MODULE 4-2 MILESTONE THREE:

ENHANCEMENT TWO: ALGORITHMS AND DATA STRUCTURE

Arturo Santiago-Rivera

Prof. Brooke Goggin, M.S., M.S., M.S., Ed.D (ABD)

CS-499 Computer Science Capstone 22EW4

Southern New Hampshire University

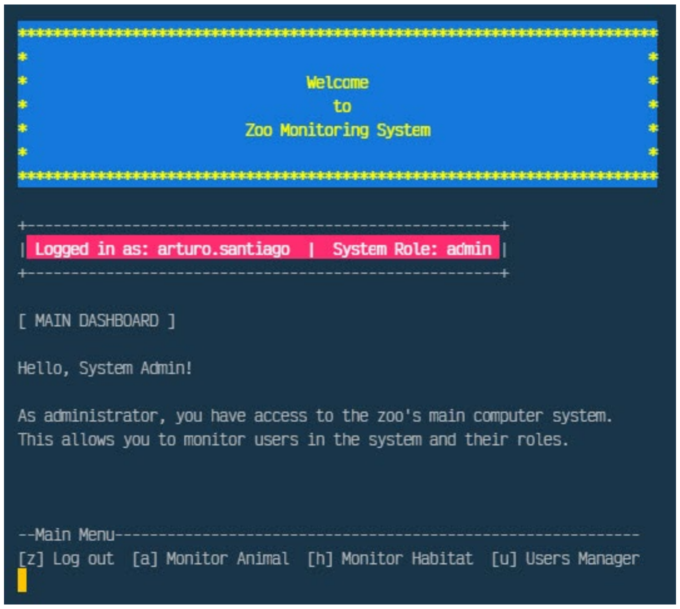
March 27, 2022

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

This paper is a narrative that accompanies the artifact enhancements for algorithms and data structure. It explains why the selected artifact was included in this category of our ePortfolio and reflects on the process used to create the artifact. The narrative focuses on the learning that happened through the artifact's creation (Southern New Hampshire University, 2022).

Prompt

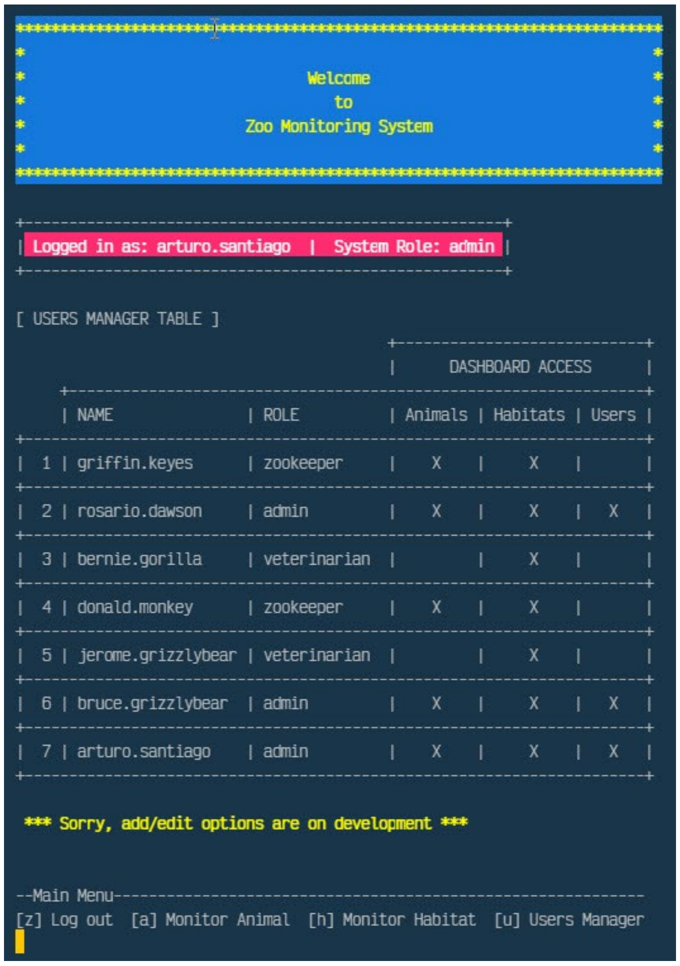
The artifact selected for the algorithms and data structure category is the Zoo Monitor System Program. The program aims to develop an authentication system that manages authentication and authorization for zookeeper users and administrators. The program was planned, designed, and develop as part of the IT145 Foundation in Application Development computer science course. The program is developed in the JAVA programming language as a standalone application running in the computer terminal. The initial development and programming tool used was Apache NetBeans IDE; however, the enhancements were worked on using a text editor. The testing and running of the app are through the computer terminal.



```
*************************************************************************
*                                                                       *
*                               Welcome                                 *
*                                 to                                    *
*                         Zoo Monitoring System                         *
*                                                                       *
*************************************************************************

+-------------------------------------------------------+
| Logged in as: arturo.santiago  |  System Role: admin |
+-------------------------------------------------------+

[ MAIN DASHBOARD ]

Hello, System Admin!

As administrator, you have access to the zoo's main computer system.
This allows you to monitor users in the system and their roles.



--Main Menu-------------------------------------------------------------
[z] Log out  [a] Monitor Animal  [h] Monitor Habitat  [u] Users Manager
```

