CM3110 Mobile App Development

# Topic 2 - The App Dev Tools

In the lecture this week we started looking at the tools used to create Android apps. In the lab this we’ll do a follow-along, walking through all the steps of getting a new Android Studio project setup for our weather app.

Below is the rough series of steps that we’ll going to follow.

Text in *italics* refers to items / things that should be visible in Android Studio.

## Task 1 – The Hello World of apps

### Creating a New Project

* Open Android Studio.
* Create a *New Project* (or select *New Project* from the *File* menu if Android Studio is already open).
* Create a new project for *Phone and Tablet* and select the *Empty Activity* option.
* Complete the *Empty Activity* form with appropriate details.
  + Remember to set the *Save Location* to somewhere sensible (although we’ll use GitHub to backup everything it’s still a good idea to store the files locally somewhere that you can find them again), and
  + **Set the *Language* to *Java****.* (the default is Kotlin, and its easy to miss!).
* Wait a few minutes for Android Studio to finish setting up, building, and displaying the new project and *MainActivity.java* file.

### Configure Github for a new project

Its never too early to start getting your code back-up sorted. Assuming you’ve created a GitHub repository (using the link on the Moodle page), get the URL for the repo you created. If your unsure what that is, go to the GitHub page for it, and copy the URL that would be used for cloning the repo.

Note – we’re assuming here that you’ve got an empty GitHub repo.

In Android Studio

* Create a local git repo on the machine you’re working on:
  + Select *VCS* then *Create Git Repository.*
  + Select the folder that contains your app code, then click *OK.*
  + Wait a couple of minutes for Android Studio to setup the local git repo. The benefit of doing this, is that it’ll create a very useful .gitignore to ensure that only the important files are uploaded to GitHub – anything that is autogenerated at build time won’t be copied to GitHub.
* Now add the remote repo:
  + From the menu, select *Git* then *Manage Remotes …*
  + Add a new option, with the *Name* set to *origin* and the *URL* set to the url of your GitHub repo.
* If you’re promoted to *Log in to GitHub*
  + **Don’t select Log In via GitHub…** this option no longer works, instead…
  + Click *Use Token…*
  + If you have a token handy, then go ahead and copy it into the dialog. Otherwise, click *Generate Token.*
  + Log in to GitHub and it’ll automatically setup a *New Personal Access Token* for you. You may want to **change the expiration**to a *custom date* of 31 Jan 2023 – this will cover you for the duration of the module.
    - If you’re creating a token yourself, then ensure you have selected the following *scopes:* 
      * *workflow*
      * *read:org* from the *admin*:*org* group
      * *gist*
  + Generate the token, and then copy it.
  + Save the token to a text file somewhere useful (such as your H: drive or OneDrive), as you’ll need it next time you use Android Studio.
  + Back in Android Studio, paste the token into the *Token* field of the *Log In to GitHub* dialog (leave the *Server* as *github.com*).
  + Then click *Log In.*
* Ok, now we should have our GitHub repo listed as a remote for the local git repository running on your machine. Click *OK* to finish that process.

### Push the Android Studio Project to GitHub

Now that you’ve got the GitHub repo setup as a remote, lets push your local code to it.

First, add and commit all the files:

* *Git* then *Commit…*
* Checkall the *Unversioned Files* and add a suitable *Commit message* (try to be more original than “first commit”).
* Then click *Commit and Push..* to both add these files to the local git repo, and push them to the remote GitHub repo.
* All being well, it’ll take a couple of minutes – Android Studio may warn you that some *Checks failed*. If this happens, take a look at the and either resolve the issues, or go ahead and hit *Commit Anyway and Push…*
  + These checks are usually for things like TODO comments in code and other minor issues that won’t impact the ability of your code to run.
* On the *Push Commits to <Repo>* dialog, hit *Push*.
* Then you should get a notification that the *pushed master to new branch origin/master* has been successful.
  + If you don’t trust it open of a web browser, go to your GitHub repo and check – the code (and various other files) should be there.

### Run the app and a Virtual Android Device

**Warning:** this should be the most boring app you ever run! But all great things must start somewhere!

If *No Devices* is being shown on the Android Studio menu bar, then you’ll need to setup an Android Virtual Device (AVD):

* From the menu, select *Tools* then *Devices Manager.*
* It’ll either open as a new dialog, or as a side tab in Android Studio. Click *Create device* then go through the Wizard, creating a new *Pixel 2 XL* Phone with the *System image* of *API 33 x86\_64*. Give it a suitable *AVD Name*, and the select *Finish.*

Once a device has been setup, it should be visible on the menu bar. Select it, then click the green run arrow.

After a couple of minutes, the virtual device should display and show the Hello World screen of Android app development. If it boots to a blank screen, then you need to turn it on.

Congratulations, you’re an app developer now!

Once its up and running, explore the AVD, the available menus (try rotating the device), etc.

## Task 2 – Adding a Doman Object

In a future lab we’ll be downloading and displaying the hourly forecast for a given location. Let’s do some forward planning by designing and creating a Domain Object to store that information within our app.

First, let’s design the domain object. Open your favourite weather app and check the **weather in your favourite location one hour from now**.

What information is displayed in the app?

