CSE2MAD LAB 5 – BLUETOOTH SCANNER WITH MENU

AIMS

Today will be making a Bluetooth Scanner app we will be exploring;

- The BluetoothAdapter class
- Dangerous permissions at Runtime
- Custom menus
- And the Switch View

We will be revisiting various concepts learned through the semester including;

- Broadcast receivers
- ArrayAdapters

STEP 1 – MAKING THE MAIN ACTIVITY UI & REQUESTING PERMISSIONS

a) Create a new project & choose the empty activity template. (See week 1 lab)
b) Create the UI in the visual DESIGN view for the main activity as in Figure 1. If you need a hint see the component tree in Figure 2 and layout code from previous labs.

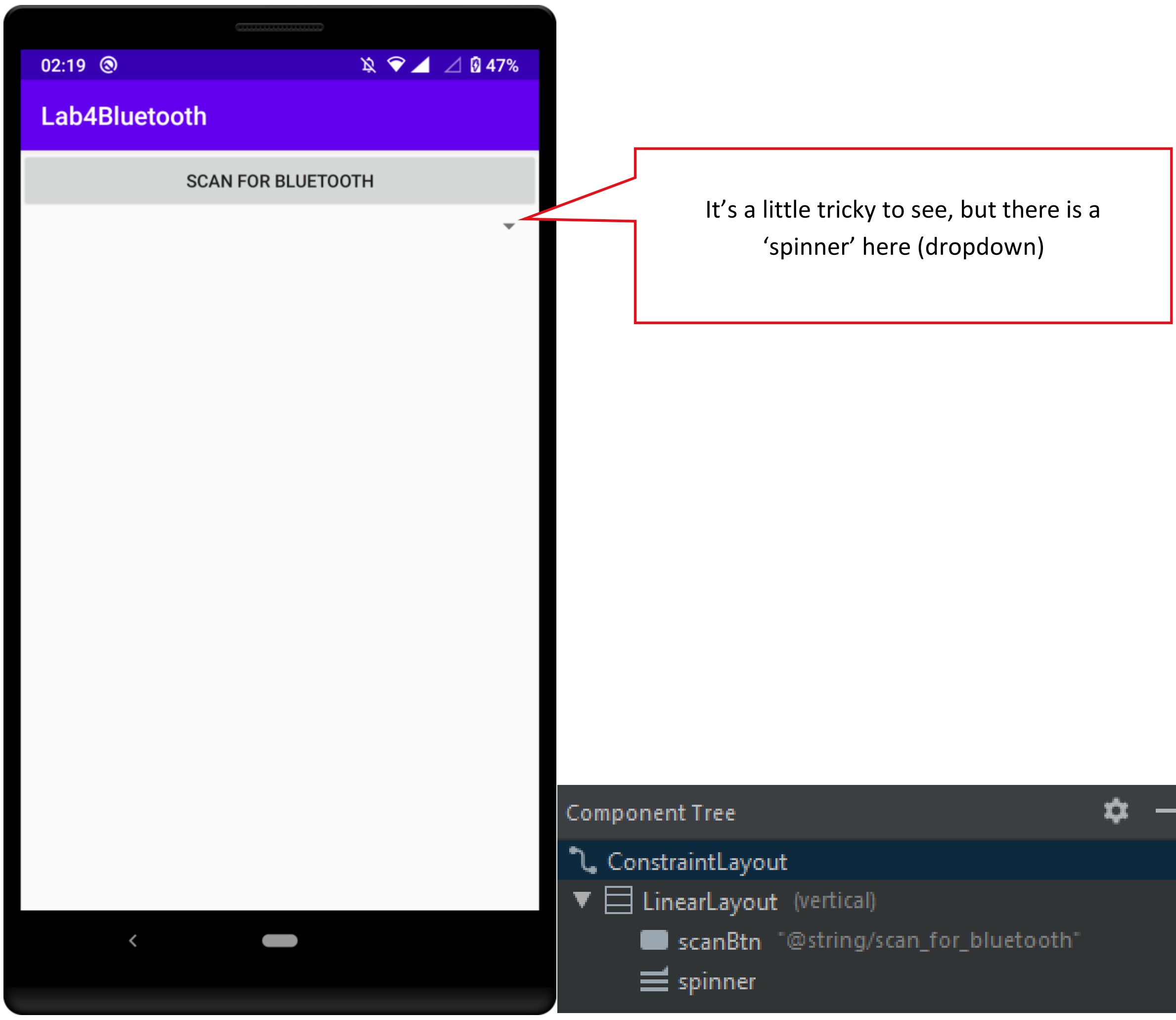

02:19
47%
Lab4Bluetooth
SCAN FOR BLUETOOTH
It's a little tricky to see, but there is a 'spinner' here (dropdown)
Component Tree
ConstraintLayout
LinearLayout (vertical)
scanBtn "@string/scan_for_bluetooth"
spinner

