**Overwhelming Android Review**

* **Anything we draw on the GUI contents layout will be transferred to XML code in the XML file**
* **R.java file contains all elements you add to your programs and gives unique number for each one of them**
* **String.xml file contains all strings you have defined for your program to be in use and this makes your program easy to be translated to other languages**
* **Src file: contains all you java file of your program and code to be compiled**
* **Gen file: contains R.java file**
* **Res file: contains all your file resources like: pictures layout drawing string.xml file**
* **To display message use this code:**

Toast.makeText (getBaseContext (), "All data saved!", Toast.LENGTH\_LONG ).show();

* **Manifest.xml file contains all data about your layout like: action and category**

1. **Action: it’s something we use to call this layout in code**
2. **Category: if two layouts have the same action, category is the way to differentiate between them**

**We write action and category inside intent-filter taps**

* **in mainfesr.xml if you didn’t put intent-filter tag inside the activity you block other programs to call this activity**

**EX:**

<activity

android:name =*".MainActivity"*

android:label=*"@string/app\_name"* >

<intent-filter>

<action android:name =*"android.intent.action.MAIN"* />

<category android:name =*"android.intent.category.LAUNCHER"* />

</intent-filter>

</activity>

* **to define new element in code:**

Button saver = (Button) findViewById (*R.id.delete\_editdelete*);

* **findViewById() is a methods take an int represent the id of your element in R.java file to return object of this element**
* **there is a method for each element called setOnClickListener() it take interface of OnclickListenr**

**EX:**

saver.setOnClickListener (**new** OnClickListener () {

@Override

**public** **void** onClick (View v) {

//your code here

}});

* **types of layout:**

**1- Linear**

**Arranges elements vertical or horizontal**

**2- Table**

**3- Relative**

**We define elements position in relative to other elements**

**4- Frame**

**We use it to put some views all in the same place above each other so we can use them to display things after each other**

**also we can put linear layout inside this frame layout to make different shapes**

**5- Scroll view**

**We use it when we need a lot of views more than the screen space**

**It gives you scrollbar when run**

**It takes only one element so it should be group view**

* **oncreate method: it will run when the activity is opened automatically**

@Override

**protected** **void** oncreate (Bundle savedInstanceState) {

**super**.onCreate (savedInstanceState);

//we use this line to hide the project name form the program

requestWindowFeature (Window.*FEATURE\_NO\_TITLE*);

setContentView (R.layout.*deleteall*);

}

* **If we don’t want the name of the activity to be visible in the user screen we add this line:**

requestWindowFeature (Window.*FEATURE\_NO\_TITLE*);

* **This line defines which layout will work onscreen with this activity:**

setContentView (R.layout.*deleteall*);

* **Activity Life Cycle for any Android activity:**

****

* **Note that:**

**If you leave an activity A and go to activity B**

1. **A will call onPause () method**
2. **B will call oncreate () method**
3. **A will call onstop () method**
4. **B will have its rest of life cycle**

* **we can add views ---> such as buttons, textview or edittext or we can add groupview which can contain some views; so View is the super class for all GUI elements we use in our Application**
* **Intent Is a class helps you to move from current activity to another one which can be another one in you program or form the Android system**
* **To use intent to go to another activity:**

Intent i = **new** Intent ();

i.setAction ("android.intent.action.deleteall");

startActivity (i);

**Notice in setAction () method we have used the action of the activity we want to move to it**

* **To send data to the activity we call by the intent we use:**

i.putExtra ("name", texter.getText());

**First argument of the method is the key we will use to get this data and the second argument is the data we want to send**

* **To get data we have sent:**

String s = i.getExtras ().getString ("name").toString ();

**Notice we have used the key “name” in the method of method getExtras to get data**

* **To open web by the intent:**

Intent i = **new** Intent (Intent.*ACTION\_VIEW*);

i.setData (Uri.*parse* ("http:// www.facebook.com"));

* **To call number by the intent:**

Intent i = **new** Intent (Intent.ACTION\_DAIL);

i.setData (Uri.*parse* ("tel: 01144010476"));

