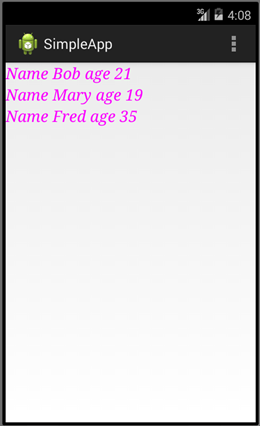
# Starting with Android Studio for Android



### Aims:

To Introduce Android Studio as our IDE for this module.

### Objectives:

* Creating projects, workspaces.
* Perspectives, Views and Editors.
* Coding and running a simple android application.
* Some debugging
* Familiarising with java

Review of Lecture Slides, and potential further reading

* What is the open software alliance?
* What is the android software stack, and what are its components?
* What are the key differences between JVM, Dalvik and ART?
* What is an APK file?
* Why would you subclass Activity and View classes?

## 1. About Android Studio

To support the lecture series we will be developing Java applications using the Android Studio IDE. It is strongly recommended that you install Android Studio at home. PCs, Macs and Linux are supported and the installation already comes with the OpenJDK <https://developer.android.com/studio>

## 2. Starting a new project

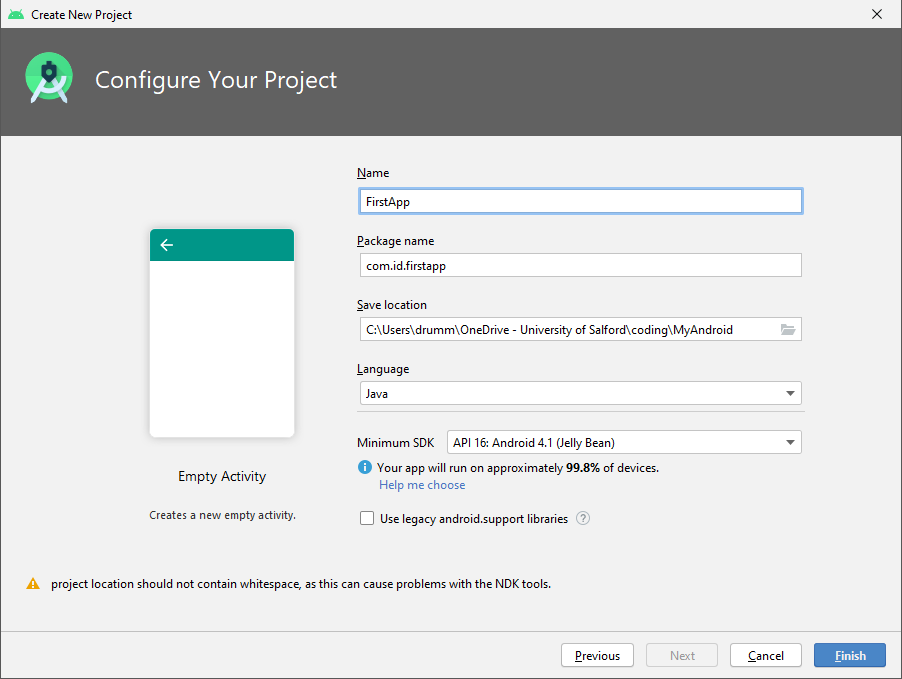
Let’s get started…

Open Android Studio and ‘Start a new Android Studio project’.

Choose an ‘Empty Activity’ and click [Next]

In the ‘Configure your project’ dialog set your Application’s …

* ‘Name’ to ‘FirstApp’ or similar.
* ‘Package name’ to com.***yourinitials***.firstapp … e.g. *com.id.firstapp*
* ‘Save location’ to a suitable directory on your OneDrive
* ‘Language’ to **Java** **(NOT Kotlin!)**
* ‘Minimum API level’ to suggested default *…e.g. API 16: Android 4.1 (Jelly Bean)*