Fig 1 Fig 2

c) Now set up the member variables for your button & spinner and binding them to your view objects, and add the button onClick Listener

```
private Spinner resultsSpinner;
private Button scanBtn;
private String TAG = "MainAct";

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    //Spinner
    resultsSpinner = (Spinner) findViewById(R.id.spinner);

    //Button
    scanBtn = (Button) findViewById(R.id.scanBtn);
    scanBtn.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            checkPermission();

        }
    });
}
```

d) As we are dealing with Bluetooth add a Bluetooth member variable

```
private BluetoothAdapter BA;
```

e) And initalise the Bluetooth adapter object in the onCreate() so that we can access its methods.

```
// initialise bluetooth Adapter
BA = BluetoothAdapter.getDefaultAdapter();
```

f) Now we need to detect the discoverable Bluetooth devices nearby, but we need to add our permissions first to the normal place, the AndroidManifest.xml

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

g) As per this weeks lecture, ACCESS_FINE_LOCATION is considered a ‘Dangerous permission” and a request must be made to the user DURING RUNTIME, add the method below OUTSIDE your onCreate() and call it from your button onClick() method.
As you can see this function checks if permission has been and if it has will start the scan with the BluetoothAdapter startDiscovery method. If permission has not been granted the permission will be requested.

```
// Function to check and request permission
public void checkPermission()
{
    // Checking if permission is not granted
    if (ContextCompat.checkSelfPermission( MainActivity.this,
Manifest.permission.ACCESS_FINE_LOCATION) == PackageManager.PERMISSION_DENIED) {
        Log.d(TAG, "Permission not set requesting now");
        ActivityCompat.requestPermissions(MainActivity.this,
                new String[]{Manifest.permission.ACCESS_FINE_LOCATION},
                FINE_LOCATION_REQUEST_CODE);
    }
    else {
        Log.d(TAG, "Permission already granted");
        BA.startDiscovery();
    }
}
```

h) You will also need to add the request code to your member variables

```
private int FINE_LOCATION_REQUEST_CODE = 8;
```

i) So far we have asked permission but we need a callback to handle the users response such as below, where if the request code matches that from task h) the results are assessed and if the user granted permissions we can again use the startDiscovery() method.

```
@Override
public void onRequestPermissionsResult(int requestCode,
                                       @NonNull String[] permissions,
                                       @NonNull int[] grantResults)
{
    super.onRequestPermissionsResult(requestCode, permissions, grantResults);

    if (requestCode == FINE_LOCATION_REQUEST_CODE) {
        // Checking whether user granted the permission or not.
        if (grantResults.length > 0 && grantResults[0] == PackageManager.PERMISSION_GRANTED) {
            BA.startDiscovery();
            Log.d(TAG, "Permission granted Initiating Scan");
        }
        else {
            Log.d(TAG, "Permission denied");
        }
    }
}
```

Now we have finally finished the setup we can start working on the scan functionality.

STEP 2 – SCAN FOR BLUETOOTH DEVICES

a) Now using the BA.startDiscovery() call in tasks g & i our phone has already started looking for bluetooth discoverable devices.

When as device has been located an "android.bluetooth.device.action.FOUND" broadcast will be generated, so we will need a broadcast receiver, and as we want to interact easily with the activity we will make it a subclass inside the MainActivity like we did last week.

```
//Detect Bluetooth state change and handle switch
public class MyBluetoothReceiver extends BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent intent) {
        String action = intent.getAction();
        //Finding devices
        Log.d(TAG, "he");
        if (BluetoothDevice.ACTION_FOUND.equals(action))
        {
             BluetoothDevice device = intent.getParcelableExtra(BluetoothDevice.EXTRA_DEVICE);
             Log.d(TAG, "Found: " + device.getName());

        }
    }
}
```

