# Important Instructions / Guidelines:

* Use proper indentation, comments, naming conventions and self-explanatory names if you want to secure better marks.
* Save all classes in separate files.
* Create .rar containing these all files and name it with your student(s) name.

Create a library Management system that manages book, authors, and patrons. The system should have the following features:

1. Book management: The system should allow librarians to add new books, edit existing books, and delete books from the library. Each book should have a title, author, ISBN, publisher, and number of copies.
2. Author management: The system should allow librarians to add new authors, edit existing authors, and delete authors from the library. Each author should have a name, date of birth, and a list of books they have written.
3. Patron management: The system should allow librarians to add new patrons, edit existing patrons, and delete patrons from the library. Each patron should have a name, address, phone number, and a list of books they have borrowed.
4. Book borrowing: The system should allow patrons to borrow books from the library.

Patrons should be able to search for books by title, author, or ISBN, and then borrow a specific number of copies of the book. If the book is not available, the system should display a message indicating the book is currently checked out.

1. Book Returning: The system should allow patrons to return books they have borrowed. Patrons should be able to search for the books they have borrowed, and then return a specific number of copies of the book.

# Deliverables:

## Code files: [30 marks]

Design and implement the code using all the requirements mentioned above. A complete running code file for each of the following is required.

1. Book: A class that represents a book. It should implement an interface `Borrowable`,

Which defines methods to borrow and return books. The book class should have properties such as title, author, ISBN, publisher, and number of copies.

1. `Author`: A class that represents an author. The `Author` class should have properties such as name, data of birth and a list of books they have written.
2. `Patron`: A class that represents a patron. The `Patron` class should have properties such as name, address, phone number, and list of books they have borrowed.
3. `Library`: A class that manages books, authors, and patrons. The `Library` class should have methods to search for books by title, author, or ISBN, and to borrow and return books.

**Challenge Part:**

1. `Borrowable`: An interface that defines methods to borrow and return books. The `Book` class should implement this interface.
2. `Status`: An Enum that represents the status of a book. The `Status` Enum should have values such as `AVAILABLE`, `CHECKED\_OUT`, and `OVERDUE`.

## Documentation: [ 10 marks]

You need to document your java project above.

1. User Documentation This include a document stating what the application is about, explanation of all the classes and their working, and how to start it/access it. Also include the class diagram with the associations between them. [6]
2. Development Documentation This includes at least the Javadocs, a description of the source code directory structure, the build process (i.e., how to compile the project), compiler time dependencies, development standards, how to set up a database for development, and how to get the source code from the repository. [2]
3. Deployment Documentation This is basically the installation manual of the application, describing any steps needed to make it run. [2]

# To demonstrate the project [10 marks]

1. Create a main class to test the library management system and its features. Add appropriate data (at least 10 records) to show the working. [5]

Hint: you can write a method, to load the data every time you run your code.

1. Record a small video to demonstrate the working of your library management system. Explain all the methods, run the code and explain the functionality and the outputs. [5]

Note: Video must not exceed 5 minutes and must not be less than 2 minutes.

**Good Luck**