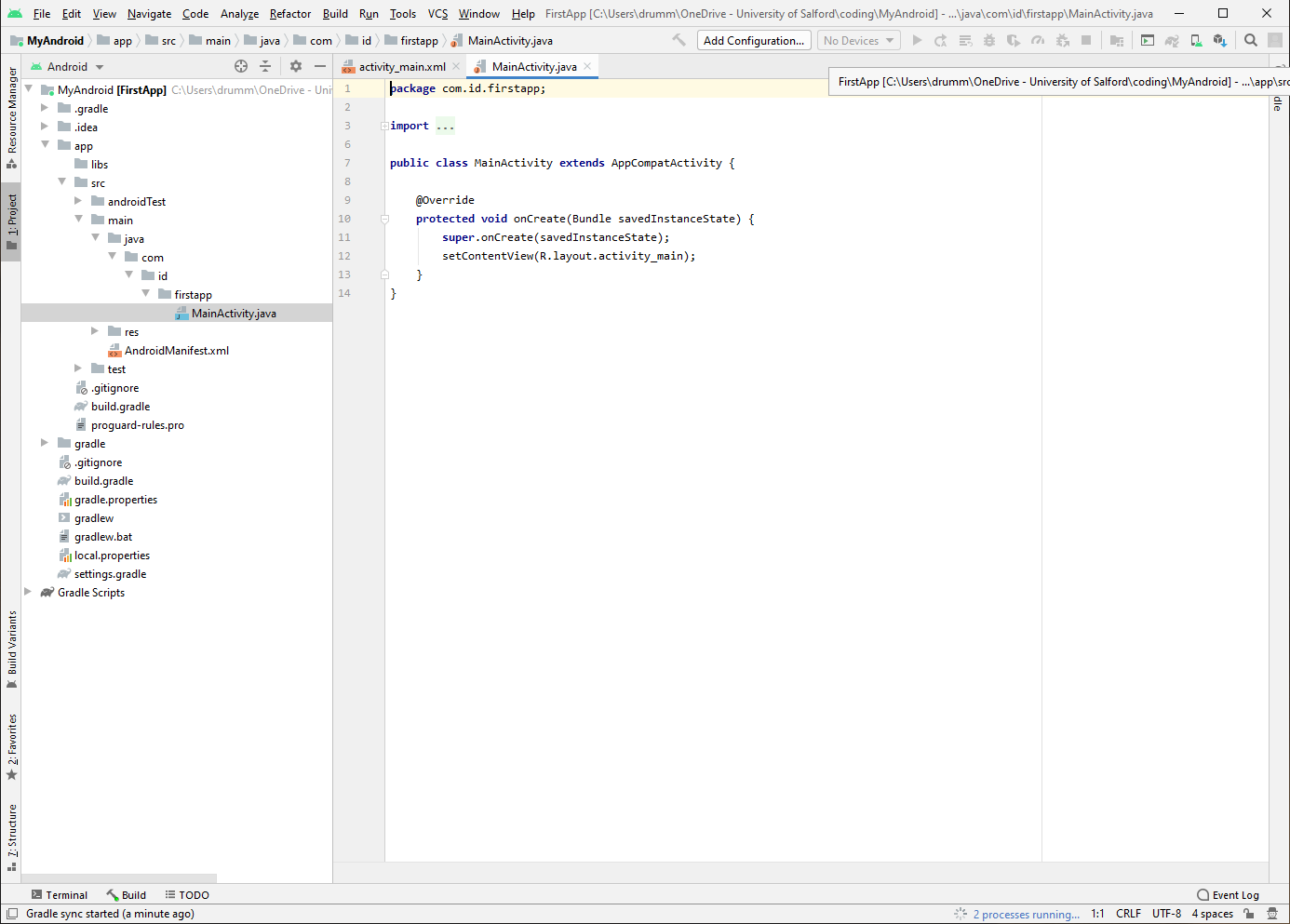


Hence click [Finish]

*Note you can ignore the warning as we’ll not be using the NDK throughout this module.*

Allow a minute or so to let Gradle complete executing tasks. Notice at the bottom right you might see ‘Processes running’ or a progress bar to indicate things are still setting up.

You will be presented with the Android Studio Integrated Development Environment.



Once working, in the resultant ‘Android ‘ pane to the left you’ll see the structure of your app.

This includes

* manifests

Containing your AndroidManifest.xml file which holds meta data associated with the app such as information about activities you’ll create, services you’ll use and permissions you’ll request from a device.

* java

Contains your package directory including your MainActvity.java source file, where you start your java programming.

* res

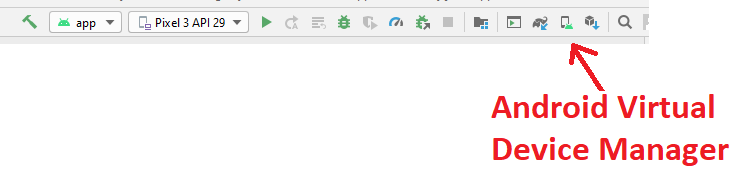
Contains directories for resources including the layout/activity\_main.xml file where you begin to define your user interface.

The default java/MainActivity.java source file extends AppCompatActivity class.

