MODULE 4-2 MILESTONE THREE:

ENHANCEMENT TWO: ALGORITHMS AND DATA STRUCTURE

Poritosh Mridha
CS-499 Computer Science Capstone

Southern New Hampshire University

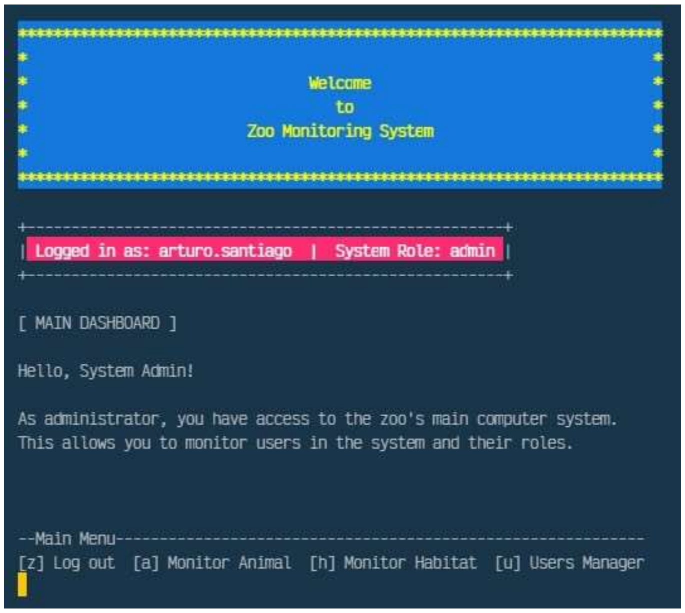
June 2, 2024

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

This document serves as a narrative to accompany the enhancements made to artifacts related to algorithms and data structures. It provides an explanation for the inclusion of the chosen artifact in this section of our ePortfolio and offers a reflection on the creation process. The narrative emphasizes the learning experiences gained during the development of the artifact (Southern New Hampshire University, 2024).

Prompt

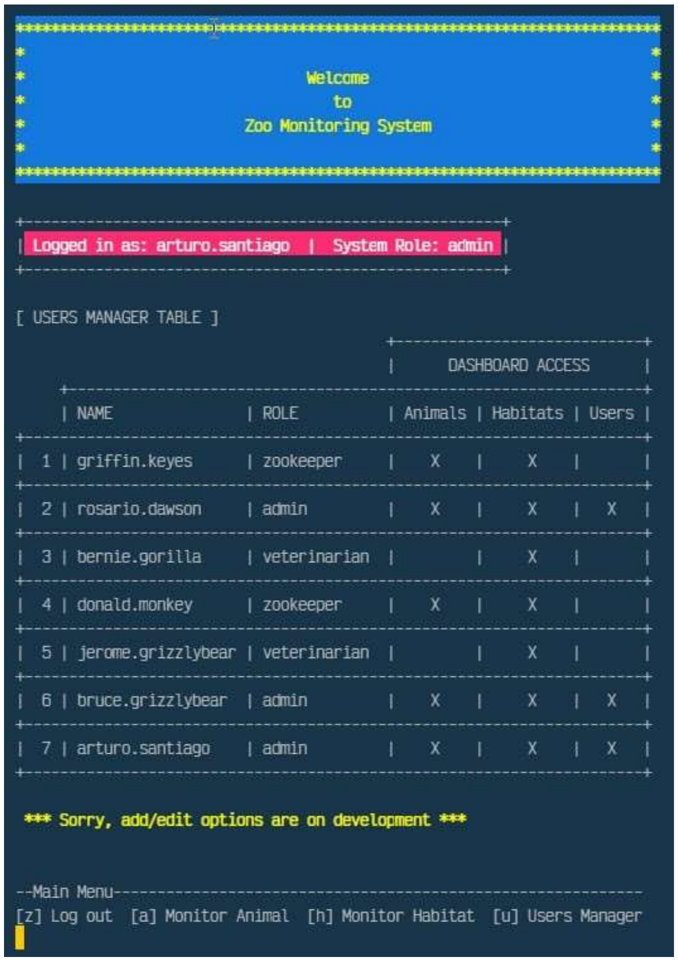
The chosen artifact for the algorithms and data structure category is the Zoo Monitor System Program. This program is designed to create an authentication system that handles authentication and authorization for zookeeper users and administrators. It was planned, designed, and developed as part of the IT145 Foundation in Application Development computer science course. The program is written in the JAVA programming language as a standalone application running in the computer terminal. Initially, Apache NetBeans IDE was used for development and programming, but the enhancements were made using a text editor. The application is tested and executed 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

