Final requirements: Library system

- A class that represents a physical book, i.e. a resource that can be borrowed etc.
    - *Data fields:* the ISBN, a library member object, i.e. a typical library user (see respective class below), and a description of the damages for the book. Plus, any other data fields you feel are required.
    - *Functionality:*
        - All getter & setter methods. The setter method for the damages should concatenate the value of its parameter to any previously recorded damages.
        - A method that checks if a book is available: returns true if the value of the library member data field is `None`. Otherwise, it returns false.
        - A method that prints all the details of a book. What if the value of the library member data field is `None`?

- A class that represents an electronic resource, e.g. e-book, journal article etc.
    - *Data fields:* a list/array that stores devices (see respective class below) that can be used by a library member in order to access the resource. Plus, any other data fields you feel are required.
    - *Functionality:*
        - All getter & setter methods, if applicable.
        - A method that prints all the details of an electronic resource.
        - A method that accepts a device object as a parameter, and appends the object at the end of the device list/array.

- A class that represents an electronic device (e.g. a computer, a tablet, an e-book reader etc.) in the library building. Library members can use a device to access electronic resources.
    - *Data fields:* a description of the device location and a boolean that indicates the availability of the device. Plus, any other data fields you feel are required.
    - *Functionality:*
        - All getter & setter methods.
        - A method that prints all the details of an electronic device. How would you print the availability data field?
        - A method that checks the availability of a device: returns true if the device is available. Otherwise, it returns false.

- A class that represents a library member, i.e. a typical user of a library (e.g. student, academic) that can borrow/return books and/or access electronic resources.
    - *Data fields:* a list/array that stores the books currently borrowed by a library member, and a field that keeps the messages/notifications received. Plus, any other data fields you feel are required.
    - *Functionality:*
        - All getter & setter methods, if applicable. The setter method for the messages should concatenate the value of its parameter to any previously recorded messages.
        - A method that prints all the details of a library member.
        - A method that prints all the messages received by a library member.
        - A method that accepts a book object as a parameter and appends this object at the end of the list that stores the books currently borrowed by a library member.
        - A method that prints the details of all books currently borrowed by a library member.
        - A method that returns the number of books currently borrowed by a library member.

- A class that represents the library.
    - *Data fields:* a list/array that stores library resources, i.e. the library catalogue. Based on the above, a library resource can be a physical book or an electronic resource. Plus, any other data fields you feel are required.
    - *Functionality:*
        - All getter & setter methods, if applicable.
        - A method that prints all the details of the library.
        - A method that checks if the catalogue contains a resource. The method accepts a parameter, i.e. the resource object. It returns true if the resource is contained in the catalogue. Otherwise, it returns false.
        - A method that simulates editing the title of a resource. The method accepts two parameters: the resource object to edit, and the new title. What if the resource is not in the catalogue?
        - A method that searches the catalogue by resource object. The method accepts a parameter, i.e. the resource object. It returns the resource object if the resource is contained in the catalogue. Otherwise, it returns `None`.

- A method that searches the catalogue by ISBN. The method accepts one parameter, i.e. the ISBN, and prints the details of all resources with this ISBN. At the end, is also prints the total amount of resources found. What if this resource is not in the catalogue? What if a resource is an electronic resource?
- A method that searches the catalogue by author name. The method accepts one parameter, i.e. the author's name, and prints the details of all resources by this author. At the end, is also prints the total amount of resources found. What if this resource is not in the catalogue?
- A method that removes a resource from the library catalogue. The method accepts a parameter, i.e. the resource object to remove. What if this resource is not in the catalogue?
- A method that removes a resource at a specific position in the library catalogue. Similar to the previous method, but the parameter represents a position in the catalogue. What if the specified position does not exist in the catalogue?
- A method that prints the details of all available books, i.e. books not currently on loan. What if a resource in the library catalogue is an electronic resource, i.e. not a physical book?
- A method that returns the number of resources in the library catalogue.
- A method that simulates adding a resource to the catalogue. The method accepts an object as a parameter and appends this object at the end of the library catalogue. If the object already exists in the catalogue, the method should reject it and print an appropriate error message.
- A method that takes a book object and a library member object as parameters, and simulates the process of lending a book: the library member data field of the book parameter is set to the value of the library member parameter, and the book parameter is added to the list of currently borrowed books for the library member parameter. The operations cannot be performed in the following cases:
    - The book is not in the catalogue.
    - The book is currently borrowed by another library member.
    - The size of the list that stores the books currently borrowed by this library member is greater than 5.
    - Appropriate error messages should be printed in all above cases!
- A method that simulates the process of accepting a book return. The method takes three parameters: the book object to return, a boolean that indicates

whether a damage is to be recorded, and the damage description. The library member data field of the book parameter is set to `None`, and if a damage is to be recorded, the damages data field of the book is set to the value of the damage parameter. The operations cannot be performed if the book is not in the catalogue, in which case an appropriate error message should be printed.

- A method that simulates sending messages/notifications to library members. The method accepts one parameter: the message. The message is sent to all library members currently holding books. What if the resource is not in the catalogue? What if a resource is an electronic resource?
- A method that prints the details of all book resources in the catalogue.
- A method that prints the details of all electronic resources in the catalogue.

A few important points:

- *The above is the set of compulsory requirements, i.e. you must complete these correctly in order to qualify for full marks.*
- *Classes: I have provided only a sub-set of data fields per class. You need more in order to qualify for full marks. For example, a library member will probably have a name data field. Use your judgement or ask me.*
- *Functionality: the requirements are strict; unless I have made a mistake (e.g. wrong parameters etc.), in which case I am happy to acknowledge and revise the requirements. Ask me if in doubt.*
- *Error checking: the above requirements contain some error checking. In other cases, error checking is implied by asking you a question.*
- *Error messages: appropriate error messages should be printed in all cases of error checking.*

Good luck!