This iteration I firstly focused on fixing problems in the previous iteration that I learned from working on workshop 3. By learning how to use an own observable patten I could remove all such things as ObservableList and Properties from the model and instead encapsulate the list by limiting the controller/views way of reaching all members to a iterator.

Then I could start with the actual assignments, first to add different type of uses by using Strategy pattern. I tried to use workshop 3 as a guideline and decided that the MainView will ask the interface UserStrategy to print the elements that differ between authorized member and unauthorized member. At login (LoginView) the choice user make will decide if the UserFactory will load the MainView with a authorized or unauthorized setup. I was choosing between this way and that the UserStrategy would decides what window will be shown. Advantages in the way I chose is that it is less code duplication, so if i.e changing color of MainWindow background will affect both users by just changing one place. On the other hand it is probably a little harder for a outsider to read the code and see what elements is handled where.

I didn't spend much effort with the login logics. It is just hardcoded
user:password
**admin:admin**
but I don't think that was important part of the exercise.

Next task to do the search I spent some time to decide what pattern to use. I didn't find any of the "concrete" patterns I studied so far to solve the task. Or rather I didn't figure out how to solve the task with those patterns. I would guess that the composite pattern together with strategy pattern could have been used. But I found a pattern called the filter pattern or sometime criteria pattern. So basically you have a interface with a meetCriteria function that takes a collection as input and return a filtered collection as output. Then in the constructor for each concrete class you can add the parameters needed for that filtering. Also you can add AndCriteria/OrCriteria which has two Criteria as fields set by the constructor. And then you can just combine multiple Criteria fields by them.

**UML:**
I added the filter package as a class diagram, but didn't redo the whole class diagram in this iteration

**Discussion:**

In the grade 2 I started doing a console based application, but after the model was done I found myself spending so much time with the user interface of the console view that I though it would be both more challenging and fruitful. But I still wanted to keep it simple and only use Model-View to not think to much about controller. But as the program grew I felt the view got messier and a controller started to be needed to give some better structure. So I first added a CRUD controller that should take care of communicating with the model. When adding the Criteria classes I made another controller to handle them. But I think that there is a lot of controller related code still in view. I think that it would take a lot of work to now design the software to be a clean MVC. So in this case I should have thinking twice before skipping the controllers and added them in a early stage to probably get a higher cohesion and less coupling.