Template code review

Overview

This is almost all the code for the request system used in our project.

Goals

To give working features for managing requests

Usage Scenarios

Code is used for working with users' request

Code

Categories for review:

1. Design decisions

1.1. [Reviewer1] Why does the document become reference in the method PlaceOutstandingRequest, after deleting the wait list? I know that it can't be checked out but, in my view, saying that it is reference is wrong since references cannot be taken at all, while docs with outstanding request are just temporarily unavailable.

1.1.1. [Owner] The plan was to make the document reference, so it cannot be checked out. When the outstanding request is finished, it should become not reference again. Though, I am going to add outstanding field, so it will be correctly. [TO BE FIXED]

1.2. [Reviewer1] When an outstanding request is placed, we should not ask patrons to return the copy immediately (line 11). We just need to prohibit all renewals and delete the queue for the doc.

1.2.1. [Owner] In fact our TA told us that it should be returned immediately (in 24 hours). Still, I suppose, it is better to clarify it on the lab session. [TO BE DISCUSSED]

1.2.2. [Owner] Discussed during lab session. Agreed on owner's solution. [CLOSED]

2. API design

3. Architecture (including inheritance hierarchy, if any)

4. Implementation techniques, in particular choice of data structures and algorithms

4.1. [Reviewer2] In method PlaceOutstandingRequest checkout is being deleted and there is added the new one with the small correction. It is better to write method ModifyCheckout.

4.1.1. [Owner] These methods were used like that because of the lack of time for implementing ModifyCheckout method, so I will fix it. [TO BE FIXED]

4.2. [Reviewer3] In method 'PriorityQueue(Document document)' on 113'th line there is invalid call of method 'public static bool comp(RawRequestData a, RawRequestData b)': 'documentRequests.Sort(comp)' instead of 'documentRequests.Sort(comp(a,b))'.

4.2.1. [Owner] In this particular case it works because of the built-in procedure sort for the list. It takes a comparator function as an argument and sort the list via it. There is no need to pass the function in a form of Compare(a,b) as by default it is defined for two arguments. [CLOSED]

4.3. [Reviewer1] In method ReturnDocument, line 36. It looks like User's debt increases only after he returns the document. It means that if he never returns it, he has no debt. Will the library be able to survive with such rules?

4.3.1. [Owner] Yes, it really works like that, which is rather bad. I suppose, I will do it this way: when librarian checks the list of checkouts in the system (what he is supposed to do every day), the system will count fines for all expired checkouts, so user will get his fine in time. [TO BE IMPLEMENTED]

4.4. [Reviewer1] In ReturnDocument, lines 37-38. Why do we delete the checkout twice: once in 'DatabaseManager.DeleteCheckout(checkout)' and then in 'checkouts.Remove(checkout)'? Why can't it be merged into one deletion?

4.4.1. It works like this, because we separated the responsibilities for classes. DatabaseManager works only with the database, while Library class, where this piece of code is located, refers to all the local content and as checkouts array is a local one, it can be reached only from Library class, so there is another line checkouts.Remove(checkout). [CLOSED]

5. Exception handling - Contracts

5.1. [Reviewer3] How are you handling exceptions? I mean, e.g. if in 'public static bool ReturnDocument(Checkout checkout)' method you cannot add a checkout, how does user will understand this? In all other methods there is no exception handling at all.

5.1.1. [Owner] In fact, all the exceptions are being handled in the interface, e.g. when we try to request the same document it gives us a message, that user cannot request one document twice. [CLOSED]

5.2. [Reviewer2] In methods PlaceOutstandingRequest and ReturnDocument there are no contracts. There is no checking that document is not null.

5.2.1. [Owner] All the checks are implemented in the interface, still I will add contracts here, so we will debug it easier. [TO BE FIXED]

5.3. [Reviewer2] In 52 line document is used before contract on document.

5.3.1. [Owner] Yes, it is definitely misordered, i will fix it. [TO BE FIXED]

6. Programming style, names

6.1. [Reviewer3] Name of 'public static bool comp(RawRequestData a, RawRequestData b)' is not appropriate.

6.1.1. [Owner] Yes, maybe it is not easy to understand, will rename it to comparator. [TO BE FIXED]

6.2. [Reviewer1] Extra space in line 92, after List.

6.2.1. [Owner] Already fixed it in the latest release. [FIXED]

6.3. [Reviewer1] Do we really need to define ‘var request’ in RequestDocument, line 68? Why not write just ‘return RequestManager.SendRequest(user, (Checkoutable)document)’?

6.3.1. It sounds reasonable. [TO BE FIXED]

7. Comments and documentation

7.1. [Reviewer2] On the line 7 what is variable checkouts? It is not clear.

7.1.1. [Owner] Actually, this array is defined in the fields of the class, from which this code was taken, so I find the commenting here superfluous. [CLOSED]

7.2. [Reviewer2] There is no documentation on the method RequestDocument.

7.2.1. [Owner] There exists documentation, but I forgot to attach it to the github. Sorry about that. [CLOSED]