**Library Class:**

The [Library.java](vscode-file://vscode-app/Applications/Visual%20Studio%20Code.app/Contents/Resources/app/out/vs/code/electron-sandbox/workbench/workbench.html) class is part of a library management system designed to manage library items, authors, and patrons. Below is a brief overview of the methods available in the [Library](vscode-file://vscode-app/Applications/Visual%20Studio%20Code.app/Contents/Resources/app/out/vs/code/electron-sandbox/workbench/workbench.html) class:

* Library()Constructor that initializes lists for library items, authors, and patrons.
* [addLibraryItem(LibraryItem item)](vscode-file://vscode-app/Applications/Visual%20Studio%20Code.app/Contents/Resources/app/out/vs/code/electron-sandbox/workbench/workbench.html" \o "Library.java): Adds a library item to the collection.
* [addAuthor(Author author)](vscode-file://vscode-app/Applications/Visual%20Studio%20Code.app/Contents/Resources/app/out/vs/code/electron-sandbox/workbench/workbench.html): Adds an author to the collection.
* [addPatron(Patron patron)](vscode-file://vscode-app/Applications/Visual%20Studio%20Code.app/Contents/Resources/app/out/vs/code/electron-sandbox/workbench/workbench.html): Adds a patron to the collection.
* [getLibraryItems()](vscode-file://vscode-app/Applications/Visual%20Studio%20Code.app/Contents/Resources/app/out/vs/code/electron-sandbox/workbench/workbench.html): Returns a list of all library items.
* [getAuthors()](vscode-file://vscode-app/Applications/Visual%20Studio%20Code.app/Contents/Resources/app/out/vs/code/electron-sandbox/workbench/workbench.html): Returns a list of all authors.
* [getPatrons()](vscode-file://vscode-app/Applications/Visual%20Studio%20Code.app/Contents/Resources/app/out/vs/code/electron-sandbox/workbench/workbench.html" \o "Library.java): Returns a list of all patrons.
* [checkOut(LibraryItem item, Patron patron)](vscode-file://vscode-app/Applications/Visual%20Studio%20Code.app/Contents/Resources/app/out/vs/code/electron-sandbox/workbench/workbench.html): Checks out a library item to a patron if the item is available.
* [checkIn(LibraryItem item, Patron patron)](vscode-file://vscode-app/Applications/Visual%20Studio%20Code.app/Contents/Resources/app/out/vs/code/electron-sandbox/workbench/workbench.html): Checks in a library item from a patron.
* [getBorrowedItems(Patron patron)](vscode-file://vscode-app/Applications/Visual%20Studio%20Code.app/Contents/Resources/app/out/vs/code/electron-sandbox/workbench/workbench.html): Returns a list of items borrowed by a specific patron.
* [getAvailableItems()](vscode-file://vscode-app/Applications/Visual%20Studio%20Code.app/Contents/Resources/app/out/vs/code/electron-sandbox/workbench/workbench.html): Returns a list of items that are currently available for checkout.
* [getCheckedOutItems()](vscode-file://vscode-app/Applications/Visual%20Studio%20Code.app/Contents/Resources/app/out/vs/code/electron-sandbox/workbench/workbench.html): Returns a list of items that are currently checked out.
* [getBorrowedItems()](vscode-file://vscode-app/Applications/Visual%20Studio%20Code.app/Contents/Resources/app/out/vs/code/electron-sandbox/workbench/workbench.html): Returns a list of all items borrowed across all patrons.
* [getBorrowedItemsByPatron(Patron patron)](vscode-file://vscode-app/Applications/Visual%20Studio%20Code.app/Contents/Resources/app/out/vs/code/electron-sandbox/workbench/workbench.html): Returns a list of items borrowed by a specific patron.

**LibraryItem:**

The [LibraryItem](vscode-file://vscode-app/Applications/Visual%20Studio%20Code.app/Contents/Resources/app/out/vs/code/electron-sandbox/workbench/workbench.html" \o "LibraryItem.java) class is an abstract representation of an item in a library that can be borrowed. It implements the [Borrowable](vscode-file://vscode-app/Applications/Visual%20Studio%20Code.app/Contents/Resources/app/out/vs/code/electron-sandbox/workbench/workbench.html) interface, requiring subclasses to implement methods for borrowing and returning items. The class encapsulates item details such as ID, title, author, ISBN, publisher, total and available copies, and status.