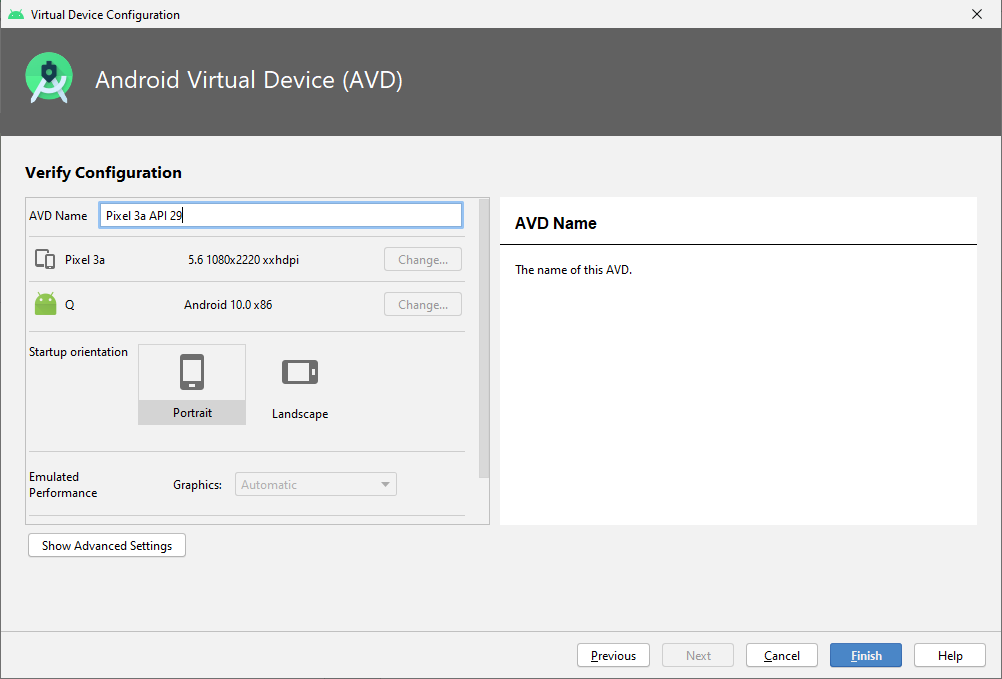
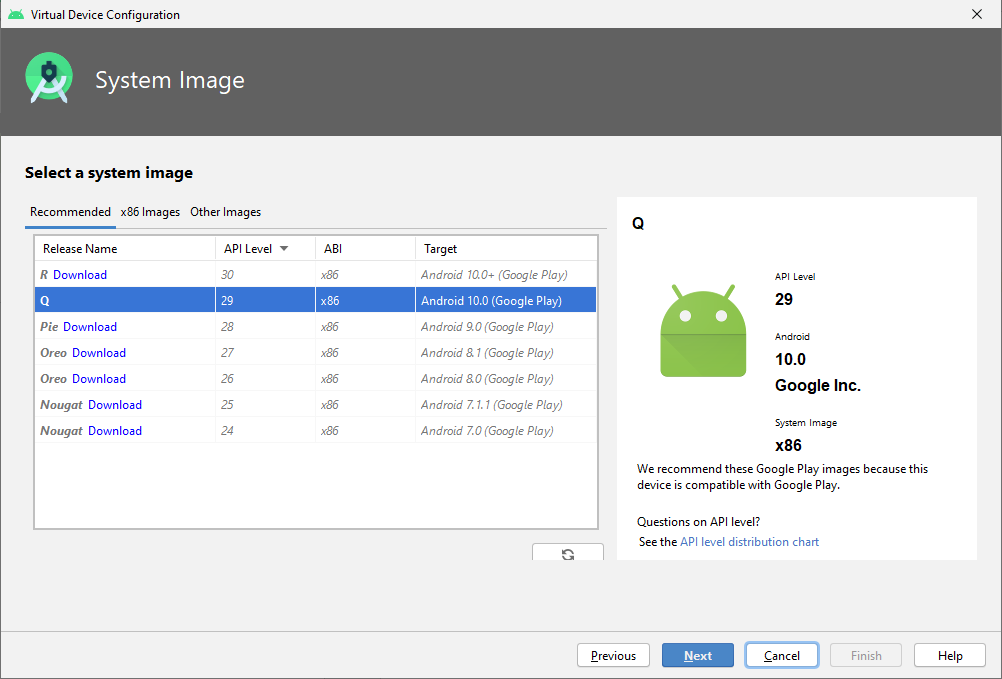
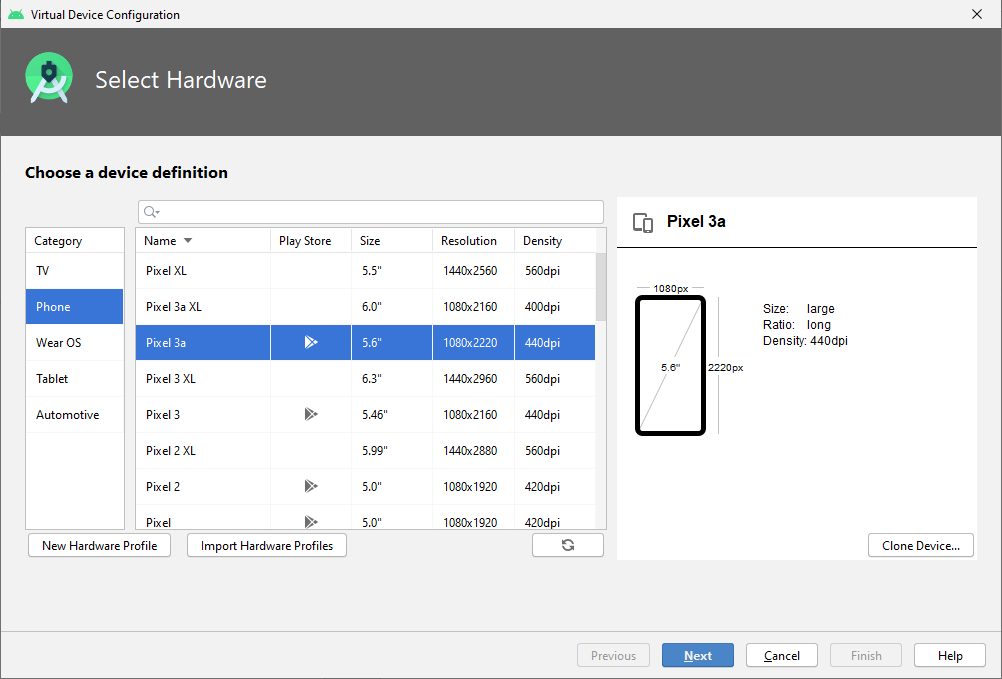