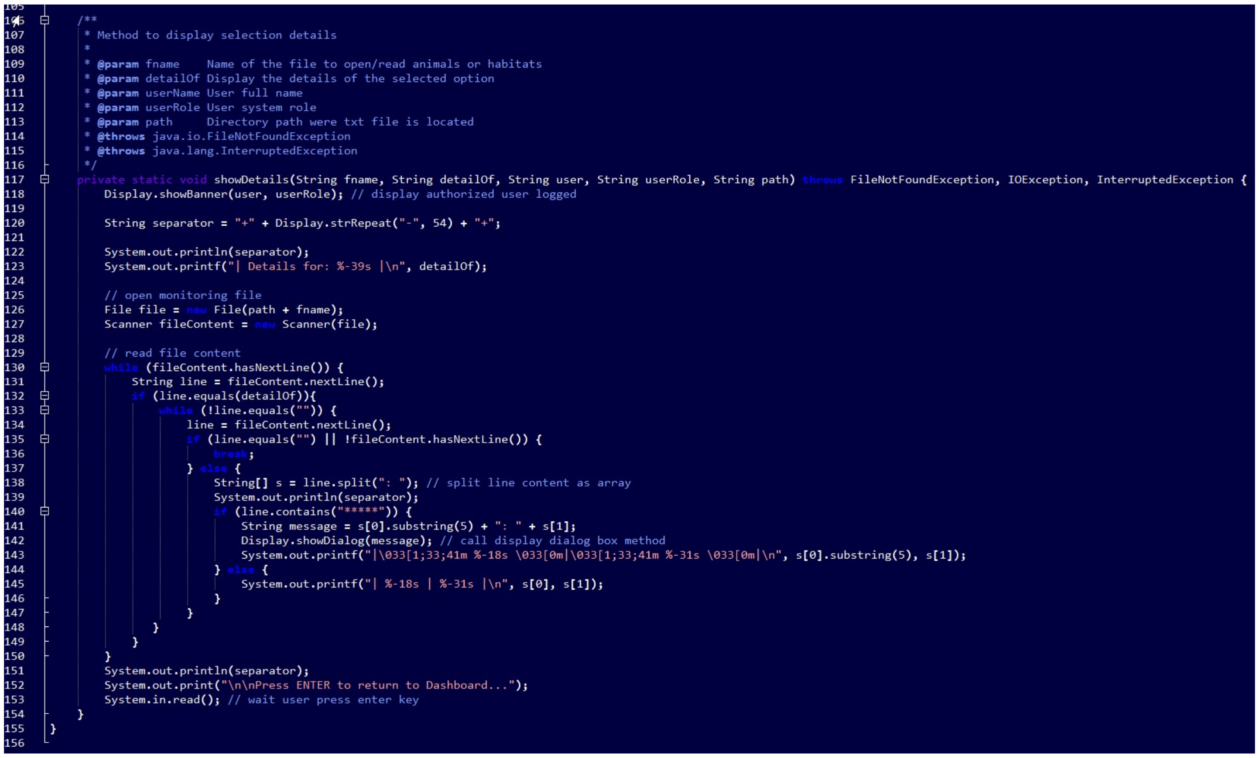
This artifact was chosen due to its requirement for comprehending a program algorithm consisting of two main systems: an authentication/authorization system and modules for a monitoring system. Once users access the program, they should only view data relevant to their role. The artifact includes design considerations for authenticating and authorizing a user into the monitoring system based on their credentials, as well as tracking user interactions with various module screens and actions according to their role within the monitoring 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

