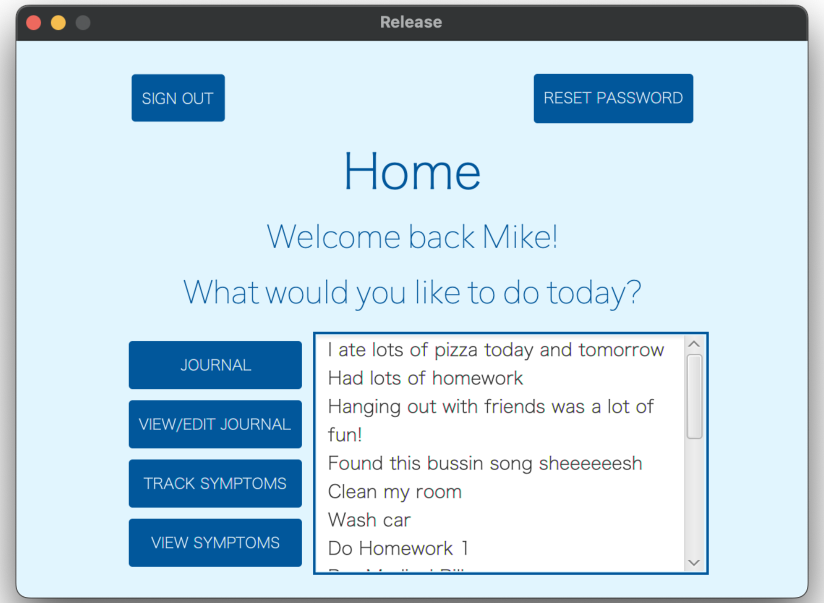
5. Sign In Page

This sign-in page takes a username and password. When the “Sign In” button is pressed, the Utility class checks whether the name and password are in the “accounts.txt” file. It has to decrypt the password to check them. If the account exists, the user is signed into the utility class and moves to the home page; otherwise, they get an alert saying the account is invalid. The “back” button brings users back to the lock page.



6. Home Page

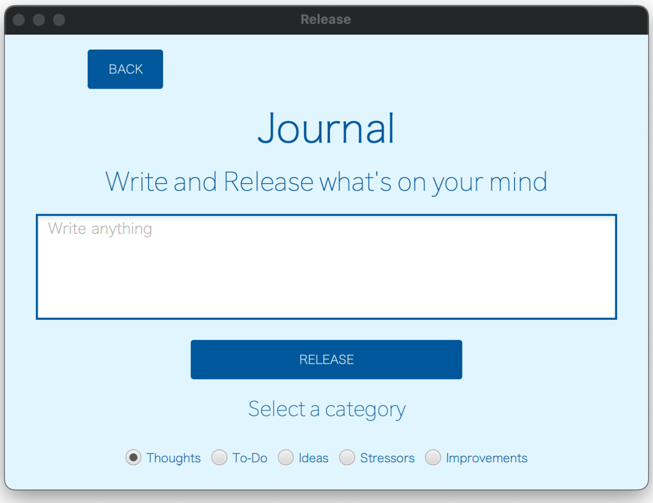
The home page is a central hub for all of the features this application provides. This page contains a personalized welcome message and a preview of your journal. The preview is pulled from the “journalData.txt” file. The “Journal”, “View/Edit Journal”, ”Track Symptoms”, “Reset Password”, and “View Symptoms” buttons all send users to their respective pages.



I ate lots of pizza today and tomorrow
Had lots of homework
Hanging out with friends was a lot of fun!
Found this bussin song sheeeeeesh
Clean my room
Wash car
Do Homework 1

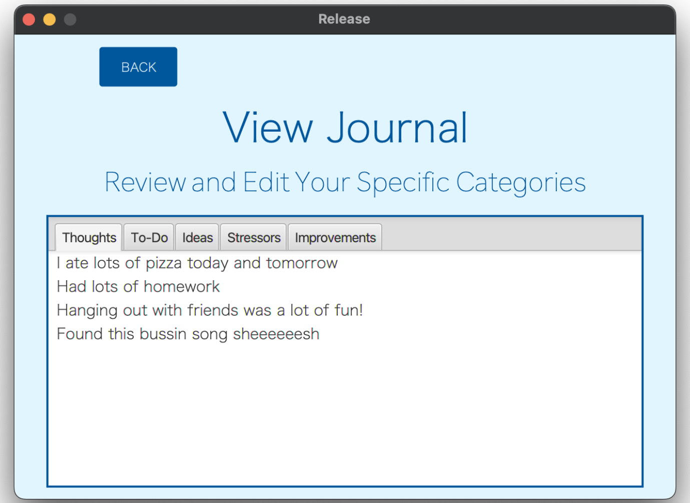
7. Journal Page

The Journaling page allows users to write and release anything on their mind and categorize the writing into five different categories. When they press “Enter” while typing or “Release,” the text will be sent to the Utility class to be encrypted and stored in the “journalData.txt” file. The TextArea cannot be empty when releasing or an alert will pop up. The “Back” button sends users back to the home page.