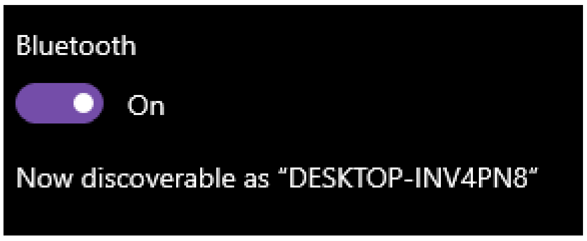
b) And of course as we have a Broadcast Receiver we need an..........? Intent filter yes and to register our Broadcast receiver. First we must declare the Broadcast receiver with the rest of our member variables

```
private BroadcastReceiver BTrx;
```

c) And add the filter and registerReceiver methond into the onCreate() in the MainActivity.

```
//Filter & Register RX
IntentFilter filter = new IntentFilter();
//receive system bluetooth broadcast
filter.addAction("android.bluetooth.device.action.FOUND");
BTrx = new MyBluetoothReceiver();
registerReceiver(BTrx, filter);
```

Now you should be able to detect discoverable Bluetooth devices BUT they will need to be set as discoverable. The devices will appear in your logcat



Now discoverable as “DESKTOP-INV4PN8”

```
2020-08-25 04:33:53.299 24082-24082/com.example.lab4bluetooth D/MainAct: Found: DESKTOP-INV4PN8
```

STEP 3 – SET UP SPINNER & ARRAY ADAPTER

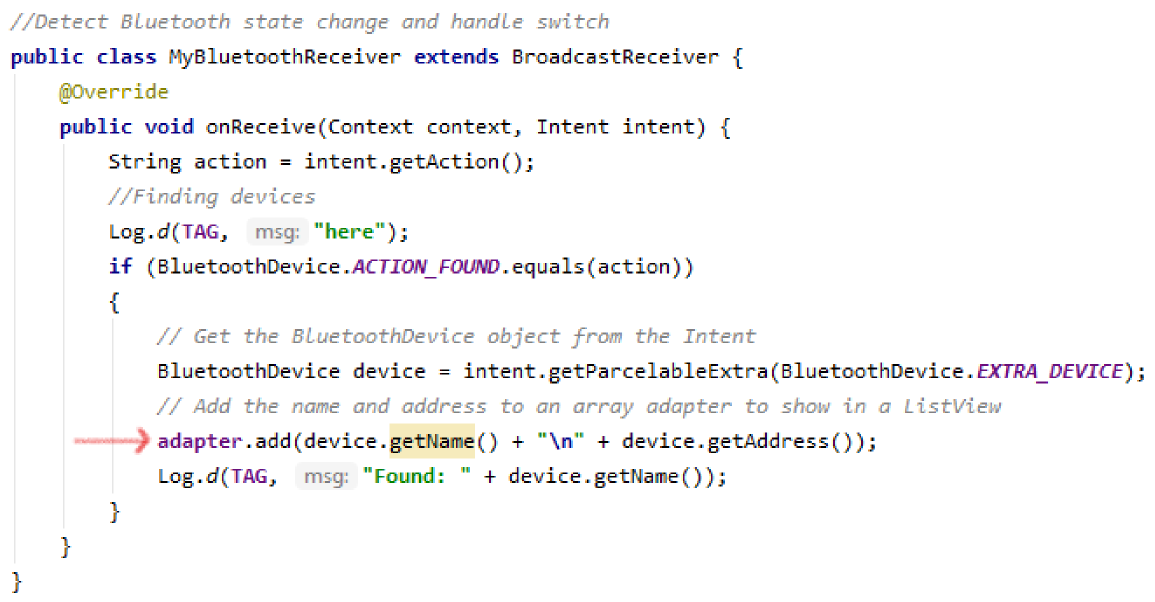
a) Now that we are getting results it would be nice to display them somewhere.. a spinner perhaps.. Thus far we have linked the spinner object to the View in the onCreate() but some additional preparation is required. Add the following code into the onCreate(). Here we are making an ArrayList to store the information we have received. Next we create an arrayAdapter where we specify the xml of the spinner and the list we created earlier. Then we set the adapter to the results spinner.

```
//prepare arraylist to store device details
ArrayList list = new ArrayList();
//set arrayList values to spinner
adapter = new ArrayAdapter<String>(
        this, android.R.layout.simple_spinner_item, list);
// Specify the layout to use when the list of choices appears
adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
// Apply the adapter to the spinner
resultsSpinner.setAdapter(adapter);
```

b) We will also need to add the adapter as a member variable !

```
private ArrayAdapter<String> adapter;
```

c) Now in the Broadcast receiver we have to add the devices to the arraylist adapter as they are found, this will propagate the spinner.