Figure 1 Zoo Monitor System Main Dashboard for Admin Role

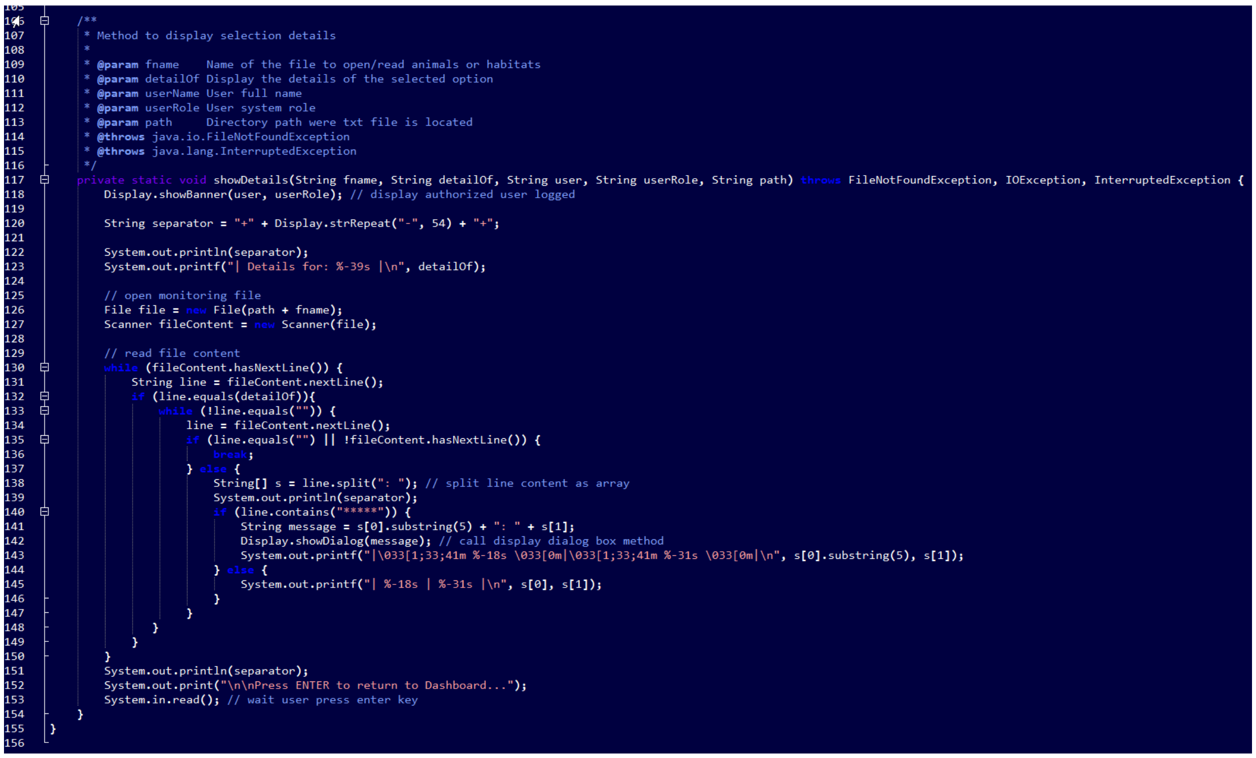
This artifact was selected because it involved understanding a program algorithm composed of two central systems, an authentication/authorization system, with the enhancement of modules for a monitoring system. Once the users enter the program, they only should see data related to their role. The artifact consists of design considerations to authenticate and authorize a user into the monitor system based on user credentials and the accountability of the user interaction with different modules screen and actions according to their role in the monitor system.



```
************************************************************************
*                                                                      *
*                               Welcome                                *
*                                 to                                   *
*                         Zoo Monitoring System                        *
*                                                                      *
************************************************************************

+--------------------------------------------------------+
| Logged in as: arturo.santiago  |  System Role: admin |
+--------------------------------------------------------+

[ USERS MANAGER TABLE ]
                                          +-----------------------------+
                                          |      DASHBOARD ACCESS       |
     +------------------------------------------------------------------+
     | NAME              | ROLE            | Animals | Habitats | Users |
+-----------------------------------------------------------------------+
|  1 | griffin.keyes     | zookeeper     |    X    |     X    |       |
+-----------------------------------------------------------------------+
|  2 | rosario.dawson    | admin         |    X    |     X    |   X   |
+-----------------------------------------------------------------------+
|  3 | bernie.gorilla    | veterinarian  |         |     X    |       |
+-----------------------------------------------------------------------+
|  4 | donald.monkey     | zookeeper     |    X    |     X    |       |
+-----------------------------------------------------------------------+
|  5 | jerome.grizzlybear | veterinarian |         |     X    |       |
+-----------------------------------------------------------------------+
|  6 | bruce.grizzlybear | admin         |    X    |     X    |   X   |
+-----------------------------------------------------------------------+
|  7 | arturo.santiago   | admin         |    X    |     X    |   X   |
+-----------------------------------------------------------------------+

 *** Sorry, add/edit options are on development ***


--Main Menu-------------------------------------------------------------
[z] Log out  [a] Monitor Animal  [h] Monitor Habitat  [u] Users Manager
```

