|  |
| --- |
| Daily Goal |

***Resources and References***

**Setting up Android Studio and Android SDK** [<http://developer.android.com/sdk/installing/index.html>](http://developer.android.com/sdk/installing/index.html)

**SDK Manager**

<http://developer.android.com/tools/help/sdk-manager.html>

**Creating Android Project**

<https://developer.android.com/training/basics/firstapp/creating-project.html>

**Running your App**

<http://developer.android.com/tools/building/building-eclipse.html>

**Simple RadioButton Usage** <http://androidexample.com/Radio_Button_Basics/index.php?view=article_discription&aid=78&aaid=102>

**Starting another Activity**

<http://developer.android.com/training/basics/firstapp/starting-activity.html>

**Using a version control system (Github)**

<https://code.tutsplus.com/tutorials/working-with-git-in-android-studio--cms-30514>

Table of Contents

[Daily Goal 1](#_Toc511348165)

[Setting up the environment for Android development 2](#_Toc511348166)

[General steps on creating a project 2](#_Toc511348167)

[Setting up and using Git repository 3](#_Toc511348168)

[Introduction to ScrollView 3](#_Toc511348169)

[Introduction to RadioGroup and RadioButton 7](#_Toc511348170)

[Introduction to Activities 9](#_Toc511348171)

[Handling the Problem Statement 14](#_Toc511348172)

## Setting up the environment for Android development

1. Please refer to the Android Studio Guide.docx
2. For this module, it is recommended to create a dedicated workspace for C347 & Android related projects.

## General steps on creating a project

1. In the Android Studio Guide, you’ll find steps on creating a new Project.
2. Generally, these are the few information needed for a new Android project:
   1. **Application Name** which is basically the name you want to give to your application. Give it a meaningful name as users would see this name with your icon on their devices. For now, let’s give it the name **My First App**.
   2. **Domain Name** typically refers to a domain a company owns. This will form part of the package name. In this example, android.myapplicationdev.com
   3. **Package Name** is not shown to the users but distinguish your app from another company’s even when two apps have the same name. Package name usually is the reverse domain name of your company. Do use different package name for each week’s problem as Android installs one app over the older one if the older one has the same package name (even when the application names are different). It is usually auto generated but you can modify it.
   4. **Minimum Required SDK** is the place you state the version of Android a device must have to run your app. We usually use the lowest possible to reach as many user as possible because user with older phone (likely do not have latest Android OS version) can use your app. But for our worksheet, we want to learn to use the relatively new SDK features so choose **API 16: Android 4.1**.
   5. **Target SDK** and **Compile With** are usually given the most **recent** version of SDK you have for your development. In this module, set to **API 28: Android 9.0 (Pie)**
3. Next few dialogs let you specify the name of the first activity and its layout file and you can leave all of them as default unless stated otherwise.
4. Do proceed to run the newly created app on an emulator. We will based the emulator on **API 28**. You might want to recreate it if necessary.

## Setting up and using Git repository

1. Please refer to the Github Guide.docx
2. For this module, you need to check in your projects into Git.
3. Do provide the URL to your Github repository here:

|  |
| --- |
| [https://github.com/NormanLeo](https://github.com/NormanLeo/C37) |

1. For an Android Project to be deposited into the repository, you need to import into a Git repository. Do refer to the guide to import the project into the repository.
2. During the development, you shall commit the changes into the repository regularly.

## Introduction to ScrollView

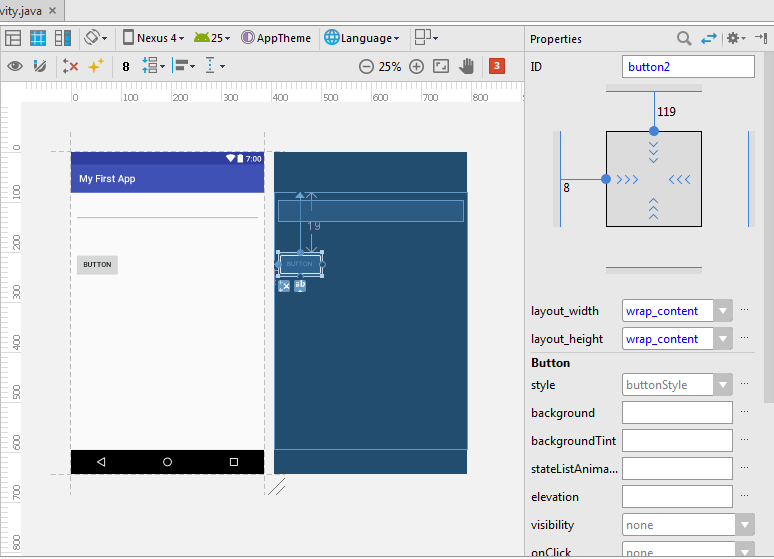
1. Create a new project and name it **Demo ScrollView**, with default values for Activity name layout name.

