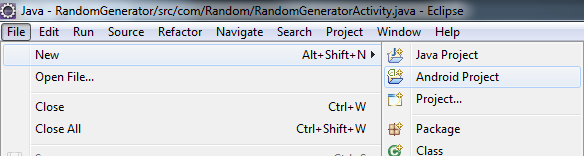
**Random Number Generator: Android Edition**

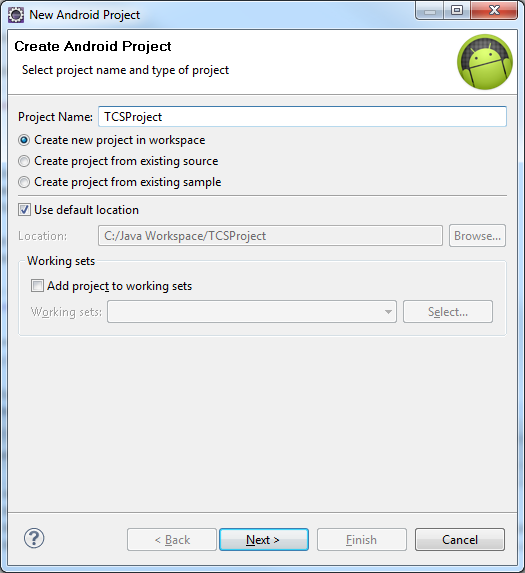
NOTE: These steps will only work if you have installed the Android SDK, Eclipse, and Android Development Tools. If you have not done so, please complete the “Getting Started” guide before proceeding!

This guide will take you through the creation of a simple Android project. You will create text objects, a button, and program it to generate a random number with each click.

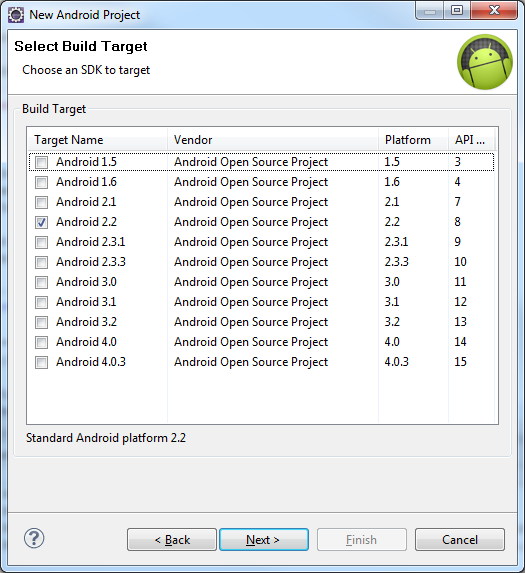
1) Start Eclipse and begin a new Android Project.



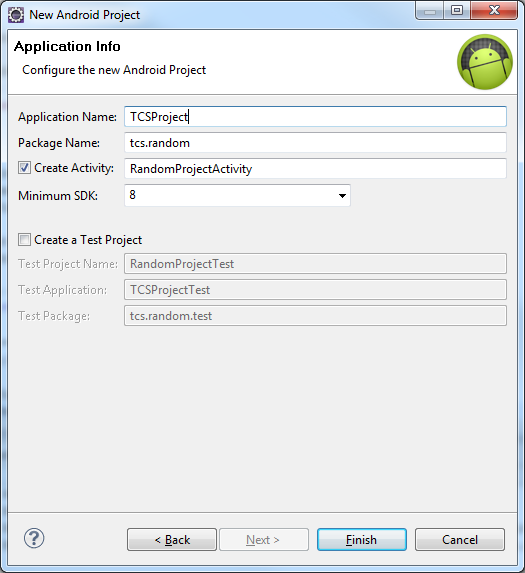
2) Enter a Project Name. For this example, we are using “TCSProject” as the name. Do not include any spaces or special characters except \_ (underscores) in the project name. Then press the “Next” button.



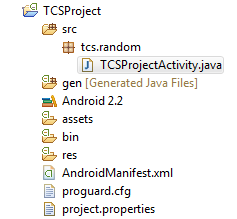
3) Select an android version. You must determine whether you want to focus on compatibility (using an older version) or functionality (using a newer version) for your project. For this example, we will use “Android 2.2” for compatibility purposes since we are only using basic Android elements. Press the “Next” button to continue.



4) Enter a package name. For this example we will use “tcs.random” though you can create your own package name. It should be all lowercase characters with each section separated by periods. It must not begin or end with a period. Numbers are also allowed. Then press the “Finish” button to create the project.



5) On the left of your Eclipse window you will see the Package Explorer. Expand your project folder, the “src” folder, and your package folder. Then double click on the Java file within the package folder. There are a number of other files in the package, but no others need to be edited for this project. You can learn more about the default generated files in the Android documents online.



