Mark Eilers
CS 499: Computer Science Capstone
4-2 Milestone Two: Enhancement Two: Algorithms and Data Structure
July 26, 2024

4-2 Milestone Two: Enhancement Two: Algorithms and Data Structure

For the second section of the ePortfolio, Algorithms and Data Structure, I have chosen to modify my zoo authentication application from my IT 145 Foundation in Application Development Course. The original artifact was created in September of 2022. In this updated version of the original assignment, I created a Java-based authentication system that was comprised of three main classes. There was the 'UserAuth' class, 'AuthHandler' class, and 'EncryptedCredentials' class. For this project, I basically started from the ground up to enhance the security measures of the assignment. The updated artifact demonstrates a basic implementation of user authentication by validating usernames and passwords, encrypting passwords using MD5 hashing, and providing role-based greetings to users upon successful login. This application was created using IntelliJ integrated development environment (IDE) specifically using Java as the coding language.

I selected this artifact for my ePortfolio because it exemplifies my skills in algorithms and data structures, particularly in the context of user authentication and password encryption. This project showcases several key components. For the algorithm portion of the project, the use of the MD5 hashing algorithm to securely encrypt user passwords demonstrates my understanding of cryptographic functions and their implementation in Java. For data structures, the use of HashMaps to store user credentials highlights my ability to effectively use data structures for efficient data retrieval and management. For the remaining portion of this assignment, I will be referring to the screenshots at the end of this document. Each screenshot is

labeled as a figure and will be referred to as such. Additionally, each screenshot provides a short description to identify what the image is showing.

For this assignment, I essentially started from the ground up. The original project lacked major components, and I wanted to create a more secure system. As a result, I created an entirely new authentication system. To start this enhancement, I began with an authorization user class. In ***Figure 1*** I imported the necessary classes for reading input and handling collections. I also initialized the public variable for predefined user access. Building on this, ***Figure 2*** shows the creation of the user credentials. This includes creating the username and password. Finally, in ***Figure 3*** a while loop statement was created to check user credentials. In this section of code, users are given a prompt to enter in their username and password for the system. If the credentials are valid, then users will be granted access to the program. If the credentials are wrong, the program increments the attempted count while informing users that their username and password do not match. Users are then prompted again. If wrong again, attempted count is incremented continuing the loop. Once the attempted count is no longer less than the max attempts, then users are presented with a maximum attempt statement asking them to try again later. The combination of ***Figures 1 – Figure 3*** showcase the creation and implementation of the user authentication class.

Once the user authentication class was completed, I then worked on the next class, AuthHandler. In this class credentials were checked. Once the credentials were correctly matched, then a welcoming statement as well as a defined role was presented to the user. If the credentials were not a match, then the user was informed of their failed login attempt. In ***Figure 4*** a method was created to validate the user's credentials. The use of HashMaps was implemented in this section to showcase my use of data structures. In ***Figure 5 – Figure 6*** the

user credentials were validated using switch and case statements. In each case, the username and password are validated for a match. Once a match is found the program displays a statement welcoming the user and informing them of their role within the computer system. If the credentials were not a match, then the user was also informed of their failed attempt.

In the final set of three screenshots, three additional classes were created to round out the final project. In ***Figure 7*** the credential class was created. This contained the setters and getters for the username, password, and encryption portions of the program. Finally, in ***Figures 8 – Figures 9*** code was written to transform the user's password into an encrypted password. This was done using MD5 Hashing to encrypt the passwords, adding an additional layer of security to the overall project. Overall, this project is a major improvement to the original artifact. Code was sectioned into different classes that would make it easier to maintain and scale for future use. Additionally, each class has appropriate comments and proper naming standards. This makes the code easier to manage, work with, scale, update, and overall use in a professional setting.

With the enhancements made to this artifact, I have successfully met the course outcomes I planned to achieve in Module One. These include demonstrating the ability to implement and improve algorithms and data structures in a real-world application. I currently have no updates to my outcome-coverage plans as the enhancement algins with my initial goals. Enhancing and modifying this artifact was a valuable learning experience. During this process, I learned several important lessons. These include the importance of code readability, user experience, and exception handling.