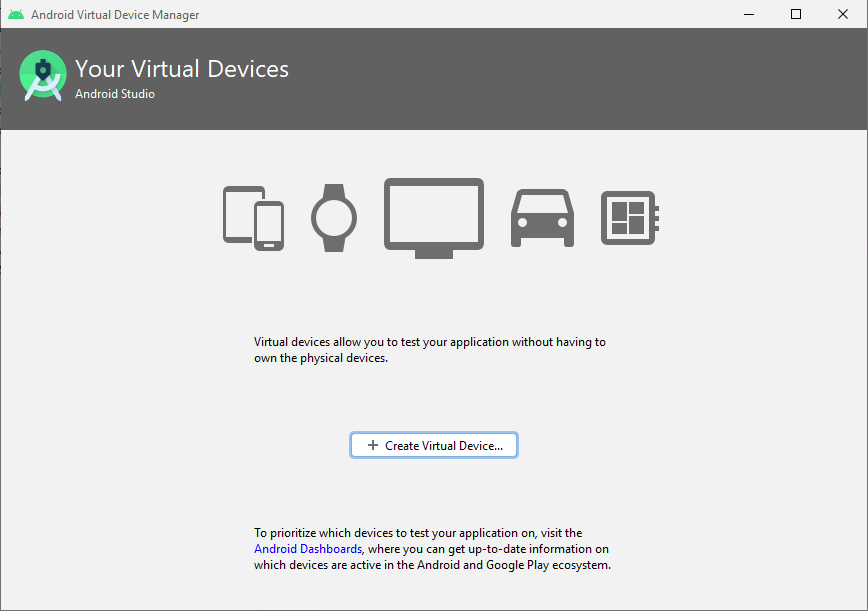
**package** com.id.firstapp;  
**import** androidx.appcompat.app.AppCompatActivity;  
**import** android.os.Bundle;  
  
