Android Bluetooth Control LED.

Deyson Rodrigues

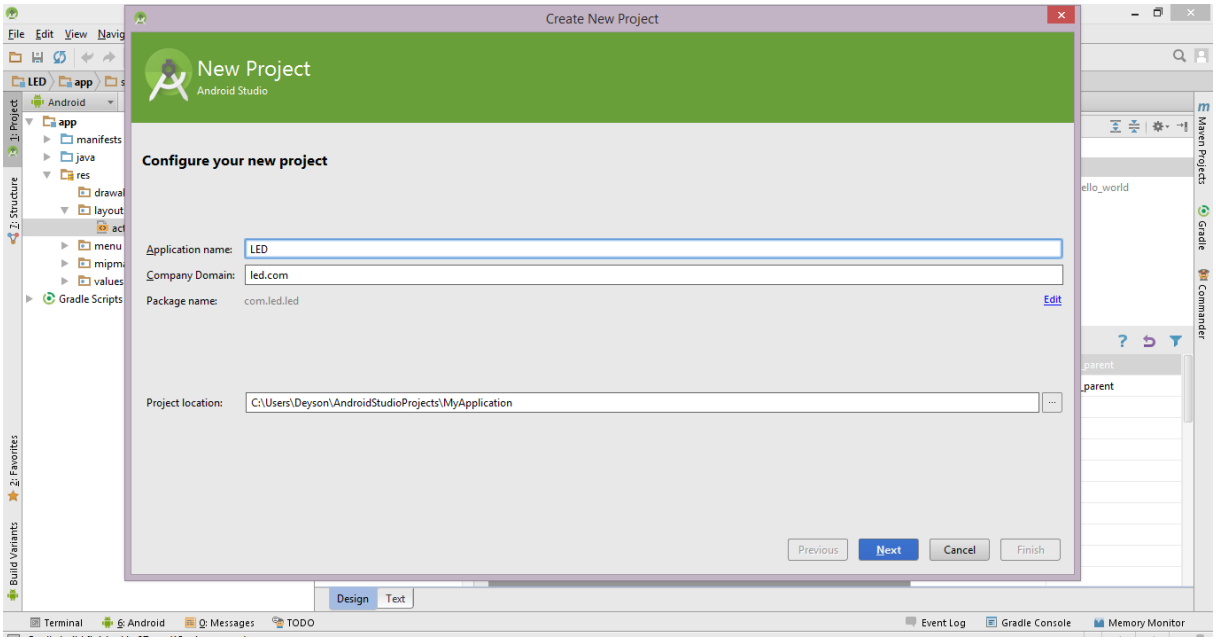
This is a step-by-step tutorial for making an android apk using bluetooth.

Before start coding,

- Download Android Studio IDE and update Java.
- Java and C programming skills will help.
- This tutorial will not explain Java Programming.
- *If you want to code using Eclipse IDE, it is almost the same.*
- The apk will send commands to turn on/turn off a LED and controls the brightness.

Android: New Project

- Open Android Studio and create a new Project: ***File > New Project.***
- A pop up Windows will appear. Change the Application Name and Company Name:

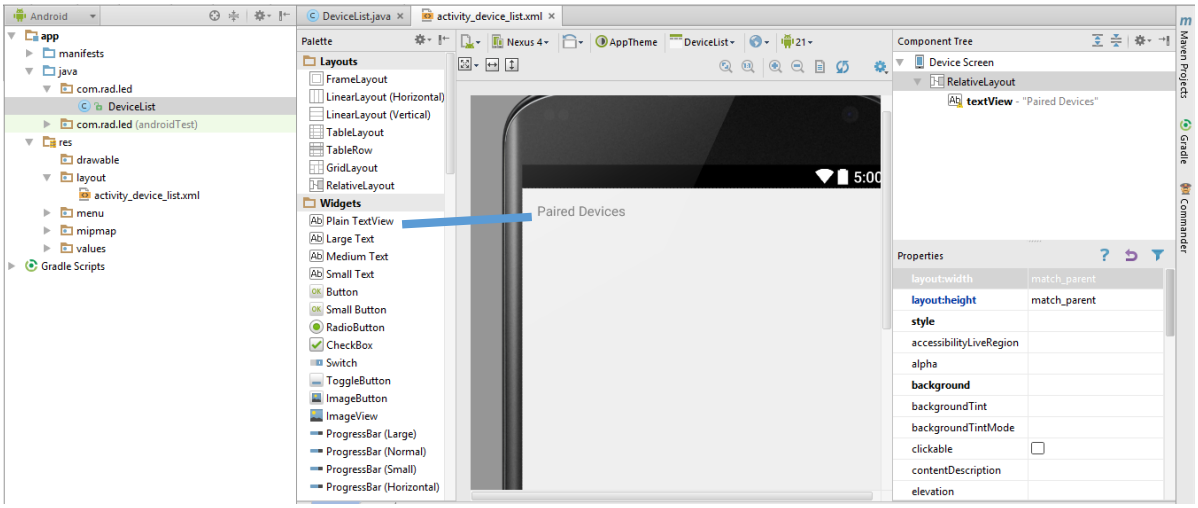


- Click next to choose the target of the application. The default is Android 4.0 (IceCream Sandwich)
- Click next and choose a **Blank Activity**.
- Click next and rename the **Activity Name** to “DeviceList”.
- Now click “finish” and the Project will be create.

Android: Layout Part 1.

When the build is finished, a “Hello world!” screen will be open. To create the layout of the apk, we need to add:

- ***TextView*** to display some hint to the user;

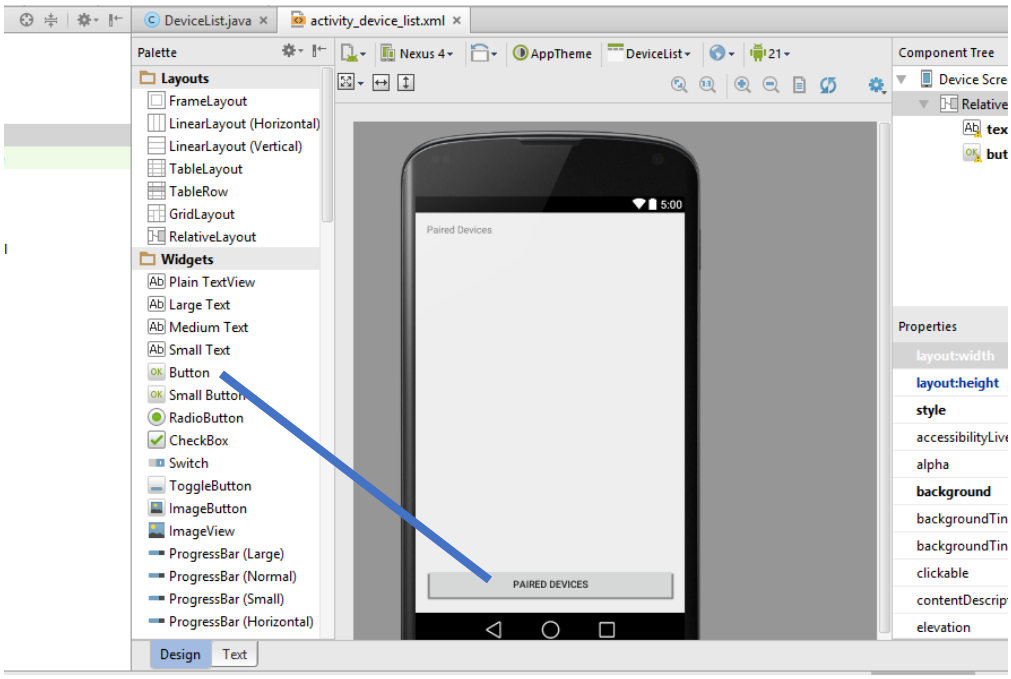


Click twice the TextView to change the text. A box will appear:

Text = The text to be displayed.

Id = the id of this widget.

- ***Button*** to show the paired devices.