6) Inspect the Java file. Your code should resemble the following:

**package** tcs.random;

**import** android.app.Activity;

**import** android.os.Bundle;

**public** **class** TCSProjectActivity **extends** Activity {

/\*\* Called when the activity is first created. \*/

@Override

**public** **void** onCreate(Bundle savedInstanceState) {

**super**.onCreate(savedInstanceState);

setContentView(R.layout.*main*);

}

}

This code constitutes the basic structure necessary for an Android application. The package and class name reflect what you entered during the project setup to distinguish your app from others on the android device. The import statements allow access to the core Android functionality to run an app. The “onCreate” method is similar to the “main” method in standard Java programs and will contain the basic runtime instructions for your app.

7) Add the following import statements to the top of the file where you see the “import android.app.Activity” and “import android.os.Bundle” lines. The order does not matter.

**import** android.widget.LinearLayout;

**import** android.widget.TextView;

**import** android.widget.Button;

**import** android.view.View;

**import** android.view.View.OnClickListener;

**import** java.util.Random;

The first three lines, the imports for LinearLayout, TextView and Button, will allow us to include the basic visual elements for our app. The next two, View and OnClickListener, will allow us to add click actions to the Button. The last, Random, will enable us to create the actual random number generator for use in the app.

8) Now we will begin creating the elements for our main screen. The first is the layout, which will contain all the other elements. All the elements will go in the onCreate method, following the lines that are already there.

**public** **void** onCreate(Bundle savedInstanceState) {

**super**.onCreate(savedInstanceState);

setContentView(R.layout.*main*);

LinearLayout layout = **new** LinearLayout(**this**);

layout.setOrientation(LinearLayout.*VERTICAL*);

}

The “this” keyword refers to the current activity, which is where the entirety of this program will run. The “LinearLayout.VERTICAL” property indicates that all the elements will be placed on screen in order, vertically, starting with the top and working down the screen.

9) The following lines will all go in the onCreate method under the LinearLayout you just created. We will first create a TextView with the instruction text.

TextView textView = **new** TextView(**this**);

textView.setText("Click the button to generate a random number!");

10) Next, we will create the button that we will press to generate the number.

Button button = **new** Button(**this**);

button.setText("Click me!");

11) Now, we will create a TextView to contain the result generated by the button press. Set some default text so you can see it on screen before any number has been generated.

**final** TextView result = **new** TextView(**this**);

result.setText("There is no result yet!");

Note the “final” keyword for this TextView. This is needed so the object can be used with the Button later on, rather than accessing the instance in this variable it will be passed a copy, then the copy will be used to update the original instance with the modified information. This will allow us to refresh it on each button click.

12) Now that all the elements have been created, we will add them to the screen.

layout.addView(textView);

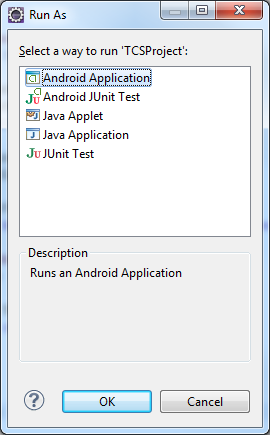
layout.addView(button);

layout.addView(result);

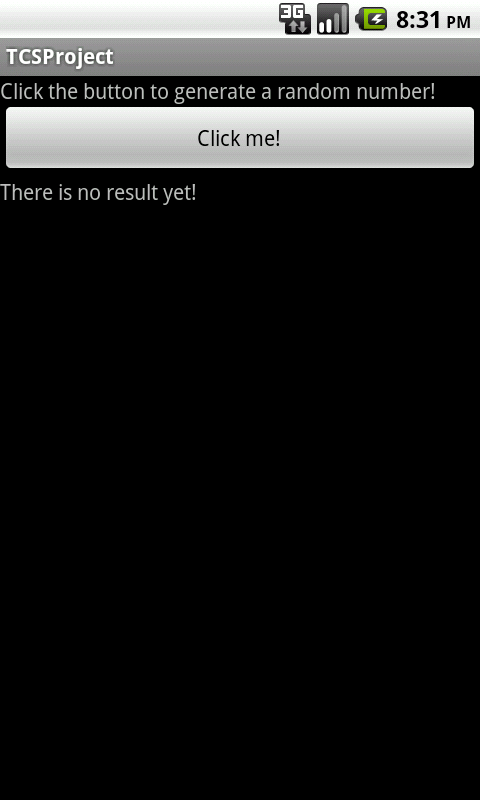
**this**.setContentView(layout);

The first three lines add the TextView and Button objects to the layout we created earlier, and the last places the layout with all the objects on the screen and makes them visible.

13) Run the project by pressing “Ctrl+F11” on the keyboard, or pressing the green circle button with the white arrow inside it. On the dialog that comes up, select “Android Application” and press the “OK” button. If Eclipse asks if you want to save, click the “Yes” button.



Here is an example of what the app should look like at this point:



14) At this point, all the elements display on screen but the button has no code to tell it what to do. We will now insert that code. Place the following code between where you created the “result” TextView lines and where you added the elements to the layout.

button.setOnClickListener(**new** OnClickListener()

{

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

{

}

});

The “OnClickListener” class provides a way for the button press to interact with the rest of the app. The “View v” in the “onClick” method returns the current View, which in this case will be the screen in which the button is contained. Anything the button does will affect that View.