**public class** MainActivity **extends** AppCompatActivity {  
  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_main***);  
 }  
}

To Run this app you need to firstly create and **Android Virtual Device** which runs in a Emulator *(or you can connect via USB a Phone with developer mode enabled, see Appendix at end of this worksheet)*.

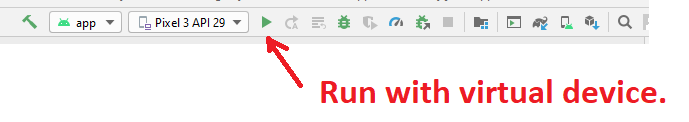
To create a virtual device select the **Android Virtual Device manager** from the main menu **Tools->AVD Manager** or the toolbar.



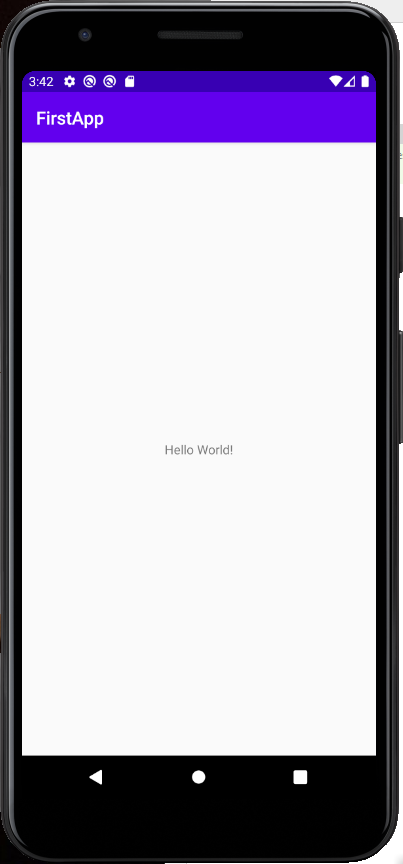
You will be prompted to “Create Virtual Device” and can hence select from a range of hardware and system images. Currently *Pixel XL API 28* is downloaded on lab machines, so go with that, though you can download other APIs for your own PC or Laptop.



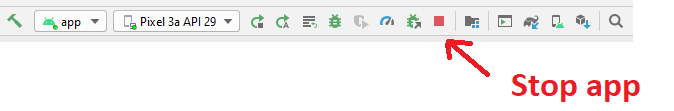
Once you’ve created your virtual device you can run it from the main menu **Run->Run ‘app’** or the toolbar.



The app shows ‘Hello World’ based on default XML code in res/layout/activity\_main.xml.



When finished click the red square to stop the app.



***Note to use an emulator your PC should support Virtualisation and have it enabled (see Appendix at the end of this worksheet).***

## 3. A Simple Hello World App

Let’s try creating the classic ‘Hello World’ application from java code.

In the MainActivity class declare as fields

LinearLayout myLayout;

TextView myTextView;

Android Studio will underline these declarations in red because it doesn’t yet know about LinearLayout or TextView. You need to add the necessary import statements for these as yet unknown classes.

Android studio can help you do this, for example click on ‘LinearLayout’ and press ‘ALT ENTER’ to import android.view. LinearLayout.

In the onCreate() method, and just above setContentView(), instantiate objects for the above references

**myLayout**=**new** LinearLayout(**this**);  
**myTextView**=**new** TextView(**this**);

call TextView’s setText() method with hello greeting.

**myTextView**.setText(**"Hello world from Java"**);

Add the TextView to the layout

**myLayout**.addView(**myTextView**);

Comment out previous call to setContentView(), hence set content view to take your layout as an argument…

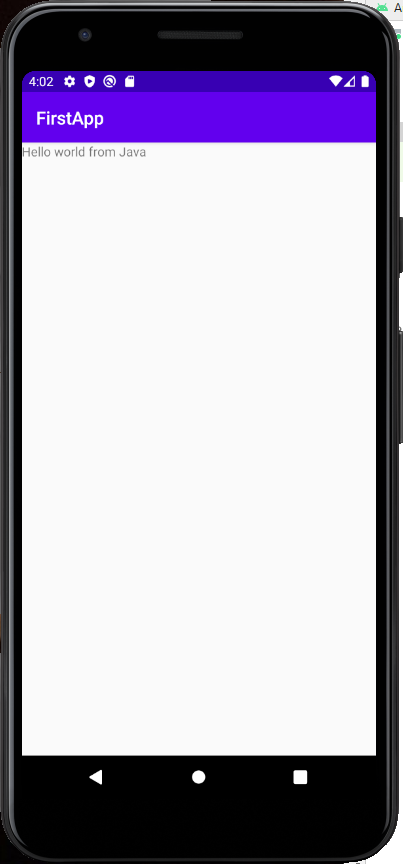
setContentView(**myLayout**);

instead of R.layout.activity\_main.

Your application should look like this…

**package** com.id.firstapp;  
**import** androidx.appcompat.app.AppCompatActivity;  
**import** android.os.Bundle;  
**import** android.widget.LinearLayout;  
**import** android.widget.TextView;  
  
**public class** MainActivity **extends** AppCompatActivity {  
  
 LinearLayout **myLayout**;  
 TextView **myTextView**;  
  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 *//setContentView(R.layout.activity\_main);* **myLayout**=**new** LinearLayout(**this**);  
 **myTextView**=**new** TextView(**this**);  
  
 **myTextView**.setText(**"Hello world from Java"**);  
 **myLayout**.addView(**myTextView**);  
 setContentView(**myLayout**);  
 }  
}