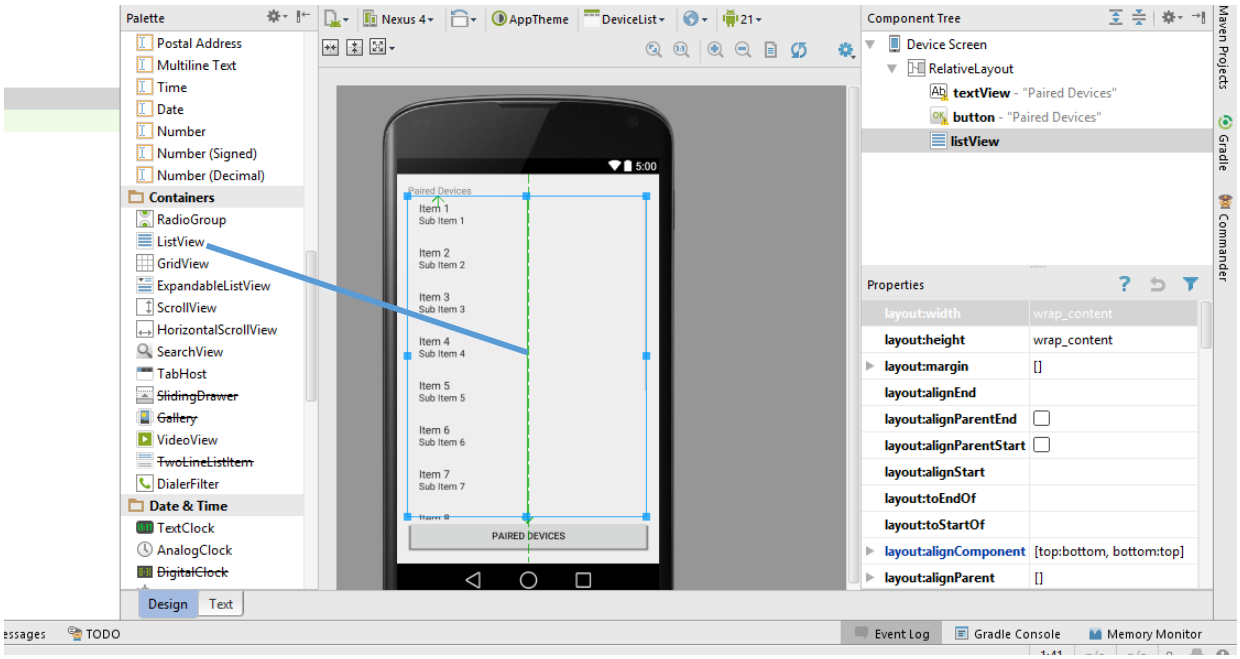


Click twice the Button to change the text. A box will appear:

Text = The text to be displayed.

Id = the id of this widget.

- ***ListView*** to show the paired devices;

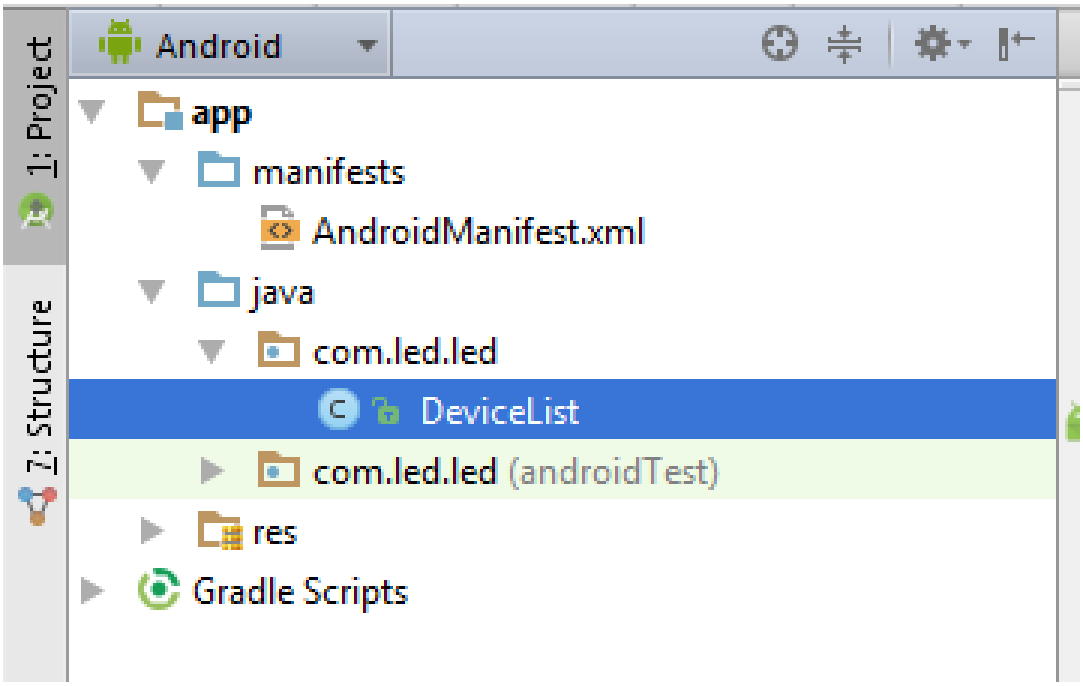


Now the main activity is finished, you can see that all widget used are shown on ***Components Tree.***

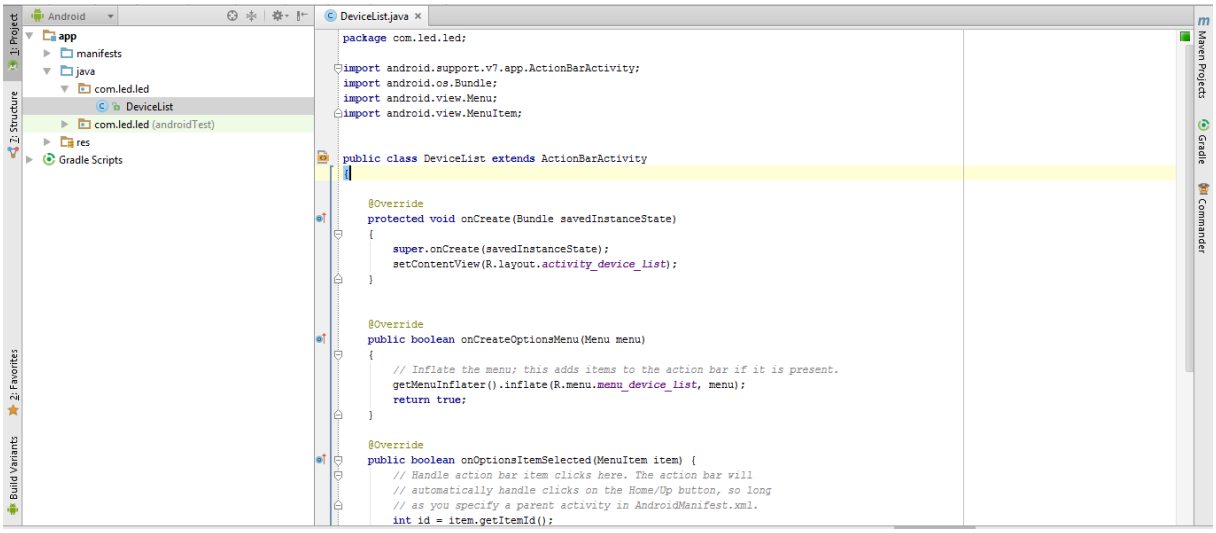
Android: Class Code Part 1.

On the left side there's a folder called "app ", open it and you'll see other folder called "java".

Java folder contains the package of the apk (com.led.led), and all the source code.



- Open DeviceList class;

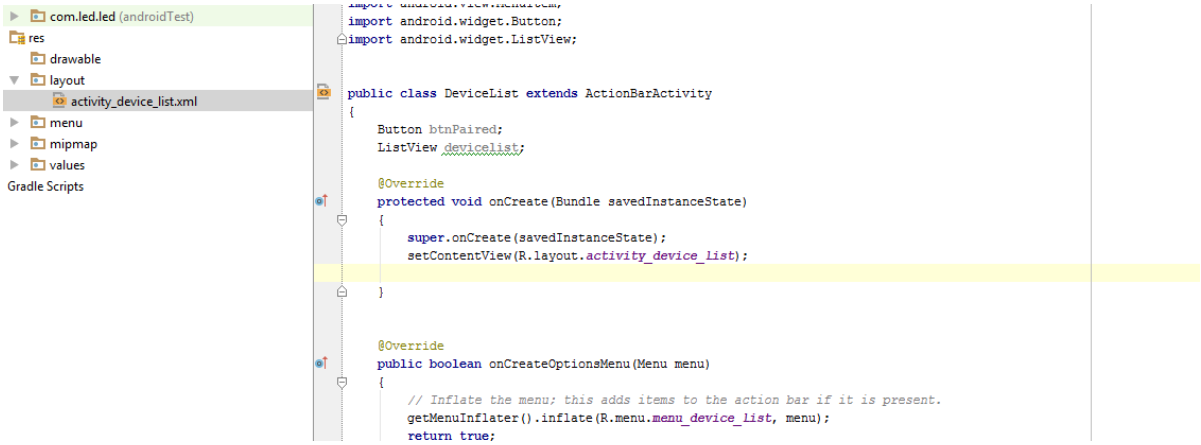


Import the followings packages:

import android.widget.Button;
import android.widget.ListView;

