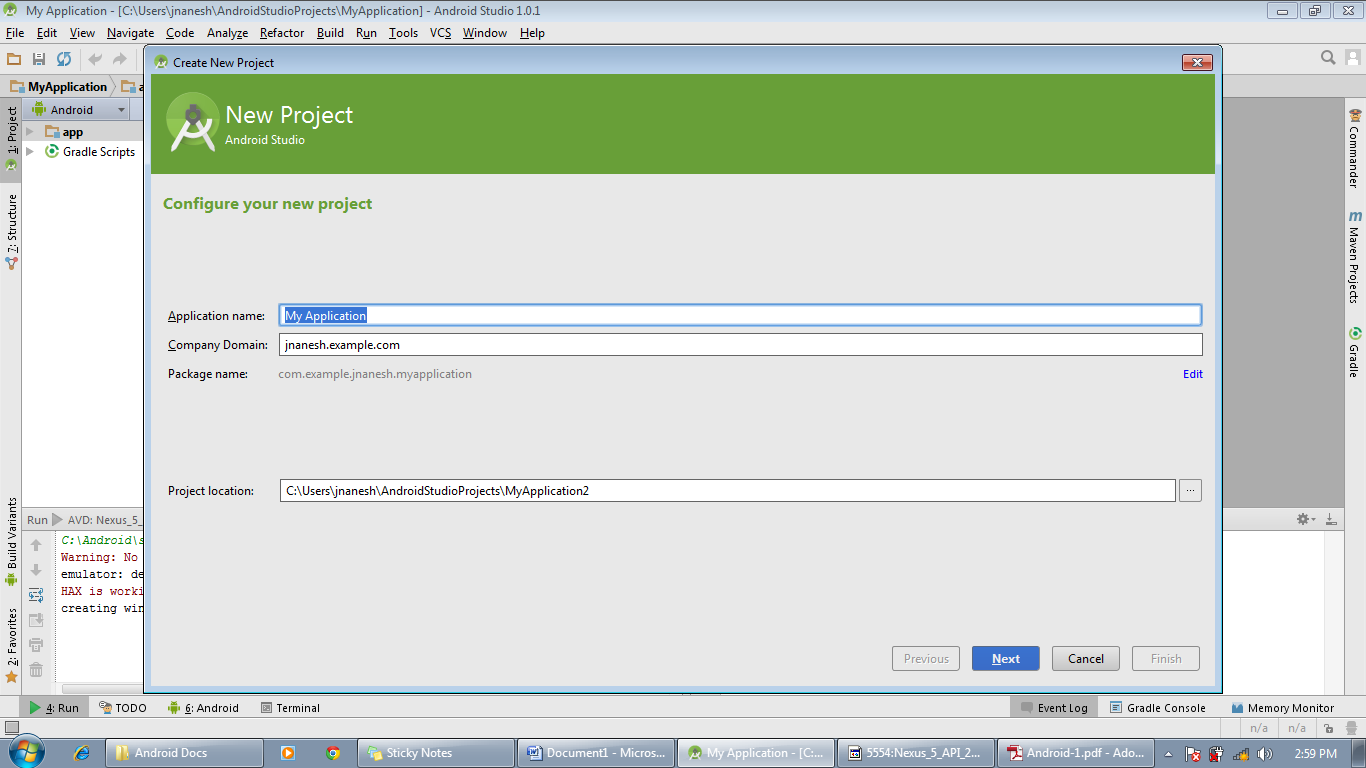
Android Exercise

Day 1

Assignment 1: Create a new project called: Multiply

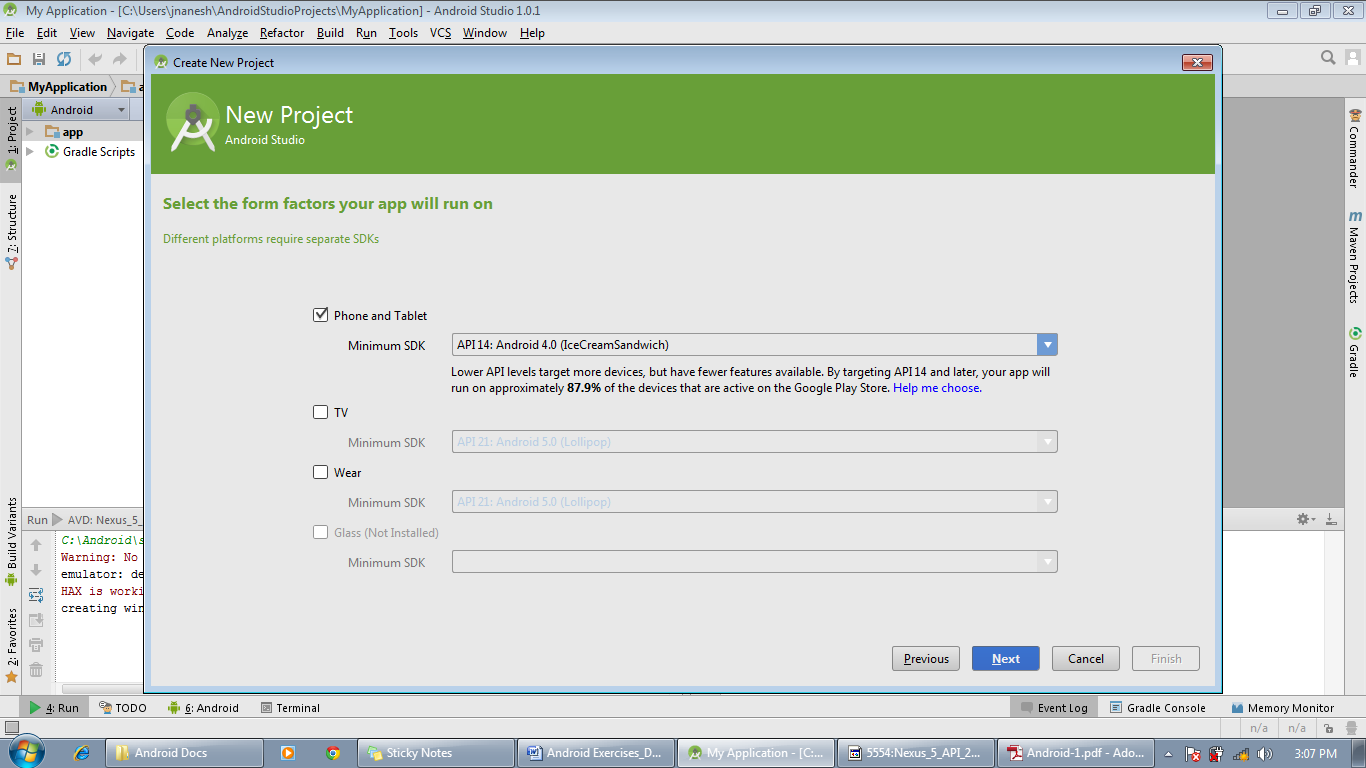
Solution 1: File -> New Project