8. View Journal Page

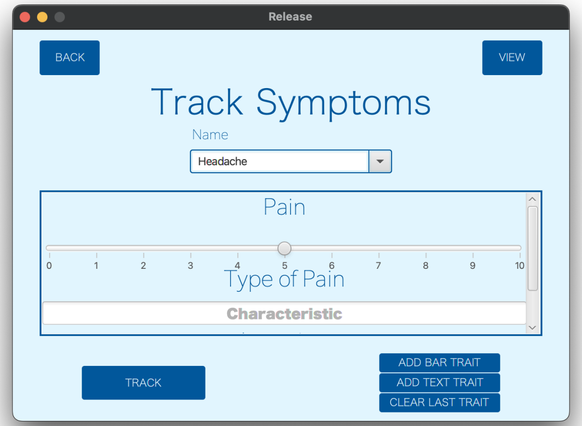
The View Journal page on initialization retrieves all of the journal data from the “journalData.txt” file and puts it all on categorized tabs based on what the user selected before. This page also allows users to edit the journal as they please. When the “Back” button is pressed, the data is resent to the Utility class, which replaces the old data on the “journalData.txt” file with the new data. The user is then sent back to the home page.



I ate lots of pizza today and tomorrow
Had lots of homework
Hanging out with friends was a lot of fun!
Found this bussin song sheeeeeesh

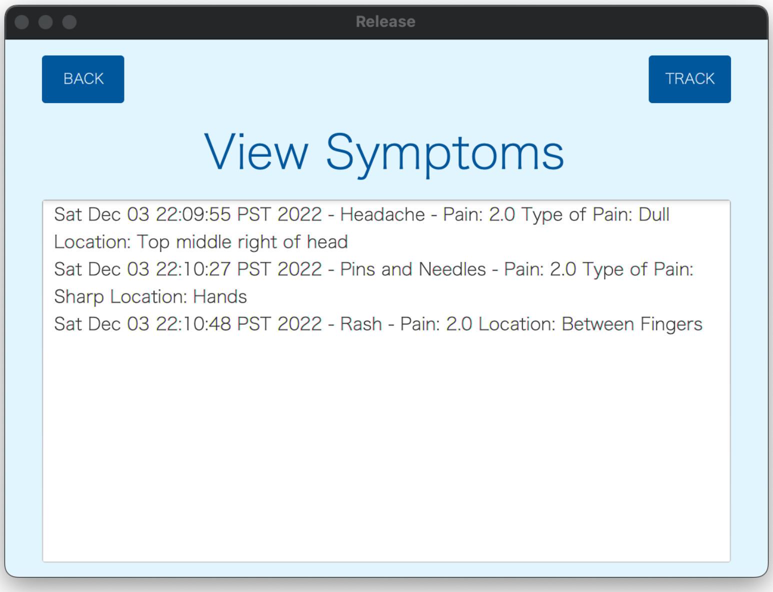
9. Tack Symptoms Page

The track symptoms page allows users to track symptoms by name and add any amount of attributes to be stored as well. On initialization, the previous symptoms the user has experienced are pulled from “userSymptoms.txt” and added to the combo box. When a previous symptom is selected on the combo box, it will show all the attributes they used before for that symptom. When “Track” is pressed, the data of the symptom is sent encrypted to “symptomData.txt”, while the symptom attributes specific to that user are sent encrypted to “userSymptoms.txt”. The buttons at the bottom right allow users to add attributes and remove them. The “Back” and “View” buttons bring users to the home page and view symptoms page, respectively.