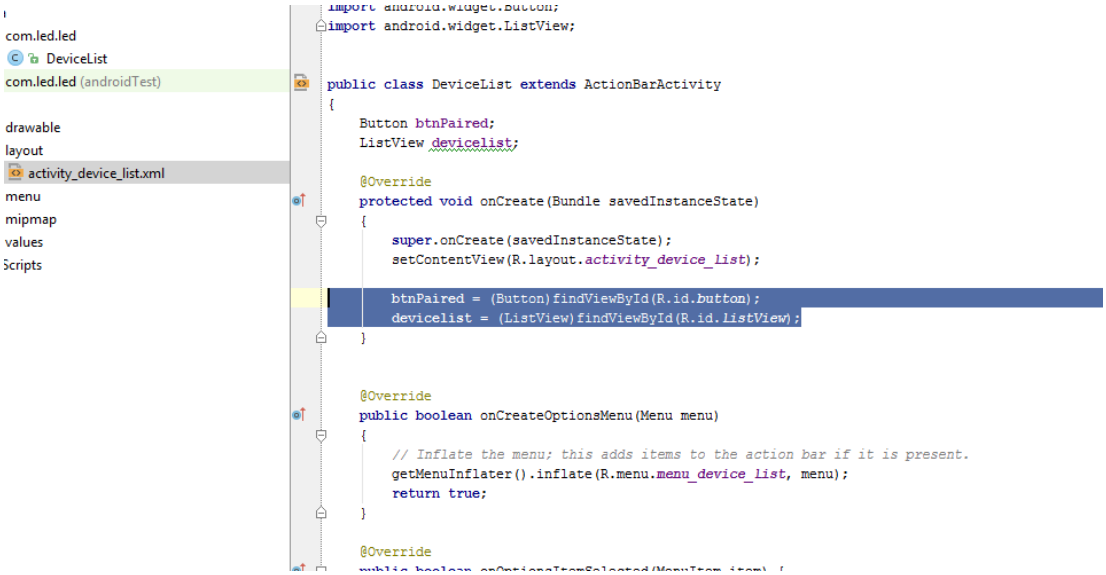
Create widgets variables to "call" the widgets used to create the layout:

Button btnPaired;
ListView devicelist;



Initialize the variables.

btnPaired = (Button)findViewById(R.id.button);
devicelist = (ListView)findViewById(R.id.listView);

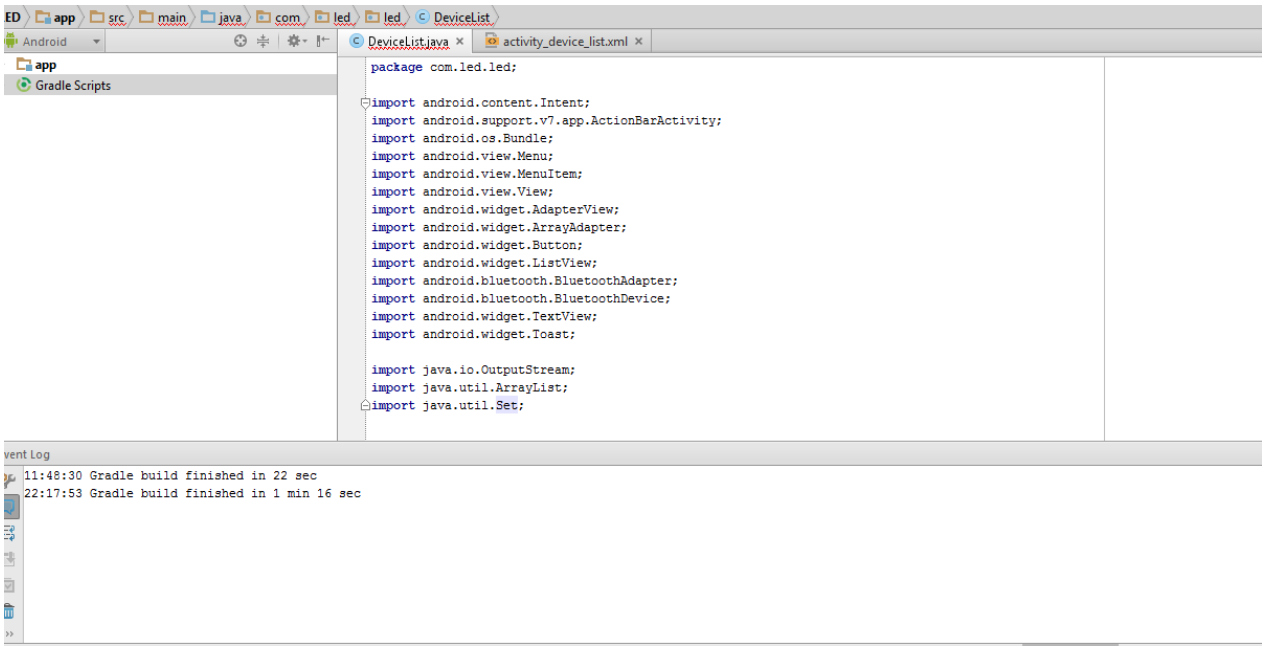


Import the following packages:

import java.util.Set;
import java.util.ArrayList;

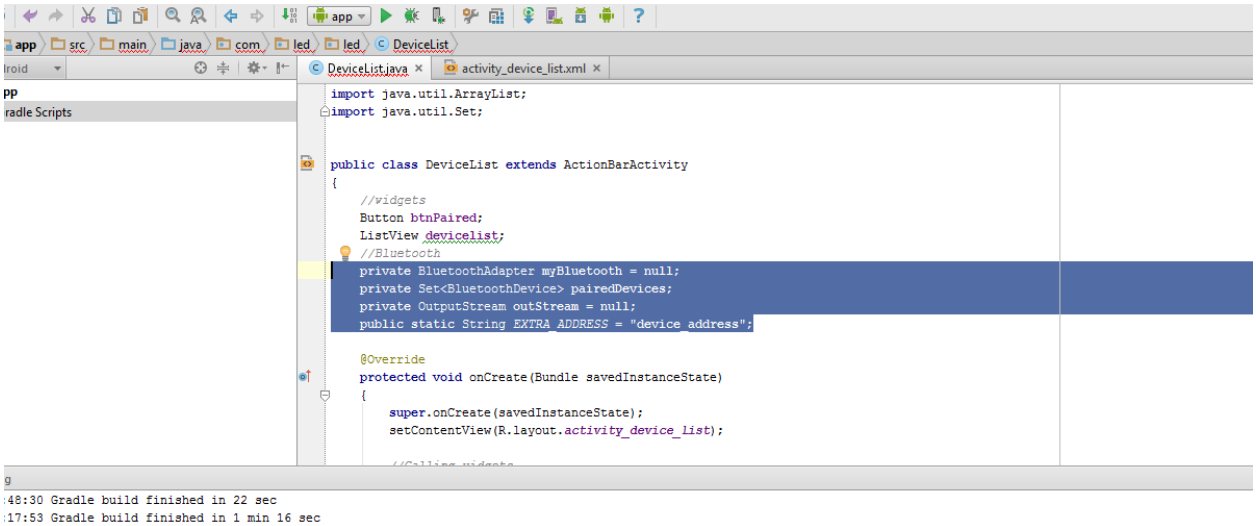
import android.widget.Toast;
import android.widget.ArrayAdapter;
import android.widget.AdapterView
import android.widget.AdapterView.OnClickListener

import android.widget.TextView;
import android.content.Intent;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;



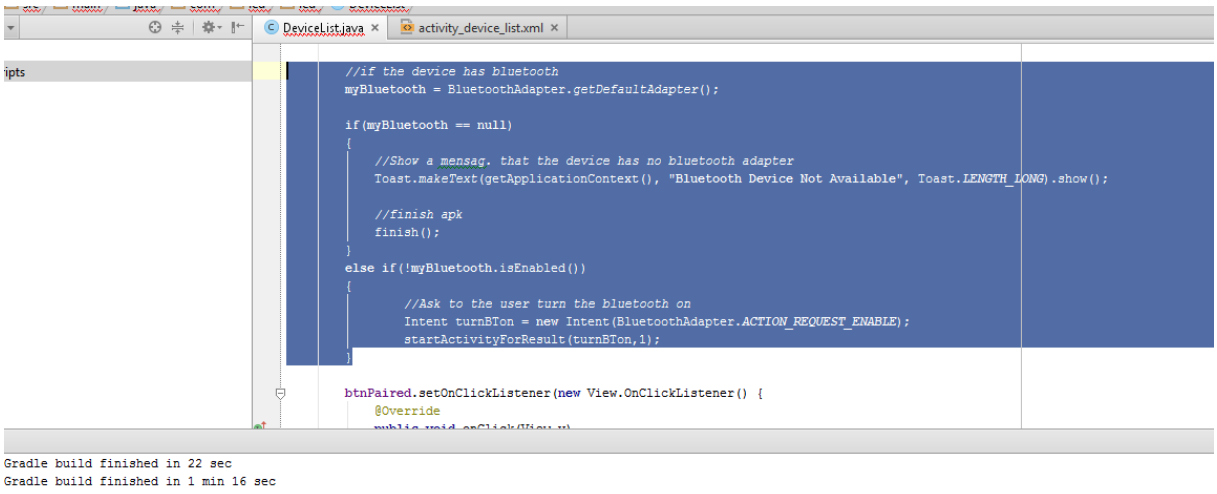
Create variables to control bluetooth:

private BluetoothAdapter myBluetooth = null;
private Set<BluetoothDevice> pairedDevices;



Writing a stable code avoids weird erros, so it's good to check if the device has bluetooth adapter and whether it's activated.

```
myBluetooth = BluetoothAdapter.getDefaultAdapter();

      if(myBluetooth == null)

      {
         //Show a mensag. that the device has no bluetooth adapter
         Toast.makeText(getApplicationContext(), "Bluetooth Device Not Available",
Toast.LENGTH_LONG).show();

         //finish apk
         finish();

      }
  else
      {
         if (myBluetooth.isEnabled())
         { }
         else
         {
            //Ask to the user turn the bluetooth on
            Intent turnBTon = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
            startActivityForResult(turnBTon,1);
         }
```



According to Android documents, an Intent is a messaging object you can use to request an action from another app component. Although intents facilitate communication between components in several ways, there are three fundamental use-cases:

- **To start an activity:**

  An `Activity` represents a single screen in an app. You can start a new instance of an `Activity` by passing an `Intent` to `startActivity()`. The `Intent` describes the activity to start and carries any necessary data.