Figure 2 Zoo Monitor System User Management Dashboard

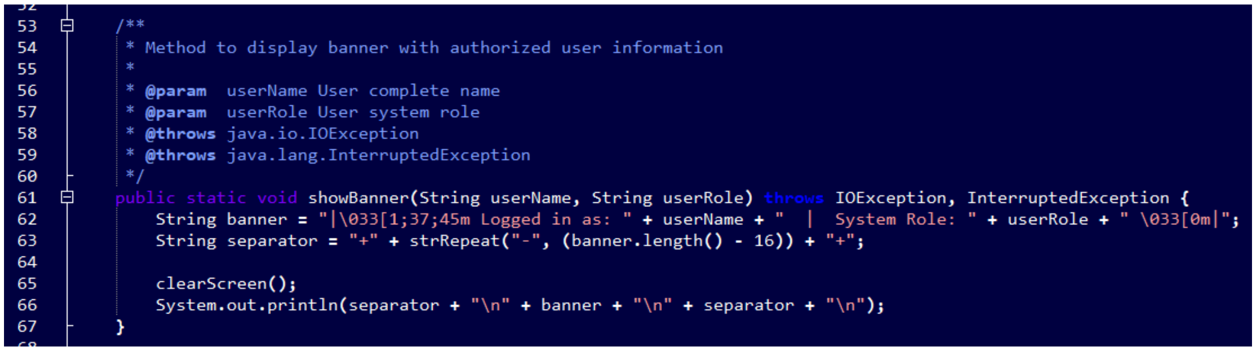
The artifact involves the engineering of practices of validating input data and architect and design with default denial. This skill in us a security mindset that anticipates adversarial exploits in software architecture and designs to expose potential vulnerabilities, mitigate design flaws, and ensure privacy and enhanced data security and resource. The source code is split into different classes and methods based on their functionality and action. We showcase the engineering considerations of relationship and functionality between the different classes and methods by using arguments, parameters, and variables in scope. Reading external files into a dynamic data structure of string array variable to evaluate user inputs' conditions, reading data files line-by-line to evaluate conditions, and reading data files to display their content on the screen. The string array is the simple linear data structure implemented over all the methods in the program classes. This approach improves the design and evaluation of computing solutions that solve a given problem using algorithmic principles and computer science practices and standards appropriate to its solution while managing the trade-offs involved in design choices.



```java
    /**
     * Method to display selection details
     *
     * @param fname    Name of the file to open/read animals or habitats
     * @param detailOf Display the details of the selected option
     * @param userName User full name
     * @param userRole User system role
     * @param path     Directory path were txt file is located
     * @throws java.io.FileNotFoundException
     * @throws java.lang.InterruptedException
     */
    private static void showDetails(String fname, String detailOf, String user, String userRole, String path) throws FileNotFoundException, IOException, InterruptedException {
        Display.showBanner(user, userRole); // display authorized user logged

        String separator = "+" + Display.strRepeat("-", 54) + "+";

        System.out.println(separator);
        System.out.printf("| Details for: %-39s |\n", detailOf);

        // open monitoring file
        File file = new File(path + fname);
        Scanner fileContent = new Scanner(file);

        // read file content
        while (fileContent.hasNextLine()) {
            String line = fileContent.nextLine();
            if (line.equals(detailOf)){
                while (!line.equals("")) {
                    line = fileContent.nextLine();
                    if (line.equals("") || !fileContent.hasNextLine()) {
                        break;
                    } else {
                        String[] s = line.split(": "); // split line content as array
                        System.out.println(separator);
                        if (line.contains("*****")) {
                            String message = s[0].substring(5) + ": " + s[1];
                            Display.showDialog(message); // call display dialog box method
                            System.out.printf("|\033[1;33;41m %-18s \033[0m|\033[1;33;41m %-31s \033[0m|\n", s[0].substring(5), s[1]);
                        } else {
                            System.out.printf("| %-18s | %-31s |\n", s[0], s[1]);
                        }
                    }
                }
            }
        }
        System.out.println(separator);
        System.out.print("\n\nPress ENTER to return to Dashboard...");
        System.in.read(); // wait user press enter key
    }
}
```