|  |  |
| --- | --- |
| Project Name | Demo ScrollView |
| Package | com.myapplicationdev.android.demoscrollview |
| Activity Name | MainActivity |
| Layout Name | activity\_main.xml |
| Min SDK | API 16 |

1. Import the created project into your Github repository. Copy the url of the project here

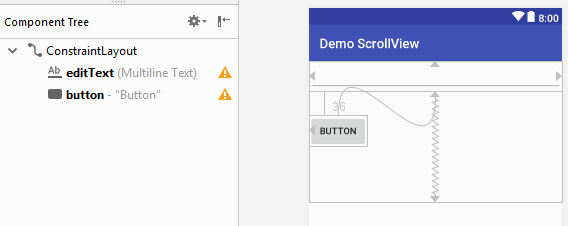
|  |
| --- |
| <https://github.com/NormanLeo/DemoScrollView> |

1. It is very important to make your app scrollable when the information it displays goes beyond the visible area. Generally, it is a good design guideline to let user scroll vertically only and not horizontally.
2. Double click on **activity\_main.xml** under your project **res/layout** folder.
3. Notice that there’s a new ViewGroup that is being used by default. It is called ContraintLayout. You could regard it as an upgraded version of the RelativeLayout.

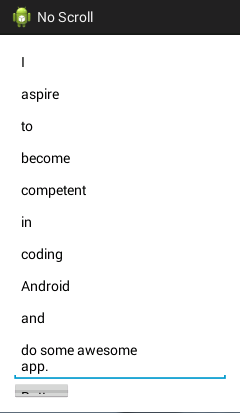


Do make use of the new interface specially meant for ConstraintLayout. Do set the constraints to each view. You could switch to LinearLayout if you wish to.

1. Delete the automatically placed **Hello world!** TextView and then add a multiline EditText as the first child in the layout (look for **multiline Text** that you can drag from the Palette). Next add a button underneath the EditText.

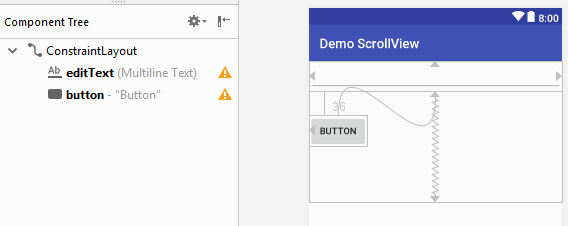
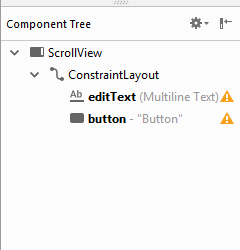
Do take note of the constraints shown below.   


1. Run the app on either a device or emulator and type into the EditText until it goes beyond the bottom of the screen. The button is hidden from view and user cannot click on the button now.



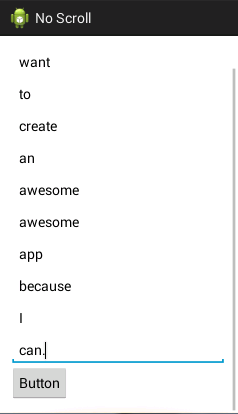
1. This is an serious blunder in user interface design. How to solve this issue? Simply use **ScrollView**.
2. ScrollView is a layout container / ViewGroup. To use ScrollView to encompass an existing layout, you need to edit it via the Text mode, in XML.

Basically, in terms of the UI components, you need to modify it to be like the sample on the right, below.

1. A sample of how the UI will look like in XML is as follow:

|  |
| --- |
| *<?*xml version="1.0" encoding="utf-8"*?>* <ScrollView xmlns:android="http://schemas.android.com/apk/res/android"  xmlns:app="http://schemas.android.com/apk/res-auto"  xmlns:tools="http://schemas.android.com/tools"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent"  tools:context="com.myapplicationdev.android.myfirstapp.MainActivity">   <android.support.constraint.ConstraintLayout  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content">   <Button  …  />   <EditText  Fill in the missing properties  …  />   </android.support.constraint.ConstraintLayout>  </ScrollView> |

1. Run the app on either a device or emulator and type into the EditText until it goes beyond the bottom of the screen. Call you scroll vertically when it happens? Thanks to ScrollView, it is so easy to have this useful feature that your users would take for granted.  
     