- **To start a service:**

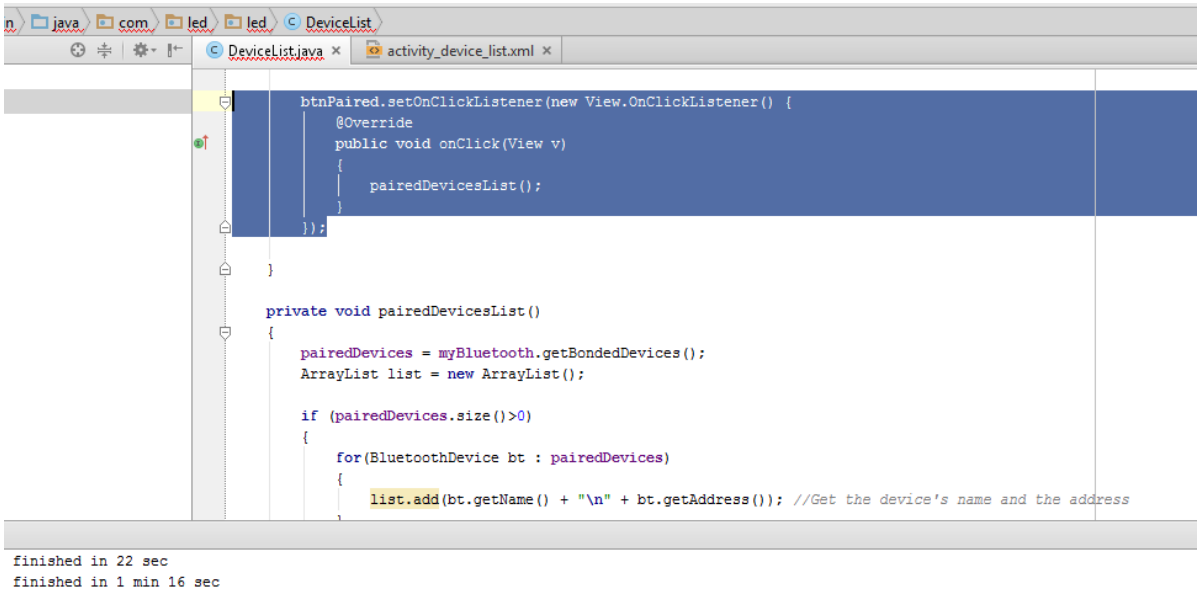
  A `Service` is a component that performs operations in the background without a user interface. You can start a service to perform a one-time operation (such as download a file) by passing an `Intent` to `startService()`. The `Intent` describes the service to start and carries any necessary data.

- **To deliver a broadcast:**

  A broadcast is a message that any app can receive. The system delivers various broadcasts for system events, such as when the system boots up or the device starts charging. You can deliver a broadcast to other apps by passing an `Intent` to `sendBroadcast()`, `sendOrderedBroadcast()`, or `sendStickyBroadcast()`.

We need to "listen" when the button is clicked to show paired devices. So *OnClickListener* Api will handle it

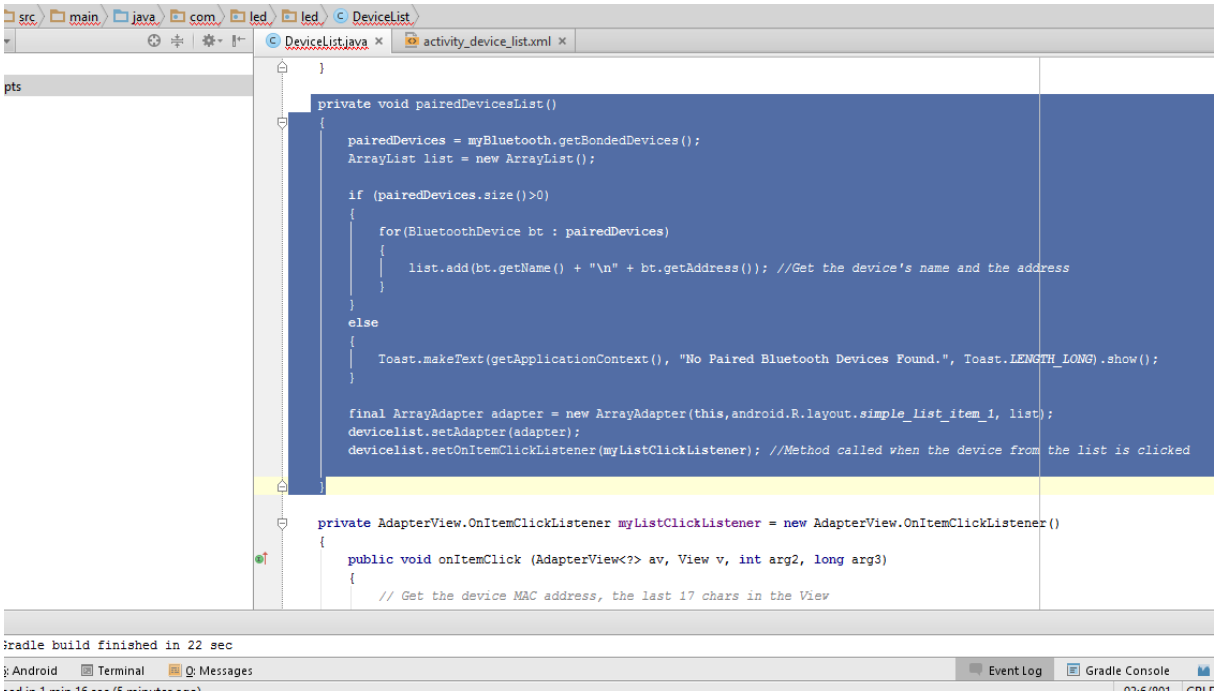
```
btnPaired.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v)
    {
        pairedDevicesList(); //method that will be called
    }
});
```



The *PairedDevicesList* method:

```
private void pairedDevicesList()
   {
      pairedDevices = myBluetooth.getBondedDevices();
      ArrayList list = new ArrayList();
```

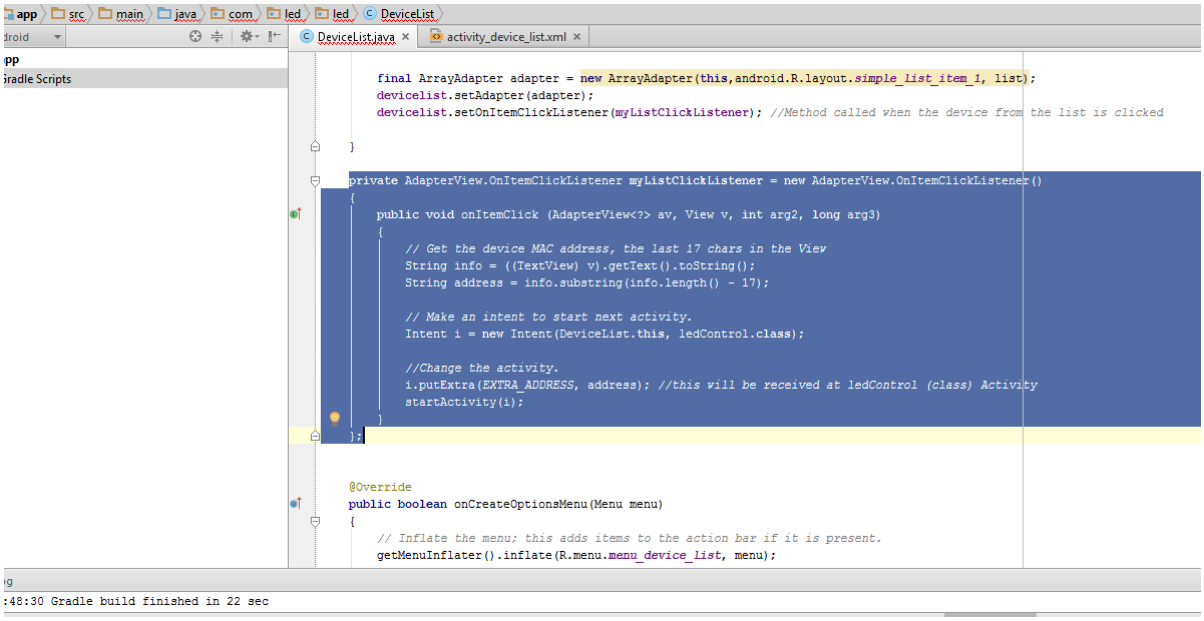
```
    if (pairedDevices.size()>0)
    {
       for(BluetoothDevice bt : pairedDevices)
       {
          list.add(bt.getName() + "\n" + bt.getAddress()); //Get the device's name and the
address
       }
    }
    else
    {
       Toast.makeText(getApplicationContext(), "No Paired Bluetooth Devices Found.",
Toast.LENGTH_LONG).show();
    }
    final ArrayAdapter adapter = new
ArrayAdapter(this,android.R.layout.simple_list_item_1, list);
    devicelist.setAdapter(adapter);
    devicelist.setOnItemClickListener(myListClickListener); //Method called when the device
from the list is clicked
  }
```