Figure 3 MonitorModule Class showDetail Method

The artifact enhancements allow the user to list animal/habitat options by reading from external animal files or habitats files, know the animals' activities in their care, and monitor their living habitats. We showcase our skills and abilities to design software, consider and interpret user needs, and implement them into a program relation of activities in an organized structure. Understanding the algorithms required by the program scenario gives the ability to translate it to pseudocode as one program code. We can determine an organized structure of a block of codes that can be separated into a primary class and four classes (modules). One of the four classes is a menu (Display Class) repetitive in the three key system modules, RoleModule, MonitorModule, and User Module. We introduce GUI actions 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. These actions align with user-centered design principles to demonstrate our ability to use well-founded and innovative techniques, skills, and tools in computing to implement maintainable computer solutions that deliver value and accomplish industry-specific goals.

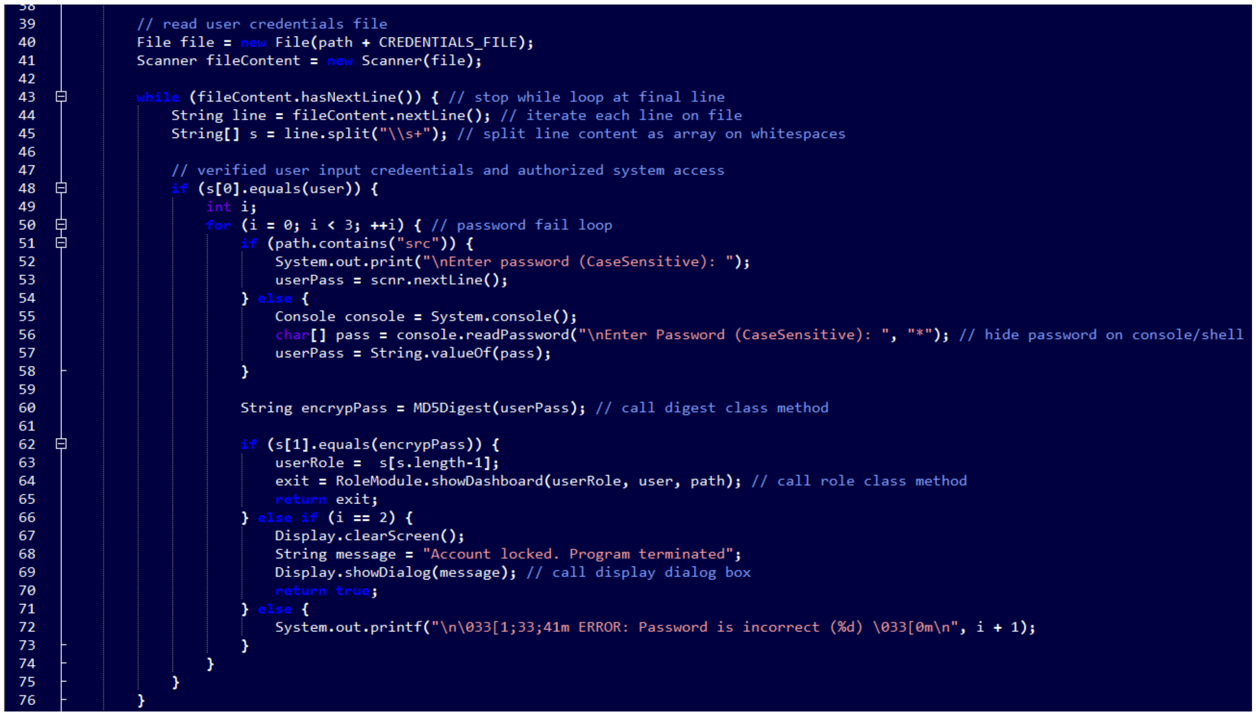


```java
    /**
     * Method to display banner with authorized user information
     *
     * @param  userName User complete name
     * @param  userRole User system role
     * @throws java.io.IOException
     * @throws java.lang.InterruptedException
     */
    public static void showBanner(String userName, String userRole) throws IOException, InterruptedException {
        String banner = "|\033[1;37;45m Logged in as: " + userName + "  |  System Role: " + userRole + " \033[0m|";
        String separator = "+" + strRepeat("-", (banner.length() - 16)) + "+";

        clearScreen();
        System.out.println(separator + "\n" + banner + "\n" + separator + "\n");
    }
```