   

Note: Commit changes into Github.

## Introduction to RadioGroup and RadioButton

1. Create a new project and name it **Demo RadioButton**.

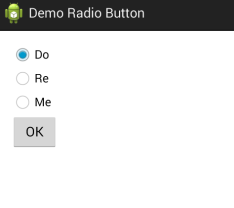
|  |  |
| --- | --- |
| Project Name | Demo RadioButton |
| Package | com.myapplicationdev.android.demoradiobutton |
| Activity Name | MainActivity |
| Layout Name | activity\_main.xml |
| Min SDK | API 16 |

1. Import into Github and do provide the URL to your Github repository here:

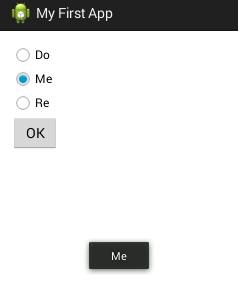
|  |
| --- |
| https://github.com/NormanLeo/DemoRadiobutton |

1. RadioGroup allows user to select only RadioButton in its group at any one time. In another word, RadioButton in a RadioGroup can only be mutually exclusively selected.
2. Aside from using ContraintLayout, you can change the root layout to **LinearLayout (Vertical)** and drag a RadioGroup into the LinearLayout.
3. Drag 3 RadioButton into the grouping. You can change the **Orientation** under **Properties** to have the radio buttons laid out horizontally or vertically.
4. Go to the **activity\_main.xml** XML code, the corresponding XML code for the RadioButton UI would look similar to the one given below.

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 | <RadioGroup android:id=*"@+id/radioGroup1"*  android:layout\_width=*"wrap\_content"*  android:layout\_height=*"wrap\_content"*  android:orientation=*"****vertical****"* >  <RadioButton android:id=*"@+id/radio0"*  android:layout\_width=*"wrap\_content"*  android:layout\_height=*"wrap\_content"*  android:checked=*"true"* android:text=*"Do"* />  <RadioButton android:id=*"@+id/radio1"*  android:layout\_width=*"wrap\_content"*  android:layout\_height=*"wrap\_content"*  android:text=*"Re"* />  <RadioButton android:id=*"@+id/radio2"*  android:layout\_width=*"wrap\_content"*  android:layout\_height=*"wrap\_content"*  android:text=*"Me"* />  </RadioGroup> |

1. Now add a button under the RadioGroup. Leave the id as **@+id/button1**. But change its text to **OK**.
2. The layout in shown in the *Design* tab should look like this:  
     
   
3. Go to **MainActivity.java** and add in the code below (in the red box), when the button is clicked, to display a Toast on the radio button chosen. Do import all relevant namespace so the compiler could find the classes for Button and RadioGroup.

|  |
| --- |
|  |

1. The code above aims to get the radio button selected in the following steps.
   1. Get the RadioGroup object using findViewById();
   2. From the RadioGroup object, get the selected radio button Id by calling RadioGroup method getCheckedRadioButtonId();
   3. Get the radio button object by passing in the radio button Id into findViewById();
   4. To get the text on the radio button, called the radio button object method getText().
2. Run the app and does it show a Toast of the radio button you had selected?  
     
   

Note: Commit changes into Github.

## Introduction to Activities

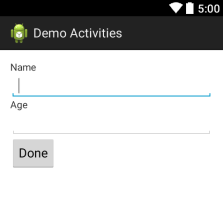
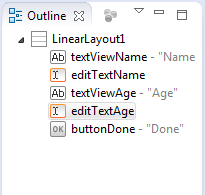
1. Create a new project and name it **Demo Activities**, its **Activity Name** as **DemoActivities** and **Layout Name** as **activity\_demo\_activities**.

|  |  |
| --- | --- |
| Project Name | Demo Activities |
| Package | com.myapplicationdev.android.demoactivities |
| Activity Name | DemoActivities |
| Layout Name | activity\_demo\_activities.xml |
| Min SDK | API 16 |

