**Object Orientated Programming with Java**

**Assignments 5: Graphics**

**OVERVIEW**

The concept: I am a dynamic person and I really enjoy doing various different sports! Sometimes I train for a long period of time for a competition and the organization of my trainings through the weeks can become quite confusing. That’s how I had this idea, build an application which would be very simple, easy to use, enabling to schedule my sport trainings through the week.

It was decided to develop an Android application; as we are already using JavaFX for our Software Engineering project, I thought it would be interesting to gain experience with a different graphical library. Android Studio was chosen to develop the application, which has the great advantage of coming with an Emulator on which you can test your application as you are building it. An online repository was also created on GitHub in order to back-up easily and regularly my work.

**KEY DESIGN STAGES**

Define Architecture

It was decided to containerise the whole application within one Activity. It layout is divided into two parts:

- A bottom navigation menu; it will always appear at the bottom of the screen

- A Fragment container; its content is dynamic and will change depending on which item is selected on the navigation menu.

The different pages are containerised in fragments, so that they can be placed within the activity fragment container.

Finally ‘pop up’ windows are created within fragments using Dialog Fragments.

Each of these pages have 2 components: a class containing the code defining the page components’ behaviour, and a layout defining the page components’ properties.

Development stages

1. **Create the Application structure**

Firstly, I focused on creating the different pages that would be accessible on the application. A *MainActivity* class was created in which the bottom navigation is created and handle on a click which page should display. Two pages are available, ‘Home’ defined by the *HomeFragment* and ‘Training’ defined by the *TrainingFragment* class. By default, the application will show the ‘Home’ page.



Two tabs were then added into the Training page: ‘Category’ and ‘Favourites’. Each of them has its own class which creates a Fragment object.



At the moment, those pages all have a layout, but completely empty – although the navigation between them works fine.

1. **Add content to Home page**

The first goal was to be able to display data on this page. A general practise which is used for any other Fragment is the following:

1st: bring up the layout as a View object

2nd: identify different widgets present on the layout using the view object and retrieve widgets with their ID.

3rd: use those widgets as it suits our need (for instance, retrieve data from an input field, or listen for a click event on a button).

Then, a Training class was created storing the facts about a training: title, category and time.

At the start, dummy data was used - stored within MainActivity class and passed to the fragment. A *ListView* widget was added to the home layout, and another layout was created specifying the layout of each instance part of the *ListView* layout. Then using an adapter *TrainingAdapter* object, the data was read from the HomeFragment object and displayed on the Home page.

Then, we added a button widget to allow the user to create a new training. On click, this button displays a popup which asks the user whether he wants to create a new Training. At this stage, if ‘confirm’ was clicked, nothing would happen because a page to create a training doesn’t exist yet.

1. **How to Create a new Training**

A first version of the “create a training” page was added to the application. It only has TextField widgets in which can input the title, Category and time of the training. A Relative layout was chosen over Constrained, so that only the title has to be fixed relative to the layout – the rest are fixed with respect to the title. This way, once the user clicks on a text field to enter data, as the screen size reduces to display the keyboard – the rest of the widget will seat invisible under the keyboard.

To do that, a new class was created as well as a layout.

1. **Implement database (tables + APIs)**

It’s quite a complicated process to always have to pass information between fragments in order to store them within the Main Activity. As well if the application shuts down, all the existing information will be lost. We therefore implemented a database with our application using SQLite.

A new class DBHandler was created, containing methods to create the database as well as the trainings table. It also contains a set of CRUD (create/retrieve/update/delete) APIs for trainings. The Home page now calls the *getAllTraining()* method to retrieve the data to add on the ListView. Also, the create Training page adds the new Training object to database using the *addTRaining()* method.

1. **Add content to Category tab**

The tab layout has a ListView widget which displays the categories names.

A new ‘Category’ table was added to the database, as well as a set of APIs. A type field was initially implemented in the case where we would have other features on the application. For instance, an option to add ‘Meals’ was considered at the start, although it wasn’t implemented in the end.