Figure 4 Display Class showBanner Method

We employ industry-standard JAVA code best practices and techniques such as in-line comments, appropriate naming conventions, formatting, and indentation in conformance with proper coding standards, making the code easy to read and enhancing the application code organization. The program code is easy to read and follows formatting best practices defined by the industry, such as indentation in conformance with appropriate coding standards. The code is clearly and adequately documented with an easy-to-maintain commenting style and consistency.

The source code is well-structured, consistent in style, and consistently properly formatted, including line breaks. We utilize appropriate syntax and conventions in terms of their best practice and use in programming. The implemented data structures are programmatic, where the stored variable values can be used efficiently in other classes methods. Method names are verbs as they represent actions being performed on something. All cases are covered in an IF- -ELSEIF or CASE block, including ELSE or DEFAULT clauses. Loops avoid manipulating the index variable or using it upon exit from the loop.

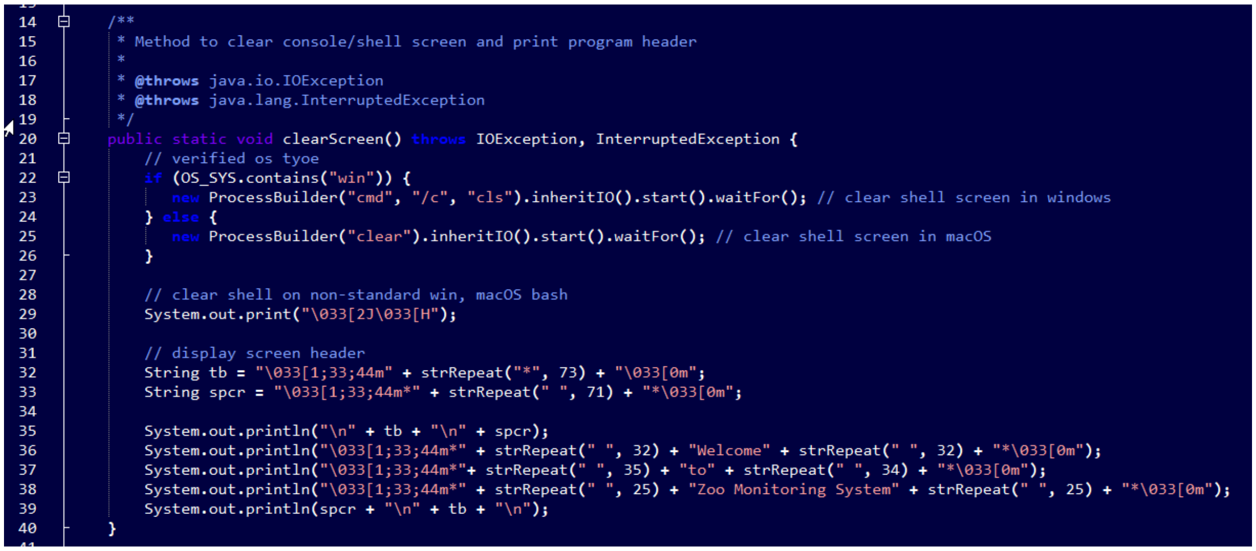


```java
        // read user credentials file
        File file = new File(path + CREDENTIALS_FILE);
        Scanner fileContent = new Scanner(file);

        while (fileContent.hasNextLine()) { // stop while loop at final line
            String line = fileContent.nextLine(); // iterate each line on file
            String[] s = line.split("\\s+"); // split line content as array on whitespaces

            // verified user input credeentials and authorized system access
            if (s[0].equals(user)) {
                int i;
                for (i = 0; i < 3; ++i) { // password fail loop
                    if (path.contains("src")) {
                        System.out.print("\nEnter password (CaseSensitive): ");
                        userPass = scnr.nextLine();
                    } else {
                        Console console = System.console();
                        char[] pass = console.readPassword("\nEnter Password (CaseSensitive): ", "*"); // hide password on console/shell
                        userPass = String.valueOf(pass);
                    }

                    String encrypPass = MD5Digest(userPass); // call digest class method

                    if (s[1].equals(encrypPass)) {
                        userRole =  s[s.length-1];
                        exit = RoleModule.showDashboard(userRole, user, path); // call role class method
                        return exit;
                    } else if (i == 2) {
                        Display.clearScreen();
                        String message = "Account locked. Program terminated";
                        Display.showDialog(message); // call display dialog box
                        return true;
                    } else {
                        System.out.printf("\n\033[1;33;41m ERROR: Password is incorrect (%d) \033[0m\n", i + 1);
                    }
                }
            }
        }
```

