

# Library Management System

## Mini Project

In this project, you have developed a console-based Library Management System by employing procedural programming (i.e., a set of functions and function calls). In this mini project, you are going to re-implement the same console-based Library Management System by using object-oriented programming practices! The interface from the user's perspective will stay the same. However, you will have a significantly different implementation in the source code level. The primary goal of this mini project is to practice using classes and objects. Hence, please use every opportunity to employ object-oriented programming concepts in your code. **At minimum**, you should create and use the following classes implemented with the below provided list of attributes and methods. If you see a need, you may add additional classes and/or methods/attributes that are not in the following list.

### List of Classes with their Attributes and Methods:

- **MenuItem**

- Attributes:

- **text**: Stores the text of the menu item, e.g., “Add a book to my book list.”
    - **number**: Stores the number of this menu item, e.g., 2 for the above item in student menu, which will be used when printing the menu and getting user selection

- Methods:

- **display()**: This method returns the current MenuItem object’s number and text properly when called.

- **Menu**

- Attributes:
  - **header:** header represents the introductory text that usually appears before the menu options, e.g., “Welcome to Admin Menu! ”
  - **menuItems:** list of MenuItem objects - Each option in a menu will be represented as a MenuItem object (see the previous class) and stored in a list.
- Methods:
  - **display(display\_header):** prints the menu on the screen, It will display the header attribute before menu items if display\_header parameter is True. Otherwise, it will not show the header.
  - **add\_menu\_item(text, number):** It should build and add a new MenuItem object to the list of “menuItems” attribute

- **User**

- Attributes:
  - **menu:** is an object of Menu class with an empty string “” as a parameter for the header. This attribute should be a class level attribute.
  - **menu\_items:** is an empty list will be used to store the proper menu items. This attribute should be a class level attribute.
  - **name:** will be used to stores the name of the user
  - **Password:** will be used to stores the password of the user
- Methods:
  - **display():** This method returns the current user’s object info (name and password).
  - **menu\_builder():** This method will be called from the init() method of Student class and Admin class by using inheritance, it should build the admin menu and student menu with all of its options by using the class level attribute menu which is an instance of Menu class.

- **Admin**

This class should be inherited from User class

- Attributes:

- **menu:** override the menu attribute to update the header parameter to “Welcome to admin menu”. This attribute should be a class level attribute.
    - **menu\_items:** override the menu\_items attribute to update the content of the list. (check the “Required Data Structures” part at the end of the manual to find the list).
    - **Role:** should be always equal to “admin”.

- **Student**

This class should be inherited

- Attributes:

- **menu\_flag:** will be used to check whether the menu has been created before or not. By default it should be False as a boolean value. This attribute should be a class level attribute.
    - **menu:** override the menu attribute to update the header parameter to “Welcome to student menu”. This attribute should be a class level attribute.
    - **menu\_items:** override the menu\_items attribute to update the content of the list. (check the “Required Data Structures” part at the end of the manual to find the list)
    - **role:** should be always equal to “student”.

- **UserDB**

- Attributes:

- **example\_users:** store the information of the users (check the “Required Data Structures” part at the end of the manual to find the dict). This attribute should be a class level attribute.
- **users\_objects:** dict of User objects where key would be user name, value would be the corresponding User object. This attribute should be a class level attribute.
- **example\_users\_flag:** this attribute will determine whether to build the example users or not. By default it should be True as a boolean value.

- Methods:

- **create\_example\_users():** This method will be called from the init() method of this class according to the flag. It should build the example users by using the add\_user method.
- **add\_user(name, password, role):** The objects of Student class and Admin class should be created in this method and added to users\_objects dict.
- **remove\_user():** It should handle deleting a user.
- **list\_user():** lists all the student users.
- **validate\_user(uid, password):** validate the user credentials by checking the username and the password. If true, It should return the object of the user. Else, it should return False.

