7-1 Final Project Submission: Authentication or Monitoring System

Arturo Santiago-Rivera

Prof. John Watson

IT-145-X5942 Found in App Development 18EW5

Southern New Hampshire University

June 17, 2018

PROBLEM STATEMENT/SCENARIO

Managing a Zoo's computer infrastructure includes two key components: controlling the access to system data and monitoring the animal/habitat activities in exhibits. As a duty of the role of managing the computer infrastructure, we must develop a working program and detailed documentation describing the development process for the authentication and authorization of the people to gain access to data in the computer systems. In a future, we can extend the program and process documentation to monitor the activities of the animals in the care of the zookeeper and monitor their living habitats.

Once the users enter the computer system, they only should see data related to their particular role. The program needs to read the information in a file with the authorized users and compare it with the user input credentials. The user should input their system username and up to three times their system password. When the user is authenticated, the program should read the corresponding user role file that describes the particular duties and activities authorized in the system. The working program should do the following:

I. Authentication

- Asks the user for a username.
- Asks the user for a password.
- Digest the password using a referenced digest five (MD5) hash.
- Check user credentials against the valid credentials in the credentials file.
- Limits failed password attempts to three before notifying the user and exiting the program.
- After successful user authentication, it gives the user access to the correct role file.

- The system information stored in the role file should be displayed.
- Allows a user to log out.
- It stays on the credential screen until either a successful attempt has been made, three unsuccessful password attempts have been made, or the user chooses to exit the program.

II. Monitoring (FUTURE)

- Asks a user if they want to monitor an animal, monitor a habitat, or exit.
- Display a list of animal/habitat options from either the animals or habitats file.
    - Asks the user to enter one of the options.
- Displays the monitoring information by finding the appropriate section in the file.
- Separates sections by the category and selection (such as “Animal - Lion” or “Habitat -Penguin”)
- Uses a dialog box to alert the zookeeper if the monitor detects something out of the normal range (These will be denoted in the files by a new line starting with *****. Do not display the asterisks in the dialog.)
- Allows a user to return to the original options

OVERALL PROCESS

Initially, we start to figure out the pseudocode of part one of the scenarios, authentication. We used the pseudocode to determine which block of codes can be separated into classes and methods. Assuring that the program worked as required by the scenario, we split the code into a primary class and two classes (modules). One of the two classes is a menu (Display Class) which is repetitive in the key system module, RoleModule. Completed the working program, we

started to introduce some GUI into the program base to clear the shell screen, display a header and banner, and use two third-party classes, one for ANSI colors and the other for wrap lines.

PSEUDOCODE

```
PROGRAM ZooMonitorSystem:

CLASS ZooMonitorSystem
      METHOD main
            WHILE not true
                  OBTAIN user first name or quit
                  IF quit THEN
                        EXIT WHILE
                  ELSE
                        OBTAIN user last name
                        DEFINE user = first name + “.” + last name
                        CALL Authentication user
                  ENDIF/ELSE
                  IF CALL return false THEN
                        DISPLAY user not registered
                  ENDIF
            END WHILE
            DISPLAY Good Bye!
      END METHOD
END CLASS

CLASS Authenticate
      METHOD userCredentials
            OPEN credential file
            WHILE file has lines
                  READ line on file
                  IF user in line THEN
                        FOR password entry three times
                              OBTAIN user password
                              CALL MD5Digest user password
                              IF password in line
                                    GET user role from line
                                    CALL Role user role
                              ELSE IF password run three times
                                    DISPLAY account lock. program terminated
                                    RETURN true
                              ELSE
                                    DISPLAY password incorrect
                              ENDIF/ELSE
                        ENDFOR
                  ENDIF
            ENDWHILE
            RETURN true/false
```

```
      METHOD MD5Digest
            // copy-paste code supplied
            RETURN password digested
      END METHOD
END CLASS

CLASS Display
      METHOD showMenu
            DEFINE menu = log out
            IF user role veterinarian THEN
                  menu += monitoring habitat
            IF user role zookeeper THEN
                  menu += monitoring animal + habitat
            ELSE
                  menu += monitoring animal + habitat + user
            ENDIF/ELSE
            WHILE false
                  DISPLAY menu
                  GET user option
                  SWITCH option
                        CASE log out
                              RETURN true
                  ENDSWITCH
            ENDWHILE
            RETURN true/false
      END METHOD
END CLASS

CLASS RoleModule
      METHOD showDashboard
            OPEN user role file
            WHILE file has lines
                  READ line on file
                  DISPLAY line
            ENDWHILE
            CALL METHOD showMenu user role
            RETURN true/false
      END METHOD
END CLASS
```