There is other method called ***myListClickListener***. It allow the ListView to be clicked.

```
  private AdapterView.OnItemClickListener myListClickListener = new
AdapterView.OnItemClickListener()
  {
    public void onItemClick (AdapterView<?> av, View v, int arg2, long arg3)
    {
      // Get the device MAC address, the last 17 chars in the View
```

```
        String info = ((TextView) v).getText().toString();
        String address = info.substring(info.length() - 17);

        // Make an intent to start next activity.
        Intent i = new Intent(DeviceList.this, ledControl.class);

        //Change the activity.
        i.putExtra(EXTRA_ADDRESS, address); //this will be received at ledControl (class) Activity
        startActivity(i);
    }
};
```



We need to create na new activity. If you're following this tutorial step-by-step, you're seeing an error at:

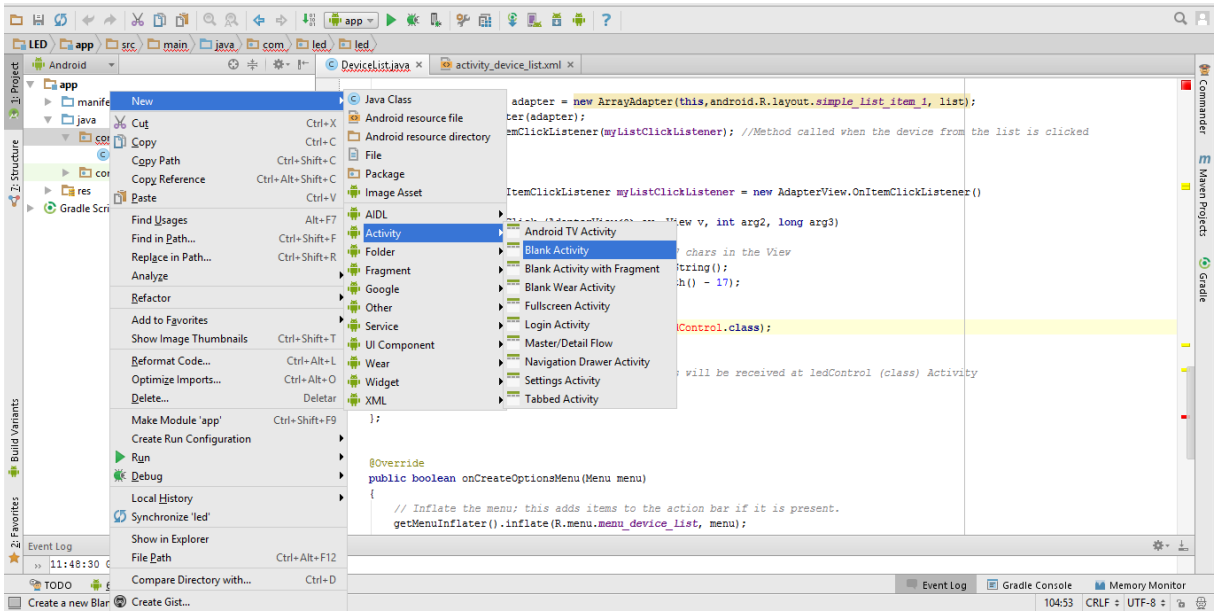
*Intent i = new Intent(DeviceList.this, **ledControl.class**);*

The error happens because there's **no ledControl class** in the package.

Let's create a new activity called **ledControl.**

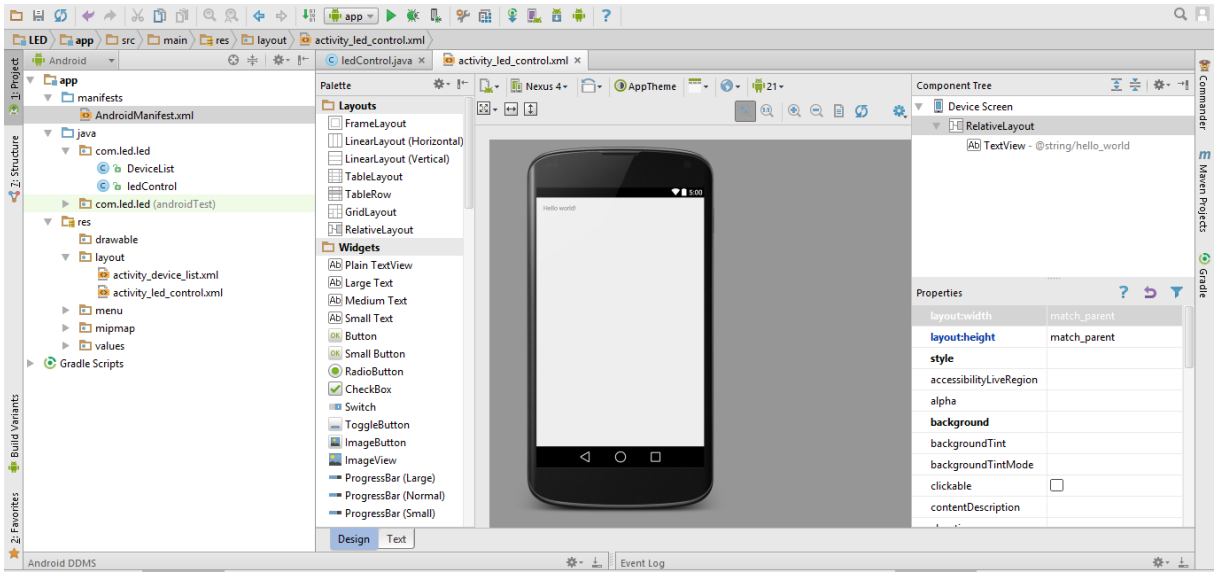
Android: Layout Part 2.

Go to app > java > com.led.led, Right click, **New Activity > Blank Activity**



Name it to **ledControl** and finish;

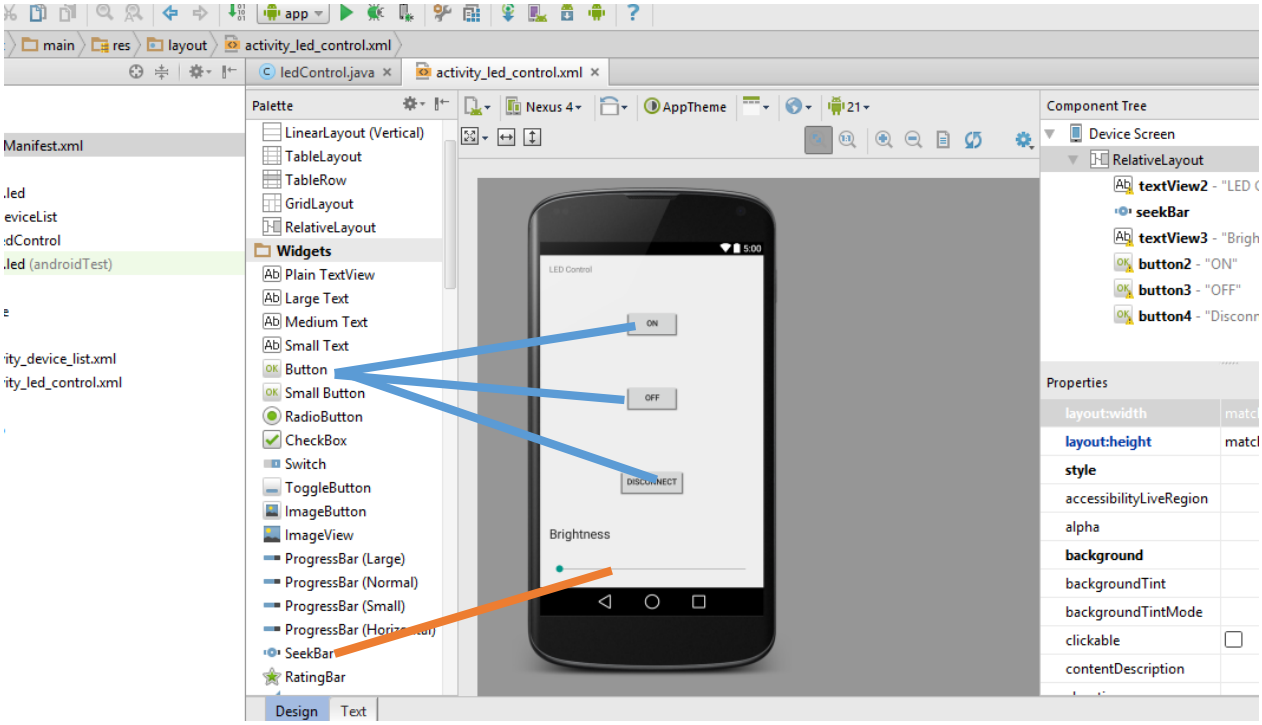
All over again, a “Hello world!” screen will be seen.