Figure 5 Authenticate Class While Loop If-Else Example

Significant errors come from splitting into methods and classes and how to classify 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 terminal shell/bash. To resolve our program code to the point we wanted, we explored different blocks of codes to show a simple GUI but of good presentation.

We use the Jansi 2.1.0 API JAVA library, which allows us to add ANSI colors to our dashboard screens.



```java
/**
 * Method to clear console/shell screen and print program header
 *
 * @throws java.io.IOException
 * @throws java.lang.InterruptedException
 */
public static void clearScreen() throws IOException, InterruptedException {
    // verified os tyoe
    if (OS_SYS.contains("win")) {
        new ProcessBuilder("cmd", "/c", "cls").inheritIO().start().waitFor(); // clear shell screen in windows
    } else {
        new ProcessBuilder("clear").inheritIO().start().waitFor(); // clear shell screen in macOS
    }

    // clear shell on non-standard win, macOS bash
    System.out.print("\033[2J\033[H");

    // display screen header
    String tb = "\033[1;33;44m" + strRepeat("*", 73) + "\033[0m";
    String spcr = "\033[1;33;44m*" + strRepeat(" ", 71) + "*\033[0m";

    System.out.println("\n" + tb + "\n" + spcr);
    System.out.println("\033[1;33;44m*" + strRepeat(" ", 32) + "Welcome" + strRepeat(" ", 32) + "*\033[0m");
    System.out.println("\033[1;33;44m*"+ strRepeat(" ", 35) + "to" + strRepeat(" ", 34) + "*\033[0m");
    System.out.println("\033[1;33;44m*" + strRepeat(" ", 25) + "Zoo Monitoring System" + strRepeat(" ", 25) + "*\033[0m");
    System.out.println(spcr + "\n" + tb + "\n");
}
```

Figure 6 Display Class clearScreen Method

We wanted to show different screens according to the menu options, clear the screen for each option, and not show everything on one screen. For this purpose, we introduce a code block that determines the OS in which the program is executed. Working with file streaming has been extremely exciting, making us think about the program's enhancements with attention to detail. We accomplished all our improvement for the program presentation and simplified the code classes and methods. We produce a working program beyond an input/output exercise that has required us to research things that we have done in other languages, which can be produced on multiple operating systems like Windows and macOS.

References

Southern New Hampshire University. (2022, March 21). *Milestone Three Guidelines and Rubric Enhancement Two: Algorithms and Data Structure* . Retrieved from Module 4-2 Milestone Three: Enhancement Two: Algorithms and Data Structure: https://learn.snhu.edu/d2l/common/dialogs/quickLink/quickLink.d2l?ou=1014915&type=coursefile&fileId=Course+Documents%2fCS+499+Milestone+Three+Guidelines+and+Rubric.pdf