1. Import into Github and do provide the URL to your Github repository here:

|  |
| --- |
| https://github.com/NormanLeo/DemoActivities |

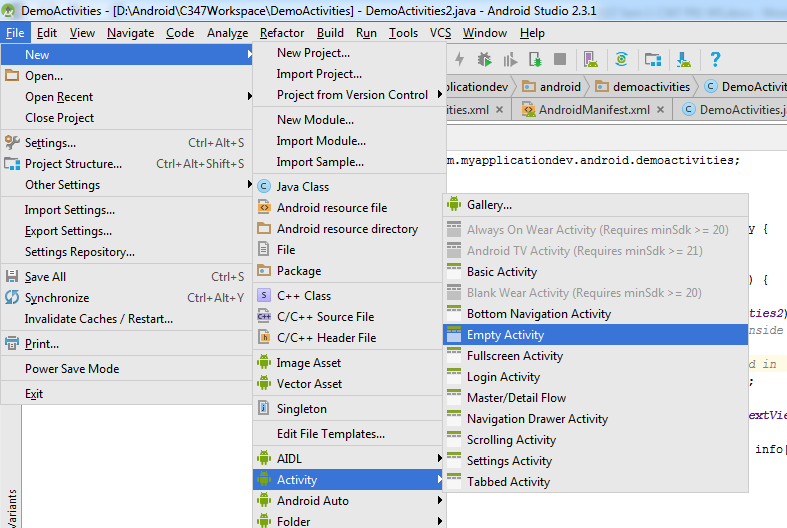
1. Open the activity layout file in Graphical Layout and delete the automatically created TextView “Hello World!”.
2. You have the option to retain the ConstraintLayout, or change the root Layout to **LinearLayout (Vertical)**.
3. Add two TextView objects, two EditText objects and a button to the layout as below. Use the same IDs as shown in Outline.

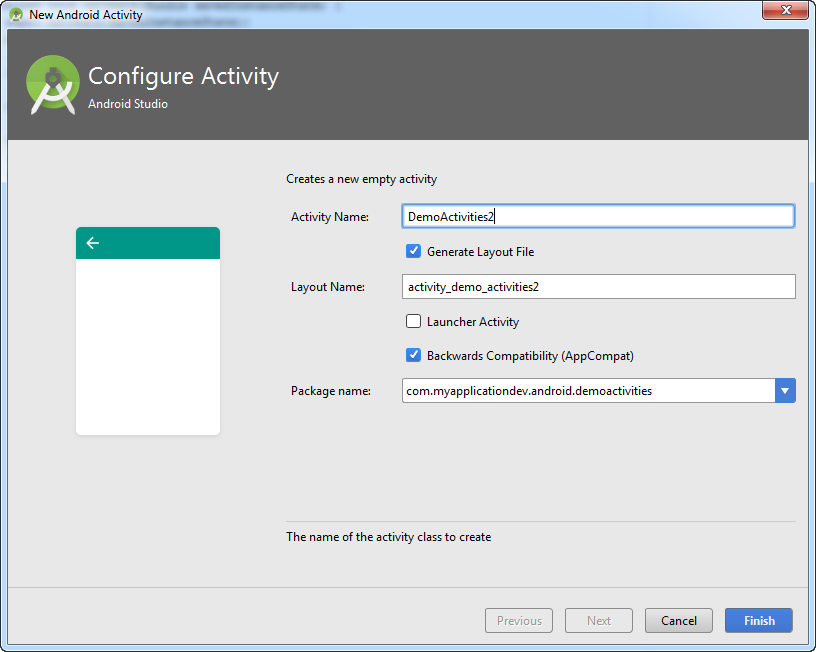
1. Next we add in the code to get the name and age from the EditText objects when the button is clicked. Then we start a new activity and also pass the name and age to this new activity.
2. Put in the code inside the **onCreate()** method of **DemoActivities** class.

|  |
| --- |
|  |

1. You would see **DemoActivities2** on line 21 above marked as error but that’s because we have not created this activity yet. Let’s do it next.
2. Select **File>** **New > Activity > Empty Activity…** from the menu.



