

# Android - Introduction

## Project and Company name

After clicking on the “new project” you will have to input your application name (which will be the application’s name in the google play store together with the company url you entered, this is your unique app id in the google play store - can be change later on and before uploading the app to the google play store)

## Activity

Is the component of an app. Activity is a single screen with the logic behind it.

Each activity has a Java file which inherits from “Activity” or “ActionBarActivity” (more options later on) if you want to use an action bar, and a proper xml file (UI) - By Using the Blank Activity template the system automaticlly creates “MainActivity.java” in the package you have enetred and an activity\_main.xml file that will holds the initial screen of that activity, the file created in the res/layout folder.

The Activity class has to overrides a function called “**onCreate**” which is used to connect the xml file to the java and do additional settings after that. When the activity first launched by the system (later we will discuss how to tell the system to launch an activity) it creates an instance of that activity and immediately calls the function “onCreate” which will then load the proper xml file that represents the initial UI for that activity.

To change the connectivity to another xml file you must change the **SetContentView** function in the onCreate to another xml file (using the “R” class) - only in later activities when you add your own Activity java file and a xml file. The **R file** contains ID of each resource - this ID’s are simply integers that allows us to make a connection between the xml file and the java code.

## xml layout file

is the way to visualize your program on the screen.

You can toggle between the code of the xml and the preview at the bottom of the screen (Design , Text).

## Layouts

The root element of each layout xml file is a Layout. All layout inherits from ViewGroup which intended to hold separate views according to a particular logic. Inside the xml file will start with a

layout (RelativeLayout,LinearLayout,TableLayout etc) The most common layouts is the “RelativeLayout” and the “LinearLayout”.

**RelativeLayout** - will position the elements relatively (to each other or to their parents).

**LinearLayout** - positions the elements in a form of a linear layout (one after the other - can vertical or horizontal and determined by the **layout\_orientation** attribute).

## layout\_width and layout\_height

for every content you must define the layouts height and width.

You can either use a constant size or a relative size.

Relative sizes:

wrap\_content - is used to use the minimal amount for it to wrap the whole content.

match\_parent - match the content size to the same size as its parent.

fixed sizes can be determined by: px(not recommended since each screen has different amount of pixels), dp or sp.

dp/dip = is a way to have a relative size for anything (except text) - device independent pixels

sp/sip = same as dp just for text - scale independent pixels - when the users ask for larger text in the settings the sp will be changed accordingly.

## id

this is unique indentidiar (per a single xml file, in different xml files you can give same id's) that will be used to retrieve the element in the java code - the id will automatically be added to the R file and by typing R.id.THE ID you can get the reference to that element.

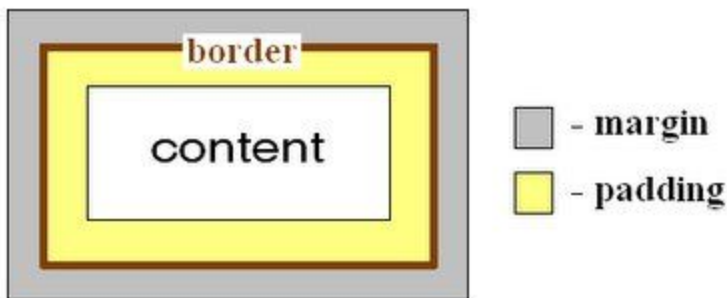
## padding and margin

Both are used to separate the inside content from the other elements in the layout.

**Padding** is the space inside the border, between the border and the actual view's content.

Note that padding goes completely around the content: there is padding on the top, bottom, right and left sides (which can be independent).

**Margins** are the spaces outside the border, between the border and the other elements next to this view. In the image, the margin is the grey area outside the entire object. Note that, like the padding, the margin goes completely around the content: there are margins on the top, bottom, right, and left sides.



## Gravity

will define the position of the selected content. You can either use just gravity to manipulate the inside content position or the “**layout gravity**” to manipulate the element in his wrapper, meaning to position the element in his layout.

## dimen xml file

dimens xml files are made for defining features in the xml to be flexible (meaning the defining features will change depending on the screen size or other)

You can create different dimens and assign them to a certain kind of machine (screen size etc.) this will be done at the creation of the dimen. Each dimen file will be chosen automatically by the system when the user running your app and the correct size will be loaded. For example in the xml file the layout padding size we can see `dimens/horizonatal_margin` and not fixed size, this will result in the different sizes according to the screen size(phone or tablet, and will be done automatically).

## string xml file

Same idea as dimen. instead of hardcoding the the string in the xml or the java file we simple give a reference to the name of the string in the string.xml file, and later on when we want to add an additional language we just need to add an appropriate string.xml file(in values-iw for example) and the system will choose the appropriate string automatically according to the device's locale

## Manifest

is the manifest of the app which defines all of the detail about the app (name of the app, apps icon, permissions, components, etc), when the user wants to download the app the play store shows the needed permissions from the manifest file. When the app is first downloaded all the classes registered in the manifest are loaded by the class loader, that is why we can listen to an

sms broadcast in the system even before we first run the app. We must register each component we add to the app(Activities,Services etc ) in the app Manifest.

## intent - filter

intent is a kind of way to send messages inside the phone. and the intent filters where the phone redirects all kinds of functions. each intent filter has an action string, when that action is broadcasted the system will automatically create an instance of the activity(or other component containing that action) and calls onCreate. For example the action MAIN which we can see in the MainActivity is used to respond to the app icon being pressed. The system detects that our app icon was pressed and searches out manifest for the proper activity to launch(later we will learn about creating custom intent-filters).

## Listener

All listeners are interfaces that responds to runtime events. For Example - Button click. When the system detects a button click for it automatically calls the onClick function in the View.OnClickListener(inner interface inside the View class). What we should do is: after placing the button in the xml file, we retrieve its reference with the **findViewById** function of the activity (the view that will automatically be searched for that id is the one set by the **setContentView** function mentioned above) and after getting that reference we can use the button **setOnClickListener** function (inherited from View class), pass a class instance that implements the **View.OnClickListener** interface and override the **onClick** function. When the button will be pressed the system will automatically call our implementation of the onClick function.

there are few common ways to implements the listeners:

1. The activity is implementing the Listener and override the onClick function. this allows a single implementation of the function per activity - same function for all the buttons and we must separate each button's action with the view's id for example. Not commonly used for buttons.
2. Create a special Inner class that implements the interface and pass an instance of it. This allows treating a few buttons in the same way by passing the same class.
3. Define an anonymous inner class the implements the interface. This is the most common way. in that way we provide a single implementation per button and cannot use it again but it simplifies the code and the need of extra classes.