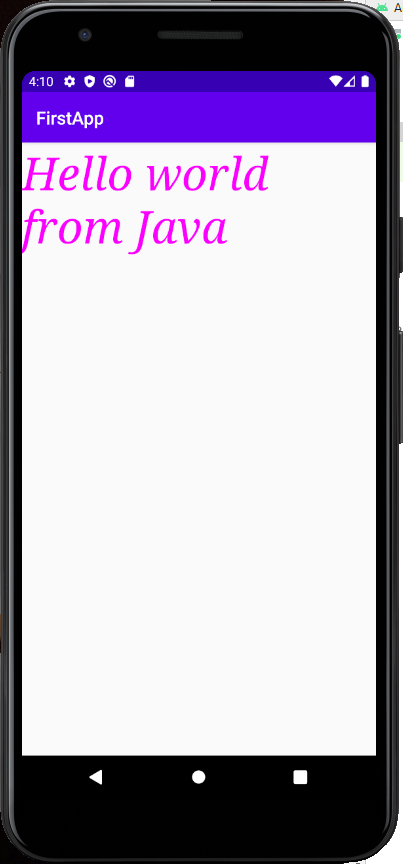
Run the Android Application.



## EXERCISE

You can use some of TextView’s numerous methods for more interesting looking text. Remember to use ‘ALT ENTER’ to resolve unknown classes, interfaces, enums, etc.

**myTextView**.setTextSize(50);  
**myTextView**.setTextColor(Color.*rgb*(250, 0, 255));  
**myTextView**.setTypeface(Typeface.***SERIF***, Typeface.***ITALIC***);



Try varying some of the parameters.

## 4. Some basic debugging with Android Studio

Android Studio has some very useful debugging facilities. For the purpose of this introduction, let’s just set a breakpoint, step through the code line by line and watch the values of variables change.

Firstly, set a break point by moving the cursor to

protected void onCreate(Bundle savedInstanceState)

and choosing from the main menu Run->Toggle Line Breakpoint.

Note a small pink circle appears on a grey vertical strip to the left of the code. Double clicking on this strip is another way of toggling breakpoints on and off.

To run the app in debug mode, from the main menu choose ‘Run’->Debug ‘app’. You can hence step over code line by line and view the state of objects, much like many other IDEs.

## Coding some Java

In the following exercises you’ll be coding Java to display data. The java syntax is identical to windows desktop apps. If you are more familiar with C# you’ll see much of Java is the same, as they both share a common C heritage.

## EXERCISE

Add to the MainActivity class the following method…

**void** showSomeNumbers()  
{  
 **int** i;  
 String str=**""**;  
 **for**(i=1;i<20;i++)  
 str=str+**"Number "** + i + **"\n"**;  
 **myTextView**.setText(str);  
}

And call the method from within onCreate() with …

showSomeNumbers();

Save and Run As an Android Application.

The method simply outputs a string containing lines separated by ‘\n’ for new line. If there are too many lines then some of the text will be off the screen.

EXERCISE

To display all you could wrap your layout in a scroll view. In the MainActivity class declare a ScrollView as a field

ScrollView **myScrollView**;

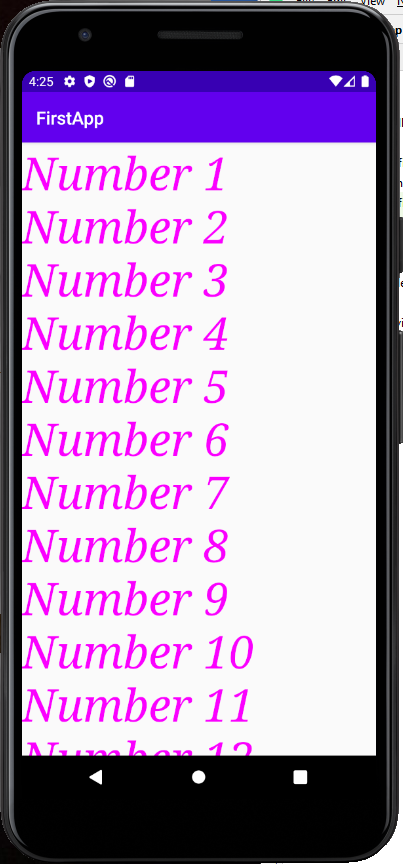
In the onCreate method add

**myScrollView**=**new** ScrollView(**this**);

comment out previous call to setContentView() and add…

**myScrollView**.addView(**myLayout**);  
setContentView(**myScrollView**);

Run the Android Application.



EXERCISE

It’ll be very useful to be able to define our own classes, subclass them and manage them in array lists.

From the main menu File->New->Java Class create a new class called Student. Android Studio should create a new Student.java file in the same package as MainActivity.java.