* You may see details such as: temperature, unit of measurement, humidity, time of day, date, wind speed, gust speed, wind direction, chance of rain, an icon indicating weather type, amongst other things.
* Select which of these attributes you’re going to add to your domain object. For each, define the type (string, int, double, float, Boolean, Class, Enum, etc) and name.
  + Note – for **icons**, the data type is likely to be String or Enum – we’ll work out how to convert that to an icon later in the module.
* For the purposes of this example, we’ll add one attribute to the Hour Forecast:
  + Temperature attribute, with type double.

### Create a new Java class

* Android Studio’s File menu is context sensitive, so select the *java* folder in the *Project* explorer. Then select *File* from the menu, then *New* then *Java Class*.
* Select the *Destination Directory* as the *main* folder that contains your app’s code (i.e. **not** the *androidTest* or *unitTest*) folders.
* Enter a name – such as *HourForecast* – and make sure *Class* is selected then press the enter key.
* If you’ve setup git in your Android Studio project, you’ll receive a prompt to *Add File to Git*. This is generally a good idea, so go ahead and *Add*.
* The Android Studio Java editor should be displayed, with a basic Java class.
* Add a new private member variable with type double and name temperature (see below).
* Then *right click* on the code editor and select the *Generate…* option from the menu that appear.
* Select *Getter and Setter*
* Select *temperature:double* (it’s probably already selected), then click *OK*.
* Your class should now look something like:

public class HourForecast {  
   
 private double temperature;  
  
 public double getTemperature() {  
 return temperature;  
 }  
  
 public void setTemperature(double temperature) {  
 this.temperature = temperature;  
 }  
}

### Commit and Push to change to GitHub

When the new class was created, Android Studio asked if we want to add it to the list of files that git tracks for us on our local machine. Now we need to commit the changes we’ve made and push them to our remote GitHub repository.

This is effectively the same process as we followed earlier:

* Select *Git* from the menu, then *Commit…*
* Make sure the new file is selected (in this case *HourForecast.java).*
  + New files are highlighted in Green
  + Edited files are highlighted Blue
* Add a *Commit Message*, then click *Commit and Push…*.

## Task 3 - Complete the Hour Forecast Domain Object

We’ve added on member variable to the hour forecast domain object, yet hourly weather forecasts are likely to have much more information. You have already identified what the are – complete the hour forecast domain object by adding member variables and associated getter and setter methods for these.

Once you’ve updated the domain object, don’t forget to push the changes to your GitHub repo.

## **Well done!**

You are now able to use the main tools involved in app development and have already started to create your first app!

Every modern native Android app is started life in the same way as you have just done. Every app developer starts off making sure they have all their tools setup together and in working order. You are now officially one of them!

## Task 4 - Changing the Hello World message – either in lab or as exercise for next week

OK, so the Hello World Message on the activity\_main.xml is fine but let’s face it – is a bit … Hello World.

Foreshadowing next week, your task now is to create the code that will change this to display information from an Hour Forecast domain object. This builds on the reading you did follow the lecture this week on Android resources (you did read up on the slides didn’t you?).

Open MainActivity.java

* In *activity\_main.xml* select the *TextView* and give the *id* attribute a value, such as “tv\_message”.
* In the MainActivity.java file, in the onCreate method, add new lines of code to create a new Hour Forecast (or whatever you called the class), and set the temperature to 32.4):

HourForecast hf = new HourForecast();  
hf.setTemperature(32.4);

* Update the text of the tv\_message TextView to be the temperature value using the following code:

TextView tv = findViewById(R.id.*tv\_message*);  
tv.setText(String.*valueOf*(hf.getTemperature()));

A few things have happened there:

* + First we created a variable of type TextView, with the name tv
  + Then we used the findViewById method – this method attempts get the TextView from the user interface with the provided ID.
  + To get the ID, we used the R class – a special class generated by Android that allows us to access the resources
  + R.id is a generated class that contains all of the id values for the views (widgets) on the UI – we used R.id.tv\_message as it corresponds to the id that we just gave the TextView in the UI editor
  + We then set the text of that TextView – i.e. what is shown on the UI
  + We convert the temperature to a String so that it can be displayed.

### Using String Resources

Really when we are displaying anything on the user interface, we should do so using a String resource. Let’s add one

* Open *strings.xml* from the *res/values* folder
* Add a new value – note the value below uses the formatted string (see <https://developer.android.com/guide/topics/resources/string-resource#formatting-strings>)

<string name="str\_tv\_message">%1$d Celsius</string>

* Open *MainActivy.java* and update the code setting the text as follows  
    
  TextView tv = findViewById(R.id.*tv\_message*);  
  String text = getString(R.string.*str\_tv\_message*, hf.getTemperature());  
  tv.setText(text);

## Task 5 - Before for next week

Last week you defined the use cases, task flow, and user flows for your weather app. Now you’re in the position to be able to start building it. However there are still a few things we need to plan out. We’ll adopt an agile approach to that, planning, implementing, and testing as we go.

Next week you’ll start creating the user interfaces for your app. Use the remainder of this lab and the time between now and then to create wireframes for the pages that your app is going to require – that way next week you’ll know both what the user experience will be like, and how the app will appear.

Share you wireframes on the provided Powerpoint file and we’ll aim to provide some feedback on them before the next lab.