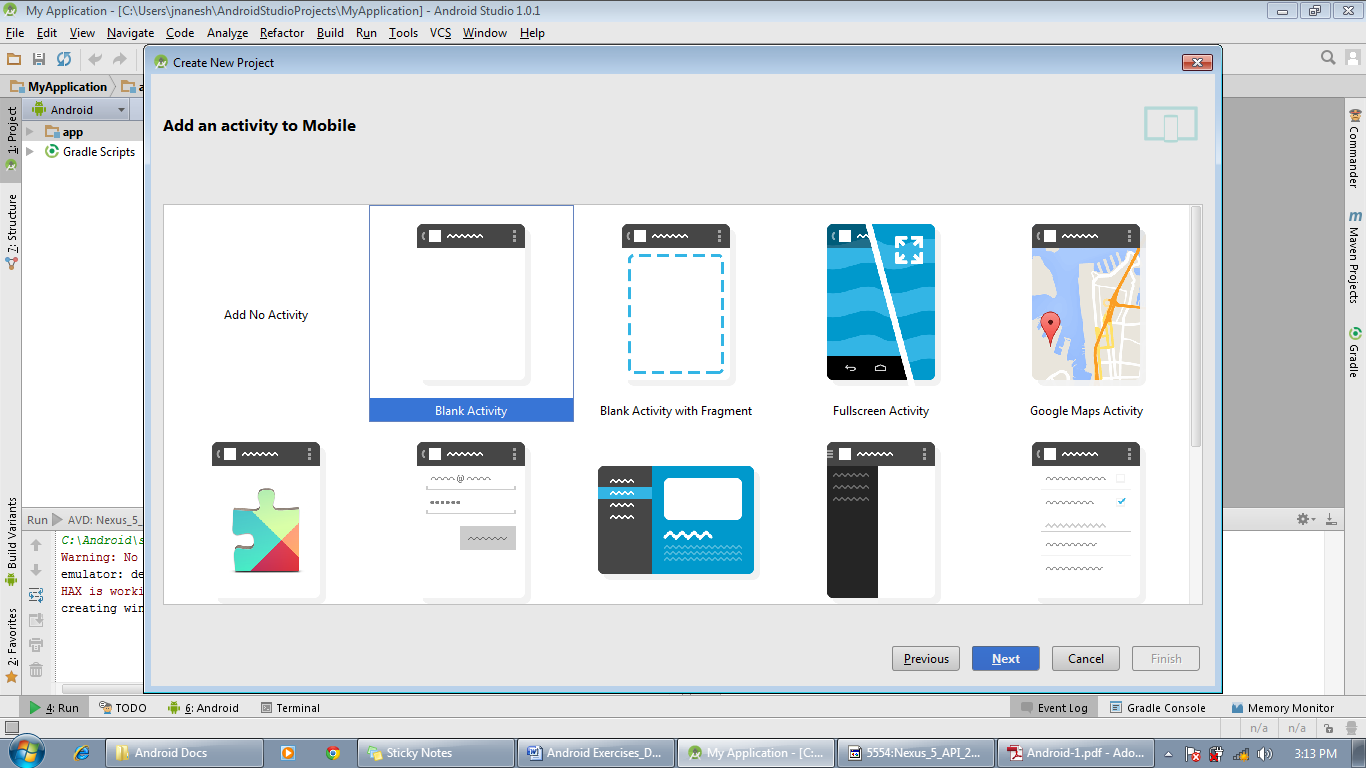
* Applicatation name: Multiply *(Name should starts with Capital letter)*
* Company Domain: techmindcraft.multiply.com