Add to the class protected name, age and grade fields.

**package** com.id.firstapp;  
  
**public class** Student {  
 **protected** String **name**;  
 **protected int age**;  
 **protected double grade**;  
}

You can have some fun letting Android Studio generate the constructor and getter and setter methods for the class. Ensure the cursor is in a suitable location in the Student class, hence from the main menu select **Code-> Generate**. Try to figure out what you need to do to autogenerate code so the Student class looks like…

**public class** Student {  
 **protected** String **name**;  
 **protected int age**;  
 **protected double grade**;  
  
 **public** Student(String name, **int** age, **double** grade) {  
 **this**.**name** = name;  
 **this**.**age** = age;  
 **this**.**grade** = grade;  
 }  
  
 **public** String getName() {  
 **return name**;  
 }  
  
 **public int** getAge() {  
 **return age**;  
 }  
  
 **public double** getGrade() {  
 **return grade**;  
 }  
  
 **public void** setName(String name) {  
 **this**.**name** = name;  
 }  
  
 **public void** setAge(**int** age) {  
 **this**.**age** = age;  
 }  
  
 **public void** setGrade(**double** grade) {  
 **this**.**grade** = grade;  
 }  
}

In the MainActivity class declare as a field an array list of type student with

ArrayList<Student> students = **new** ArrayList<Student>();

Also define the following method…

**void** showSomeStudents()  
{  
 **students**.add(**new** Student(**"Bob"**,21, 55.5));  
 **students**.add(**new** Student(**"Mary"**,19, 70.3));  
 **students**.add(**new** Student(**"Fred"**,35, 66.0));  
  
 String str=**""**;  
 **for**(Student s:**students**)  
 str=str+**"Name "** + s.getName() + **" age "** + s.getAge() + **"\n"**;  
  
 **myTextView**.setText(str);  
}

From onCreate() call the showSomeStudents() method.

Run the Android Application.

What does the method do?

Student objects are added to the array list then displayed with a ‘for each’ loop.

EXERCISE

Add to the Student class doubles for height, weight and a boolean for gender (male is true, female is false). Try let Android Studio generate the new constructor and appropriate getter methods.

In the showSomeStudents() method hard code five new students and display all data values of the students.

Also display the average age, height and width of the students (how might you code this?).

## Complete Code Example

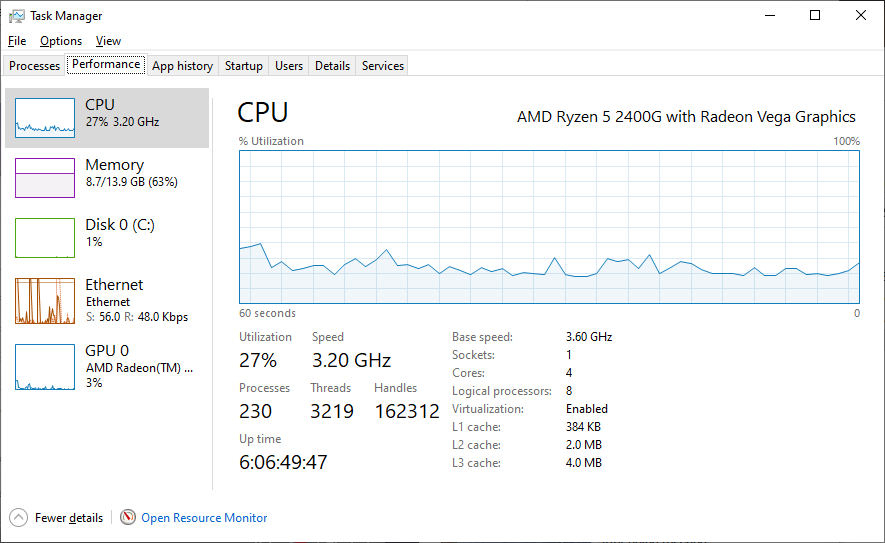
**package** com.id.firstapp;  
**import** androidx.appcompat.app.AppCompatActivity;  
  
**import** android.graphics.Color;  
**import** android.graphics.Typeface;  
**import** android.os.Bundle;  
**import** android.widget.LinearLayout;  
**import** android.widget.ScrollView;  
**import** android.widget.TextView;  
  
**import** java.util.ArrayList;  
  
**public class** MainActivity **extends** AppCompatActivity {  
  
 LinearLayout **myLayout**;  
 TextView **myTextView**;  
 ScrollView **myScrollView**;  
 ArrayList<Student> **students** = **new** ArrayList<Student>();  
  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 *//setContentView(R.layout.activity\_main);* **myLayout**=**new** LinearLayout(**this**);  
 **myTextView**=**new** TextView(**this**);  
 **myScrollView**=**new** ScrollView(**this**);  
  