- **Book**

- Attributes:

- **bid:** stores the book id
- **name:** stores the name of the book
- **no\_of\_copies:** stores the number of the copies of the book
- **list\_of\_authors:** stores the a list of authors names of the book

- Methods:

- **display():** This method returns the current book's object info.

- **Library**

- Attributes:

- **example\_books:** store the information of the example books (check the “Required Data Structures” part at the end of the manual to find the dict). This attribute should be a class level attribute.
- **author\_to\_books:** class level attribute dict of author names where key would be the author name, value would be a list including the corresponding book object.
- **book\_objects:** class level attribute dict of Book objects where key would be the id of the book, value would be the corresponding Book object.
- **example\_books\_flag:** this attribute will determine whether to build the example books or not. By default it should be True as a boolean value.

- Methods:

- **create\_example\_books():** This method will be called from the init() method of this class according to the flag. It should build the example users by using the add\_book method.
- **add\_book(bid, name, copies, authors):** The objects of Book class should be created in this method and added to book\_objects dict and author\_to\_books dict.
- **remove\_book():** It should handle deleting a book, it should delete the given book and remove it from authors\_to\_books dict.
- **list\_book():** list all the books in the library
- **search\_book():** This method searches the library for a specific book by the book name or the author name.

- **update\_book\_copies()**: This method updates the number of copies of a specific book after checking the number of the students holding the book

- **Main**

- Attributes:

- **library**: a Library object with flag True
    - **userDB**: a UserDB object with flag True
    - **current\_user**: will hold the current\_user object. By default is None

- Methods:

- **login()**: This is the central method that runs the primary logic of the system. It will call methods of other objects whenever required and handle user credentials.
    - **show\_admin\_menu()**: This method will control the main flow of the admin menu when the user signs in as admin.
    - **show\_student\_menu()**: This method will control the main flow of the student menu when the user signs in as a student.

## Required Data Structures:

- **menu\_items for Admin class**
  - `menu_items = ["List books", "Create a book", "Delete a book", "Search for a book", "Change number of a copies of a book", "Show students borrowed a book by ""ID", "List Users", "Create User", "Delete User", "Exit"]`
- **menu\_items for Student class**
  - `menu_items = ["Search for a book", "Add a book to my book list", "delete a book from my book list", "show my borrowed books", "Exit"]`
- **example\_users for UserDB class**
  - `example_users = {'Ahmet': ['1234', 'student'], 'Ayse': ['4321', 'student'], 'admin': ['0000', 'admin']}`
- **example\_books for Library class**
  - `example_books = {"001": ["Biology", 2, ["Alice", "Bob"]], "002": ["Chemistry", 3, ["Alice"]]}`

## Implementation Notes:

- You must use the given template.
- You should fill the attributes and the methods of the given classes.
- “TODO” statements in the template will help you for implementation.
- To have a clear understanding please use the attached executable file to test the project and check how it works
- **Outside of class definitions, only 2 lines of code should exist** that do the following:
  - In one line an object of the Main class will be created.
  - In the other line, a proper method of the Main object will be called to start the program.
  - **Everything else should be under a class.**
- Please note that although `__init__` methods are not specified, for every class in the above list, you should have an `__init__` method in each class.
- In each method of a class, `self` should be the first input parameter. Self parameter is not

included in the above method specifications for brevity.

- You should actively use all the methods listed under each class at least once or more. Make sure that you eliminate repeated or similar codes by defining new methods for them.

### **How to run the executable file of the project;**

- **For Windows;**
  - Download the corresponding file “mp4\_win.exe”.
  - Right click on the file and run it as administrator to start using the project.
- **For Mac;**
  - Download the corresponding file “mp4\_mac”.
  - Open a terminal in the same directory with the file.
  - Write “chmod +x mp4\_mac” and press enter.
  - Double click on the file itself “mp4\_mac” and start using the project.
- **For Linux distros;**
  - Download the corresponding file “mp4\_linux”.
  - Open a terminal in the same directory with the file.
  - Write “chmod +x mp4\_linux” on the terminal and press enter.
  - Write “./mp4\_linux” and press enter to start using the project.





