Alexandria University
Faculty of Engineering
Computer and Communication Engineering
Specialized Scientific Programs

CC272
Programming II
Fall 2022

LAB 10

MVVM (Model View ViewModel)

Introduction:

MVVM pattern is widely used in practice, if you are willing to become a mobile developer (someone who develops mobile applications) or Frontend web developer (someone who develops the front-end of the website or web apps that we use in our daily life) you will see the word “MVVM” widely used, the concept is straightforward, you create 3 packages one for the model, one for the view and one for the ViewModel.

The model package contains the classes that represent our data, for example: in previous labs, we had a Librarian that is saved in the Database, the Librarian is the unit that we use to keep in the database, so for simplicity, any class that will be saved into the database is called a model.

The view package contains simply our views, for example, any class that holds User Interface like JFrames or any GUI-related classes

The ViewModel package contains the classes that hold the business logic for each screen in our application, so if we have any calculations or any logic performed on any screen, this logic should be put in the ViewModel, in practice, there is also what is called a shared ViewModel that holds business logic for multiple screens (by screen here it is meant a JFrame Window).

The pattern applies the concept of separation of concerns if you wonder why are we doing all these extra steps. Why shouldn’t we just use one package for everything? Here is the answer: in practice, you will build huge applications that have multiple screens of multiple models, we may have hundreds of classes in one project, so by separating this huge number of classes we make our lives easier as developers, it also helps in something called unit testing which is out of course’s scope, another big benefit is that by having structure for our code, we reduce the number of bugs that may be produced.

So when we change one class, our changes will only be reflected on View or ViewModel or our models, this paradigm also helps us in teamwork it’s like having two packages in previous labs: one for the front and one for the backend.

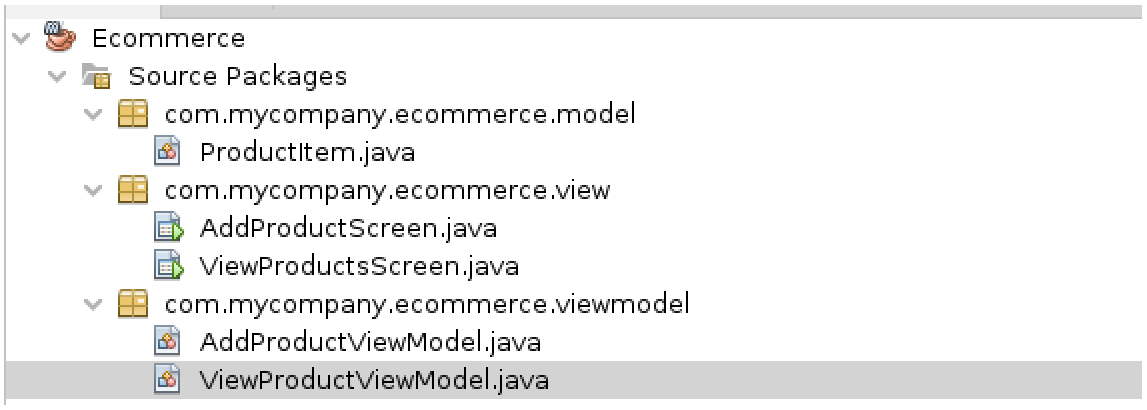
Dr. Layla Abou-Hadeed

Eng. Ahmed El-Sayed
Eng. Tarek Salah
Eng. Ahmed Ashraf
Eng. Michael Said
Eng. Minoura Waguih
Eng. Mostafa Ibrahim
Eng. Noha Mahmoud
Eng. Adel Meleka

Requirements:

You are required to develop a simple java application that contains only two screens, one screen for adding a new product and the other screen for displaying the products already added, the product should have (a unique id, name, and price), and you may also try to add a picture for each product (adding a picture for each product is a **BONUS**)

The project Hierarchy should be the same as the following screenshot:



```
Ecommerce
  Source Packages
    com.mycompany.ecommerce.model
      ProductItem.java
    com.mycompany.ecommerce.view
      AddProductScreen.java
      ViewProductsScreen.java
    com.mycompany.ecommerce.viewmodel
      AddProductViewModel.java
      ViewProductViewModel.java
```

ProductItem is our model that will be saved into the database, the two screens are normal JFrame screens like the ones you used in previous projects, any function or calculation should be added to the view model of that class, you should not do any calculations or code in the screen classes, so for example if we have a button in AddProductScreen that should add a new product to the database, the code or the business logic for adding this new product should present in viewModel and here is a small pseudocode example:

```
Class AddProductScreen ….. {
        void onButtonClicked(event e){
                // do validations to product
                // if ok then connect to database
                // add product to database
                // navigate to main screen (ViewProductsScreen)
        }
}
```

The previous approach is the approach that you used in previous projects, but we want to change this code to our new mvvm pattern, so instead what we will do:

```
Class AddProductScreen ….. {
        void onButtonClicked(event e){
                // boolean result =
        addProductViewModel.validateThenAdd(product_to_be_validated)
                // if product validation was ok and product is added, navigate to main screen
                // else display error dialog or any other option
        }
}

Class AddProductViewModel {

        Boolean Method validateThenAdd(String productName, id, …..){
                // do validation and if ok connect to db if not return false
                // connect to database
                // add product to database  (this can be a separate function instead of having a
very long function
                // if successfully added return true else return false
        }

}
```

The previous approach is what you are required to do, you can put the navigation code in View classes as you normally do, and you can also use observer pattern for navigation and this will also be **a BONUS (using observer pattern)** **check this link for observer pattern**

Procedure:

1. Design the two required screens (any design should be ok, this depends on your style but you have to do validation and navigation correctly), also when designing a new screen try to mimic the applications that we use in our daily lives, this is an important field of science called User Experience.
2. Implement ProductItem (this is a simple class)

3. Add logic for validation (this code will also be an easy thing for you) and database operations (by database operations it's meant: adding a product to the database or retrieving products from database
4. For the database, in this lab you will use a real database called sql database, it's the same as the text files you used and same here means same concept but not same code, this part requires you to use your research skills to add or implement sql database code in the project. (you should do this on your own) hint: search for **JDBC** or any other valid database of your choice

Notes and Remarks

- There is no right and wrong in this project, any option is valid if you separated the code as discussed.
- Again, the design is yours you can make it any style you want.
- If you face any difficulties, use your own research skills for solving the problem and in the discussion, you can mention the difficulty and how you solved it (again there is no right and wrong as this is your first experience)
- Make yourself comfortable and don't panic, this is an easy lab and it's ok to see new things that you didn't know previously as this will be your daily life when you work on real projects.
- You can work in groups of 2 - 3, please note that each team member should be aware of the full project (try learning git and github in your free time, as this is widely used in practice and will make your life easier when working in teams)

Grading Criteria:

- Separation of concerns and using MVVM as explained in above sections
- Doing validations and navigation
- Saving and retrieving product to and from the database (any valid sql or nosql database but not a textfile)
- Don't cheat, this is an opportunity to learn something that you will use in daily work, you shouldn't build your career on cheating, also divide the work evenly, each team member should face challenges, the one genius approach is a very bad approach and will make other team members face difficulties in work in the future.
- Have fun and increase your research skills, don't panic again, it's ok to face new challenges.