While I would consider this project to be a success. I did run into some issues. Namely, one of the biggest issues to start was finding the IDE that worked for me. I initially started this work using eclipse. However, I quickly found that to be tedious and problematic for me. While I

can appreciate the features that eclipse offers, it is not a program that works best for me. As a result, I switched over to IntelliJ IDEA. Once I made the switch, I found that creating the project was easier to manage. This was due to the features that IntelliJ offers, and admittedly the overall look and feel of the IDE. Another issue I ran into was figuring out exactly how I wanted to plan this project. This artifact requires a lot of different classes. As a result, a lot of working parts are running to make the program function. I needed to find a way to plan out this project to effectively manage my time to meet the deadline. I found that pseudocode was beneficial. It allowed me to create an overall plan that I could work off. Pseudocode gave me the chance to get everything mapped out, so that I was not updating and changing major portions of already completed code. As a result, it allowed me to effectively manage my time. Overall, this process has enhanced my skills in writing clean, maintainable code and reinforced the importance of considering both functionality and user experience in software development.

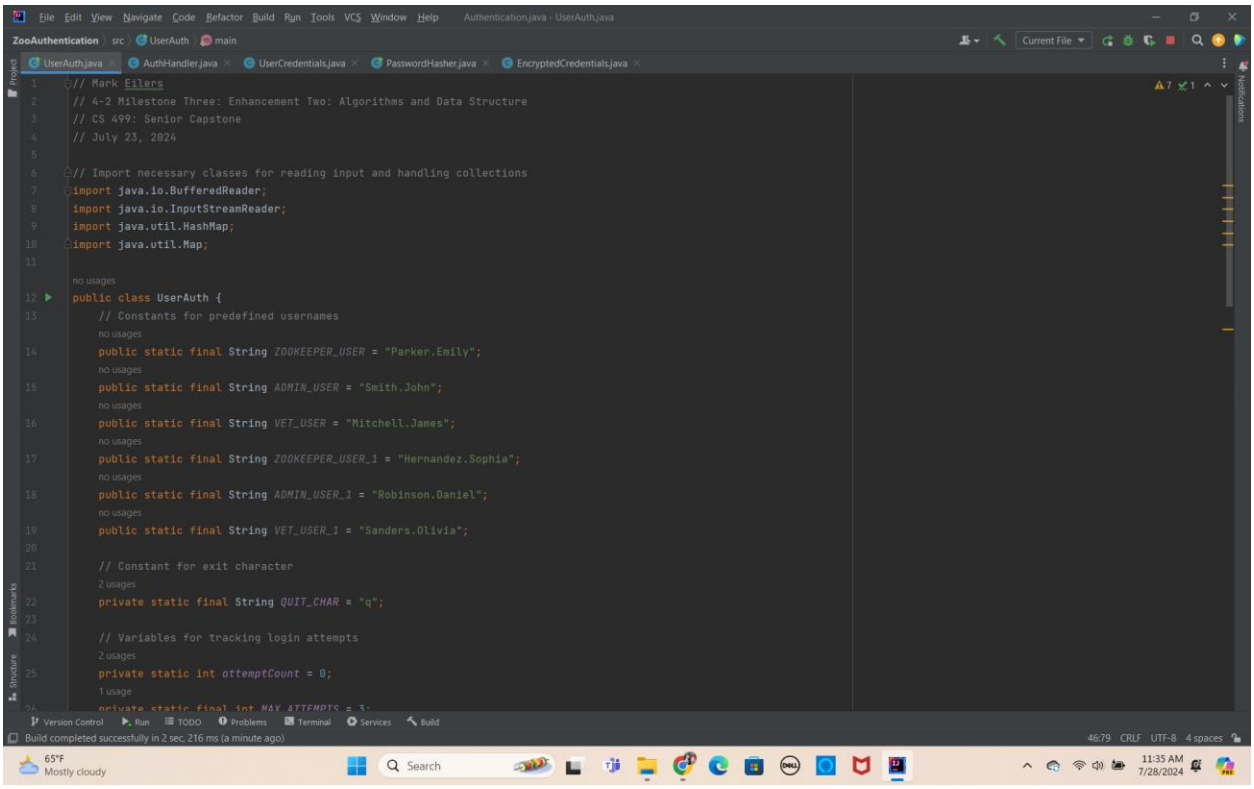

Figure 1: This screenshot shows the beginning stages of the main function of the code. In this image, the necessary imports are shown as well as the creation of the public variables for the different users in the computer system. This screenshot begins the user authentication process.

