Augustus Mendy

Southern New Hampshire University

CS-499-10450-M01

Professor Ramsey

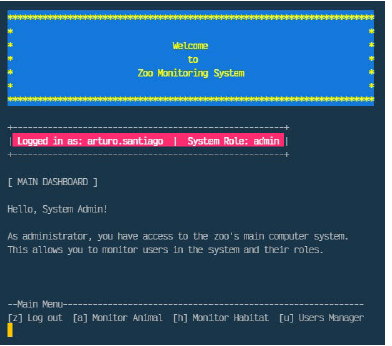
September 21, 2025

4-2 Milestone Three Enhancement Two Algorithms and Data Structure

This paper tells the story of the artifact improvements for software engineering and design. It provides insight into the process of creating the chosen artifact and explains why it was included in this section of our ePortfolio. The story is centered on the knowledge gained via the development of the artifact.

Prompt

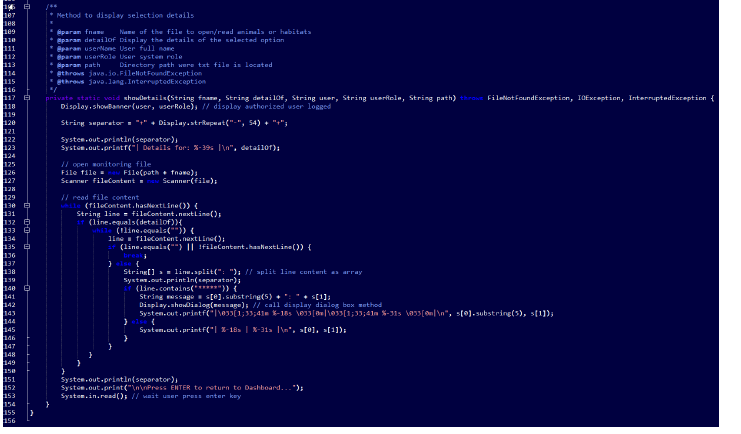
For the category of algorithms and data structures, the Zoo Monitor System Program was chosen as the artifact. Developing an authentication system that controls authorization and authentication for zookeeper administrators and users is the program's goal. As a component of the computer science course IT145 Foundation in Application Development, the software was created, planned, and designed. It is a stand-alone software that runs at the computer terminal and was created using the Java programming language. Although a text editor was utilized to work on the improvements, the Apache NetBeans IDE was the original development and programming tool. The computer terminal is used to test and operate the application.



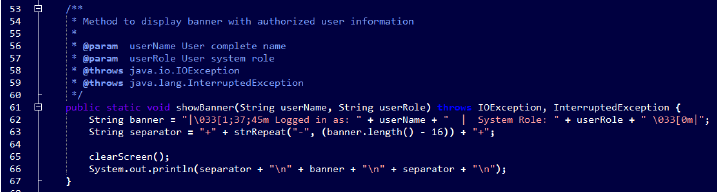
This artifact was chosen because it required comprehension of a program algorithm made up of two main systems: an authorization/authentication system and modules for a monitoring system. Only information relevant to their position should be visible to users after they enter the program. Design considerations for authenticating and authorizing a user into the monitor system based on user credentials and the responsibility of the user's interaction with various modules' screens and actions in accordance with their roles within the monitor system comprise the artifact.



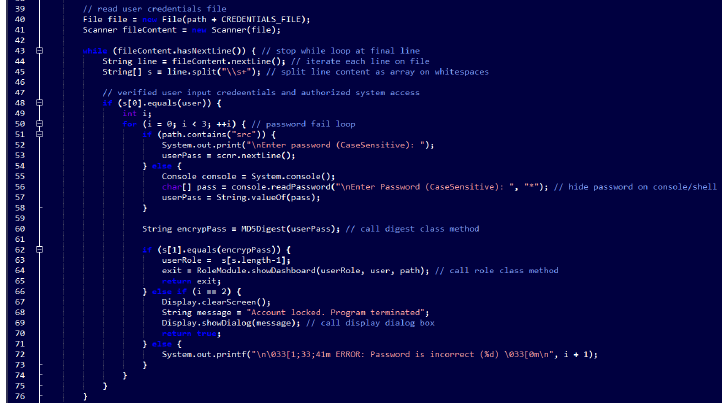
Engineering procedures for checking input data, as well as architect and design with default denial, are all part of the artifact. By anticipating adversarial exploits in software architecture and designs, this skill helps us develop a security mentality that can identify potential vulnerabilities, minimize design faults, and  protect privacy while improving data security and resource utilization. The source code is divided into many classes and methods according to their action and functionality. The technical concerns of the link and functionality between the many classes and methods are demonstrated by the use of variables, parameters, and arguments in scope. To analyze the conditions of user inputs, external files are read into a dynamic data structure of string array variable. Data files are read line-by-line to evaluate conditions and to display their information on the screen. All of the program classes' methods use the string array, a straightforward linear data structure. Through the management of design trade-offs, this method enhances the creation and assessment of computational solutions that address a specific problem by applying algorithmic concepts, computer science techniques, and standards relevant to its solution.