METHODS AND CLASSES

Defined our complete pseudocode and program executing correctly, I split the code in the following classes with respective methods:

- `CLASS ZooMonitoringSystem`
    - `METHOD main`
- `CLASS Authenticate`

    - METHOD userCredentials
    - METHOD MD5Diegest
- CLASS Display
    - METHOD clearScreen
    - METHOD strRepeat
    - METHOD showBanner
    - METHOD showDialog
    - METHOD showMenu
- CLASS RoleMonitor
    - METHOD showDashboard
    - METHOD wrapLines

As stated before, we initially worked the program pseudocode and then started to split the code into blocks of codes that are repetitive and can be defined as methods and classes.

ERROR DOCUMENTATION

Significant errors come from splitting into methods and classes and how NetBeans classified each of them and their location when they are imported into the program. Because of this form of classification, we worked the program to handle errors by determining if it runs through the NetBeans output shell or OS cmd or terminal shell/bash.

Another significant error is with issues of a third-party class, AnsiConsole, which is called through a specific path. Since NetBeans work for classes in a specific directory, any class out of this directory is shown as a package that does not exist. If we run the program in the output windows of NetBeans, fits show an error of compilation because the package, AnsiConsole, does not exist in the classes directory. First, we need to compile the program in the OS cmd or terminal shell/bash.

SOLUTION DOCUMENTATION

To resolve our program code to the point we wanted, we explored different blocks of codes to show a GUI simple but good presentation. We wanted to show different screens according to the menu options and to clear the screen for each option and not show everything on one screen. For this purpose, we introduce a block of code that the determine the OS in which the program is executed:

```
// get computer OS type
private static final String OS_SYS = System.getProperty("os.name").toLowerCase();
// verified OS type
if (OS_SYS.contains("win")) {
	// clear screen shell/bash on windows
	new ProcessBuilder("cmd", "/c", "cls").inheritIO().start().waitFor();
} else {
	// clear screen shell/bash on macOS
	new ProcessBuilder("clear").inheritIO().start().waitFor;
}
```

A similar approach was tasked to determine if the program is executed through the NetBeans output windows or the OS cmd or terminal shell/bash:

```
// verification of run mode, NetBeans, or OS terminal
File dir = new File("src/txt_files");
if (!dir.exists()) {
	// cmd or terminal shell path
	dirPath = "txt_files/";
} else {
	// NetBeans debug path
	dirPath = "src/txt_files/";
}
```

References

Southern New Hampshire University. (2018, June 03). *5-2 Milestone One: Pseudocode Submission*. Retrieved from Module Five: File Streaming: https://learn.snhu.edu/d2l/le/content/55717/viewContent/1351330/View

Southern New Hampshire University. (2018, May 13). *IT 145 Final Project Guidelines and Rubric.* Retrieved from Module Seven: 7-1 Final Project Submission: Authentication or Monitoring System: https://learn.snhu.edu/d2l/lor/viewer/viewFile.d2lfile/55717/13546,-1/

Southern New Hampshire University. (2018, June 03). *IT 145 Guide to Pseudocode.* Retrieved from Module Five: 5-2 Milestone One: Pseudocode Submission: https://learn.snhu.edu/d2l/lor/viewer/viewFile.d2lfile/55717/20091,1/

Southern New Hampshire University. (2018, June 03). *IT 145 Milestone One Guidelines and Rubric.* Retrieved from Module Five: 5-2 Milestone One: Pseudocode Submission: https://learn.snhu.edu/d2l/lor/viewer/viewFile.d2lfile/55717/13543,-1/