*Format for company domain (company\_domain\_name.application\_name.com)*

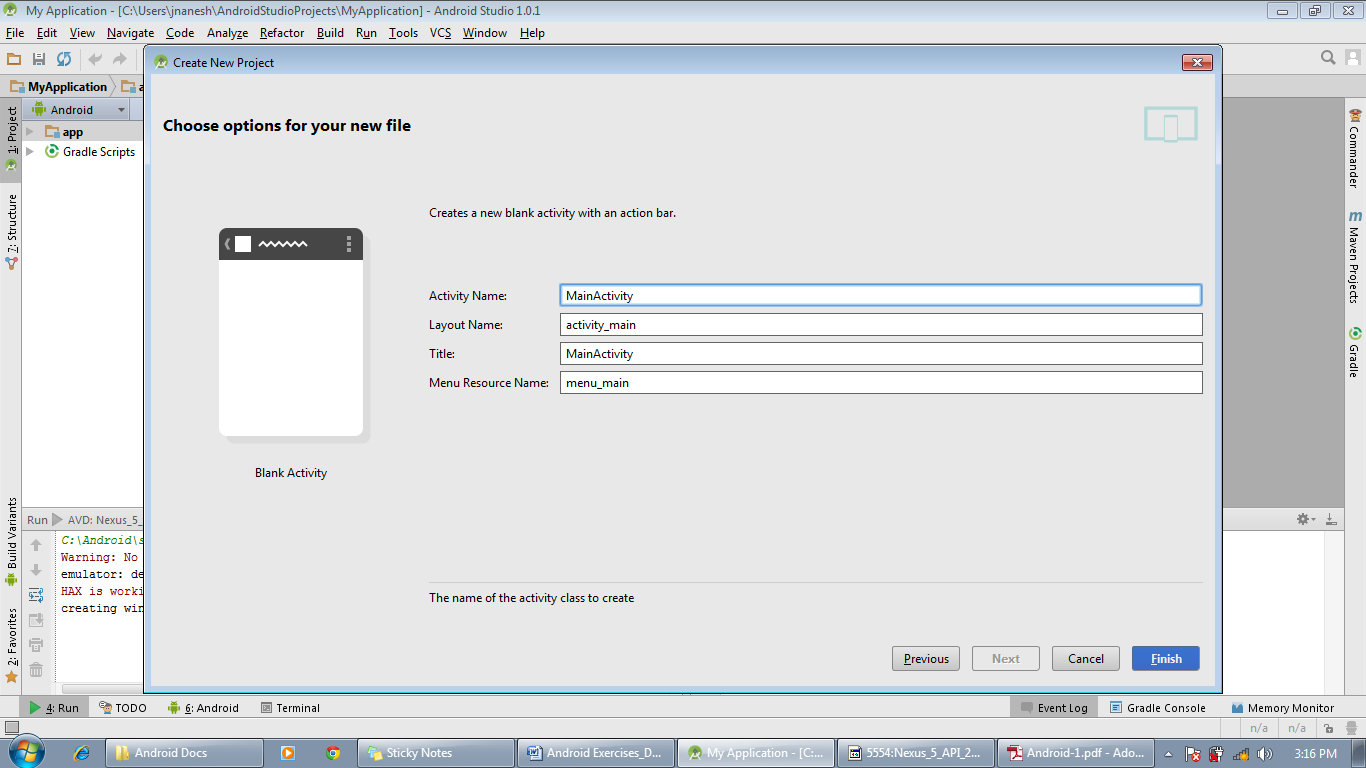
* Click on Next



* Select Phone and Tablet *(It will be selected by default)*
* Minimum SDK: Select as per your requirement, it is advisable to select the minimum API level to API14.
* Click on Next



* Select Blank Activity
* Click on Next



Activity Name: MainActivity *(Refers to name of the class, hence do not give any spaces in b/w)*

Layout Name: activity\_main *(Don’t alter the first component, second component will be same as first component of your activity name )*

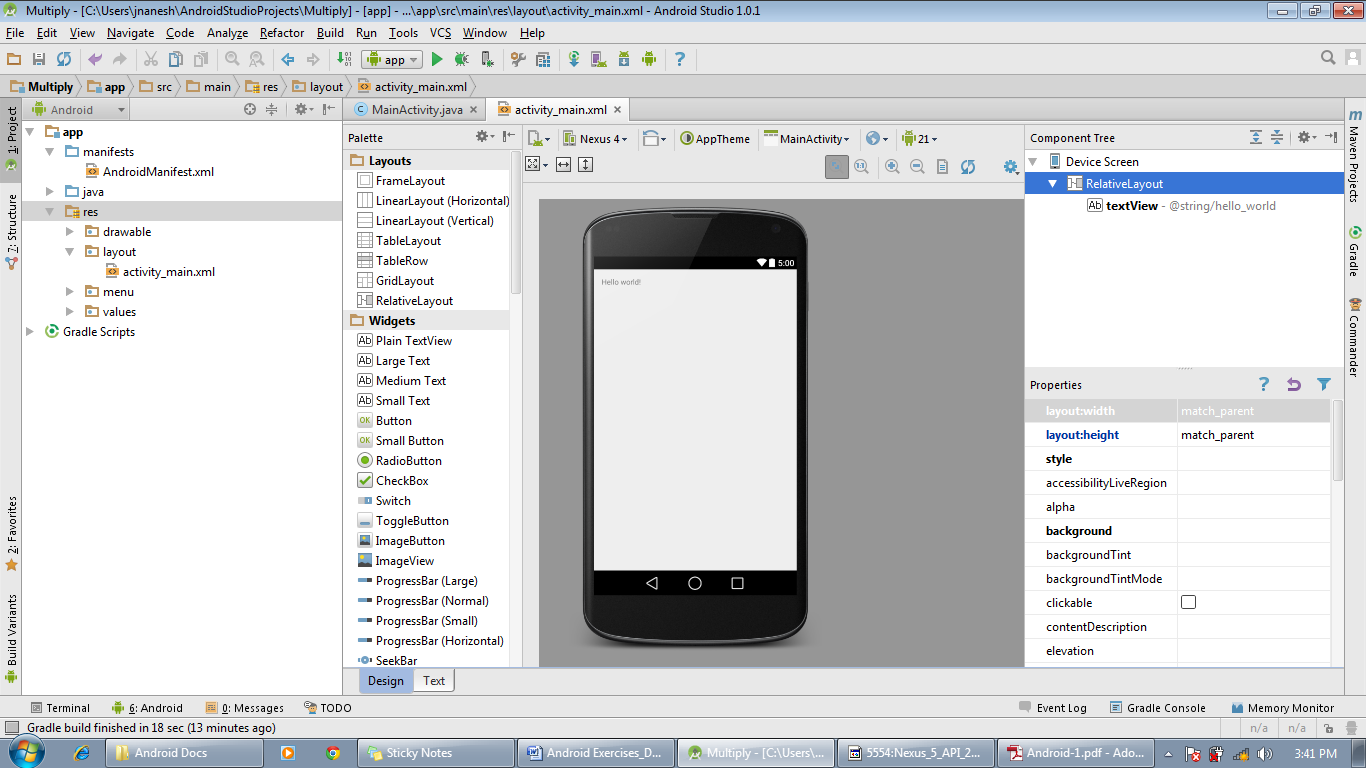
Title: Multiply Main Activity *(User defined – The application title)*

Menu Resource Name:menu\_main

* Click Finish

**Overview/Structure of an android project:**

The ***Multiply*** project created in the above mentioned steps will looks similar as below:



Now let’s take a brief look on structure of an android project.

