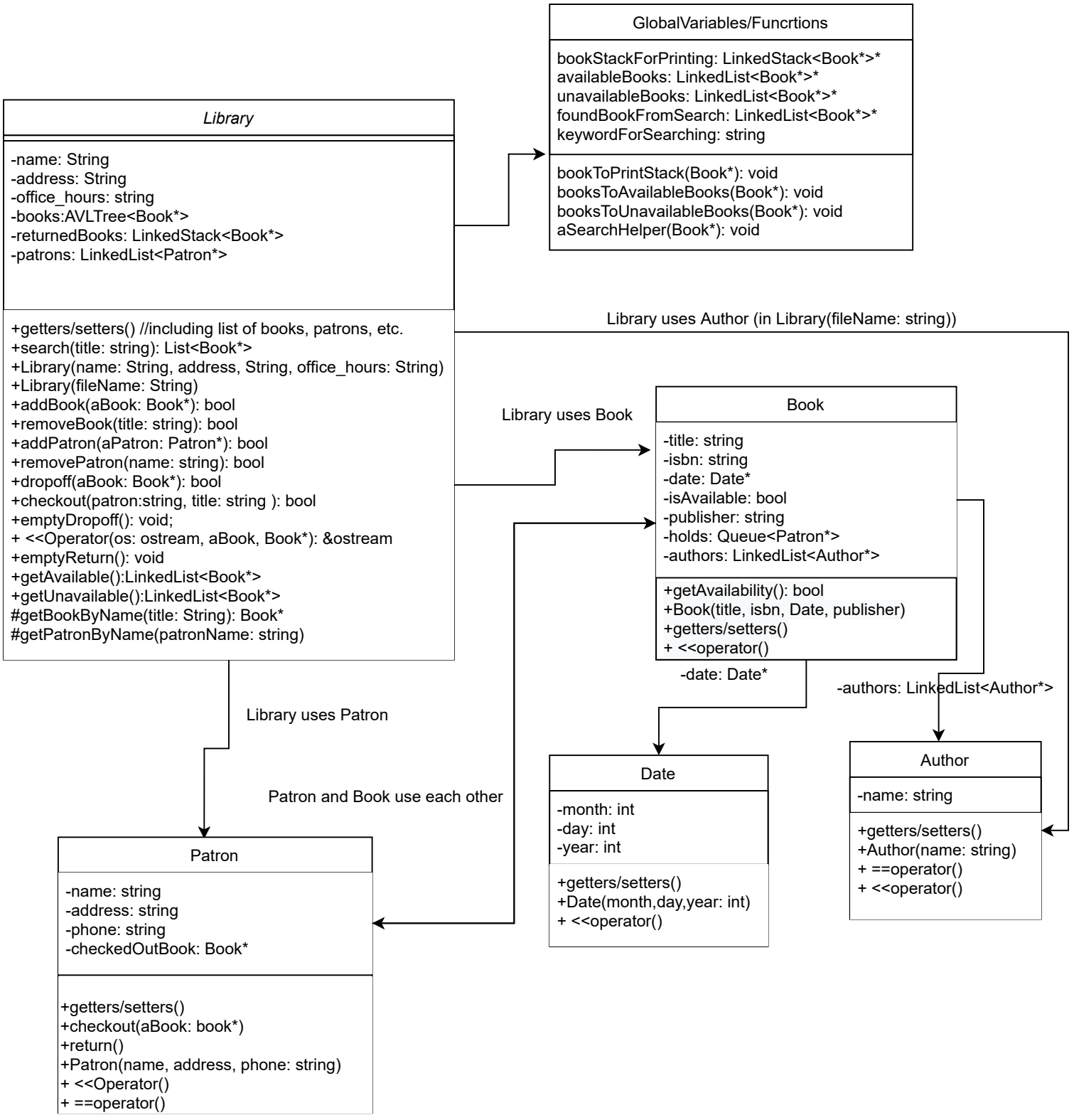
PA UML Design Update and Report



CSCI 2421 PA Design

(changes/additions in red)

Primary Classes:

- *Library*
  - Constructors/getters/setters
  - *search* function will take a string to search the AVL of Books that the Library class will hold and find those that match and are available. Uses global function *aSearchHelper()* and global variable *keywordForSearching* that is passed to the inOrderTraverse function of the Books AVL
  - *addBook* will add a new smart pointer to a Book object that is then added to the AVL of Books
  - *removeBook* will delete the node of the Book corresponding to the string entered, ~~will use *search* to find the book.~~
  - *addPatron* will add a new pointer to a Patron object to the LinkedList of Patrons held by the Library class
  - *removePatron* will remove a Patron from the List
  - *dropoff* will take a Book pointer and add it to the Stack of returned Books, which will then be set at available upon emptying of dropbox.
  - *checkout* will get a Patron using their name (string), and a Book using its title (string). If both are found, it will call the Patron object's *checkout* function, and set the Book to unavailable.
  - *emptyDropoff* empties the returnedBook stack and sets each Book in the stack to available
  - *getAvailable* passes the global function booksToAvailableBooks in the inOrderTraverse to add each available book from the AVLTree of Books to the global variable of availableBooks, and returns it
  - *getUnavailable* passes the global function booksToUnavailableBooks in the inOrderTraverse to add each available book from the AVLTree of Books to the global variable of unavailableBooks, and returns it
  - *getBookByName* looks for a book in the AVL tree of Books using a modified *getEntry()* called *getEntryWithPointerItems*() that handles pointers. Uses a string to search
  - *getPatronByName* looks for a book in the LinkedList of Patrons by iterating through each entry and checking for a string match
  - *addHold* will get a Patron using their name (string), and a Book using its title (string). It will look for a Book that is unavailable that match the title. If both the Book and Patron are found, it will add the Patron to Book's hold Queue
  - Overloaded << operator will print out name of the Library, open/close hours, address, all Books, and all Patrons
- *Book*
  - Constructors/getters/setters
  - *getAvailability()* gets whether the Book is checked out or not
  - Notes: the *holds* Queue saves a queue of Patrons who wish to check out the book. They are added to this queue if they wish to check out the book, but it is not currently available at the Library

- *Patron*
  - Constructors/getters/setters
  - *checkout* checks the available Books at the Library, if it is not available, then they are added to the Book's *holds* queue. If it is available, the Book will be added to the checked out Books stack held by the Patron.
  - *return* will pop the Book from the stack of checked out Books and add

    - it to the returned Books stack at the Library
- *Date*
    - Constructors/getters/setters
    - Notes: Date will be used by Book as an object to store the publish date of the Book
- *Author*
    - Constructors/getters/setters
    - Notes: Author will be used by Book: Book will hold a list of Authors who wrote the book.
- ~~*Time*~~
    - ~~Constructors/getters/setters~~
    - ~~Notes: a simple time object to keep track of whether we are currently within the operating hours of the Library. Will increment upon each~~

Development Plan

1. Kai will develop Book, Patron, Author, and Date classes because they all have close relationships.
2. Matthew will develop the Library and Time classes, since they are exclusive to one another.
3. All will be developed on a GIT repo so we can ensure our classes work together. The final step will be ensuring that all our classes work together perfectly, and implementing the pseudo-simulation in main().

Status of Assignment

1. All expected functions of program are implemented.
2. main() assumes perfect user entry (no strings where ints are expected etc.)
3. A few issues with regex and regex_match cause searching to sometimes be inconsistent.