* **To open an activity by the intent only to get data:**

1. **Call startActivityForResult() in main activity, it takes two argument the first is the intent and the second is the request number which is a unique number for this activity with this intent:**

Intent i = **new** Intent();

i.setAction ("android.intent.action.second");

i.addCategory ("aboahmed.dataproject");

startActivityForResult (i, *77*);

1. **Create onActivityResult () you must build it and it take 3 arguments the first is the request number, the second is resultCode which is number we get to check the data is valid and the third is the returning intent to get data form it:**

**public** **void** onActivityResult(**int** requestCode , **int** resultCode , Intent data)

{

//request code to specify the activity sends data

//resultCode to check data is valid

**if** (requestCode == *request\_code\_second* && resultCode == Activity.*RESULT\_OK*)

{}

}

1. **In the another activity we should call setResult () method for our intent it takes two arguments first is resultcode and the second is the intent**

Intent x = **new** Intent ();

setResult (Activity.*RESULT\_OK*, x);

1. **We end the anther activity be the method finish()**

* **To make the activity appears as a dialog in your app go to manifest and add this line for you activity description:**

<activity

android:name=*".deleteall"*

android:theme=*"@android:style/Theme.Dialog"*

>

* **we can put space between the elements and page border by adding that in the xml:**

android:layout\_marginTop*="20dp"*

* **we can specify the width and height of layout to fill the page or with specific distance**

**Fill -->** android:layout\_height=*"fill\_parent"*

**Wrap the content of the page -->** android:layout\_height="*wrap\_content"*

**Dot per pixel -->** android:layout\_height =*"150dp"*

* **any element in the page can have property called layout weight to specify the amount of space it would take from the layout GUI**

**write --->** android:layout\_weight = *""*

**It takes double number from 0 to 1**

* **always adjust id after putting any element because its id defines the position of other elements in relative layout**
* **we can make array of strings in string.xml file**

<string-array name="work">

<item >1st one</item>

<item >2nd one</item>

<Item>3rd one</item>

</string-array>

**We can use it by code and use it in program and that make it easy to translate our app to another languages**

String y [] = resourse.getStringArray (R.array.work);

* **we can save colours in xml file using code**

**notice: we can define our xml file an put it in values file in res file and put any strings, colours or data in it and R.java will see all data we put in this file**

<color name="red">#ff00f3</color>

<color name="yellow">#ff02dd</color>

**And we can call it from code with**

int color = resourse.getColor (R.color.red);

**Or get it from xml files with** // this way is good for fast change or program style

android:textColor*="@color/yourcolorname"*

* **resources reference syntax:**

**Get any element of resources in the xml files**

*@type/name*

**But in xml files, if we wrote**

*@+type/name*

**The + sign means if the element is not exist creat it**

**To make element in the xml file without using the + before the id**

**We go to any xml file**

**We define it the id we want:**

<item type="id" name="eleName"></item>

**Then i use it in the xml file**

**EX:**

<EditText name="@id/eleName">

**Notice we didn't use + sign**

* **when we set the dimension of any element we should use dp –dot per pixel -unit so the program has the same relative size for each small or big device, in some projects event if we have used dp the elements are not very visible due to different dots per inch differences**
* **in smart devices dots per inch are divided into groups**

**low / medium / high / xhigh**

* **it's good to use sp unit with font in elements because keeps the scale of font for different screen size –the same as dp but for fonts-**
* **any photo we intend to use in our app should be loaded in drawable xml file**
* **we should put various copies of our photo with different size to help android use suitable photos for each device due its screen size and its dots per unit**
* **to add image background in a layout by code in xml file:**

android:background=*"@drawable/your\_pic\_name"*

* **we can set background color using color drawable**

**We set in xml a drawable using**

<drawable name="goal">#ddffdd</drawable>

**Then we call it using in code in any java file**

ColorDrawable cd = (ColorDrawable) resourse.getDrawable (R.drawable.goal);

btn.setBackgroundDrawable (cd);

* **to save dimensions in a file**

**In an xml file**

<dimen name="x"> 12dp </dimen>

**In the property xml file**

android:layout\_width=*"@dimen/xtra"*

**So i can change it for the program easily**

Btnx.setFontSIze (R.dimen.x);