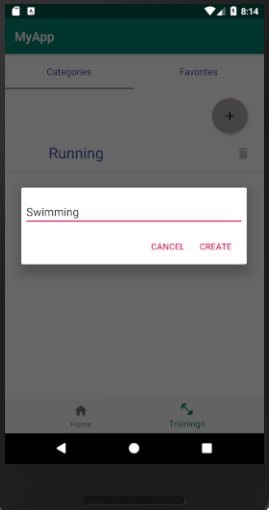
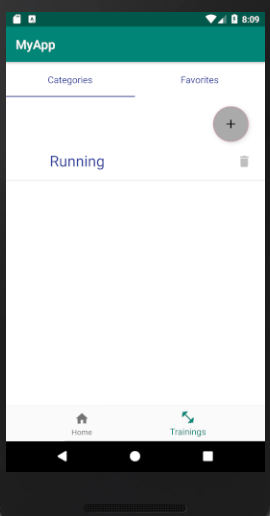
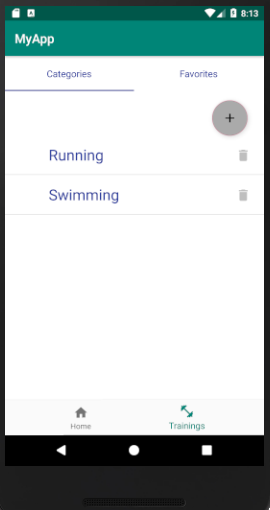
Then, within the category tab, a button widget was added on the layout to enable the user to create a new category, and on click, a popup displays asking for the new category. Because only the category name needs to be specified, a popup seemed a better choice than a fragment in terms of user experience.

A delete button was also added next to each category, calling the delete API from DBHandler.

The main Challenge here was to update the categories directly without having to refresh page when a new one was created or one was deleted. In the end, this problem was solved in the PopUpCreateCategory class directly: using the getActivity() method, it was possible to retrieve the ListView of the category tab and set a new adapter with the updated ArrayList of categories. Hence once returning to the category tab, the new category appears directly.

Add

Delete



1. **Modified ‘ Create a Training ‘ page**

* Category input:

The TextField was replaced with a dropdown list containing the existing category (see pictures below).

Also, appears by default ‘Select a Category’ to ease the understanding of the user.

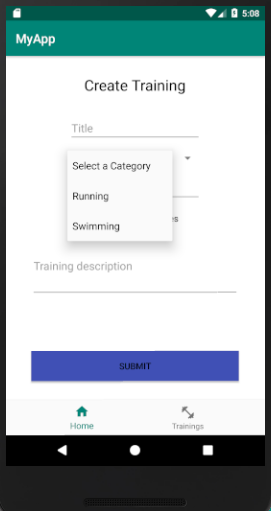
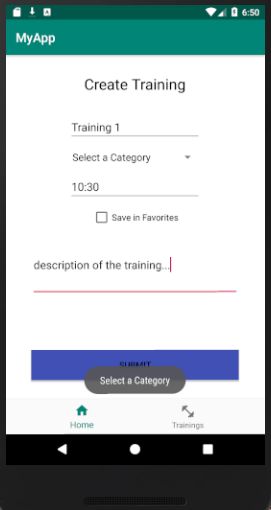
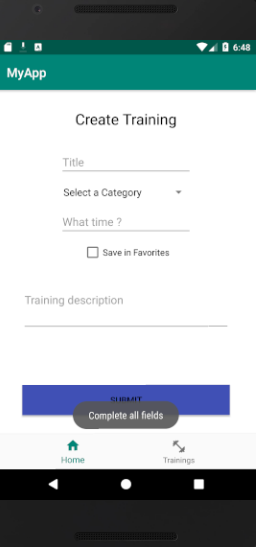
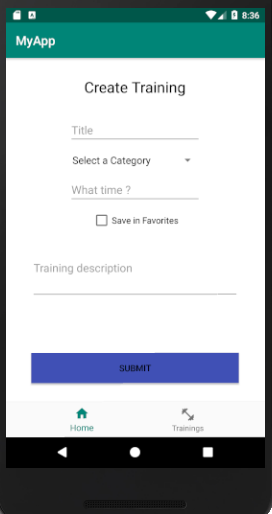
* A tick box was added so that the training can be saved as favourites

This required to add a Boolean field *isFavorite* to the Training class and modify the DBHandler APIs