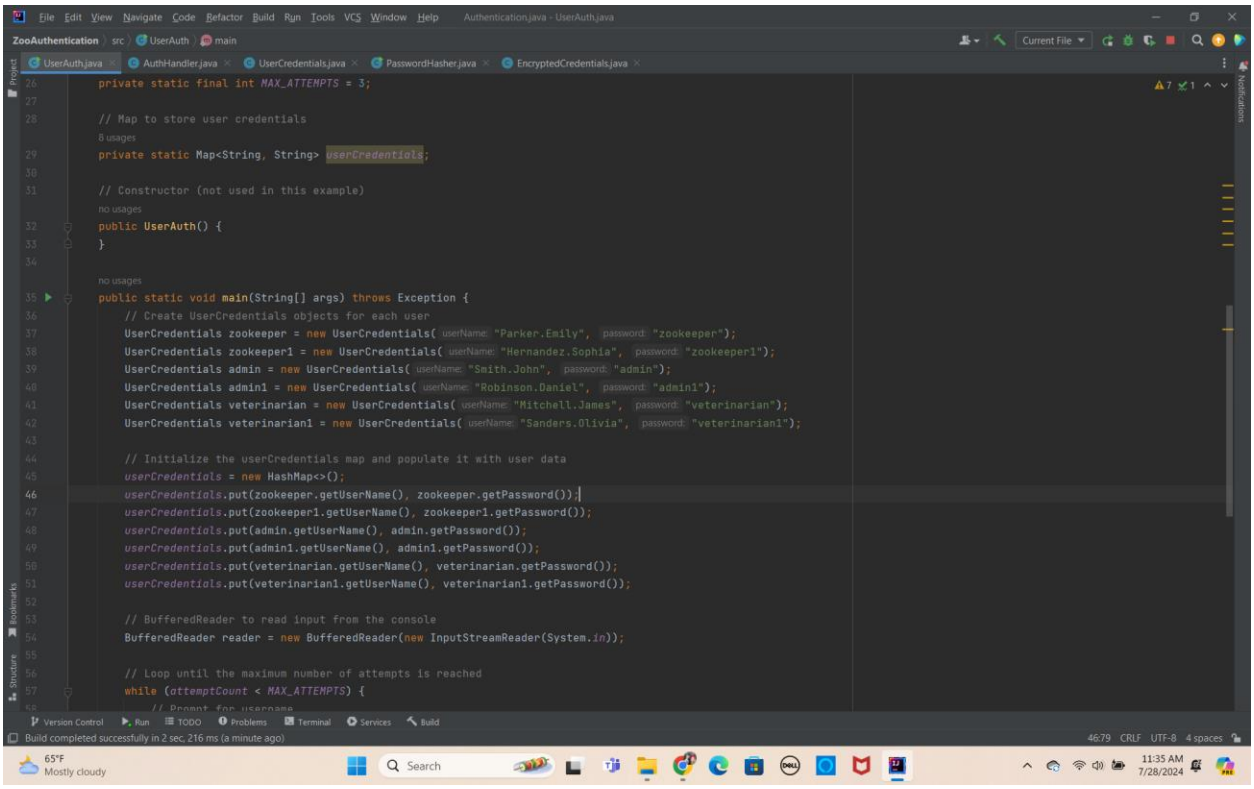

Figure 2: This is continuation of the user authentication class creating user credentials.

