 

Books management system

using Linked List Structure

Data Structure Course

Academic Year (2023-2024) – Second Semester

**Team Members:**

Zahra Alshowiheen -2210003655

Aisha Alshayie - 2210003560

Hoor Alsawaf - 2210003647

Batool al Salim - 2210003514

Fatimah Al Jaffal - 2210003646

**Supervised by:** Dr. Enas El-Sharawy

## TABLE OF CONTENT

1. [Introduction 2](#_TOC_250004)
2. [Challenges 2](#_TOC_250003)
3. [Solution 2](#_TOC_250002)
4. Algorithms& Explanation 3
5. [conclusion 5](#_TOC_250000)

## INTRODUCTION

The Books Management System is a software solution developed to streamline the organization and management of books using a circular linked list data structure. This case study offers an in-depth analysis of the system's development process, focusing on its functionalities, challenges encountered during implementation, the solution devised to address these challenges, algorithms employed, and the resultant outcomes achieved.

## CHALLENGES

Throughout the development phase of the Books Management System, several challenges were identified and addressed:

* **User Interface Design**: Designing an intuitive and user-friendly interface that allows users to interact seamlessly with the circular linked list operations posed a significant challenge. Ensuring that users can easily add, remove, and display books requires careful consideration of interface elements and user experience.
* **Algorithm Implementation**: Implementing robust algorithms for adding, removing, and displaying books within the circular linked list was another challenge. These algorithms are needed to efficiently handle various operations while maintaining the integrity of the data structure.
* **Rotation Optimization**: Ensuring efficient rotation functionality to optimize book access within the circular linked list posed a technical challenge. The rotation operation needed to be implemented in a manner that minimizes computational complexity and facilitates seamless access to books.

## SOLUTION

To overcome the challenges encountered during development, a Java-based GUI application was developed as the solution. The application leverages the CircularLinkedList class to manage books effectively. Key components of the solution include:

* **Add Book**: Enables users to add books to the circular linked list by entering the book title via a user-friendly interface.
* **Remove Book**: Provides users with the ability to remove books from the list based on their titles, enhancing the system's flexibility and usability.
* **Display Books**: Offers a clear and concise display of the list of books currently stored in the system, allowing users to visualize and manage their collection effortlessly.
* **Rotate**: Implements a rotation functionality that reorders the circular linked list, optimizing book access and enhancing system performance.

1. ALGORITHMS & EXPLANATION

# The CircularLinkedList class encapsulates the core algorithms and functionalities of the system. Key algorithms include:

Node class:

This inner class represents a node in the circular linked list.

It contains two fields: value, representing the data stored in the node, and next, representing the reference to the next node in the list.

CircularLinkedList class:

This class represents the circular linked list data structure.

getSize() method:

Returns the size of the circular linked list.

Algorithm: Simply returns the value of the size variable.

first() method:

Returns the value of the first node in the list.

Algorithm: Returns the value field of the head node.

last() method:

Returns the value of the last node in the list.

Algorithm: Returns the value field of the tail node.

isEmpty() method:

Checks if the circular linked list is empty.

Algorithm: Checks if the size variable is equal to 0.

display() method:

Returns a string representation of the circular linked list.

Algorithm: Iterates through each node in the list, appending its value to a StringBuilder along with a counter for display purposes.

findNode(E key) method:

Finds the position of a node with the given value in the list.

Algorithm: Iterates through each node in the list, comparing its value with the given key. If found, returns the position of the node; otherwise, returns a message indicating that the value was not found.

removeFirst() method:

Removes the first node from the list.

Algorithm: Updates the head reference to point to the next node in the list, updates the tail reference if necessary, and decrements the size variable.

addFirst(E value) method:

Adds a new node with the given value to the beginning of the list.

Algorithm: Creates a new node with the given value and sets its next reference to the current head. Updates the head reference to the new node, updates the tail reference if necessary, and increments the size variable.

addLast(E value) method:

Adds a new node with the given value to the end of the list.

Algorithm: Creates a new node with the given value and sets its next reference to null. If the list is empty, sets both head and tail references to the new node. Otherwise, sets the next reference of the current tail node to the new node, updates the tail reference to the new node, and increments the size variable.

removeNode(E key) method:

Removes the node with the given value from the list.

Algorithm: Handles special cases if the node to remove is the first node. Otherwise, iterates through the list to find the node with the given value and removes it by updating the next reference of the previous node. Decrements the size variable accordingly.

rotate() method:

Rotates the list by moving the head to the next node.

Algorithm: Updates the head reference to the next node in the list, effectively rotating the list. If the list is empty, no rotation occurs.

## CONCLUSION

In conclusion, the Books Management System provides a practical and efficient solution for organizing and managing books using a circular linked list data structure. By addressing initial challenges through robust implementation and intuitive design, the system offers users a seamless experience for book management. Further enhancements and optimizations can be explored to improve the system's functionality and user satisfaction, ensuring its continued relevance and effectiveness in book management scenarios.