10. View Symptoms Page

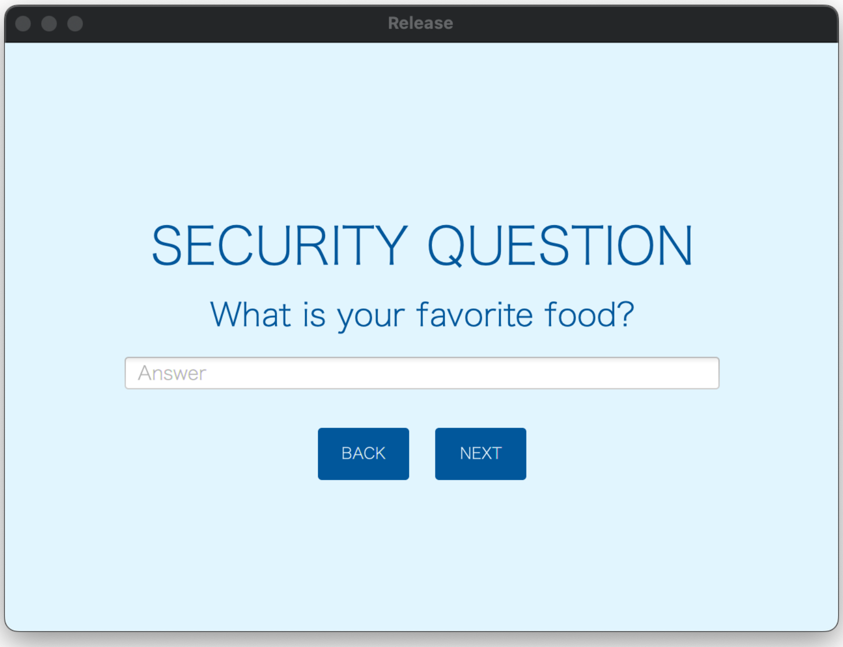
This page pulls the user's symptoms on initialization from "symptomData.txt" and shows it to the user. The "Back" and "Track" buttons bring users to the home page and track symptoms page, respectively.



Sat Dec 03 22:09:55 PST 2022 - Headache - Pain: 2.0 Type of Pain: Dull Location: Top middle right of head
Sat Dec 03 22:10:27 PST 2022 - Pins and Needles - Pain: 2.0 Type of Pain: Sharp Location: Hands
Sat Dec 03 22:10:48 PST 2022 - Rash - Pain: 2.0 Location: Between Fingers

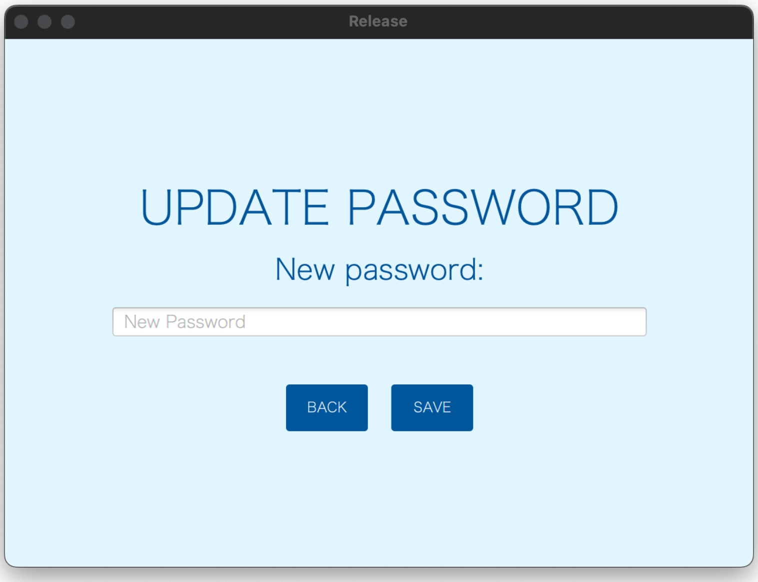
11. Reset Password Check Page

This page asks you the security question you chose and asks you for the answer. If correct, on "Next" the user is sent to the reset password page. The "Back" button goes back to the home page.



12. Reset Password Page

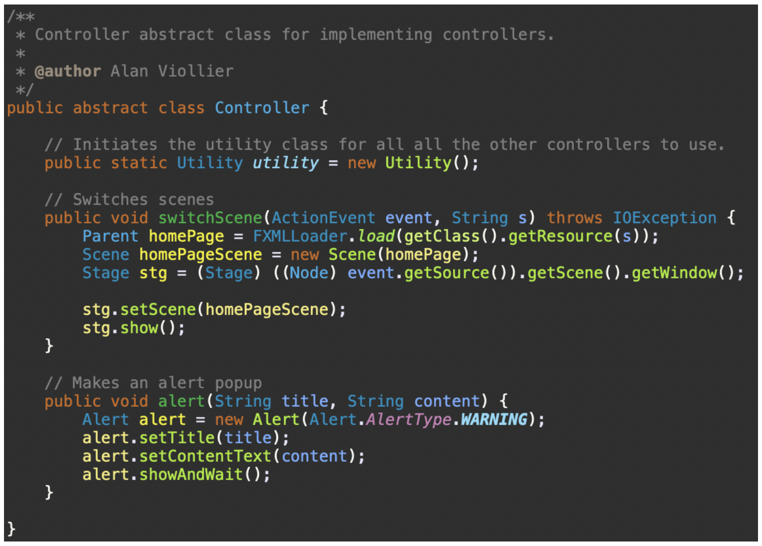
This page takes an input and, on “Next,” resets the password for the user in the “accounts.txt” file. The “Back” button brings the user back to the home page.