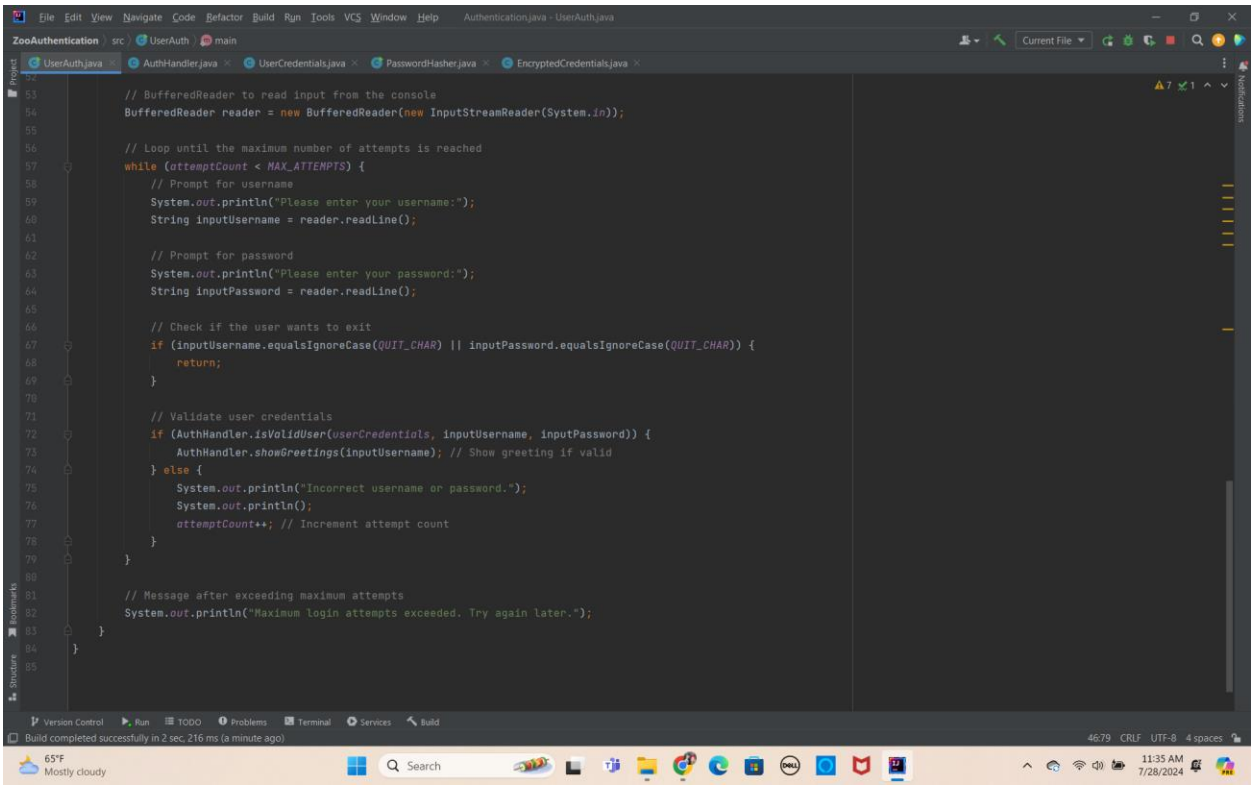

Figure 3: Screenshot shows a while loop statement to check for user credentials. In this section of code, users are given a prompt to enter in their username and password for the system. If the credentials are valid, then users will be granted access to the program. If the credentials are wrong, the program increments the attempted count while informing users that their username and password do not match. Users are then prompted again. If wrong again, attempted count is incremented continuing the loop. Once the attempted count is no longer less than the max attempts, then users are presented with a maximum attempt statement asking them to try again later.