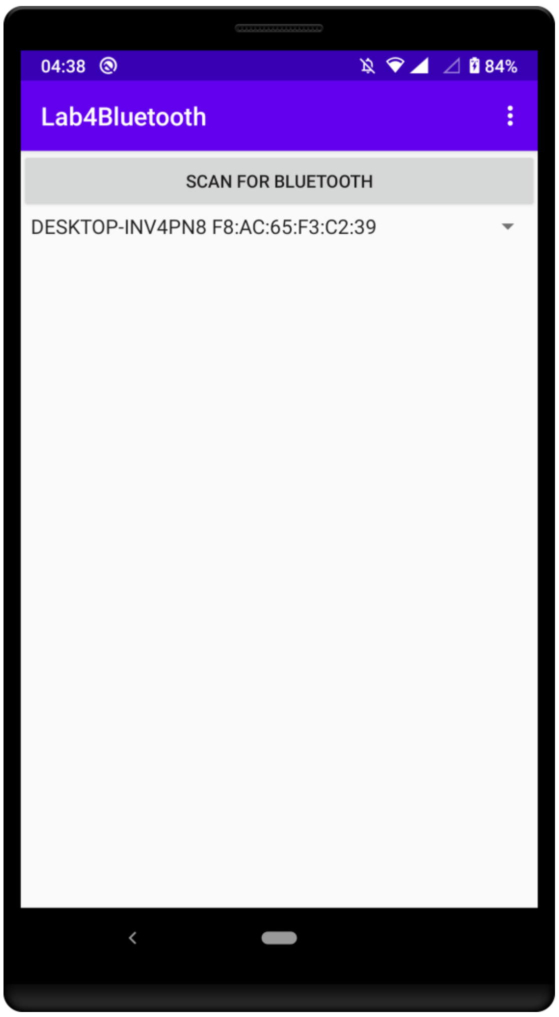


```
//Detect Bluetooth state change and handle switch
public class MyBluetoothReceiver extends BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent intent) {
        String action = intent.getAction();
        //Finding devices
        Log.d(TAG,  msg: "here");
        if (BluetoothDevice.ACTION_FOUND.equals(action))
        {
            // Get the BluetoothDevice object from the Intent
            BluetoothDevice device = intent.getParcelableExtra(BluetoothDevice.EXTRA_DEVICE);
            // Add the name and address to an array adapter to show in a ListView
            adapter.add(device.getName() + "\n" + device.getAddress());
            Log.d(TAG,  msg: "Found: " + device.getName());
        }
    }
}
```

d) And lastly we must unregister the broadcast receiver in the onDestroy()

```
//Unregister the broadcast receiver when it is no longer needed:
@Override
protected void onDestroy() {
    super.onDestroy();
    unregisterReceiver(BTrx);
}
```

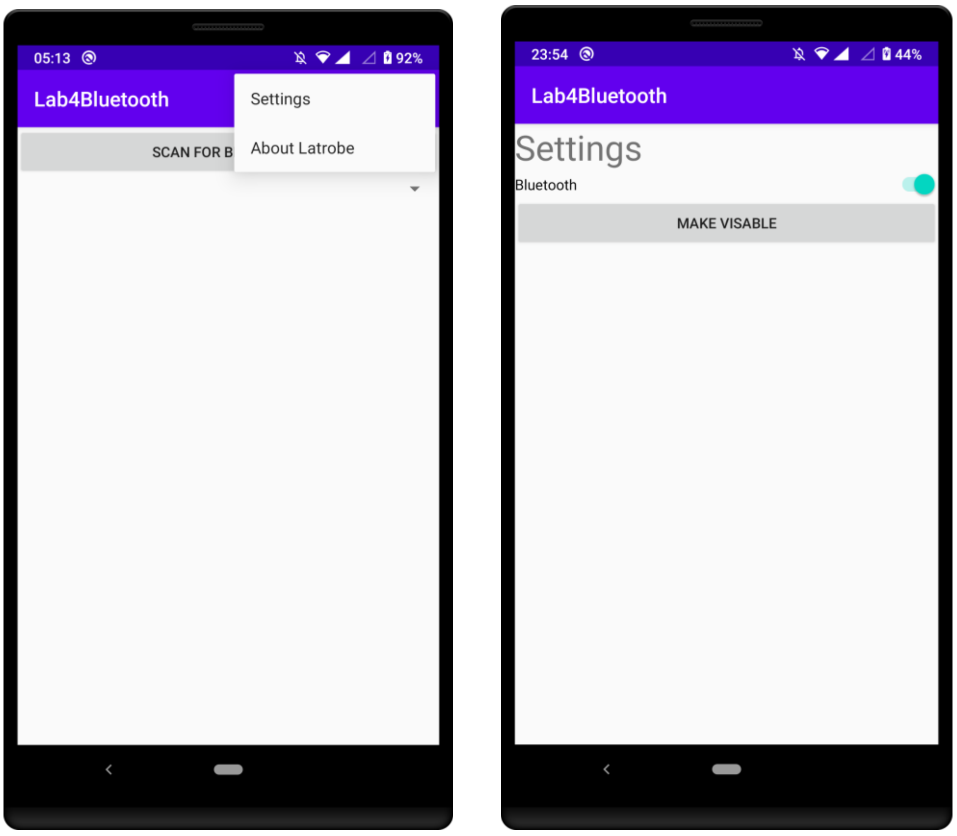
Now you should have a functioning app ready to scan all your devices 😉