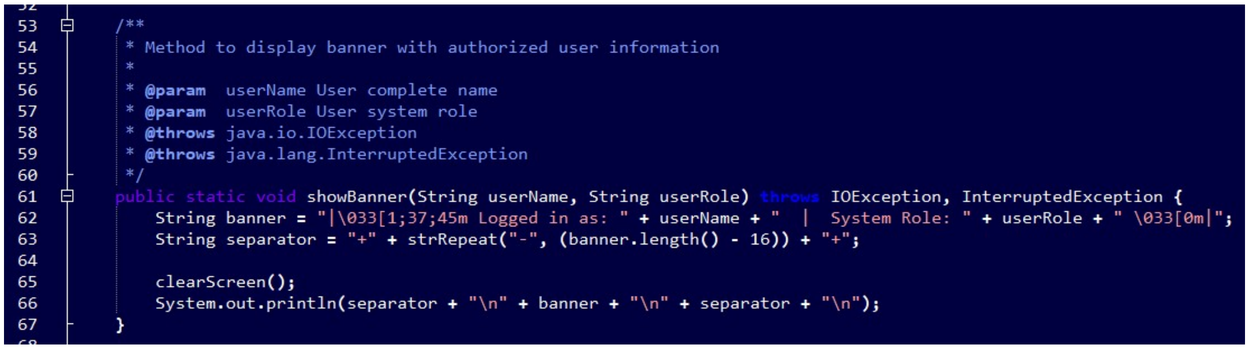
The artifact encompasses engineering practices focused on validating input data and designing with a default denial approach. This skill instills a security mindset that anticipates adversarial exploits in software architecture and designs, aiming to identify potential vulnerabilities, mitigate design flaws, and ensure privacy and enhanced data security and resources. The source code is divided into various classes and methods based on their functionality and purpose. We illustrate the engineering considerations of relationships and functionalities between different classes and methods through the use of arguments, parameters, and scoped variables. The program reads external files into a dynamic data structure of string arrays to evaluate user input conditions, reads data files line-by-line for condition evaluation, and displays file content on the screen. The string array, a simple linear data structure, is implemented across all methods in the program classes. This approach enhances the design and assessment of computing solutions to solve specific problems using algorithmic principles and computer science practices and standards, while effectively 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 enable users to list animal and habitat options by reading from external animal or habitat files, track the activities of animals in their care, and monitor their living environments. We demonstrate our skills and abilities to design software, consider and interpret user needs, and implement them into a structured program with organized activities. Understanding the algorithms required for the program scenario allows us to translate it into pseudocode and eventually into a coherent program code. We can determine an organized code structure that separates into a primary class and four modules. One of these modules is a menu (Display Class), which is repeated in the three key system modules: RoleModule, MonitorModule, and UserModule. 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 line wrapping. These actions align with user-centered design principles, showcasing our ability to employ well-founded and innovative techniques, skills, and tools in computing to implement maintainable computer solutions that deliver value and meet industry-specific goals.

