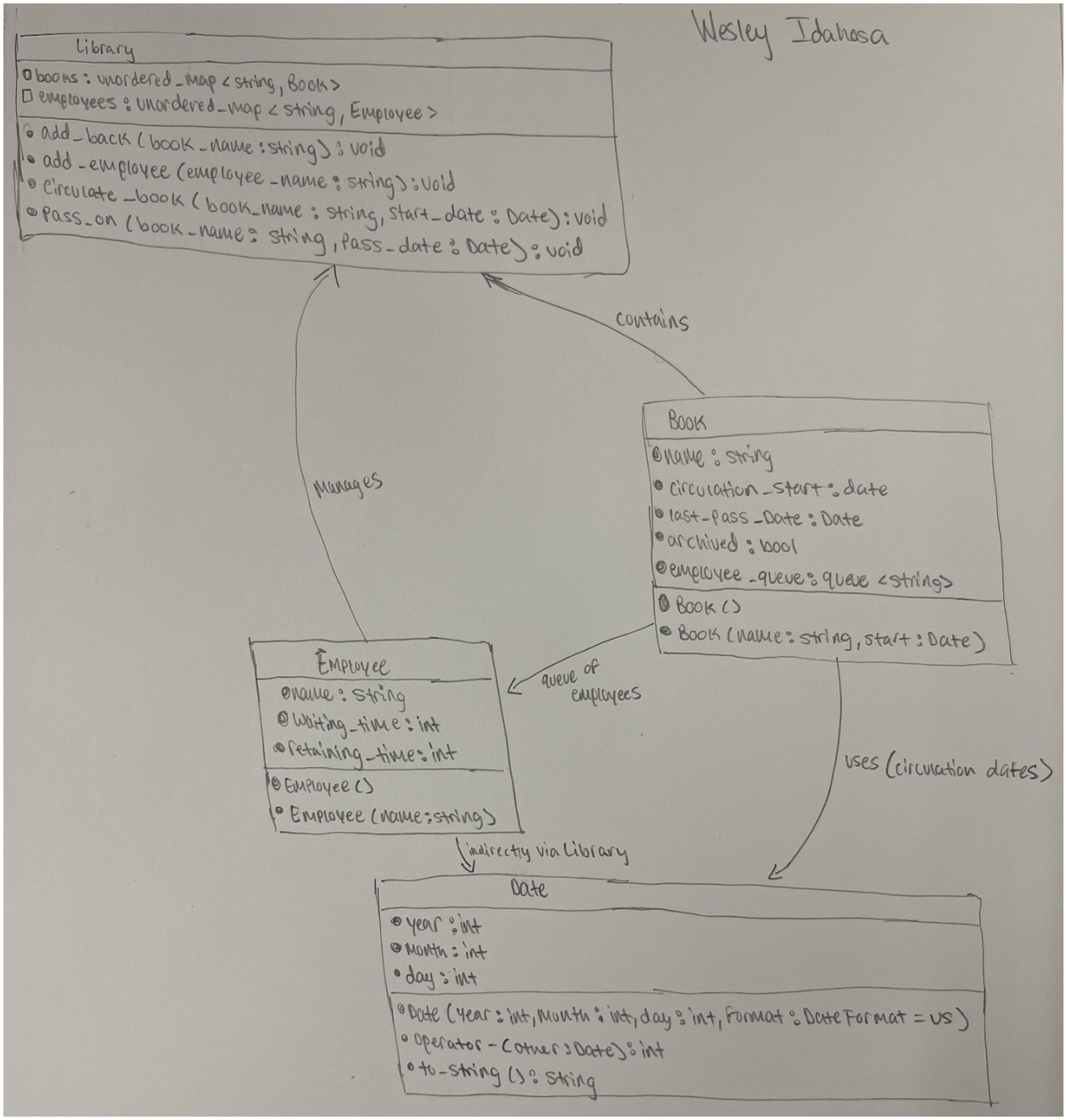
Wesley Idahosa
May 8 2025
Project Report - Library Book Circulation Systems

Assumptions

- All book names and employee names are unique identifiers. No two books or employees will have the same name
- The circulation process begins only after a book has been added and employees are already in the system
- Dates are always valid and in proper chronological order. There is no need for error checking for invalid or reverse dates
- Once a book is archived, it will no longer be part of the active circulation process
- Employees pass the book only on the specified dates provided and do not keep the book longer than instructed

These assumptions helped simplify logic and avoid edge case handling that wasn't mentioned in the project description

UML Class Diagram



Efficiency of Algorithms

1. add_book(string book_name)
    a. Time complexity: O(1) Insertion into an unordered_map is average O(1))
    b. Very efficient due to constant-time hash map lookup and insert
2. add_employee(string employee_name)
    a. Time complexity: O(1)
    b. Very efficient due to constant-time hash map lookup and insert

3. circulate_book(string book_name, Date start_date)
    a. Time Complexity: O(n), where n is the number of employees
    b. It loops through all employees and pushes them into the book's queue in priority order. Sorting can take O(n log n), but the current version may not include sorting logic yet
    c. Optimization: If priority sorting were used, the sorting step could be made more efficient using a heap
4. pass_on(string book_name, Date pass_date)
    a. Time complexity: O(1) for popping from the front of the queue
    b. Updating the employee data and printing status is also O(1), assuming date subtraction is constant time

Overall, the program is very efficient, and all major operations are either O(1) or O(n) where necessary. There is little to no redundancy, and performance is suitable for the main idea of the project

Individual Contributions

This was an individual project, where I completed all the design, coding, and testing myself. My contributions include:

1. Designing the structure of the program including choosing to use unordered_map, queue, and Date handling
2. Writing and testing the Book, Employee, and Library classes
3. Debugging errors such as constructor initialization orders and missing includes
4. Adding comments to make the code more readable and organized
5. Writing this report and manually verifying the output to match expected results

This project gave me hands-on experience in object-oriented programming, class relationships, and STL containers like undered_map, queue, and vector

References

- Class lectures and slides - for understanding class designs
- https://www.visual-paradigm.com/guide/uml-unified-modeling-language/uml-class-diagram-tutorial/ - for understanding class diagrams better and how to construct mine
- ChatGPT - For errors I was encountering in my code and how to fix them