ENHANCEMENT TWO: ALGORITHMS AND DATA STRUCTURE

Abigail Sibazeu

CS-499 Computer Science Capstone

Southern New Hampshire University

02/02/2025

This document serves as an explanation accompanying the improvements made to the artifact related to algorithms and data structures. It provides justification for its inclusion in this section of our ePortfolio and offers insights into the methodology applied during its development. Additionally, it highlights the knowledge gained throughout the artifact's creation process.

The Zoo Monitor System was chosen as the representative artifact for the algorithms and data structures category due to its focus on implementing secure authentication and access management. This system is designed to control user authentication and authorization, ensuring that zookeepers and administrators have appropriate access based on their roles. It establishes a structured login mechanism that differentiates between standard users and those with higher privileges, reinforcing security within the application.

This project was initially conceptualized and developed as part of the IT145 Foundation in Application Development course, where it served as a practical exercise in programming fundamentals, object-oriented design, and security principles. The system was built as a standalone terminal-based application using Java, allowing for efficient execution within a command-line environment.

The development process began with designing the system's architecture, which involved structuring the authentication framework, defining user roles, and implementing secure access controls. Apache NetBeans IDE was utilized during the initial development phase to streamline coding, debugging, and project organization. However, as enhancements were introduced, a basic text editor was used to refine the code, optimize performance, and implement additional features.

Testing and execution of the Zoo Monitor System are conducted entirely within the computer terminal, ensuring a lightweight yet effective application that can run on various platforms without requiring a graphical interface. Through this project, key programming concepts such as data handling, conditional logic, and authentication mechanisms were reinforced, contributing to a deeper understanding of algorithms and data structures in real-world applications.

This artifact was chosen because it required comprehending a program algorithm structured around two main components: an authentication and authorization system, along with an enhanced monitoring module. The system ensures that once users log in, they can only access information relevant to their assigned roles. The design of this artifact involved carefully planning how users would be authenticated and granted appropriate access based on their credentials. Additionally, it incorporated accountability measures to track user interactions across various system modules, ensuring that actions and data visibility were restricted according to their designated permissions within the monitoring system.

This artifact incorporates engineering best practices for validating input data and designing software with a default-denial security model. This approach fosters a security-focused mindset that anticipates potential exploits, allowing for the identification of vulnerabilities, mitigation of design flaws, and reinforcement of data privacy and resource protection.

The program's source code is organized into distinct classes and methods, each serving a specific function. The engineering design emphasizes the relationships and interactions between these components through the use of arguments, parameters, and scoped variables. A key aspect of the implementation involves reading external files into a dynamic data structure, specifically a string array, to process user inputs. The system evaluates conditions by reading data files line by line, ensuring logical decision-making and displaying relevant content as needed.

The use of a string array as a simple linear data structure is consistently applied across various methods within the program's classes. This structured approach enhances the design and efficiency of computing solutions by leveraging algorithmic principles and adhering to established computer science standards. Additionally, it enables careful evaluation of trade-offs in design choices, ensuring that the program not only meets functional requirements but also maintains security and performance considerations throughout its implementation.

Text

Description automatically generated

*Figure 3 MonitorModule Class showDetail Method*

The enhancements made to this artifact enable users to access a list of animal and habitat options by reading data from external files. Additionally, users can track the activities of animals under their care and monitor the conditions of their habitats. These improvements demonstrate our ability to design software solutions that effectively interpret and address user needs by structuring program functionalities in a well-organized manner.

A deep understanding of the necessary algorithms for this program scenario allows for an efficient translation into pseudocode, which is then implemented as a cohesive program. The code is structured into a primary class and four supporting modules, ensuring modularity and maintainability. One of these modules, the Display Class, is a recurring menu system integrated across three key components: RoleModule, MonitorModule, and UserModule. This modular design improves code organization and facilitates streamlined functionality.

Further refinements introduce graphical user interface (GUI) elements that enhance user interaction. These include features for clearing the shell screen, displaying headers and banners, and integrating two third-party classes—one for ANSI color formatting and another for line wrapping. These enhancements align with user-centered design principles, demonstrating our ability to employ industry-standard tools, techniques, and innovative approaches to develop reliable, maintainable, and value-driven computing solutions tailored to industry-specific requirements.

A picture containing graphical user interface

Description automatically generated

*Figure 4 Display Class showBanner Method*

The implementation follows industry-standard Java best practices, ensuring that the code remains well-structured, readable, and maintainable. In-line comments, proper naming conventions, and consistent formatting and indentation are applied in accordance with established coding standards, enhancing overall code organization and clarity. The program adheres to industry-defined formatting guidelines, maintaining uniform indentation, structured line breaks, and a clearly documented, maintainable commenting style. The source code is structured consistently, allowing for seamless modifications and updates without sacrificing readability or functionality.

Appropriate syntax and conventions are utilized to align with programming best practices. Data structures are designed to optimize efficiency, allowing stored variable values to be effectively used across various methods and classes. Method names are verb-based to accurately represent the actions being performed. Logical control structures are implemented using IF-ELSEIF or CASE blocks, ensuring that all conditions are covered, including ELSE or DEFAULT clauses. Loops are designed to prevent unnecessary manipulation of index variables and to avoid reliance on them after loop termination, improving code reliability and reducing the risk of unintended behavior. These structured coding principles contribute to the development of a robust, efficient, and industry-compliant software solution.

Text

Description automatically generated

*Figure 5 Authenticate Class While Loop If-Else Example*

A significant challenge arises from the classification of methods and classes, particularly in determining their proper structure and location when imported into the program. Errors often stem from this classification process, requiring careful organization to ensure seamless integration. To address these challenges, the program was designed to handle errors based on whether it is executed in the NetBeans output shell or an OS terminal shell/bash. To refine the program and achieve the desired functionality, various code structures were explored to incorporate a simple yet visually appealing graphical user interface (GUI). The implementation focused on improving the program's presentation while maintaining simplicity and efficiency.

The Jansi 2.1.0 API Java library was utilized to enhance the visual appeal of the dashboard screens by integrating ANSI color formatting. This addition improves the user experience by providing a more structured and visually distinct interface, aligning with best practices in software presentation and usability.

Text

Description automatically generated

*Figure 6 Display Class clearScreen Method*

The program design incorporates dynamic screen transitions based on menu selections, ensuring that each option displays relevant content separately rather than presenting everything on a single screen. To achieve this functionality, a code block was implemented to detect the operating system on which the program is running, allowing for appropriate screen-clearing commands tailored to different environments. The process of working with file streaming has been particularly engaging, requiring careful consideration of program enhancements with a focus on precision and efficiency. Significant improvements have been made to the program’s presentation by refining its structure, simplifying code organization, and optimizing the arrangement of classes and methods.

The final implementation extends beyond a basic input/output exercise, resulting in a functional and adaptable program. The development process involved extensive research into techniques used in other programming languages, adapting them for Java while ensuring cross-platform compatibility. The program is designed to run on multiple operating systems, including Windows and macOS, demonstrating its versatility and adherence to software development best practices.

## References

Southern New Hampshire University. (2024). *Milestone Three Guidelines and Rubric Enhancement Two: Algorithms and Data Structure* . Retrieved from Module 4-2

Milestone Three: Enhancement Two: Algorithms and Data Structure