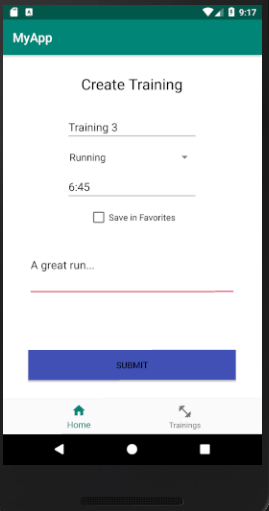
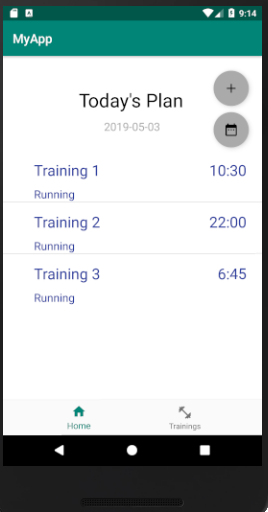
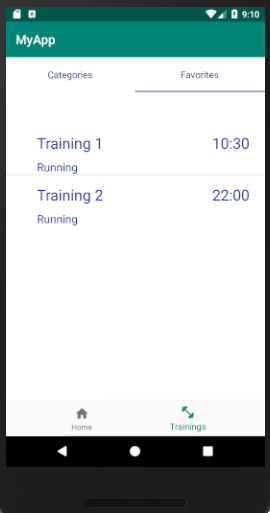
* A new field was added: training description.

This required to add a new Text field *description* to the Training class and modify the DBHandler APIs

* All fields were made mandatory, thus the user cannot submit the new Training unless he complete all the fields and select a Category. See below the small popup hat appear at the bottom of the screen in case the user click the submit button without having filled everything.
* A hint string was added to each TextField to indicate the user what to enter



1. **Add Content to Favourites tab**

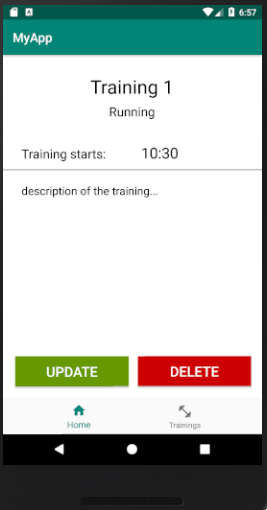
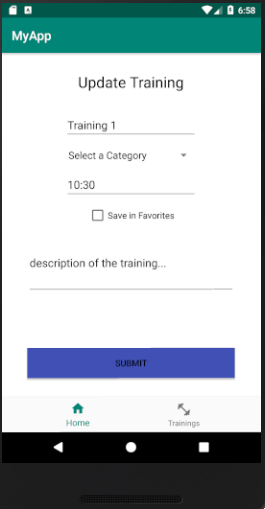


The same adapter and layout as the Home page was used. The difference is that it only displays the training for which ‘Favourites’ was ticked. To do that, a new API was added within DBHandler class.

On the images below, notice that Training 3 doesn’t appear on the Favourites tab as the box hasn’t been ticked. Unlike Training 1 & 2 who both have it ticked and appear in the favourites.

1. **Individual training page**

On the Home page*,* thelistelements are now clickable. Then, passing the Id of the training that was just clicked to a new fragment, we display the training details.

Hence a new class and Layout were created; the layout retrieves the Training data with the id, using a new API getOneTraining() implemented within DBHandler. Within the layout, we use TextView widgets to show the Title, Category, Time and description of the training. In addition, buttons widgets *delete* and *update* were added at the bottom. The delete button uses a DBHandler API to delete the training from the database, however the update one doesn’t work yet.

1. **Update Training Page**

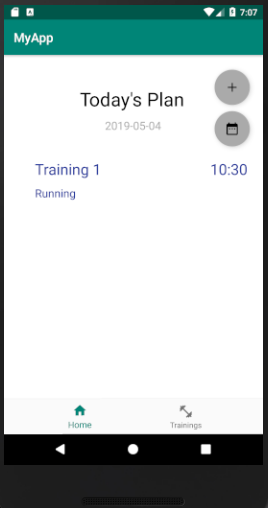
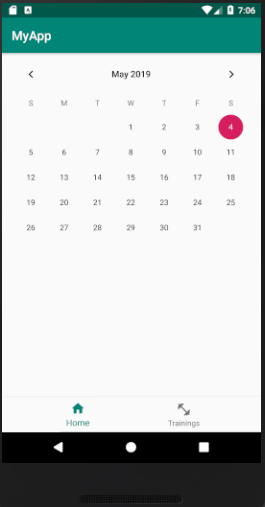
This page uses the exact same layout as *create a training*, except that fields are pre-filled with the training existing data. To update the database, it uses an existing API from DBHandler. The only problem here that remains is the category which needs to be reselected by user, as it will show by default ‘Select a Category’.