* **Methods**:
  + Constructors, getters, and setters are provided for all fields.
  + [borrowItem(int numberOfCopies)](vscode-file://vscode-app/Applications/Visual%20Studio%20Code.app/Contents/Resources/app/out/vs/code/electron-sandbox/workbench/workbench.html): Abstract method to borrow a specified number of copies of the item.
  + [returnItem(int numberOfCopies)](vscode-file://vscode-app/Applications/Visual%20Studio%20Code.app/Contents/Resources/app/out/vs/code/electron-sandbox/workbench/workbench.html): Abstract method to return a specified number of copies of the item.
  + [toString()](vscode-file://vscode-app/Applications/Visual%20Studio%20Code.app/Contents/Resources/app/out/vs/code/electron-sandbox/workbench/workbench.html): Overridden to provide a string representation of the item.

**Periodical**

* **Periodical**: An abstract class that extends [LibraryItem](vscode-file://vscode-app/Applications/Visual%20Studio%20Code.app/Contents/Resources/app/out/vs/code/electron-sandbox/workbench/workbench.html), representing periodicals in the library such as magazines and newspapers. It requires the implementation of methods to borrow and return items.
* **PrintedPeriodical**: An extention/subclass of **Periodical.**
  + **coverType is an extra component for PrintedPeriodical.**
* **ElectronicPeriodical: An extension/subclass for Periodical.**
  + **fileFormat is an extra component for Periodical.**

**Book:**

* **LibraryItem**: An abstract class representing an item in a library that can be borrowed. Implements the Borrowable interface.
* **Book**: A concrete implementation of [LibraryItem](vscode-file://vscode-app/Applications/Visual%20Studio%20Code.app/Contents/Resources/app/out/vs/code/electron-sandbox/workbench/workbench.html), representing a book in the library.
* **EBook**: An extension/subclass of **Book.**
  + **fileSize and fileFormat are extra components for EBook.**
* **AudioBook: An extension/subclass of Book.**
  + **fileSize and audioFormat are extra components for AudioBook.**
* **PrintedBook: An extension/subclass of Book.**
  + **pageNumber and coverStyle are extra components for PrintedBook.**

**Author:**

**Author**: Represents an author with a name, date of birth, and a list of library items they have written.

**Getters:**

* [getName()](vscode-file://vscode-app/Applications/Visual%20Studio%20Code.app/Contents/Resources/app/out/vs/code/electron-sandbox/workbench/workbench.html): Returns the name of the author.
* [getDob()](vscode-file://vscode-app/Applications/Visual%20Studio%20Code.app/Contents/Resources/app/out/vs/code/electron-sandbox/workbench/workbench.html): Returns the date of birth of the author.
* [getItems()](vscode-file://vscode-app/Applications/Visual%20Studio%20Code.app/Contents/Resources/app/out/vs/code/electron-sandbox/workbench/workbench.html): Returns the list of [LibraryItem](vscode-file://vscode-app/Applications/Visual%20Studio%20Code.app/Contents/Resources/app/out/vs/code/electron-sandbox/workbench/workbench.html" \o "LibraryItem.java)s associated with the author.
* [addWrittenItem(LibraryItem item)](vscode-file://vscode-app/Applications/Visual%20Studio%20Code.app/Contents/Resources/app/out/vs/code/electron-sandbox/workbench/workbench.html" \o "Author.java): Adds a [LibraryItem](vscode-file://vscode-app/Applications/Visual%20Studio%20Code.app/Contents/Resources/app/out/vs/code/electron-sandbox/workbench/workbench.html) to the author's list of items.

**Patron:**

* **Patron**: An abstract class representing a library patron with personal details and a list of borrowed library items. It provides methods to manage these attributes.

**Getters and Setters:**

* [getName()](vscode-file://vscode-app/Applications/Visual%20Studio%20Code.app/Contents/Resources/app/out/vs/code/electron-sandbox/workbench/workbench.html" \o "Patron.java): Returns the patron's name.
* [getPhoneNumber()](vscode-file://vscode-app/Applications/Visual%20Studio%20Code.app/Contents/Resources/app/out/vs/code/electron-sandbox/workbench/workbench.html): Returns the patron's phone number.
* [getAddress()](vscode-file://vscode-app/Applications/Visual%20Studio%20Code.app/Contents/Resources/app/out/vs/code/electron-sandbox/workbench/workbench.html" \o "Patron.java): Returns the patron's address.
* [getBorrowedItems()](vscode-file://vscode-app/Applications/Visual%20Studio%20Code.app/Contents/Resources/app/out/vs/code/electron-sandbox/workbench/workbench.html): Returns the list of library items currently borrowed by the patron.
* [setName(String name)](vscode-file://vscode-app/Applications/Visual%20Studio%20Code.app/Contents/Resources/app/out/vs/code/electron-sandbox/workbench/workbench.html): Sets the patron's name.
* [setPhoneNumber(String phoneNumber)](vscode-file://vscode-app/Applications/Visual%20Studio%20Code.app/Contents/Resources/app/out/vs/code/electron-sandbox/workbench/workbench.html): Sets the patron's phone number.
* [setAddress(String address)](vscode-file://vscode-app/Applications/Visual%20Studio%20Code.app/Contents/Resources/app/out/vs/code/electron-sandbox/workbench/workbench.html): Sets the patron's address.
* [setBorrowedItems(List<LibraryItem> borrowItems)](vscode-file://vscode-app/Applications/Visual%20Studio%20Code.app/Contents/Resources/app/out/vs/code/electron-sandbox/workbench/workbench.html): Sets the list of borrowed library items.

Subclasses for Patron:

* Employee: Has employeeID and inherets from Patron.
* Student: Has studentID and inherits from Patron.

**Introduction:** "Good [morning/afternoon/evening], everyone. Today, I'm excited to introduce our library management system, a comprehensive solution designed to streamline the management of library resources, including books, periodicals, authors, and patrons. This system is built with flexibility and efficiency in mind, catering to the diverse needs of our library's stakeholders."

**Library Class Overview:** "Let's start with the heart of our system, the Library class. This class acts as the central hub, managing the various entities within our library. It allows us to add and remove library items, authors, and patrons. Furthermore, it facilitates the checking in and out of items, ensuring a smooth borrowing process. The Library class also provides functionalities to view available items, items currently checked out, and items borrowed by specific patrons, making it easier to track the circulation of resources."

**LibraryItem and Its Subclasses:** "Moving on to the LibraryItem class, this abstract entity represents any item that can be borrowed from our library. It includes essential details such as ID, title, author, and status. Subclasses of LibraryItem, such as Book, Periodical, EBook, AudioBook, and PrintedBook, extend this functionality to cater to specific types of library materials, each with unique attributes like file size, format, and cover style."

**Author and Patron Entities:** "Our system also emphasizes the importance of authors and patrons. The Author class captures details about the authors of library items, including their name, date of birth, and the items they've written. On the other hand, the Patron class represents our library users, detailing their personal information and the items they've borrowed. This class is further specialized into Employee and Student subclasses, recognizing the diverse roles within our library community."

**Conclusion:** "In conclusion, our library management system is a robust tool designed to enhance the efficiency and effectiveness of library operations. From managing the lifecycle of library items to facilitating user interactions, this system is equipped to handle the dynamic needs of modern libraries. Thank you for your attention, and I look forward to any questions you may have."

**Introduction (Charity)**

**Charity:** "Good [morning/afternoon/evening], everyone. Today, I'm excited to introduce our library management system, a comprehensive solution designed to streamline the management of library resources, including books, periodicals, authors, and patrons. This system is built with flexibility and efficiency in mind, catering to the diverse needs of our library's stakeholders."

**Library Class Overview (Ed)**

**Ed:** "Let's start with the heart of our system, the Library class. This class acts as the central hub, managing the various entities within our library. It allows us to add and remove library items, authors, and patrons. Furthermore, it facilitates the checking in and out of items, ensuring a smooth borrowing process. The Library class also provides functionalities to view available items, items currently checked out, and items borrowed by specific patrons, making it easier to track the circulation of resources."

**LibraryItem and Its Subclasses (Charity)**

**Charity:** "Moving on to the LibraryItem class, this abstract entity represents any item that can be borrowed from our library. It includes essential details such as ID, title, author, and status. Subclasses of LibraryItem, such as Book, Periodical, EBook, AudioBook, and PrintedBook, extend this functionality to cater to specific types of library materials, each with unique attributes like file size, format, and cover style."

**Author and Patron Entities (Ed)**

**Ed:** "Our system also emphasizes the importance of authors and patrons. The Author class captures details about the authors of library items, including their name, date of birth, and the items they've written. On the other hand, the Patron class represents our library users, detailing their personal information and the items they've borrowed. This class is further specialized into Employee and Student subclasses, recognizing the diverse roles within our library community."

**Quick Demo (Dawson)**

**Dawson:** "Now, let's see a quick demo of our system using the LibraryMenu class. We'll walk you through adding a new library item."

**Dawson:** "First, we select the option to add a library item. Let's add a PrintedBook. We enter the item details such as ID, title, author name, date of birth, ISBN, publisher, total and available copies, page number, and cover style."

**Dawson:** “Now that we added our first item let view to make sure it worked.”

**Dawson:** “Now that we can see the item, we can add a Patron which can be a Student or an Employee.

**Dawson:** "After entering these details, the system creates the item and adds it to the library. Now, let's check out this item to a patron. We select the option to borrow an item, enter the item ID and the patron's ID. The system checks out the item and updates its status."

**Dawson:** "Next, we can return the item by selecting the return item option, entering the item ID and the patron's ID again. The system checks the item back in and updates its status to available."

**Conclusion**

**Ed:** "In conclusion, our library management system is a robust tool designed to enhance the efficiency and effectiveness of library operations. From managing the lifecycle of library items to facilitating user interactions, this system is equipped to handle the dynamic needs of modern libraries. Thank you for your attention, and we look forward to any questions you may have."

**JavaScript Script:**

Hello! This is our presentation of our Full Stack JavaScript project for the midterm sprint

Our Team is comprised of Charity Smith, Dawson Bursey and Ed Spurrell.

I will begin with talking about our token.js file:

This Node.js script manages tokens, with functions for creating, counting, updating, and displaying tokens, and logs these actions. Here’s a simplified overview:

1. **Dependencies: Imports necessary modules for file operations, date formatting, CRC32 hashing, and event handling.**

1. **Event Emitter: Creates an event emitter (myEmitter) to log actions.**

1. **Main Functions:**
   * **tokenCount: Counts and logs the number of tokens.**
   * **tokenList: Displays and logs the list of tokens.**
   * **newToken: Creates a new token for a user, updates the token file, and logs the creation.**
   * **updateToken: Updates token information (email or phone) and logs the update.**
   * **fetchRecord: Displays a token for a given user and logs the action.**
   * **searchToken: Searches and logs tokens by username, email, or phone.**

1. **Utility Functions:**
   * **addDays: Adds days to a date.**
   * **handleNewTokenResult: Handles new token creation results in a web context.**
   * **generateToken: Generates a new token from a web form.**

1. **CLI Handling:**
   * **tokenApp: Handles command-line arguments to call the appropriate function (e.g., count, list, new, update, fetch, search).**

1. **Exports: Makes the main functions available for use in other parts of the application.**

This script logs all token-related activities and supports both CLI and web-based operations.

I will now pass it over to ED to speak on our server file.

**(Edward)**

**(Dawson)**

MYAPP>JS FILE: This JavaScript code uses Node.js to handle different command-line arguments and perform specific tasks based on those arguments. Here's a brief summary of what it does:

1. **Imports Required Modules:**
   * **fs (File System) for reading files.**
   * **initializeApp from init.js for initializing the app.**
   * **configApp from config.js for configuring the app.**
   * **tokenApp and app from token.js for generating a token.**
2. **Command-Line Arguments:**
   * **Retrieves command-line arguments excluding the first two (node and script name) and stores them in myArgs.**
3. **Global Debug Mode:**
   * **Sets a global DEBUG flag to true.**
4. **Debug Logging:**
   * **If in debug mode and more than one argument is passed, logs the arguments.**
5. **Argument Handling:**
   * **Uses a switch statement to handle different first arguments:**
     + **init or i: Initializes the app by calling initializeApp().**
     + **config or c: Configures the app by calling configApp().**
     + **token or t: Generates a token by calling tokenApp().**
     + **--help or --h or any other value: Reads and prints the content of usage.txt for help instructions.**
6. **File Reading:**
   * **Reads and displays the content of usage.txt if the argument is not one of the specified commands. If there's an error reading the file, it throws an error.**

The code's primary function is to handle initialization, configuration, token generation, and help display based on command-line input.

And over to Charity to discuss the init.js file

**(Charity)**

INIT.js The init.js script  sets up a Node.js application using Express and provides command-line argument handling for creating folder structures and configuration files. This code includes the following:

1. **Module Imports:**
   * **fs and path for file system operations.**
   * **fsPromises for promise-based file system operations.**
   * **yargs for command-line argument parsing.**
   * **express for setting up a web server.**

1. **Global Variables:**
   * **app as an instance of Express.**
   * **port for the server port (3000).**
   * **folders, configjson, and tokenjson templates from a local module.**
   * **DEBUG flag set to  true for debug mode.**

1. **Command-Line Arguments Setup:**
   * **Defines command-line options (--all, --mk, --cat) using yargs.**

1. **Log Function:**
   * **log(level, message) to log messages with timestamps.**

1. **Folder Creation Function:**
   * **createFolders() to create directories defined in folders.**
   * **Logs creation attempts and results.**

1. **File Creation Function:**
   * **createFiles() to create config.json and tokens.json files if they don't exist.**
   * **Logs file creation attempts and results.**

1. **Initialization Function:**
   * **initializeApp() handles command-line arguments to decide which operations to perform:**
     + **--all creates both folders and files.**
     + **--cat creates only files.**
     + **--mk creates only folders.**
     + **--help or any other argument displays usage instructions.**
   * **Sets up an Express route to serve an index.html file.**
   * **Starts the Express server on the specified port.**

1. **Export:**
   * **Exports the initializeApp function for external use.**

The code is designed to initialize the app by setting up necessary directories and configuration files based on command-line input, and it also starts a basic web server using Express.

And back to Ed to discuss the Usage.txt file

**(Edward)**