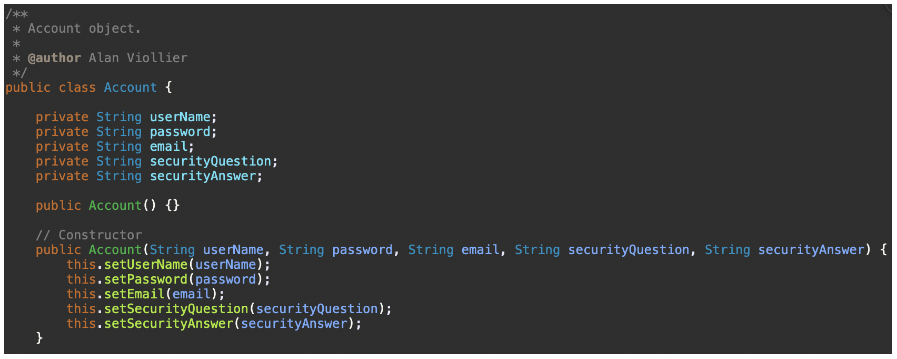
Object-Oriented Paradigm

In this project, I used the techniques of object-oriented paradigm through java. Here are some examples in my code:

Here we can see the Controller class is an abstract class. An abstract class is a template definition of the methods and variables of a class. Meaning a class can inherit this class and can use these methods as part of their class. This object-oriented paradigm is doing well, as I do not have to rewrite these methods every time I want to create a new controller for a page.



```java
/**
 * Controller abstract class for implementing controllers.
 *
 * @author Alan Viollier
 */
public abstract class Controller {

    // Initiates the utility class for all all the other controllers to use.
    public static Utility utility = new Utility();

    // Switches scenes
    public void switchScene(ActionEvent event, String s) throws IOException {
        Parent homePage = FXMLLoader.load(getClass().getResource(s));
        Scene homePageScene = new Scene(homePage);
        Stage stg = (Stage) ((Node) event.getSource()).getScene().getWindow();

        stg.setScene(homePageScene);
        stg.show();
    }

    // Makes an alert popup
    public void alert(String title, String content) {
        Alert alert = new Alert(Alert.AlertType.WARNING);
        alert.setTitle(title);
        alert.setContentText(content);
        alert.showAndWait();
    }

}
```



```java
/**
 * Account object.
 *
 * @author Alan Viollier
 */
public class Account {

    private String userName;
    private String password;
    private String email;
    private String securityQuestion;
    private String securityAnswer;

    public Account() {}

    // Constructor
    public Account(String userName, String password, String email, String securityQuestion, String securityAnswer) {
        this.setUserName(userName);
        this.setPassword(password);
        this.setEmail(email);
        this.setSecurityQuestion(securityQuestion);
        this.setSecurityAnswer(securityAnswer);
    }
```

Here we can see the account class, which can be used to make specific account objects which can hold different states. This is another core feature of object-oriented design. You have a blueprint of certain things and can create as many objects of that thing, each with its own state. All I need is the code of this account class, and I can have as many accounts as I want.

Through the object-oriented paradigm, I could write a collection of classes that create objects that can communicate with each other. There is more emphasis on the data rather than the procedure. Object-oriented programming allows my code to be more modular, which makes it easier to reuse and troubleshoot. I can use super and subclasses with inheritance to pass variables and methods to more specific versions of that class and again reuse code and make things clearer. Although not used in this project, polymorphism allows flexibility in function arguments, and a subclass can pass as its superclass. Overall the object-oriented paradigm was a great choice for this project as it made it simpler, more efficient, and reusable.

Conclusion

In conclusion, through the use of the object-oriented paradigm in java, I was able to make a fully functional journaling and symptom-tracking application. This application can help those who need to improve their mental health do so through the benefits of journaling.

Through this project, I learned more about the different types of programming paradigms. I refined my knowledge of object-oriented programming. I improved my Java language skills.

Some things I would like to improve in the future for the journaling application is to make it mobile. I don’t think many people would use a computer-based journaling app as it may take too long to open up the computer and start journaling. A mobile-based application would be a lot faster and easier to use. I would also use hashed passwords next time, as encrypting a password has to have a key or a way that can be cracked. I was initially going to use JavaScript, but as the difficult semester progressed, I chose Java as I knew the libraries. I think JavaScript is a better laguage for modern-day applications, and I would learn it and use it next time.

Thank you for checking out my CS 152 final project!