Introduction: Queues and Linked Lists

Toy Scenario: We're building a digital portal for a library. To start off, we have two basic entities in the picture, that is, the library and the customers. The first assignment will look at some rudimentary operations:

1. Borrow book
2. Return book

This is quite simple. You are provided a queue class for the book requests that come in. Furthermore, you have to maintain a linked list of pending requests, that is, books which couldn't be lent due to someone else already having borrowed them.

Assignment Questions

You must solve all the problems below, preferably before your lab test. The lab evaluation questions will build upon these assignment questions, and therefore it is strongly recommended that you try these questions beforehand.
You are given the following classes:

- public class BookData: This class has the following attributes:
  - public boolean BorrowedStatus: true indicates that the book is currently borrowed and not in the library.
  - public int BorrowedByUserID: UserID of the person who has borrowed the book. If the book isn't borrowed, it is set as -1.
  - public int ISBN: The ISBN of the book.
  - public String Publisher: Name of the publisher.
  - public String Author: Name of the author.
  - public MyDate DateOfReturn: Date on which the book is to be returned. If the book isn't borrowed, the date will be the last returned date.

- public class MyDate: This class has the following attributes:
  - public int month: Date is represented as MM/YYYY. This is the MM component.
  - public int year: Date is represented as MM/YYYY. This is the YYYY component.

- public class Node<T>: This class has the following attributes:
  - public T data: The data associated with this node.
  - public Node<T> previous: A reference to the previous node.
  - public Node<T> next: A reference to the next node.

- public class RequestData: This class has the following attributes:
  - public int ISBN: ISBN of the book that is being requested by the user.
  - public int UserID: User ID of the user who has made the request.
  - public MyDate RequestDate: The date on which the request has been made. Note that the DateOfReturn for a book should be set to 1 month after the date that it is issued.

    The class has the following methods:
  - public String toString(): The returned string will be printed if you print the object.

- public class Response: This class has the following attributes:
  - public int WaitingTime: The amount of time that it would take for the request to be processed. First response would be sent out the moment a request is received and it would account for 5 seconds of processing time for each element in the request queue. The second response would be sent after the request has been processed at the head of the queue and would account for the return date of the book in case it is currently borrowed.
  - public boolean Available: true indicates that the book is currently available with the library and not borrowed. Set this accordingly in both responses.
  - public int PositionInQueue: For first response, this should be the number of requests in the queue. For the second response, this should be the number of people waiting to borrow this book.

- public class UserData: This class has the following attributes:
  - public int UserId
  - public int Name
  - public String Address
  - public int Age

- public class RequestQueue: This class has the following attributes:
  - private Node<RequestData> front: Reference to the front of the queue.
  - private Node<RequestData> back: Reference to the back of the queue.
  - private int length: Number of requests in the queue.

    The class has the following methods:
  - Predefined:
    - public RequestData getFront(): Returns the front of the queue.
    - public String toString(): Returns a string containing length and data in queue.
  - To be implemented:

    - public int getLength(): Returns the length of the queue.
    - public void push(int ISBN, int UserID): Adds new request to the back of the queue.
    - public void pop(): Removes request from the front of the queue.

- public class PendingRequests: This class has the following attributes:
  - private int length: Number of requests in the linked list.
  - private Node<RequestData> front: Reference to the front of the linked list.
  - private Node<RequestData> back: Reference to the back of the linked list.

  The class has the following methods:
  - Predefined:
    - public String toString(): Returns a string containing length and data in the linked list.
  - To be implemented:
    - public boolean insert(Node<RequestData> insnode): Adds request to the end of the linked list.
    - public boolean delete(Node<RequestData> delnode): Deletes the given request from the linked list.
    - public Node<RequestData> find(int ISBN): Returns the first node with the given ISBN.

- public class LibraryStorage: This class has the following attributes:
  - private HashMap<Integer, BookData> storage: Map of ISBN and BookData for all the books in the library.
  - private RequestQueue rqQueue: The queue of requests.
  - private PendingRequests prLinkedList: The linked list of pending requests.

  The class has the following methods:
  - Predefined:
    - public LibraryStorage(): Constructor
    - public void push(int ISBN, int UserID): Calls push for queue of requests.
    - public String rqQueueToString(): Returns rqQueue's toString().
    - public String prLinkedListToString(): Returns prLinkedList's toString().
  - To be implemented:
    - public boolean processQueue(): Processes the request at the front of the queue by checking if the book is available in the library (returns true) or adding request to pending requests if required (returns false). Updates storage and book details accordingly.

        - public boolean processReturn(BookData book): Checks if the returned book has been requested in pending requests and updates the storage and book details accordingly.

- public class DriverCode: This class has the following methods:
    - Predefined, but modify as per your requirements:
        - public static void main(String[] args): Sets up a LibraryStorage and performs some trivial testing.

Instructions

- Do not change the accessibility, names or signatures of the attributes and methods in the driver code. Try not to add any new attributes or methods to any of the classes - it is possible to implement all required methods without doing so.
- The default constructor is used to instantiate objects of all the above classes. It is your responsibility to ensure appropriate initialization of the attributes of a newly created object.