Peer Review

Group: Filip Rydberg

The view currently has a domain responsibility in regards to the boat type. If you change the entire view layer, you have to hope that the next developer will implement the exact same boat types as strings. It should be moved to the model, preferably as an Enum.

The unique id is assigned from randomizing a number in the Member model and then checked if it is unique in the controller, and then looped in the controller until it is unique. This will work on a small scale application where you won't have as many members. But when you reach 10,000 members the program will crash in an infinite loop. Don't underestimate the number of users, try to make it scale as large as possible.

See: http://www.nextrollout.com/psy-gangnam-style-music-video-hits-crossed-int32-maximum-value-in-youtube/

The randomization of the id should be a part of member or memberDAL, not the controller. And try to keep it in only one place (preferably the memberDAL).

We found some "dead code" at the end of model.Member that throws "System.NotImplementedException();".

The application is overall well written and nicely executed. A simple elegant solution. Your class diagram is much simpler than ours (in a good way).

A few things that can be fixed:

1. We could not find a "View specific member" option.
2. The title says "Edit" when deleting a boat.
3. Empty names are allowed.
4. Use lambda or linq expressions to optimize code in deleteMember instead of looping through it.

See Larman [chapter 17 section 11] on Information Expert as it should probably be the memberDALs responsibility on handling these methods in the controller:

- MasterController.deleteMember()
- MasterController.updateMember()
- MasterController.createMember()

Larman - Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development, Third Edition