**Library Management System Documentation**

Presented by:

Dai Ahmed Tag

Maged Aziz

Tasneem Eid

Shahd Mahmoud

User Class:

Functionality:

This class represents a user in a system. It stores basic information about the user, including their name, ID, and contact information.

Methods:

* Getters and Setters:
  + getName(): Returns the user's name.
  + setName(String name): Sets the user's name.
  + getId(): Returns the user's unique ID.
  + setId(int id): Sets the user's unique ID. (Note: Modifying the ID might not be a good practice depending on the system.)
  + getContactInformation(): Returns the user's contact information.
  + setContactInformation(String contactInformation): Sets the user's contact information.

Book Class:

This class defines the structure and behavior of individual books within the library system:

Member Variables:

* title: String containing the book's title.
* author: String containing the book's author.
* ISBN: String representing the book's unique identification number.
* genre: String indicating the book's genre.
* available: Boolean flag indicating whether the book is currently available for borrowing (true) or borrowed (false).
* borrowedBy: Reference to the User object who currently has borrowed the book (null if available).

Methods:

* Getters and Setters:
  + getTitle(), setTitle(String title): Access and modify the book's title.
  + getAuthor(), setAuthor(String author): Access and modify the book's author.
  + getISBN(), setISBN(String ISBN): Access and modify the book's ISBN.
  + getGenre(), setGenre(String genre): Access and modify the book's genre.
  + isAvailable(): Returns a boolean indicating whether the book is currently available.
  + getBorrowedBy(): Returns a reference to the User object who borrowed the book (null if available).
  + setBorrowedBy(User borrowedBy): Assigns the book to a borrowing user and updates the borrowedBy reference.
  + setAvailability(boolean availability): Modifies the book's availability flag (available) to true or false.

Behavior:

* The Book class provides basic functionalities for managing book information, including title, author, ISBN, genre, and availability status.
* It allows setting the borrowing user and updating the availability status accordingly.
* It interacts with the Library class for notifying observers about changes in availability.

## BookAvilabilityObserver interface:

## his interface defines a contract for objects that are interested in receiving updates about changes in book availability within a library system. In other words, any class that implements this interface will be notified whenever the availability status of a book changes .

## Library class:

## The library class provides a comprehensive set of functionalities for managing books, users, and their interactions within the library system. The observer pattern effectively keeps interested parties informed about book availability changes, while the singleton pattern ensures centralized data management.

Variables:

* books: A list containing all available books in the library.
* users: A list containing all registered users.
* observers: A list of objects implementing
* the BookAvailabilityObserver interface: which will be notified whenever a book's availability changes.
* singleInstance: Singleton instance of Library class to ensure only one library object exists throughout the program.

Methods:

* addObserver(BookAvailabilityObserver observer): Adds a new observer to the observers list.
* removeObserver(BookAvailabilityObserver observer): Removes a specified observer from the observers list.
* notifyObservers(Book book): Notifies all registered observers about a change in the availability of the provided book.
* addBook(Book book): Adds a new book to the books list and calls notifyObservers to inform observers about the addition.
* deleteBook(Book book): Removes the specified book from the books list and calls notifyObservers to inform observers about the removal.
* getAllBooks(): Returns a copy of the books list for read-only access.
* borrowBook(Book book, User user): Checks if the book is available. If yes, sets its availability to false, assigns the borrowing user to book, and calls notifyObservers.
* returnBook(Book book, User user): Verifies if the user matches the book's borrowing information. If yes, sets the book's availability to true, and calls notifyObservers.
* editBook(Book book): Updates the information of the provided book and calls notifyObservers to inform observers about the change.
* addUser(User user): Adds a new user to the users list.

Front end Class:

This class represents an front end and input validation to the library system.

1. Managing Book Information:

* + administrator: Stores a reference to the administrator object representing the entire administrator system.

Methods:

addBook() :- The method prompts the user for information about the book, including title, author, ISBN, and genre.

Each input is validated for various conditions like emptiness, special characters, and specific requirements like ISBN containing numbers.

If any validation fails, an error message is displayed and the user is prompted to re-enter the information.

Upon successful validation, a new Book object is created with the provided information.

This object is then added to the library system using the administrator.addBook method.

a success message is displayed and the list of all books is shown.

The validation logic seems thorough, ensuring accurate and consistent book data.

The reRunAdmin method is called on some error conditions, hinting at a possible loop structure for re-attempting book addition.

deleteBook():- Prompts the user for the ISBN of the book to delete.

Uses findBookByISBN to search for the book in the library.

If the book is found, it is deleted using administrator.deleteBook and notifications are sent to observers using administrator.notifyObservers.

A success message is displayed and the updated list of books is shown.

If the book is not found, an error message is displayed.

The validation logic seems thorough, ensuring accurate and consistent book data.

The reRunAdmin method is called on some error conditions, hinting at a possible loop structure for re-attempting book addition.

borrowBook(): Checking Book Availability:

It first checks if there are any books in the library using administrator.getAllBooks().isEmpty(). If not, it displays an error message and restarts the administrator interface (reRunAdmin).

Book Search:

If books are available, it prompts the user for the book title using JOptionPane.showInputDialog.

The provided title is searched for using FindBookByTitle and the retrieved book object is stored in bookToBorrow.

Book Availability and Borrowing:

If the book is found and available (bookToBorrow and bookToBorrow.isAvailable() are true), it proceeds to user information collection.