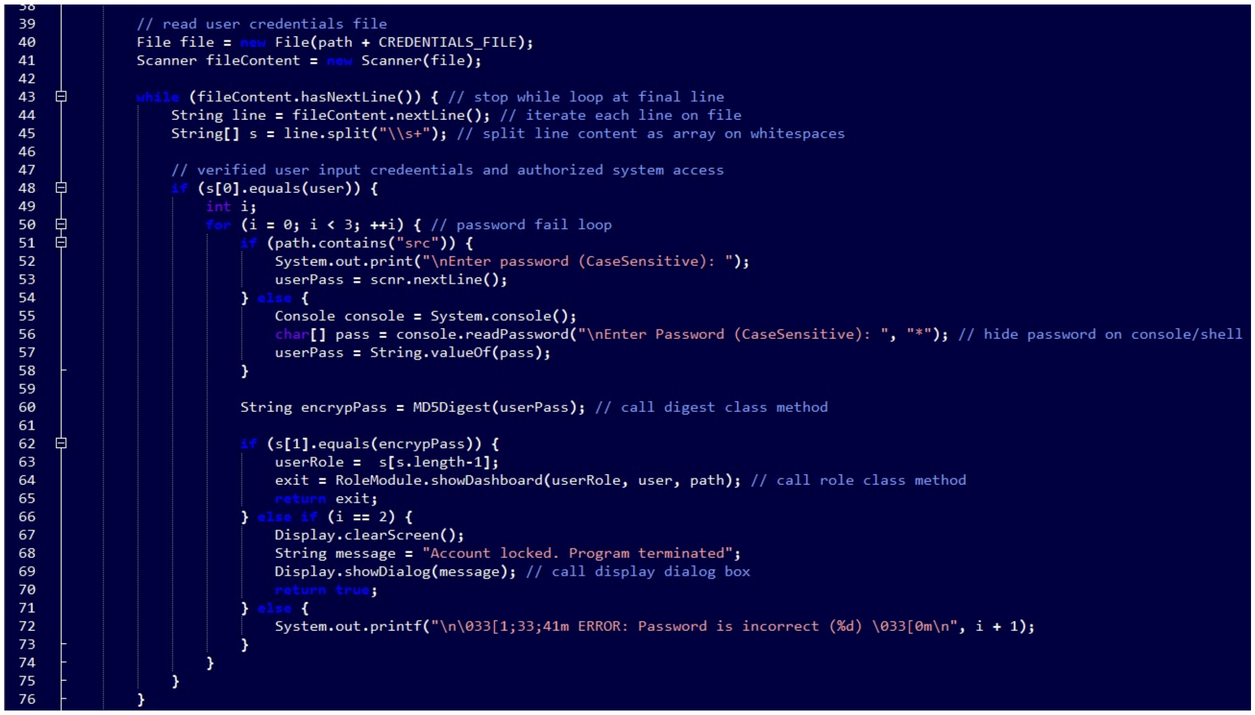


```java
/**
 * Method to display banner with authorized user information
 *
 * @param  userName User complete name
 * @param  userRole User system role
 * @throws java.io.IOException
 * @throws java.lang.InterruptedException
 */
public static void showBanner(String userName, String userRole) throws IOException, 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 adhere to industry-standard JAVA code best practices and techniques, including in-line comments, appropriate naming conventions, formatting, and indentation, in line with proper coding standards. This approach makes the code easy to read and enhances the organization of the application code. The program code is designed for readability, following industry-defined formatting best practices such as consistent indentation according to appropriate coding standards. The code is clearly and adequately documented with a maintainable commenting style and consistency.

The source code is well-structured, maintaining a consistent style and proper formatting, including line breaks. We use appropriate syntax and conventions according to best practices in programming. The implemented data structures are designed programmatically, allowing stored variable values to be used efficiently in other class methods. Method names are verbs, representing actions performed on something. All cases are covered in IF-ELSEIF or CASE blocks, including ELSE or DEFAULT clauses. Loops avoid manipulating the index variable or using it upon exit from the loop, ensuring clean and reliable code.

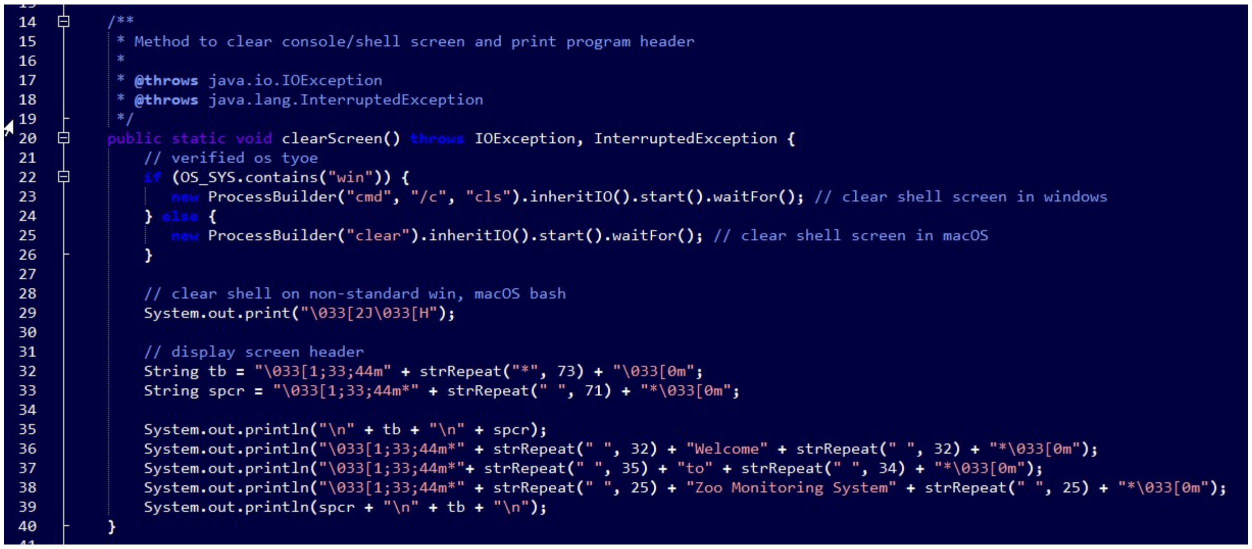


```java
        // read user credentials file
        File file = new File(path + CREDENTIALS_FILE);
        Scanner fileContent = new Scanner(file);

        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 challenges arose from dividing the program into methods and classes and determining the appropriate classification and location for each when imported into the program. Due to this classification approach, we modified the program to handle errors by checking if it runs through the NetBeans output shell or the OS terminal shell/bash. To refine our program code to the desired state, we explored various code blocks to create a simple yet well-presented GUI.

We utilized the Jansi 2.1.0 API JAVA library, which enabled us to incorporate ANSI colors into 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 aimed to display different screens based on the menu options, clearing the screen for each selection, rather than displaying everything on one screen. To achieve this, we introduced a code block that detects the operating system on which the program is running. Working with file streaming has been particularly exciting, prompting us to focus on enhancing the program with meticulous attention to detail. We achieved all our improvements for the program's presentation and streamlined the code classes and methods. We developed a functional program that goes beyond a simple input/output exercise, requiring us to research techniques we've used in other languages, and ensuring compatibility across multiple operating systems, such as Windows and macOs