1. **Added calendar Page**

The aim is to give the user the possibility to choose the date at which he wants to add a Training, therefore he can plan in advance his trainings and not have to do it every day. Thus, we added a new button on the Home page which enables to go to see the calendar. This calendar is a new page that we created within which we put a CalendarView widget.

At the moment, nothing happens when a date is clicked, although we intend to implement it later. Also, whenever we click the Calendar button, the circled date (pink circle as shown below) is always the same (currently today’ date).

Go to Calendar



1. **Add date to a training**

We added a date field to the Training class, database table and the training APIs also had to be modified. A date is saved as a SimpleDateFormat object, in the format yyyy-MM-dd.

Then, once a user wants to create a training, the date of today is passed to the ‘create a training’ fragment and assign it to the new Training object.

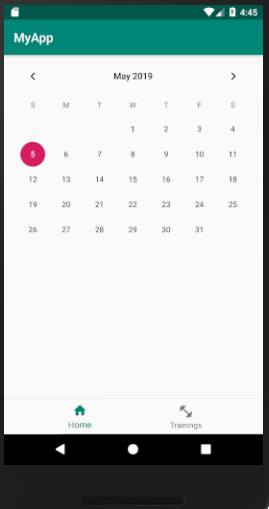
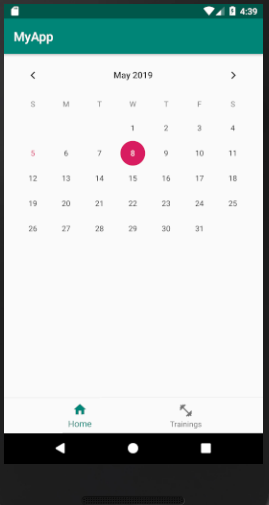
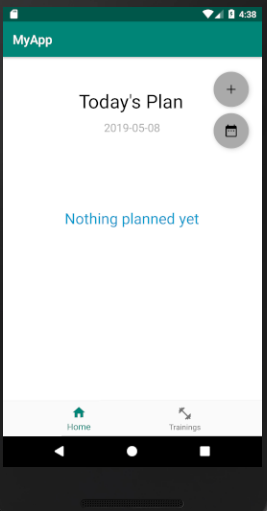
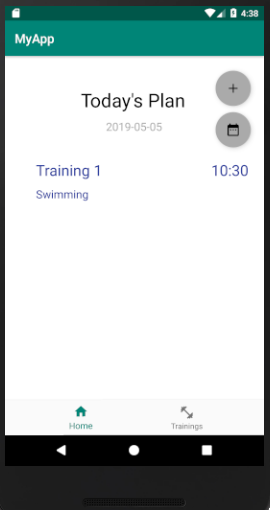
However, this is not what we are truly aiming for as we want the date of a Training to be variable and not be always today’s date.

1. **Keep track of Home page’s date**

A variable dateHome was added within the MainActivity class, as this object is only ever created once. It wouldn’t be possible to store it within a fragment as its value would wiped out every time the fragment is changed. Although, its value can be retrieved/modified from other fragments using getters and setters. When the application is started, the dateHome default value will be set to today’s date. Then, once select of a date on the calendarView, it changes the date variable in main activity – the home page is now allowing to create trainings for that new day.

In addition, the Pink circle on the calendarView now indicates the current date of the home page; as you can see on the pictures below – it varies as the user choose a new date.

Finally, the trainings displayed on the Home page are now filtered by date – only the one for the home date will appear. The only issue here is that I couldn’t get an API to query trainings by date: instead we are retrieving all the training then sort out in a new list the one with the correct date.



**CONCLUSION / FUTURE WORK**

- Implement ‘Meals’ – would be exact same content as Trainings – in fact the Training classes could be renamed ‘Item’ and the Item class would have an extra parameter ‘type’. Some work would need to be done in order to display them differently on the Home page.

- Have the application customized for a given user: would require an authentication system and relate trainings to a user (ex: adding author\_id to Training model)

- User experience could be enhanced with the calendar: android doesn’t provide any styling library – however external libraries can be used. For instance, an icon could be added under a date if it contains training/meal, at the moment it’s quite confusing for the user to easily see when his trainings are scheduled.

- Also, a reminder option could be added, which could send a notification to the user reminding him about his training the day/few hours before.