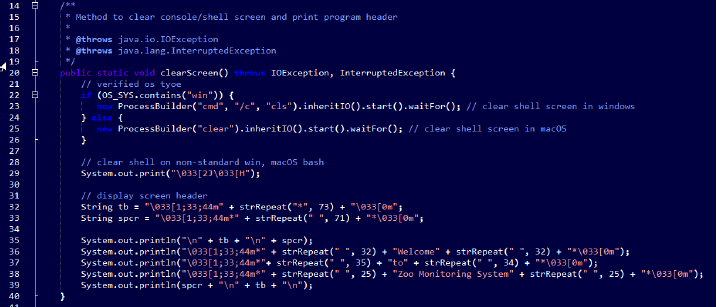
The modifications to the artifacts enable the user to monitor the living habitats of the animals under their care, list animal/habitat options by reading from external animal or habitat files, and learn about the behaviors of the animals. We demonstrate our aptitude for software design, user needs analysis and interpretation, and program linkage of tasks in a structured manner. Comprehending the algorithms needed for the program scenario enables one to convert it to pseudocode as a single program code. We can identify the structured arrangement of a block of codes that can be divided into four classes (modules) and a principal class. The RoleModule, MonitorModule, and User Module are the three main system modules that use the Display Class, one of the four classes, as a menu. In order to display a header and banner, clear the shell screen, and use two third-party classes—one for ANSI colors and the other for wrap lines—I add GUI operations to the program foundation. These efforts, which are in line with the principles of user-centered design, show that we can apply sound and creative computing methods, abilities, and resources to create scalable computer systems that add value and achieve industry-specific objectives.



I used industry-standard JAVA code best practices and approaches, like in-line comments, proper naming conventions, formatting, and indentation, to improve the organization of the application code and make the code easy to understand. The computer code adheres to industry-defined formatting best practices, including indentation in accordance with relevant coding standards, and is simple to understand. With a consistent and easy-to-maintain commenting style, the code is well and concisely described. The source code is uniformly formatted, including line breaks, and is well-structured and consistent in style. We employ proper syntax and rules based on best practices and programming applications. Programmatic data structures are used, allowing the values of stored variables to be effectively utilized in methods of other classes. Because they describe actions being taken on something, method names are verbs. An IF--ELSEIF or CASE block covers all instances, even those involving ELSE or DEFAULT clauses. Loops refrain from modifying the index variable or utilizing it when they are finished.



The division into classes and methods, as well as the classification of each one and its location upon importation into the program, result in significant inaccuracies. This type of classification allowed us to figure out if the application runs through the OS terminal shell/bash or the NetBeans output shell, which allowed us to handle failures. In order to get our program code to the desired point, we experimented with several code blocks to display a straightforward GUI with a polished appearance.   
Our dashboard screens can have ANSI colors added thanks to the Jansi 2.1.0 API Java package.



My goal was to display distinct screens based on the menu choices, clean the screen for each option, and avoid displaying everything on a single screen. We accomplish this by introducing a code block that establishes the operating system that the program runs on. Working with file streaming has been quite interesting and has forced us to consider the program's improvements carefully. We simplified the code classes and procedures and completed all of our improvements for the program presentation. Beyond mere input/output exercise, we create a functional software that can be run on many operating systems, such as Windows and macOS, and has compelled us to investigate work we have done in other languages.

**References:**

“Southern New Hampshire University. (2025, September 19). *Milestone Two Guidelines and Rubric Enhancement One: Software Design/Engineering*. Retrieved from Module 3-2 Milestone Two: Enhancement One: Software Design and Engineering: <https://learn.snhu.edu/d2l/common/dialogs/quickLink/quickLink.d2l?ou=1014915&type=coursefile&fileId=Course+Documents%2fCS+499+Milestone+Two+Guidelines+and+Rubric.pdf>.”

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>.