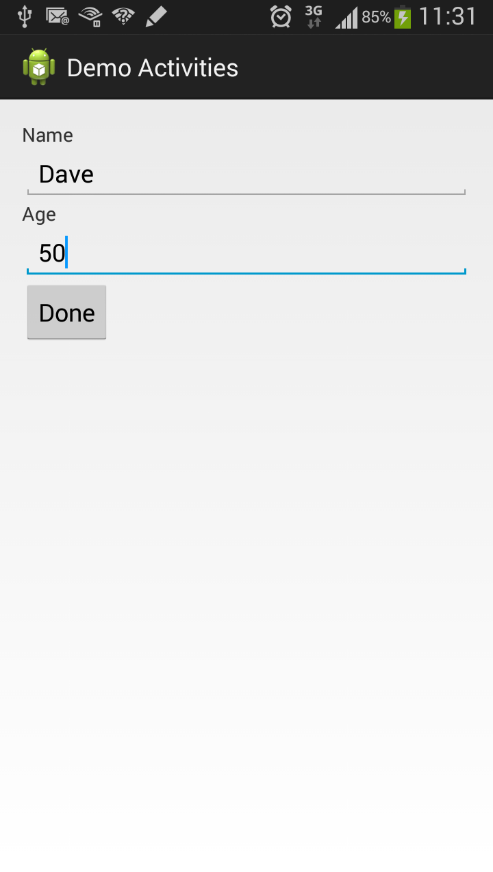
1. At the following dialog enter the new Activity name as well as the corresponding Layout name. Leave the rest of the attributes of the activity in default values but do make sure that the package is correctly filled too!



|  |  |
| --- | --- |
| Package | com.myapplicationdev.android.demoactivities |
| Activity Name | DemoActivities2 |
| Layout Name | activity\_demo\_activities2.xml |

1. Open its layout file **activity\_demo\_activities2.xml** and change its root layout to **LinearLayout (Vertical)** or retain it as ContraintLayout.
2. Add a TextView under the LinearLayout and leave its ID as **textView1**.
3. Now add the code to retrieve the name and age we passed to this activity a while ago. Also we would make **textView1** display this info.

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 8 9 10 11 12 13 | @Override  **protected** **void** onCreate(Bundle savedInstanceState) {  **super**.onCreate(savedInstanceState);  setContentView(R.layout.*activity\_demo\_activities2*);  // Get the intent so as to get the "things" inside the intent  Intent i = getIntent();  // Get the String array named "info" we passed in  String[] info = i.getStringArrayExtra("info");  // Get the TextView object  TextView tv1 = findViewById(R.id.*textView1*);  // Display the name and age on the TextView  tv1.setText("You are " + info[0] + ", age " + info[1]);  } |

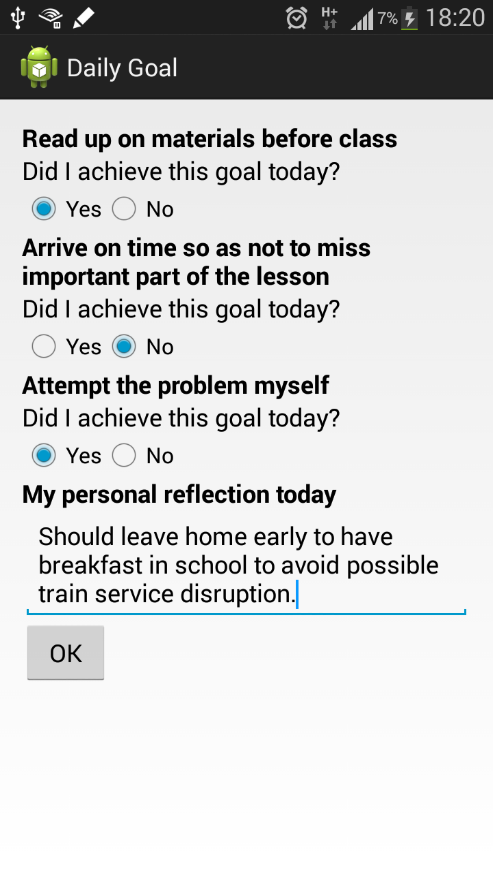
1. Run the app and do you achieve the following?  
     
    

## Handling the Problem Statement

1. You could use the suggested values below for the project.

|  |  |
| --- | --- |
| Project Name | P01-DailyGoals |
| Package | com.myapplicationdev.android.p01\_dailygoals |
| Activity Name | change as necessary |
| Layout Name | change as necessary |
| Min SDK | API 16 |

1. The problem statement states a **Daily Goal** app that shows a few goals, radio buttons and a personal reflection.

Identify all the View objects from the activities below.  
  
 

EditText (Multiline)

Button

RadioButton

TextView

TextView (Large)

1. Use a ScrollView so that the user can scroll down when the reflection is too long and hides the button.
2. How do you achieve the layout of the app that View objects placed row after row?

mop

1. You would need to create a second activity to show the summary. And how do you pass the values user enters in the first activity to the second activity?

By using Intent