15) Now, we need to create the random number generator. To do this, we must create a global Random object that can be used throughout the app. Place it in the top of your class, above the “onCreate” method:

**public** **class** TCSProjectActivity **extends** Activity {

Random random = **new** Random();

/\*\* Called when the activity is first created. \*/

@Override

**public** **void** onCreate(Bundle savedInstanceState) {

...

16) Finally, we can write the code to update the “result” TextView when we click the button. Insert the following code in the onClick method to generate the number and change the text:

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

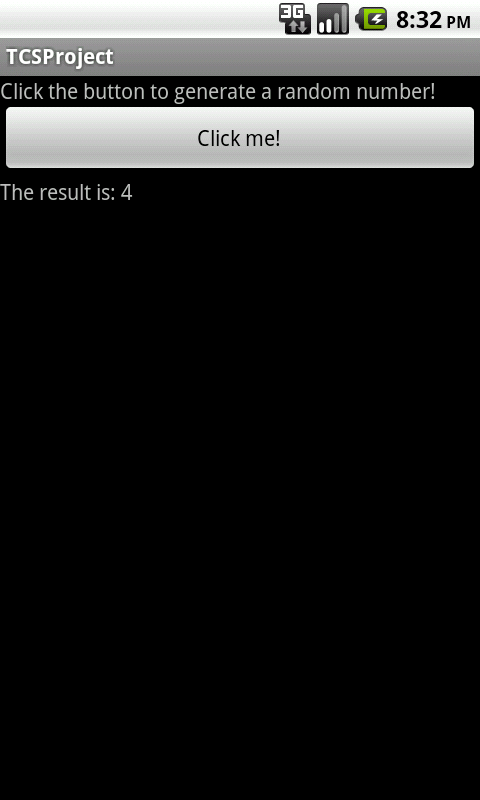
{

result.setText("The result is: " + random.nextInt(10));

}

In this example, we are using the “nextInt(int)” method to limit the available options for the random number generation. The “10” indicates the upper limit, exclusive; so for this example, all the integers selected will be from 0 to 9.

17) Run the app again, and this time the button should replace the result text with a random number.



If the app is working correctly, the first time you press the button it should update similar to the screen above, and each time after it should update the result line with a new random integer.

**Additional Steps (time permitting)**

18) Because of the nature of the Random integer generation, there may be instances when it returns the same integer multiple times as a result. Therefore, we can add a counter to keep track of how many times the button has been pressed to make sure everything is working correctly. Add a global integer variable to the class:

**public** **class** TCSProjectActivity **extends** Activity {

Random random = **new** Random();

**int** count = 1;

19) Now, we must modify the result text in the button’s “onClick” method to include the count:

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

{

result.setText("The result is: " + random.nextInt(10) + " (Count: " + count + ")");

count++;

}

Using “count++” will cause the count to be increased by 1, giving a different incremental value on each click. Run the program now to show the count.

20) We can also extend the result text to make a log. To do this, we will use a series of String variables to make the code more readable by splitting it into multiple functions. Then we will concatenate the Strings in the result text setting method. This will replace the current code in the “onClick” method.

String result1 = "The result is: " + random.nextInt(10) + " (Count: " + count + ")";

String result2 = "\n" + result.getText();

result.setText(result1 + result2);

count++;

The “\n” adds a newline to the String which is visible in the TextView. Run the program now to show the log.

21) Notice that the initial line of having no result is still visible. We can add simple logic to replace the text on the first click rather than concatenate it. This is also in the “onClick” method.

String result1 = "The result is: " + random.nextInt(10) + " (Count: " + count + ")";

String result2 = "\n" + result.getText();

**if**(count == 1)

result.setText(result1);

**else**

result.setText(result1 + result2);

count++;

Run the program now to show the log with the initial text in the “result” TextView being replaced. Generate enough numbers to lead into the following steps, showing why we need the ScrollView.

22) If you generate too many numbers, they will begin to flow off the screen. In order to keep them available for viewing, we can create a scrollable field. First, we must import the appropriate class:

**import** android.widget.ScrollView;

23) Now we must create a variable within the activity for a ScrollView:

**super**.onCreate(savedInstanceState);

setContentView(R.layout.*main*);

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

LinearLayout layout = **new** LinearLayout(**this**);

layout.setOrientation(LinearLayout.*VERTICAL*);

24) Finally, we must change the last few lines in the activity to add the LinearLayout to the ScrollView “scroller” instance, then set the view for the app to “scroller” instead of “layout” to accommodate the new arrangement.

layout.addView(textView);

layout.addView(button);

layout.addView(result);

scroller.addView(layout);

**this**.setContentView(scroller);

The ScrollView will automatically adjust its size to fit the expanding TextView, and you can now view the entire history of all the generated numbers, even after it grows beyond the size of the screen. This should be the final run of the program, to show that the log flows off the screen but you can scroll to see the other entries.

Here are example screenshots of the four states: single line with count; multiple lines; multiple lines without initial text; scrollable window with overflowing text.

