Implementation of a Library Functionality Using Priority Queues in C++

Rand Brown

**Assumptions:**

It is important to clarify exactly what you need to understand before you hop into a project, thus it is necessary to make assumptions on what is going to be important in the project. A way to do that is to find and list assumptions that are needed to make the project true and the code efficient. There are a few assumptions that I made before I started this project.

**Employee Identification:**

* Each employee's name is unique and thus is the main identifier for them as a person. This helps define the queue and show where the books are going next.
* Each book title is the identifier for the book. This means that we can see what books are needed where, it also means that there is no multiple copies of books and each one is a single copy in the library.

**Circulation of Books**

* There is only one copy of each and every book. This allows for easier identifiers of who needs what.
* Employees can only have one book at a time. This helps us understand who needs what books at what time.
* The next person in line for the book is based off of the time that they have been waiting. This means that there is no premium or other reasons that you could be prioritised in the queue.

**Priority System:**

* The next person in line is calculated by their waiting time and retaining time. This helps ensure that the next person who gets the book is the next person that deserves to get the book. This ensures equity and justice in the selection process.
* This system ensures also that there is efficiency in the selection process and people who deserve to get the books get them when they should.

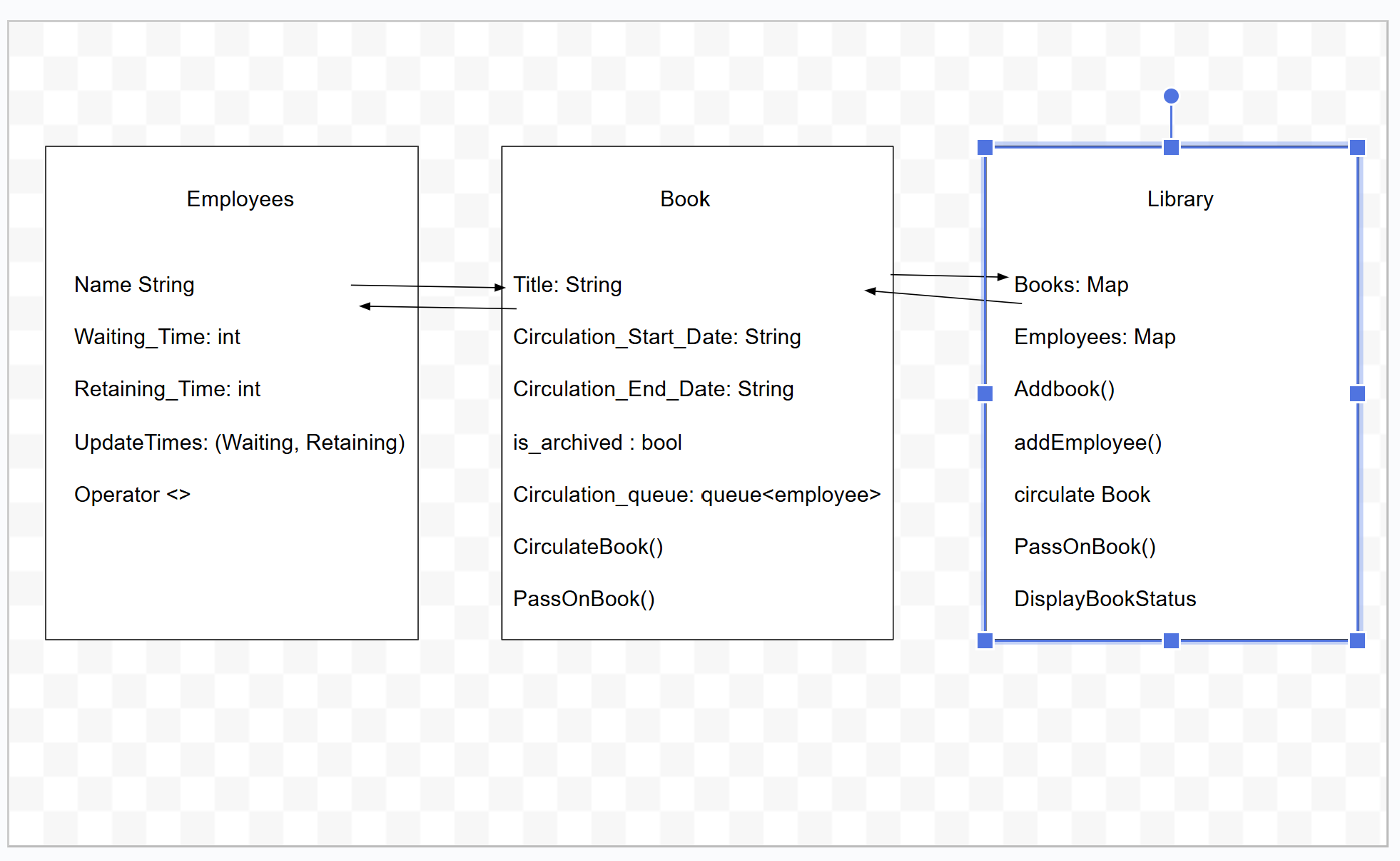
**Queue**:

* Each book has its own queue for the employees waiting for it. This means that the queues are based on the book rather than the employees as this helps keep track of exactly who gets what book and when.

**Library:**

* When an employee returns a book it is instantly passed onto the next employee this helps ensure maximum efficiency in the process.

**UML Class Diagram**



**Class and Relations Explanation:**

**Employee**:

- Represents an employee in the system. The class includes the name, waiting time, and retaining time of the employee.

- It provides an updateTimes method to adjust the times, and an overloaded operator< to prioritize employees in the queue based on their waiting time minus retaining time.

**Book**:

* Represents a book being circulated in the library system. It contains details like the title, circulation dates, archived status, and a queue for employees waiting for the book.
* The circulateBook() method is used to add employees to the book's queue, and passOnBook() is used to pass the book from one employee to the next.

**Library:**

* Manages the books and employees in the system. It allows for adding books and employees, circulating books to employees, passing books between employees, and displaying the status of books and employees.

**Efficiency of Algorithms:**

There are a few algorithms in the program that I feel it is important to examine more. These include the circulateBook, PassOnBook, and Displaying Employee and Displaying book functions.

* The circulateBook algorithm has a time complexity of O(1) because it is just a basic push function that adds an employee to a queue.
* The PassOnBook function has a time complexity of O(1) because all of the things that are achieved by this function are basic functions in a queue that is a time complexity of 0(1).
* Displaying both Book and Employee functions are O(n) as it has a time complexity of however many items that are being displayed.

**Conclusion:**

This report shows the design and implementation of a program that shows queues and sharing of books between employees at an employee library. It makes several assumptions over the situation and even shows the complexity of the major functions that are in the code. It created its own priority queue function to ensure equity in the distribution of the books to employees.