This second layout will have three buttons, one TextView and a seekbar:

- Turn On = Turns the LED On;
- Turn Off = Turns the LED Off;
- Disconnect = Closes Bluetooth Connection;
- Indicator;
- Brightness = control the brightness.

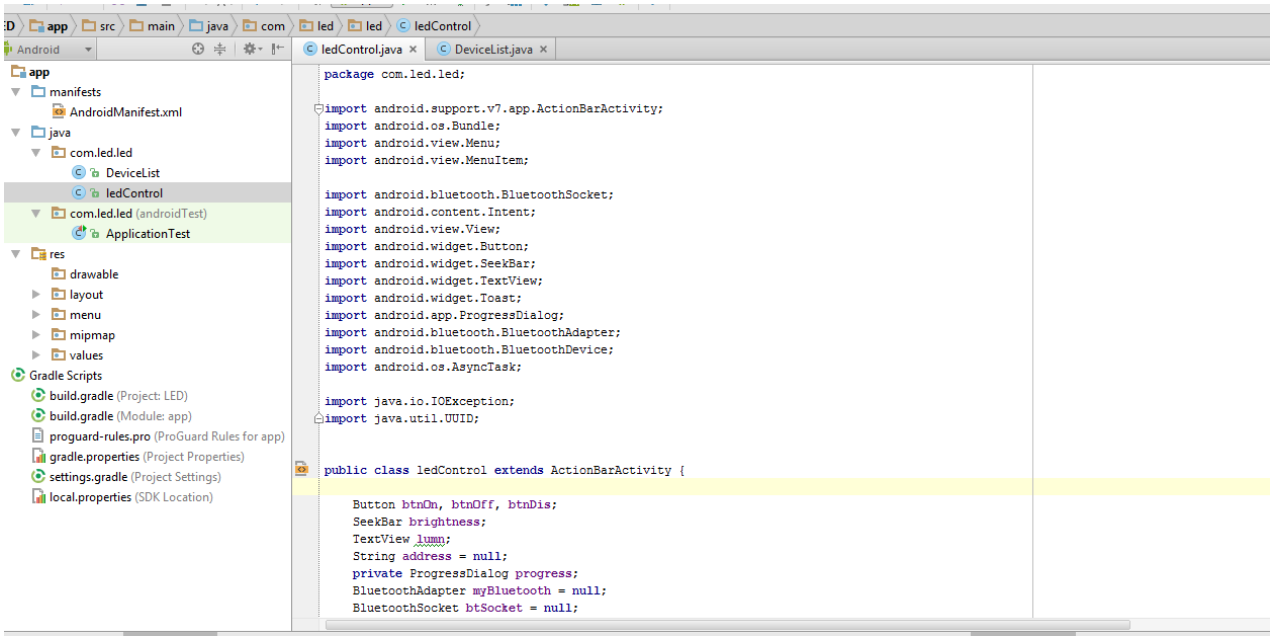
So Let’s add them:



Android: Class Code Part 2.

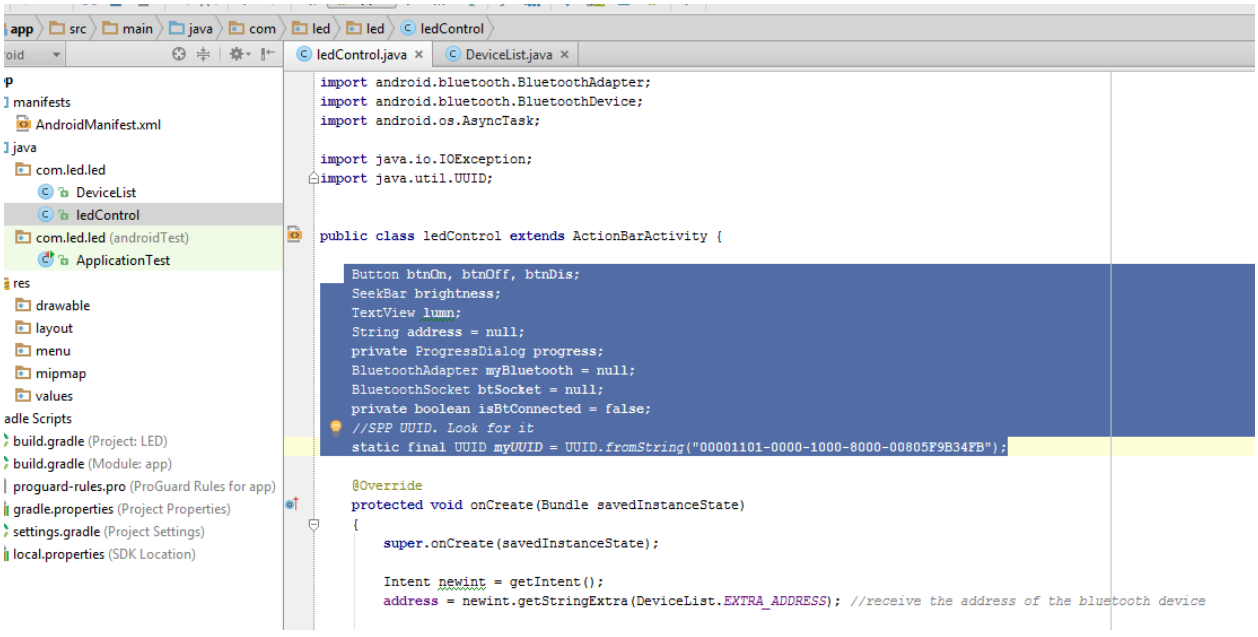
Open ledControl class and import the following packages:

import android.bluetooth.BluetoothSocket;
import android.content.Intent;
import android.view.View;
import android.widget.Button;
import android.widget.SeekBar;
import android.widget.TextView;
import android.widget.Toast;
import android.app.ProgressDialog;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.os.AsyncTask;
import java.io.IOException;
import java.util.UUID;



Create the following widget variables:

Button btnOn, btnOff, btnDis;
SeekBar brightness;
String address = null;
private ProgressDialog progress;
BluetoothAdapter myBluetooth = null;
BluetoothSocket btSocket = null;
private boolean isBtConnected = false;
static final UUID myUUID = UUID.fromString("00001101-0000-1000-8000-00805F9B34FB");



We have to initialize the variables and retrieve the bluetooth device address got in DeviceList class.

//receive the address of the bluetooth device

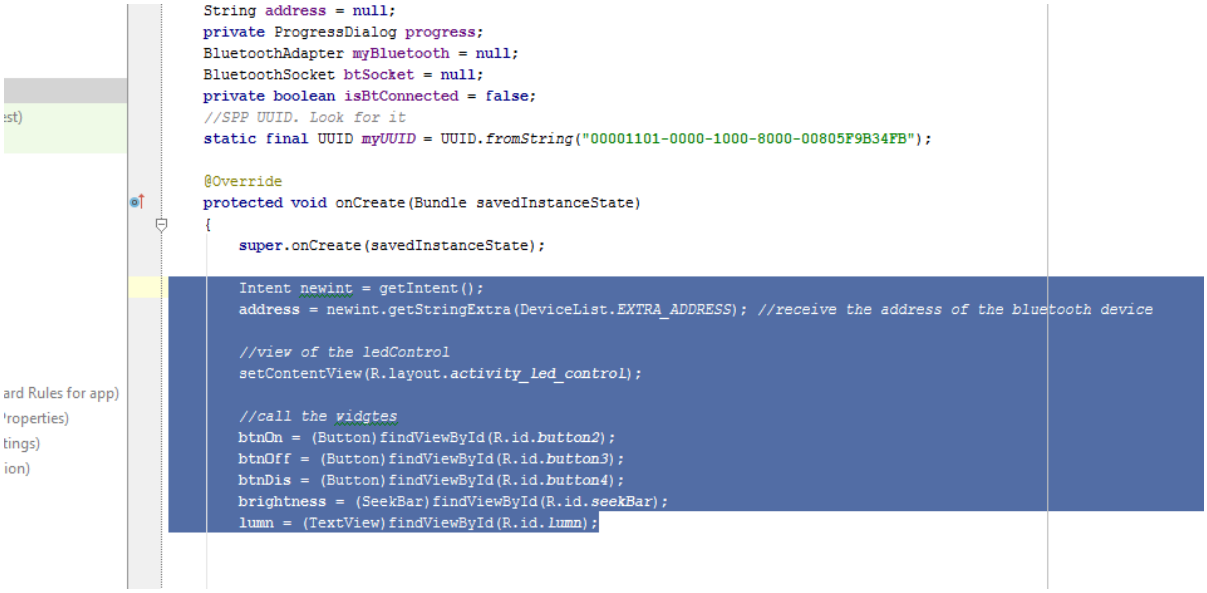
```
Intent newint = getIntent();
address = newint.getStringExtra(DeviceList.EXTRA_ADDRESS);
```