 **myTextView**.setText(**"Hello world from Java"**);  
 **myLayout**.addView(**myTextView**);  
 *//setContentView(myLayout);* **myTextView**.setTextSize(50);  
 **myTextView**.setTextColor(Color.*rgb*(250, 0, 255));  
 **myTextView**.setTypeface(Typeface.***SERIF***, Typeface.***ITALIC***);  
 showSomeNumbers();  
  
 **myScrollView**.addView(**myLayout**);  
 setContentView(**myScrollView**);  
  
 showSomeStudents();  
 }  
  
 **void** showSomeNumbers()  
 {  
 **int** i;  
 String str=**""**;  
 **for**(i=1;i<20;i++)  
 str=str+**"Number "** + i + **"\n"**;  
 **myTextView**.setText(str);  
 }  
  
 **void** showSomeStudents()  
 {  
 **students**.add(**new** Student(**"Bob"**,21, 55.5));  
 **students**.add(**new** Student(**"Mary"**,19, 70.3));  
 **students**.add(**new** Student(**"Fred"**,35, 66.0));  
  
 String str=**""**;  
 **for**(Student s:**students**)  
 str=str+**"Name "** + s.getName() + **" age "** + s.getAge() + **"\n"**;  
  
 **myTextView**.setText(str);  
 }  
}

Student.java

**package** com.id.firstapp;  
  
**public class** Student {  
 **protected** String **name**;  
 **protected int age**;  
 **protected double grade**;  
  
 **public** Student(String name, **int** age, **double** grade) {  
 **this**.**name** = name;  
 **this**.**age** = age;  
 **this**.**grade** = grade;  
 }  
  
 **public** String getName() {  
 **return name**;  
 }  
  
 **public int** getAge() {  
 **return age**;  
 }  
  
 **public double** getGrade() {  
 **return grade**;  
 }  
  
 **public void** setName(String name) {  
 **this**.**name** = name;  
 }  
  
 **public void** setAge(**int** age) {  
 **this**.**age** = age;  
 }  
  
 **public void** setGrade(**double** grade) {  
 **this**.**grade** = grade;  
 }  
}

##### APPENDIX

If you are using an **emulator** and it is running too slowly you may need to ensure your PC has virtualisation technology, and it’s enabled. For example, search for **Task Manager**, [Performance], check that **Virtualization** is Enabled.



If not, you will need to reboot boot your computer to bios, (e.g. pressing delete key or F2 key when you restart) and enable virtualization, e.g. via Advanced->CPU Configuration enabling either …

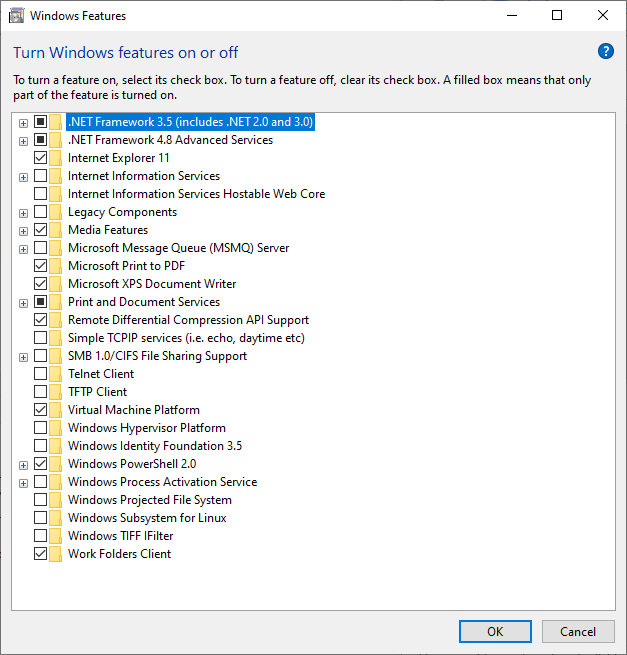
* Intel Vitualization Technology … or
* SVM mode (if you have an AMD cpu).

When windows is restarted…

Search for **Turn Windows features on or off**,

* Check **Virtual Machine Platform**
* Uncheck **Windows Hypervisor Platform**

Open Android Studio and create a Virtual Device and try it out.



**If you don’t have a PC with Virtualisation,** you can instead develop to a smart phone.To do so your phone you’ll need to enable for your phone for mobile development. How to enable this differs for different phones, so you might need to google around. For example, via your phone’s **Settings->About Phone->Version** press Build number 7 times. When enabled search Settings for **Developer Options** hence enable **USB Debugging**.