An android project consists of 3 major components:

* manifests
* java
* res

**manifests:** The ***manifest*** file presents essential information about your app to the Android system, information the system must have before it can run any of the app's code.

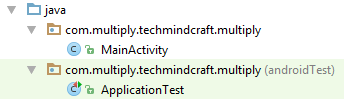
The ***manifest*** file organizes an Android application into a well-defined structure that is shared by all applications and enables the Android operating system to load and execute them in a managed environment.



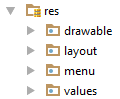
**AndroidManifest.xml:** The manifest file does the following things:-

* It names the Java package for the application. The package name serves as a unique identifier for the application.
* It describes the components of the application – the activities, services, broadcast receivers, and content providers that the application is composed of.
* It determines which processes will host application components.
* It declares which permissions the application must have in order to access protected parts of the API and interact with other applications.
* It also declares the permissions that others are required to have in order to interact with the application’s components.
* It declares the minimum level of the Android API that the application requires.
* It lists the libraries that the application must be linked against.

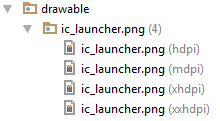
**java:** This folder contains all of our source java files and test files.



**res:** This folder contains all the resources file information of our project. There are 4 major components in resource folder which are as follows:



**drawable:**  All the bitmap files (PNG, JPEG, or GIF), 9-Patch image files, and XML files that describe Drawable shapes or Drawable objects are residing in this folder with different format of hdpi (high dots per inch), mdpi (medium dots per inch), xhdpi (extra high dots per inch)and xxhdpi.



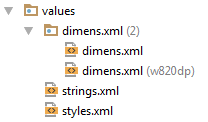
**layout:** This folder contains XML files that are compiled into screen layouts (or part of a screen).



**menu:** This folder contains XML files that define application menus



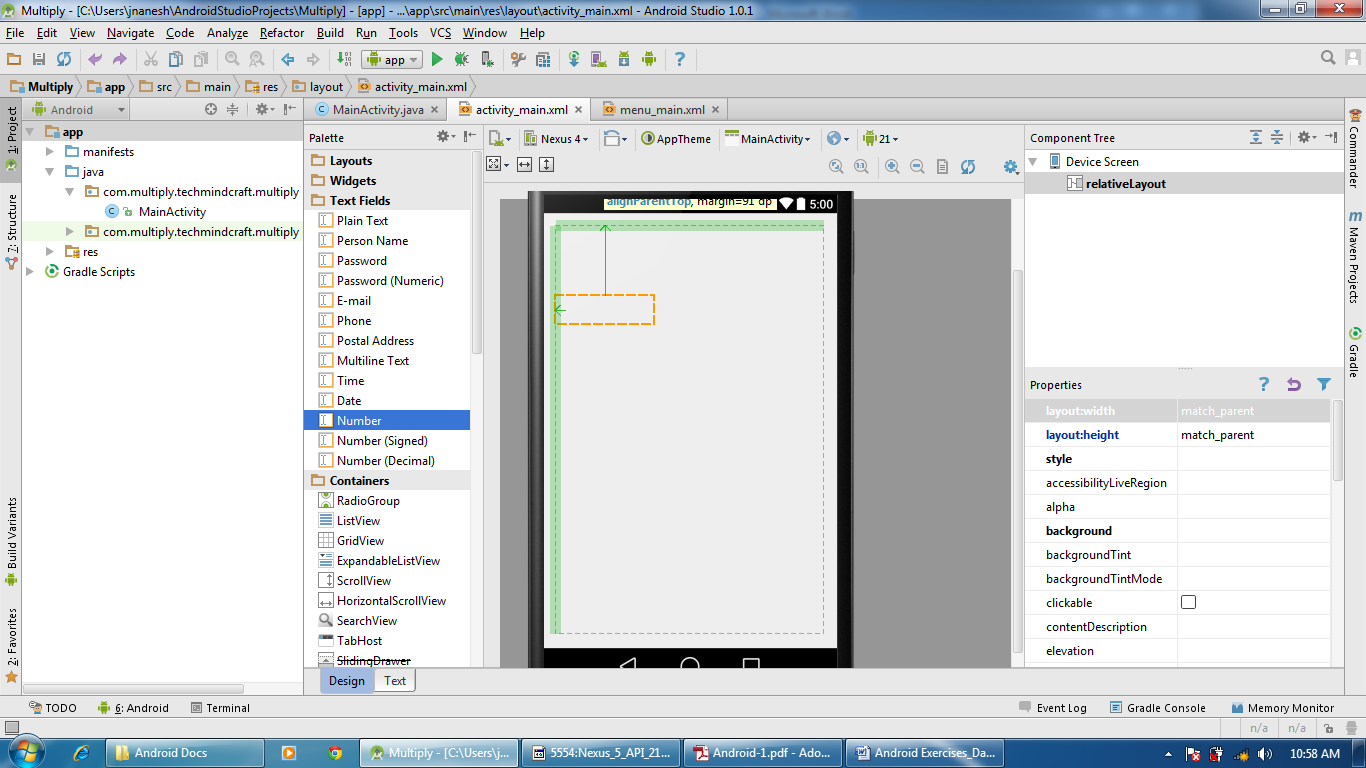
**values:** This folder contains XML files that define resources by XML element type. Unlike other resources in the res/ directory, resources written to XML files in this folder are not referenced by the file name. Instead, the XML element type controls how the resources defined within the XML files are placed into the R class.