//view of the ledControl layout

```
setContentView(R.layout.activity_led_control);
```

//call the widgtes

```
btnOn = (Button)findViewById(R.id.button2);
btnOff = (Button)findViewById(R.id.button3);
btnDis = (Button)findViewById(R.id.button4);
brightness = (SeekBar)findViewById(R.id.seekBar);
```



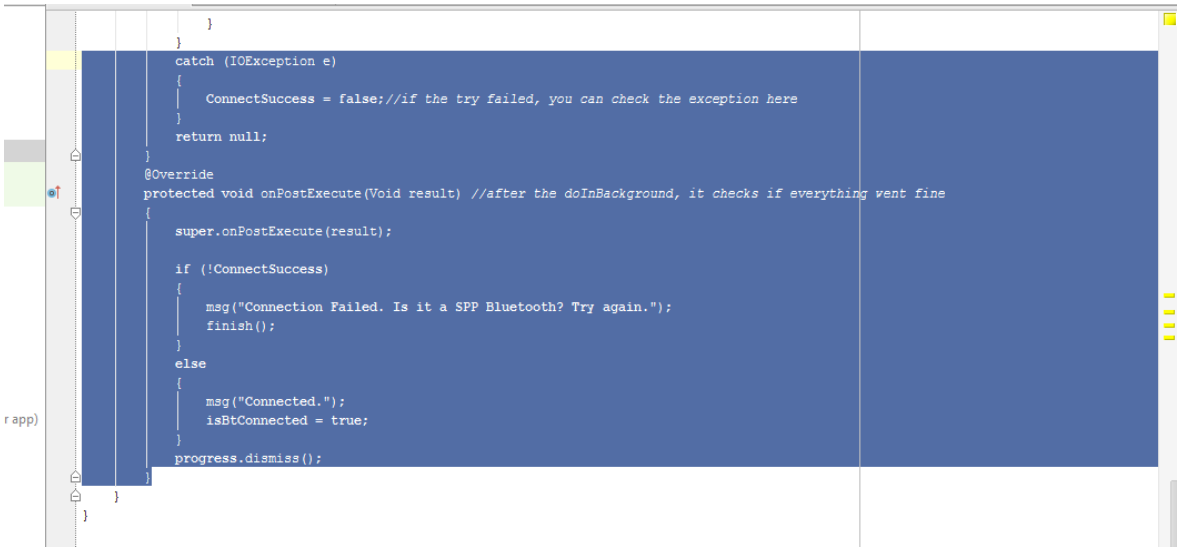
```
String address = null;
private ProgressDialog progress;
BluetoothAdapter myBluetooth = null;
BluetoothSocket btSocket = null;
private boolean isBtConnected = false;
//SPP UUID. Look for it
static final UUID myUUID = UUID.fromString("00001101-0000-1000-8000-00805F9B34FB");

@Override
protected void onCreate(Bundle savedInstanceState)
{
    super.onCreate(savedInstanceState);

    Intent newint = getIntent();
    address = newint.getStringExtra(DeviceList.EXTRA_ADDRESS); //receive the address of the bluetooth device

    //view of the ledControl
    setContentView(R.layout.activity_led_control);

    //call the widgtes
    btnOn = (Button)findViewById(R.id.button2);
    btnOff = (Button)findViewById(R.id.button3);
    btnDis = (Button)findViewById(R.id.button4);
    brightness = (SeekBar)findViewById(R.id.seekBar);
    lumn = (TextView)findViewById(R.id.lumn);
```

Let's create a class to start the connection:

```
private class ConnectBT extends AsyncTask<Void, Void, Void>  // UI thread
   {
      private boolean ConnectSuccess = true;
      @Override
      protected void onPreExecute()
      {
         progress = ProgressDialog.show(ledControl.this, "Connecting...", "Please wait!!!");
      }
      @Override
      protected Void doInBackground(Void... devices)
      {
         try
         {
            if (btSocket == null || !isBtConnected)
            {
```

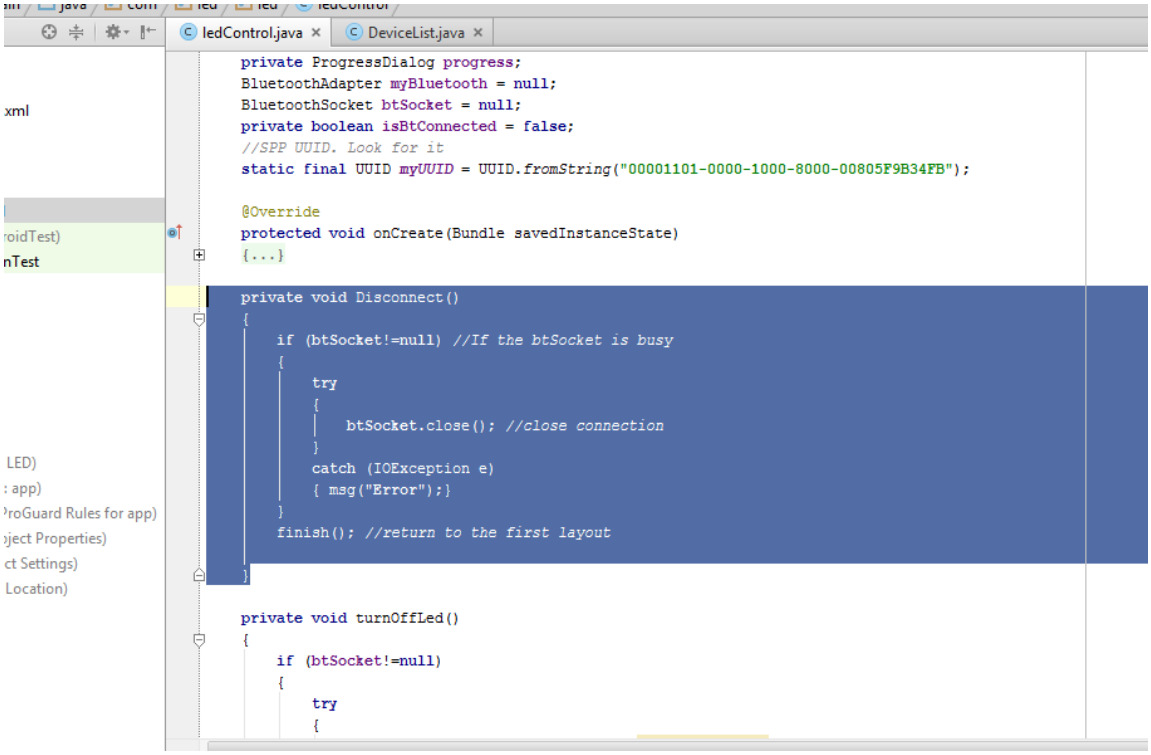
```
            myBluetooth = BluetoothAdapter.getDefaultAdapter();
            BluetoothDevice dispositivo = myBluetooth.getRemoteDevice(address);
            btSocket =
dispositivo.createInsecureRfcommSocketToServiceRecord(myUUID);
            BluetoothAdapter.getDefaultAdapter().cancelDiscovery();
            btSocket.connect();
        }
      }
      catch (IOException e)
      {
        ConnectSuccess = false;
      }
      return null;
    }
    @Override
    protected void onPostExecute(Void result)
    {
      super.onPostExecute(result);
      if (!ConnectSuccess)
      {
        msg("Connection Failed. Is it a SPP Bluetooth? Try again.");
        finish();
      }
      else
      {
        msg("Connected.");
        isBtConnected = true;
      }
      progress.dismiss();
    }
  }
```

```
    public boolean onCreateOptionsMenu(Menu menu) {...}

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {...}

    private class ConnectBT extends AsyncTask<Void, Void, Void>  // UI thread
    {
        private boolean ConnectSuccess = true; //if it's here, it's almost connected

        @Override
        protected void onPreExecute()
        {
            progress = ProgressDialog.show(ledControl.this, "Connecting...", "Please wait!!!");  //show a progress dialog
        }

        @Override
        protected Void doInBackground(Void... devices) //while the progress dialog is shown, the connection is done in background
        {
            try
            {
                if (btSocket == null || !isBtConnected)
                {
                 myBluetooth = BluetoothAdapter.getDefaultAdapter();//get the mobile bluetooth device
                 BluetoothDevice dispositivo = myBluetooth.getRemoteDevice(address);//connects to the device's address and checks if it's a
                 btSocket = dispositivo.createInsecureRfcommSocketToServiceRecord(myUUID);//create a RFCOMM (SPP) connection
                 BluetoothAdapter.getDefaultAdapter().cancelDiscovery();
                 btSocket.connect();//start connection
                }
            }
            catch (IOException e)
            {
                ConnectSuccess = false;//if the try failed, you can check the exception here
```