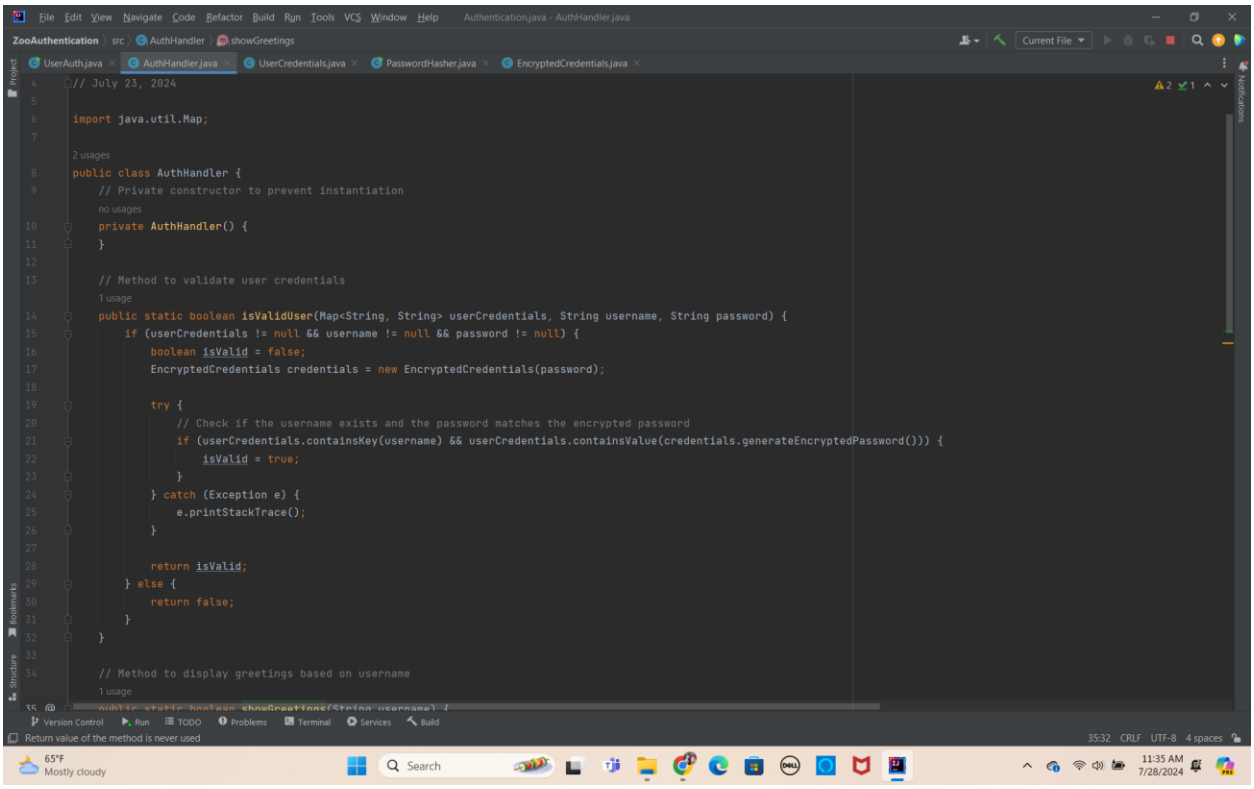

Figure 4: This is the creation of the second class, AuthHandler. In this screenshot, a method is created to validate user credentials.

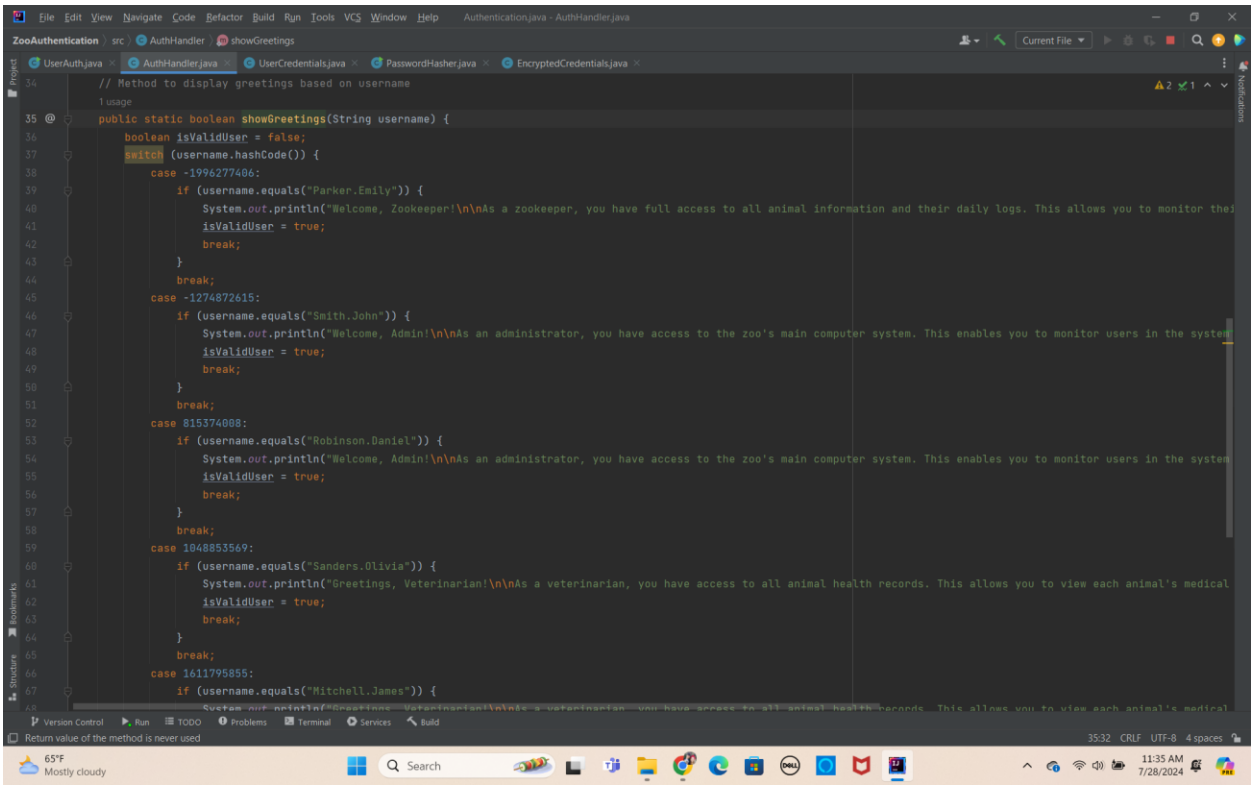

Figure 5: Once the credentials have been determined to be accurate, a welcome statement including the role of the individual is displaced. In this section, switch cases were used to check for the username. If the username was equal to a predetermined username and password from the user authentication class, then the program welcomed the user, as well as informing them of their role within the company.