Assignment 2: Add a TextField to your Activity’s xml.

Solution 2:

* + Select Text Fields from the Palette
  + Drag Number and drop on the mobile screen



Assignment 3: Create an object of the TextField in the Activity. Name it: numberTextField.

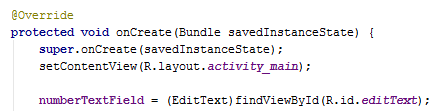
Solution 3:

* + Go to MainActivity.java
  + Declare the TextField as below:

public EditText numberTextField;



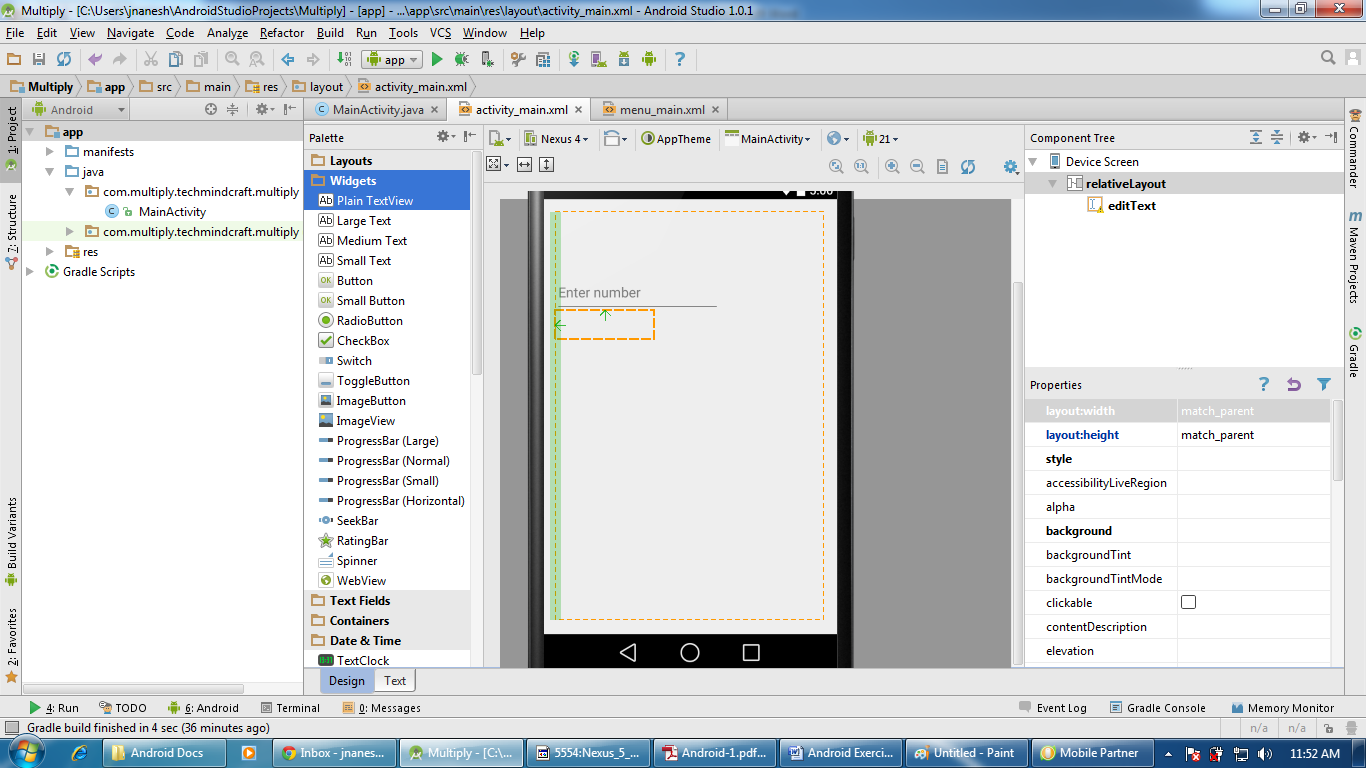
* Create an object of the numberTextField as below:



Assignment 4: Add a Label to your Activity’s xml

Solution 4:

* + Select Widgets from the Palette
  + Drag Plain TextView and drop on the mobile screen.



Assignment 5: Create an object of the label in the Activity. Name it: multiplierLabel.

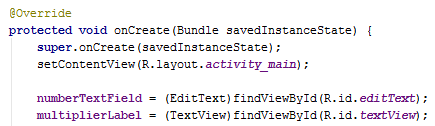
Solution 5:

* + Go to MainActivity.java
  + Type the below line after the class begins:

public TextView multiplierLabel;



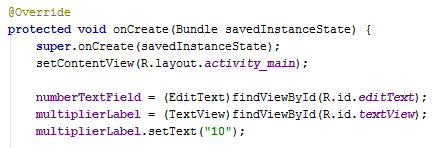
* Create an object of the multiplierLabel as below:



Assignment 6: Set the text of the label to 10 via its properties

Solution 6:

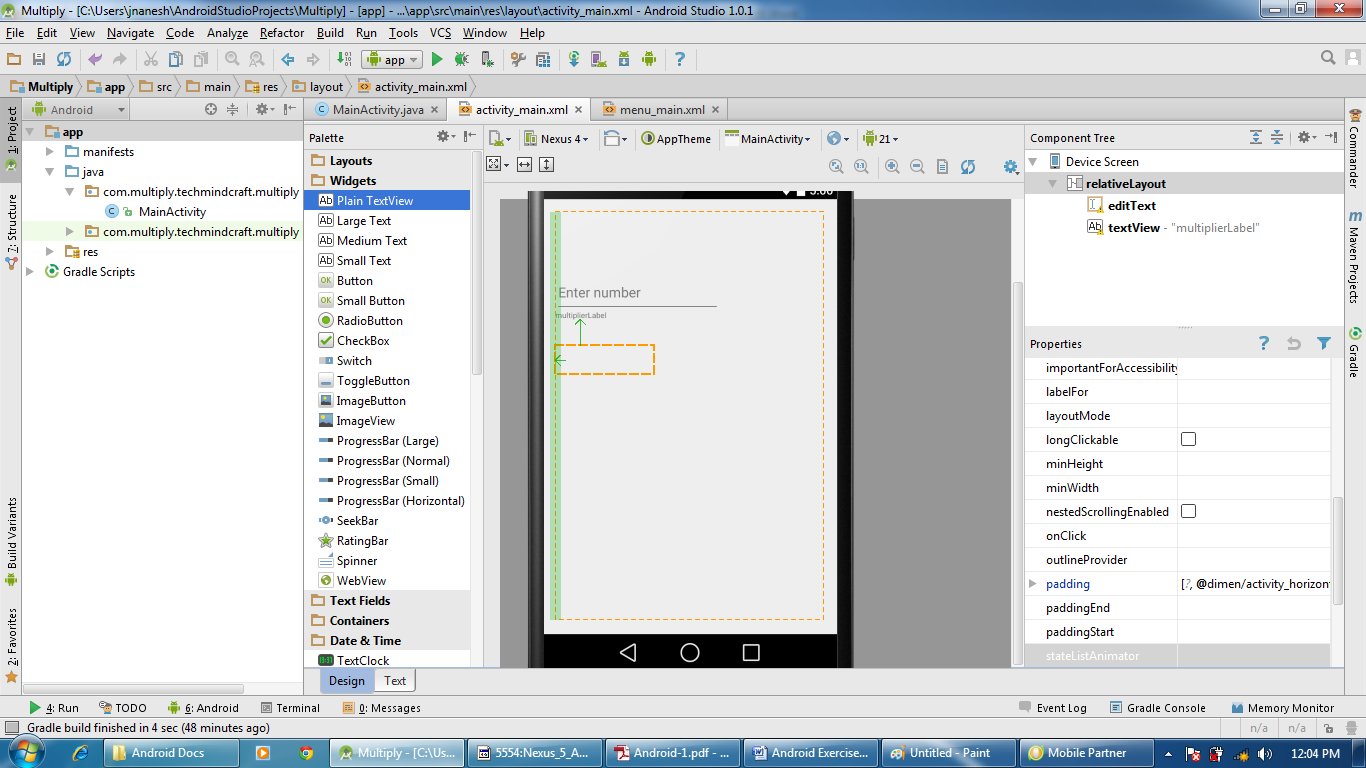
* + Go to MainActivity.java
  + Type as below in onCreate method :



Assignment 7: Add another Label to your Activity’s xml

Solution 7:

* + Select Widgets from the Palette
  + Drag Plain TextView and drop on the mobile screen

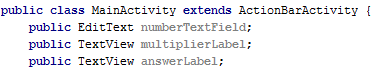


Assignment 8: Create the object for second label and name it: answerLabel

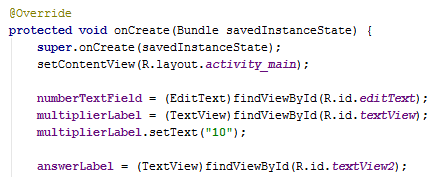
Solution 8:

* + Go to MainActivity.java
  + Type the below line after the class begins:

public TextView answerLabel;



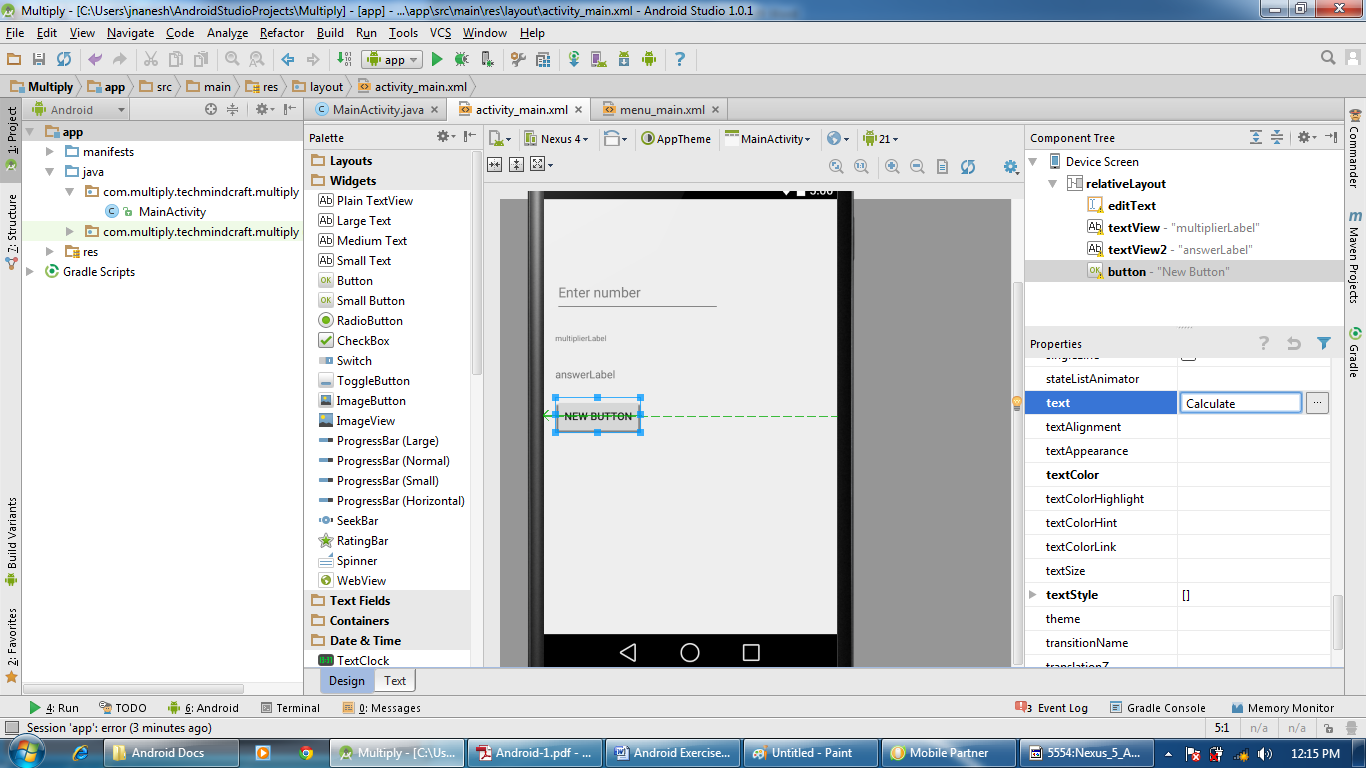
* Create an object of the answerLabel as below:



Assignment 9: Add a Button to your Activity’s xml scene in Storyboard and set its title text: Calculate

Solution 9:

* + Select Widgets from the Palette
  + Drag Button and drop on the mobile screen
  + Go to Button properties and change the text to Calculate

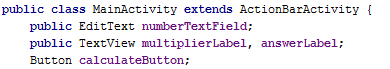


Assignment 10: Create an object for the button in your Activity. Add a method for its click event and name the method: onCalculateButtonPressed

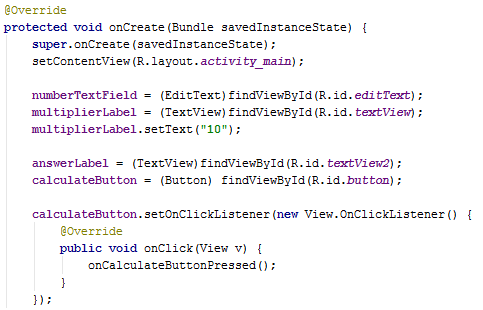
Solution 10:

* + Go to MainActivity.java
  + Type the below line after the class begins:

Button calculateButton;



* Create an object of the calculateButton and add a method for its click event and name the method: onCalculateButtonPressed



Assignment 11: In your onCalculateButtonPressed: method body in the Activity file, add code that retrieves the text from numberTextField, convert it to an int, and then assign it to a local variable.

Solution 11:

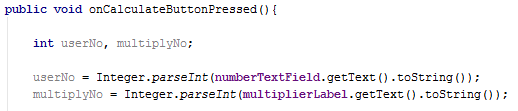
* + Go to MainActivity.java
  + Create a method onCalculateButtonPressed



Assignment 12: In your onCalculateButtonPressed: method body in the Activity file, add code to retrieve the text from your multiplierLabel, convert it to an int, and then assign it to a local variable.

Solution 12:

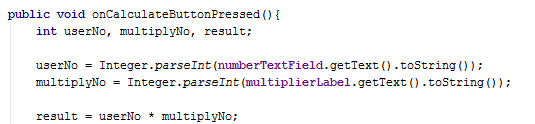
* + Go to MainActivity.java
  + Modify the method onCalculateButtonPressed as below :



Assignment 13: In your onCalculateButtonPressed: method body in the ViewController.m file, add code to multiply your two local variables and assign the result into a final local variable..

Solution 13:

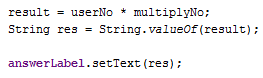
* + Go to MainActivity.java
  + Modify the method onCalculateButtonPressed as below



Assignment 14: In your onCalculateButtonPressed: method body in the ViewController.m file, add code to assign the value of the final variable containing the product of your multiplication to answerLabel.

Solution 14:

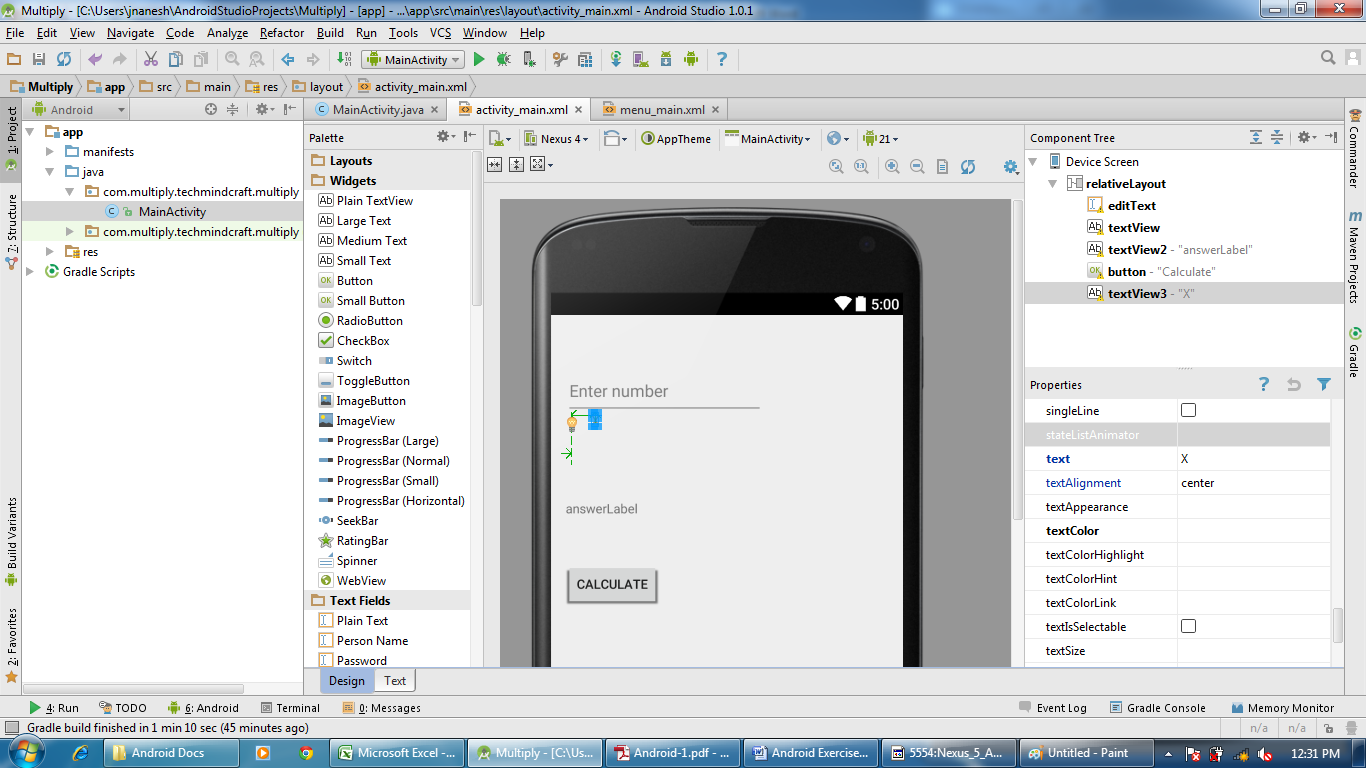
* + Go to MainActivity.java
  + Modify the method onCalculateButtonPressed as below