We need to “listen” when the button is clicked to write a command to turn on/turn off the led, disconnect and control the brightness.

```
private void Disconnect()
   {
      if (btSocket!=null) //If the btSocket is busy
      {
         try
         {
            btSocket.close(); //close connection
         }
         catch (IOException e)
         { msg("Error");}
      }
      finish(); //return to the first layout}
```




```
private void turnOffLed()
{
   if (btSocket!=null)
   {
      try
      {
         btSocket.getOutputStream().write("TF".toString().getBytes());
      }
      catch (IOException e)
      {
         msg("Error");
      }
   }
}
```


```
private void turnOnLed()
{
   if (btSocket!=null)
   {
      try
      {
         btSocket.getOutputStream().write("TO".toString().getBytes());
      }
      catch (IOException e)
      {
         msg("Error");
      }
   }
}
```


```
brightness.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {
    @Override
    public void onProgressChanged(SeekBar seekBar, int progress, boolean fromUser)
{
        if (fromUser==true)
        {
            lumn.setText(String.valueOf(progress));
            try
            {
                btSocket.getOutputStream().write(String.valueOf(progress).getBytes
());
            }
            catch (IOException e)
            {
            }
        }
    }
    @Override
    public void onStartTrackingTouch(SeekBar seekBar) {
```

```
    }
    @Override
    public void onStopTrackingTouch(SeekBar seekBar) {
    }
});
```

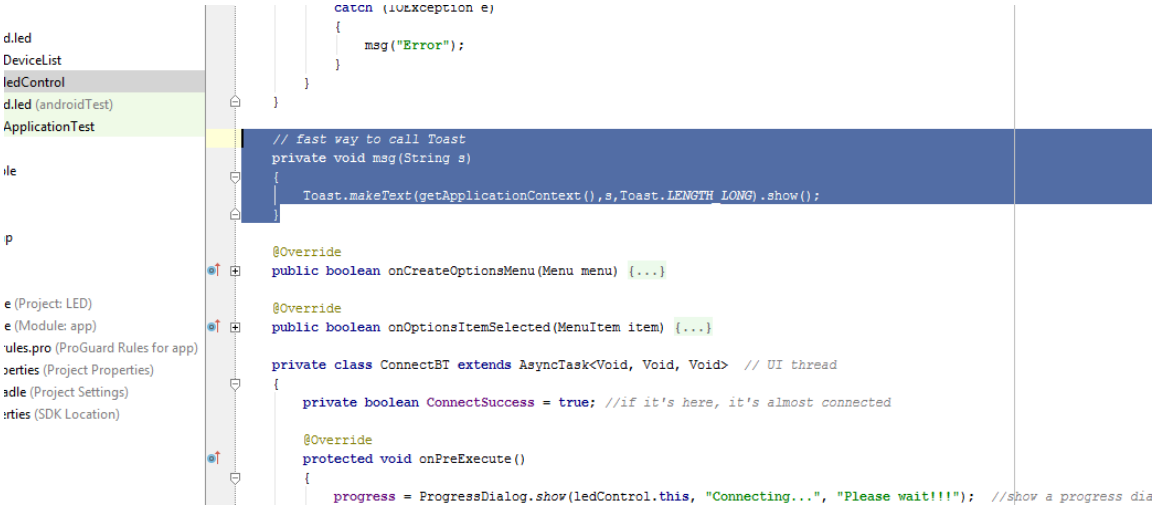
There is a method called *msg();* This method calls Toast.maketext();

private void msg(String s)

{

Toast.makeText(getApplicationContext(),s,Toast.LENGTH_LONG).show();

}



Android Manifest:

Every application must have an AndroidManifest.xml file (with precisely that name) in its root directory.

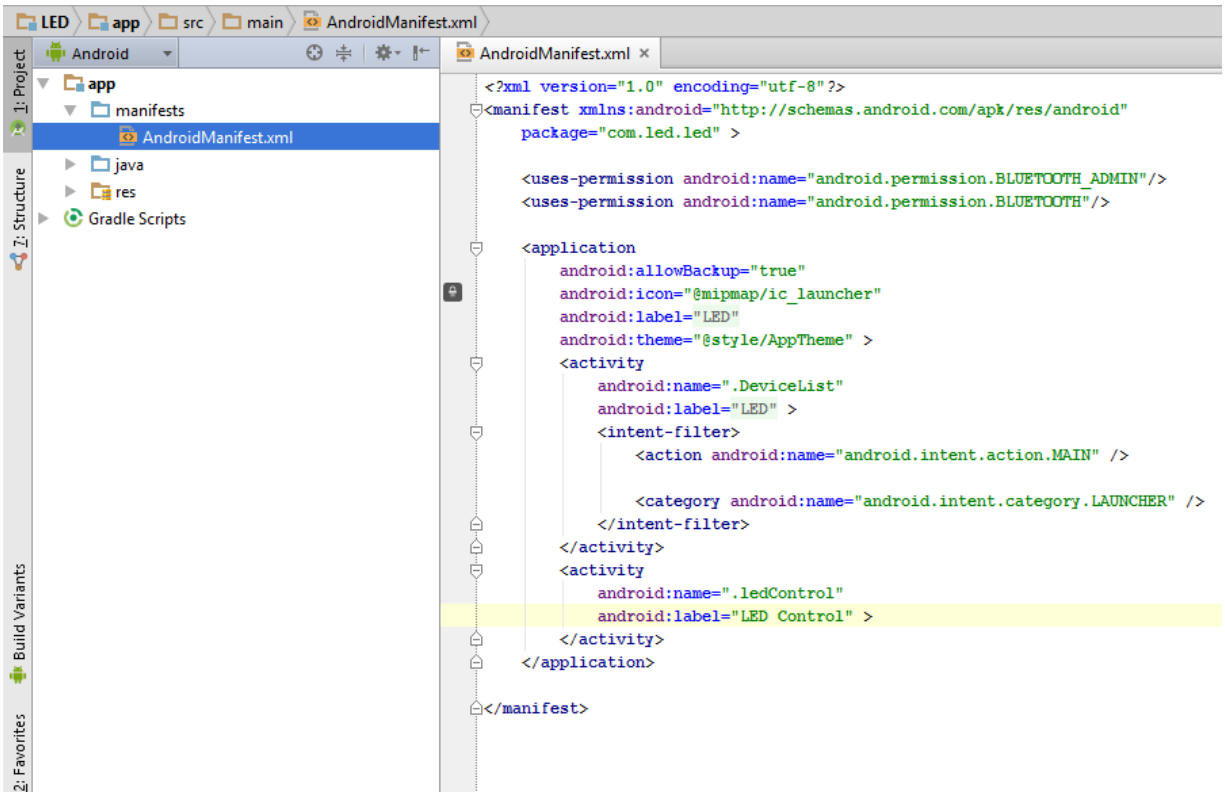
The manifest file presents essential information about your app to the Android system, information the system must have before it can run any of the app's code.

This apk uses Bluetooth Adapter and it is not available in emulator, you must test it in a running device, but before you have to add some users-permissions, otherwise the apk will crash.

In App folder, open manifests > AndroidManifest.xml

- Add the following code to allow user-permision to use Bluetooth Device

<uses-permission android:name= "android.permission.BLUETOOTH_ADMIN"/>
<uses-permission android:name= "android.permission.BLUETOOTH"/>



- Rebuild the project at: Build Menu > Rebuild Project

Now you can run it in your device.

Aruino Code

The arduino C code is very simple, no need to explain it:

```
char command;
String string;
boolean ledon = false;
```

```
#define led 5

  void setup()
  {
    Serial.begin(9600);
    pinMode(led, OUTPUT);
  }
  void loop()
  {
    if (Serial.available() > 0)
    {string = "";}
    while(Serial.available() > 0)
    {
      command = ((byte)Serial.read());
      if(command == ':')
      {
        break;
      }
      else
      {
        string += command;
      }
      delay(1);
    }
    if(string == "TO")
    {
       ledOn();
       ledon = true;
    }
    if(string =="TF")
    {
       ledOff();
       ledon = false;
       Serial.println(string); //debug
    }
    if ((string.toInt()>=0)&&(string.toInt()<=255))
    {
      if (ledon==true)
      {
        analogWrite(led, string.toInt());
        Serial.println(string); //debug
        delay(10);
      }
    }
 }

void ledOn()
   {
      analogWrite(led, 255);
      delay(10);
```

```
  }
void ledOff()
{
    analogWrite(led, 0);
    delay(10);
}
```