User Information Collection:

It prompts the user for their name (userName), ID (userId), and contact number (userContact).

Each input is validated for emptiness, special characters, and specific requirements like user name potentially containing numbers and phone number having 11 digits (validateElevenDigits function might be used).

If any validation fails, an error message is displayed and the user is prompted to re-enter the information (reRunUser).

Borrowing Confirmation and Completion:

Upon successful validation, a User object is created with the provided information.

Finally, the book is borrowed by calling administrator.borrowBook with bookToBorrow and the newly created User object.

A success message is displayed and the updated list of books is shown using displayAllBooks.

* ReturnBook():This method allows users to return a borrowed book.

It first checks if there are any books in the library using administrator.getAllBooks().isEmpty(). If not, it displays an error message and restarts the administrator interface (reRunAdmin).

Then, it prompts the user for the title of the book they want to return using JOptionPane.showInputDialog.

The provided title is validated for various conditions like emptiness, special characters, and presence of numbers. Additionally, it checks if the book is already available (meaning not borrowed). If any validation fails, an error message is displayed and the user is asked to re-enter the information (reRunUser).

If validation passes and the book is found to be borrowed, it prompts the user for their name, ID, and contact number. Similar validation checks are performed on each user input.

Upon successful user information validation, a User object is created and the book is returned using administrator.returnBook.

Finally, a success message is displayed and the updated list of books is shown using displayAllBooks

.

* editBook():This method allows editing the details of a book in the library.

It prompts the user for the ISBN of the book they want to edit using JOptionPane.showInputDialog.

The provided ISBN is validated for emptiness, special characters, and presence of numbers. If any validation fails, an error message is displayed and the user is asked to re-enter the information (reRunAdmin).

The book object is retrieved using findBookByISBN. If the book is not found, an error message is displayed and the user is asked to re-enter the information.

If the book is found, it prompts the user for new values for title, author, ISBN, and genre. Each input is validated for specific conditions like emptiness, special characters, and presence of numbers for author and ISBN.

If any validation fails for the new values, an error message is displayed and the user is asked to re-enter the information.

Upon successful validation of all new values, the corresponding fields in the Book object are updated using setter methods. Additionally, the book availability is set to true as it's being edited and potentially made available again.

A success message is displayed and the updated list of books is shown using displayAllBooks.

* displayAllBooks(List<Book> books ):This method displays a formatted list of all books in the library using JOptionPane.showMessageDialog.

It iterates through a list of Book objects and constructs a string containing information for each book (title, author, ISBN, genre, availability).

The string is then displayed in a dialog box with appropriate formatting

## Searching methods:

* findBookByISBN(String ISBN) : This method searches for a book based on its ISBN number.

It ensures the provided ISBN is not null, empty, or blank, and also verifies the absence of special characters using hasNoSpecialCharacters function.

If the ISBN meets the criteria, it iterates through all books in the library using a loop.

Within the loop, it compares the ISBN of each book with the provided ISBN using equals method.

If a match is found, the corresponding Book object is returned.

If no match is found, the method returns null.

* FindBookByTitle(String title): This method searches for a book based on its title.

It follows a similar structure to findBookByISBN, performing validation on the provided title and then iterating through all books.

Instead of comparing ISBNs, it compares the title of each book with the provided title using equals method.

If a matching title is found, the corresponding Book object is returned.

Otherwise, the method returns null.

* public repeatedISBN(String ISBN):- This function checks if a provided ISBN number is already present in the library.

It iterates through all books using administrator.getAllBooks() and retrieves each book object.

For each book, it compares its ISBN with the provided ISBN using equals method.

If a matching ISBN is found, it returns true indicating the ISBN is already used.

Otherwise, it continues iterating until all books are checked, and if no match is found, it returns false.

* containsNumbers:

This function checks if a string contains any digits.

It iterates through each character in the string and checks if it's a digit using Character.isDigit(c).

If any digit is found, it returns false indicating the string contains numbers.

Otherwise, it returns true meaning the string only has letters or special characters.

* containsCharacters:

This function is similar to containsNumbers, but it checks if a string contains any letters.

It iterates through each character and checks if it's a letter using Character.isLetter(c).

If any letter is found, it returns false indicating the string contains characters.

Otherwise, it returns true meaning the string only has digits or special characters.

* hasNoSpecialCharacters:

This function checks if a string contains only letters, digits, or whitespace characters.

It iterates through each character and checks if it's a letter, digit, or whitespace using Character.isLetterOrDigit(c) and Character.isWhitespace(c).

If any special character is found (punctuation, symbols, etc.), it returns false.

Otherwise, it returns true meaning the string only has valid characters.

* validateElevenDigits:

This function simply checks if a string has exactly 11 characters.

It verifies if the string is not null and its length is equal to 11.

If both conditions are met, it returns true, otherwise false

* Application flow control methods:
* run function:

This function serves as the main entry point for the system, presenting the initial user/administrator selection menu.

It uses JOptionPane.showInputDialog to display options and capture the user's choice.

Based on the selected option ("User" or "Administrator"), it navigates to the corresponding user interface (reRunUser or reRunAdmin functions).

* reRunUser and reRunAdmin functions:

These functions handle the respective user and administrator interfaces with sub-menus for specific actions like borrowing, returning, or managing books.

They follow a similar structure to run, displaying menus and reacting to user choices using JOptionPane.

Each function calls the corresponding method for the chosen action and then continues looping through the sub-menu until the user exits.