Assignment 15: Add a Label to your Activity’s xml, assign the value of the label to a multiplication sign: \*.

Solution 15:

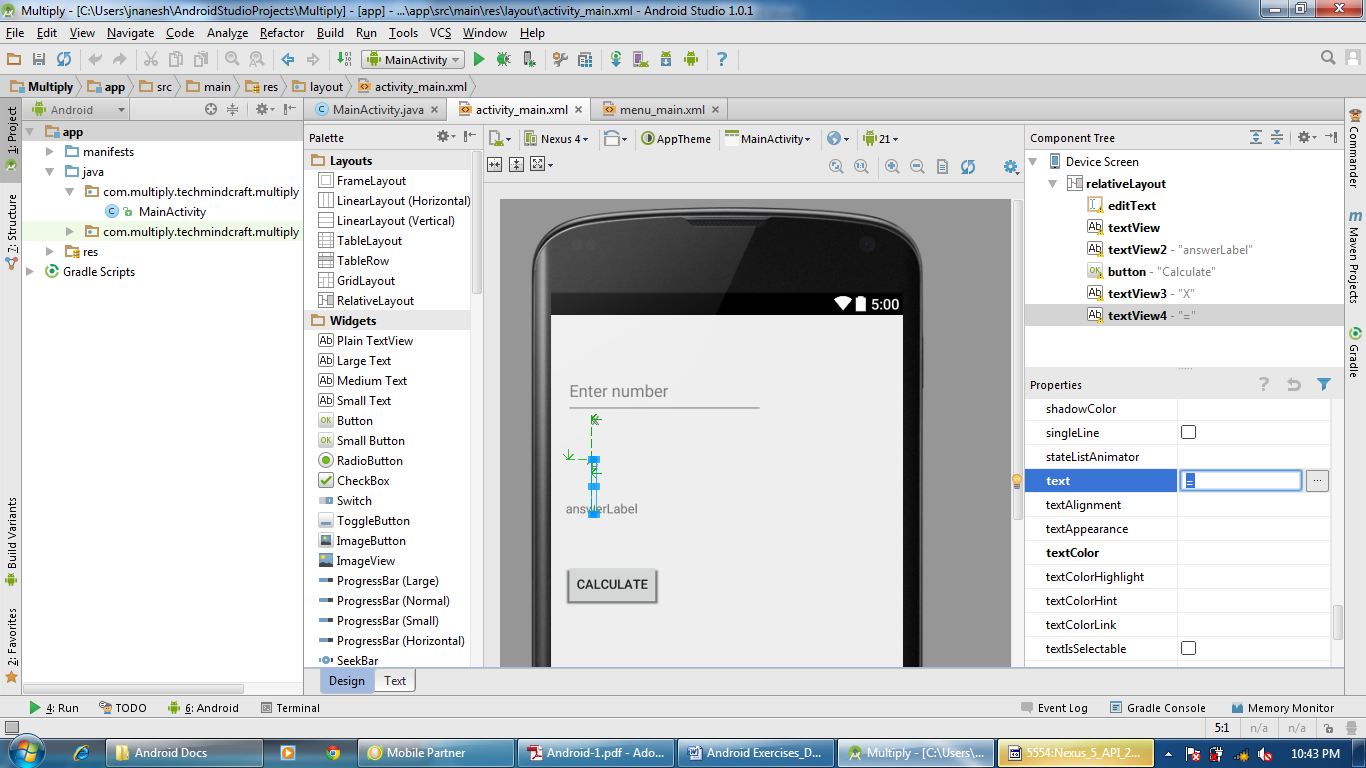
* + Select Widgets from the Palette
  + Drag Plain TextView and drop on the mobile screen
  + Go to Properties and set the text as “X”



Assignment 16: Add a Label to the Activity’s xml, assign the value of the label to an equal sign: =

Solution 16:

* + Select Widgets from the Palette
  + Drag Plain TextView and drop on the mobile screen
  + Go to Properties and set the text as “=”



Assignment 17: If the result is equal to or greater than 20, change the background color of the whole Activity to green.

If the result is less than 20, the background color should be white.

Solution 17:

* + Go to MainActivity.java
  + Modify the method onCalculateButtonPressed as below: