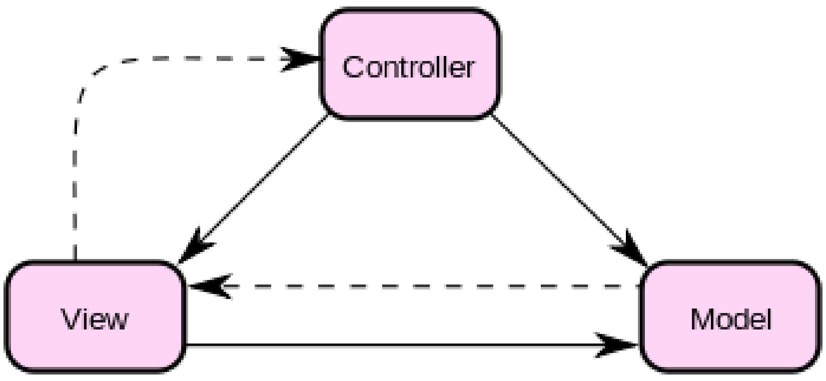
Change-Log Workshop 2

Feedback on the project:

The feedback below are the comments received for both Amelie's and Johans workshop 2 submission. The result section shows what kind of changes we did and how we implemented the criticism in our code.
Code:

1. **Fix**: member.amountOfBoats is not needed why bother having a separate attribute for this?
   Result: We removed the member.amountOfBoats variable everywhere since it was unnecessary.
2. **Fix**: member.setId is not a good function to have as anyone can set an invalid id to a member in the registry now
   Result: We removed the setId() method in the Member class to make sure that people cannot set their own Ids. Instead we added this information to the member constructor and simply parsed it in the filehandler methods when trying to use saved information.
3. **Fix:** Seems like a mixed responsibility for the controller to both persist the registry and to handle the user scenarios
   Result: The private methods for the file handling (create file, save file and init file) are now in their own classes and located in the filehandler folder.
4. **Fix**: I don't really know what the point of the interfaces are since they are not used
   Result: We removed all of the unnecessary interfaces in the model such as boatInterface and memberInterface.
5. **Fix**: View responsibility in the model
   Result: We made sure to remove all of the printouts from the model.Furthermore the functionality in the view class we moved to the member register. Both the view and the other model classes will now only be used when they get called on the controller class when needed.
6. **Fix:** Hidden dependencies between view and controller/ Proper encapsulation in the model
   Result: We did several changes to fix the errors in the code. Firstly we removed all of the functionality such as adding a boat, changing a name etc from the controller and added it to the member register. This was done to more accurately represent the MVC-pattern where the controller shouldn't really do any functionality and instead just delegate/call the View and Model. Since the models purpose is managing the data of our application The controller simply receives the user input which it validates and then sends to the model and view. The view then prints out the results for the user to see.



Diagrams:

1. **Fix:** Class diagram is missing dependencies. You can remove the attribute corresponding to the association in the class diagram.
2. **Fix:** Sequence diagram should use the actual method calls found in the implementation (addMember). Do not forget to actually create the member.
3. **Fix:** Sequence diagrams do not match the implementation
4. **Fix**: Sequence diagram should use the actual method calls found in the implementation (addMember).
   Result: We did new sequence diagrams for the cases addMember and printCompact as well as updating the class diagram, to add the changed dependencies.