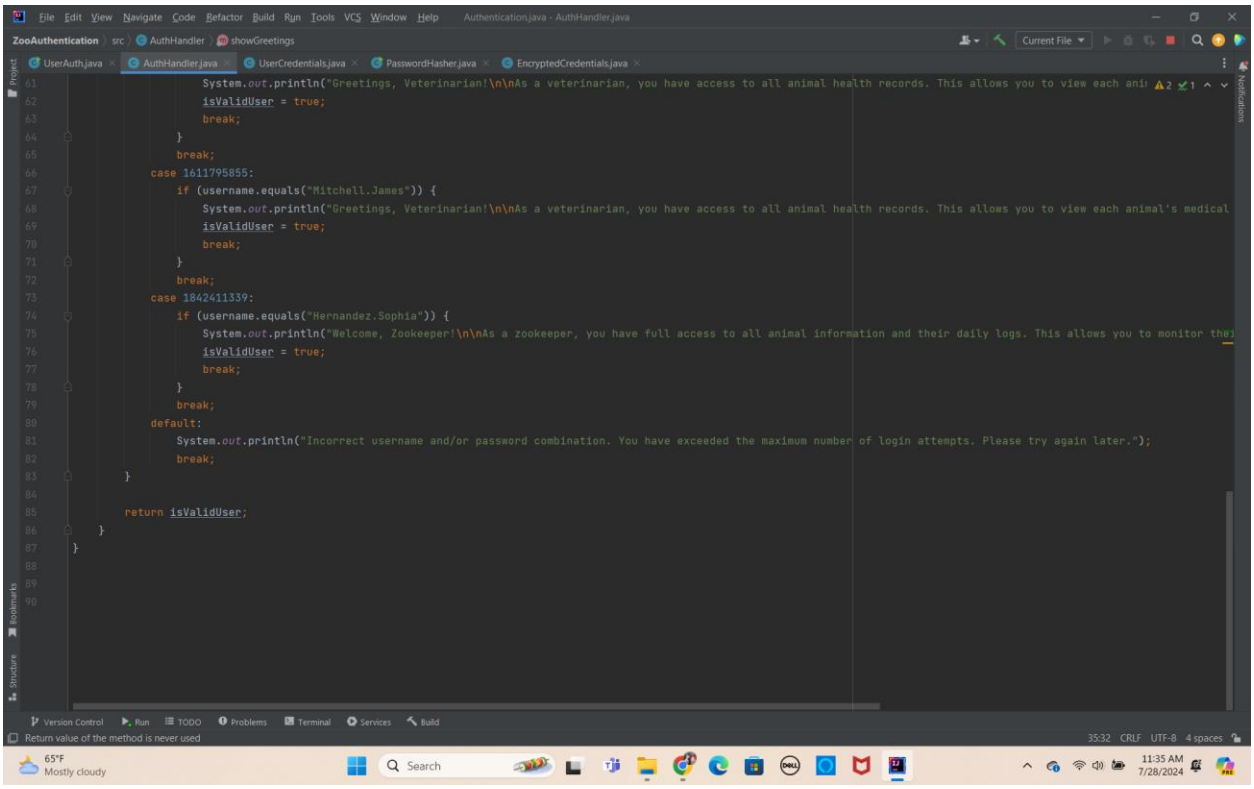

Figure 6: Screenshot is a continuation of Figure 5 with an added statement if none of the credentials are valid. If none of the credentials are valid, then a statement informing the incorrection username and/or password is displayed.

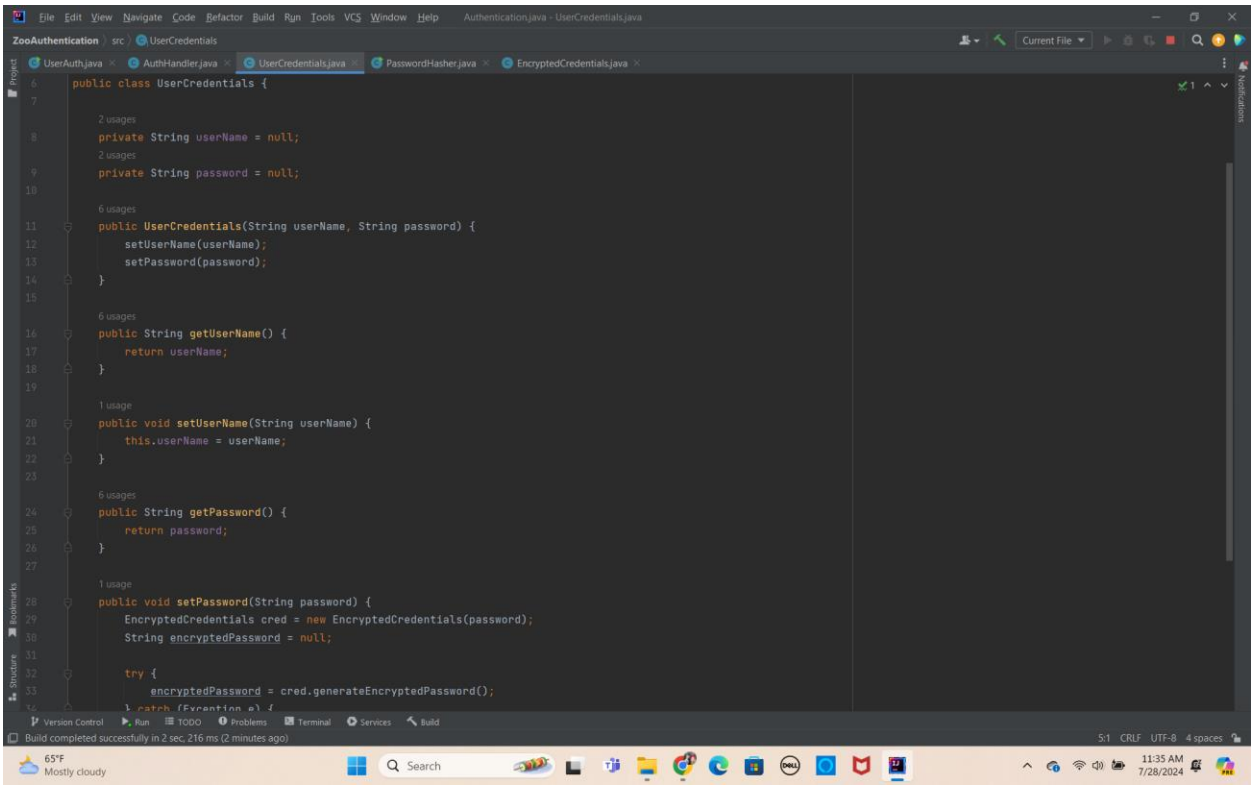

Figure 7: The screenshot shows the user credential class. In this class, the setters and getters with created to make the username and passwords for user credentials.

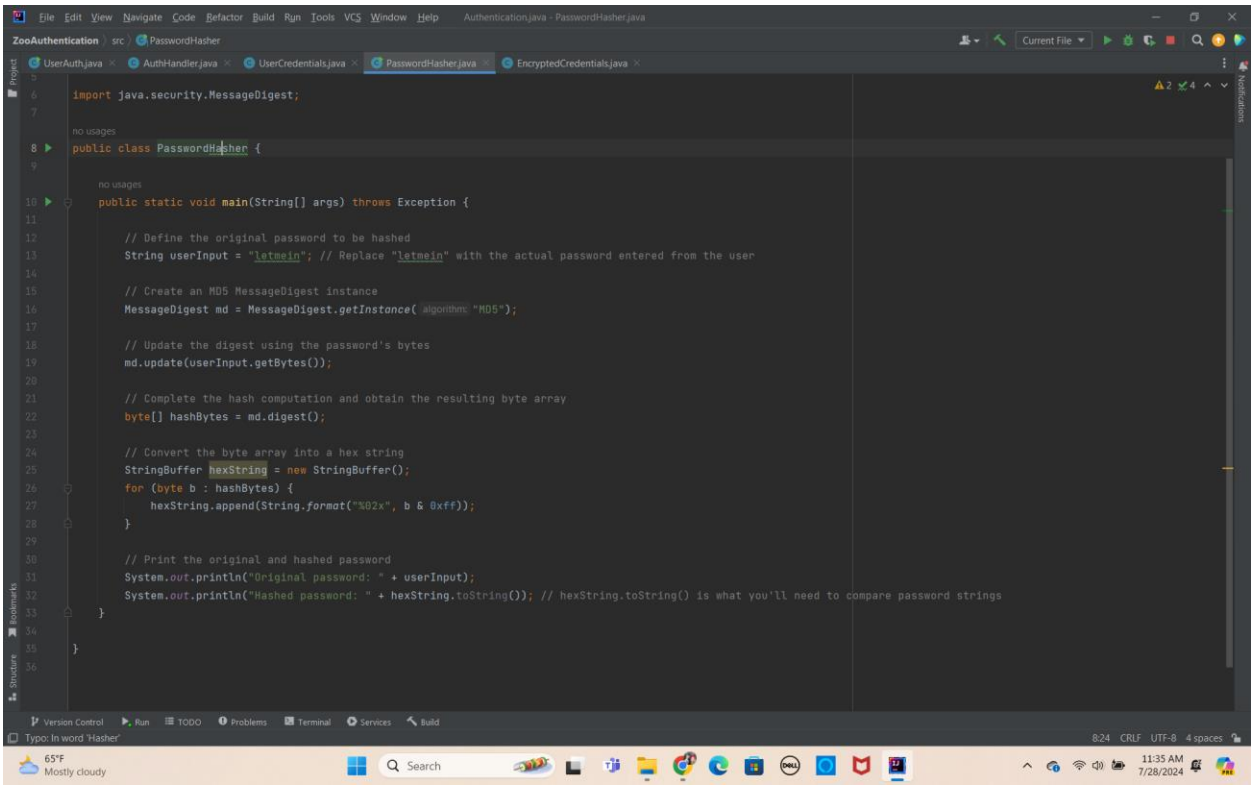

Figure 8: This section of code shows the PasswordHasher class. In this class, to add to a layer of security, the original password a user creates it turned into a hashed password. This class is used to inform users in the end what their password was, as well as their new encrypted password.

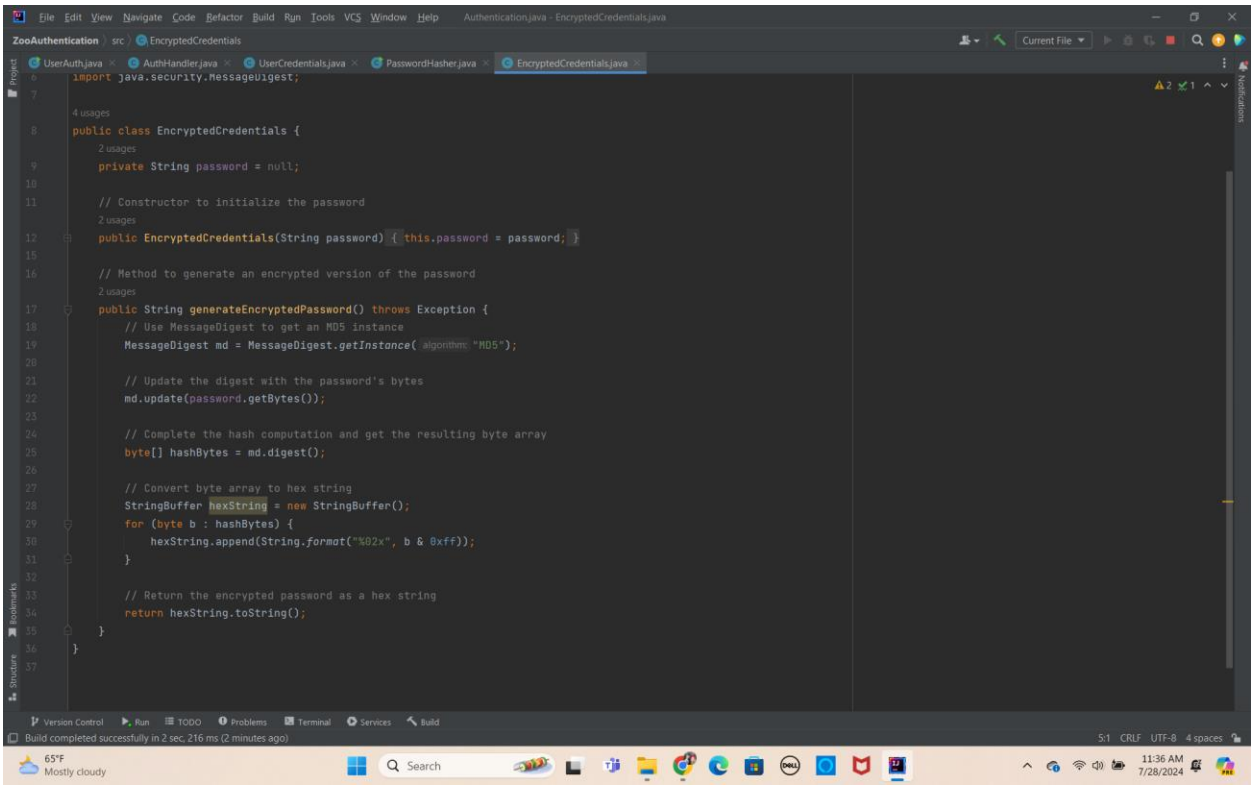

Figure 9: The final screenshot shows the encrypted credentials code to encrypt user's passwords. This process uses MD5 Hashing to encrypt user passwords, adding a layer of security to the overall program.