DESKTOP-INV4PN8 F8:AC:65:F3:C2:39

CSE2MAD LAB 5 – OPTIONAL ADD MENU & SETTINGS ACTIVITY

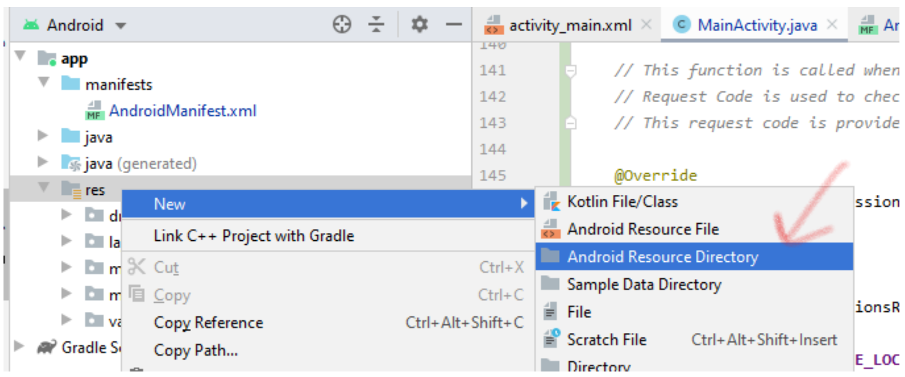
Now we are going to add a custom menu and Settings activity.



STEP 4 – SET UP THE MENU

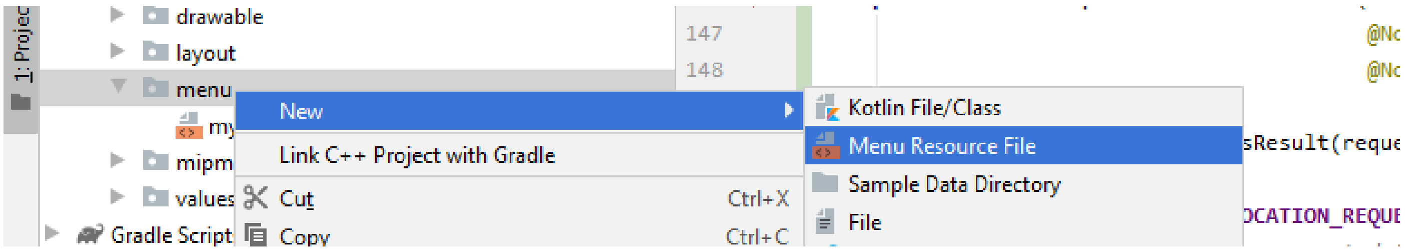
First we need to create the menu Directly and xml.

a) Create a new Android resource directory named menu (Resource type menu)



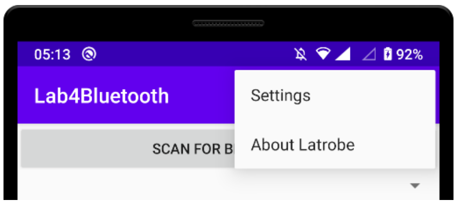
```
// This function is called when
// Request Code is used to chec
// This request code is provide

@Override
```

b) Click on the menu directory you just created and make a Menu Resource File named my_menu.xml



Open the file and add the 2 items that we will include in our menu. You can open the design view and manually drop them in or use the xml below.

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item
        android:id="@+id/settings"
        android:title="Settings" />
    <item
        android:id="@+id/about"
        android:title="About Latrobe" />
</menu>
```



c) Now we have created a lout for our menu we need to add it to our MainActivity so we can see/use it . Just like am activity it has its onCreate that inflates the xml you just created to make the menu appear at the top right.

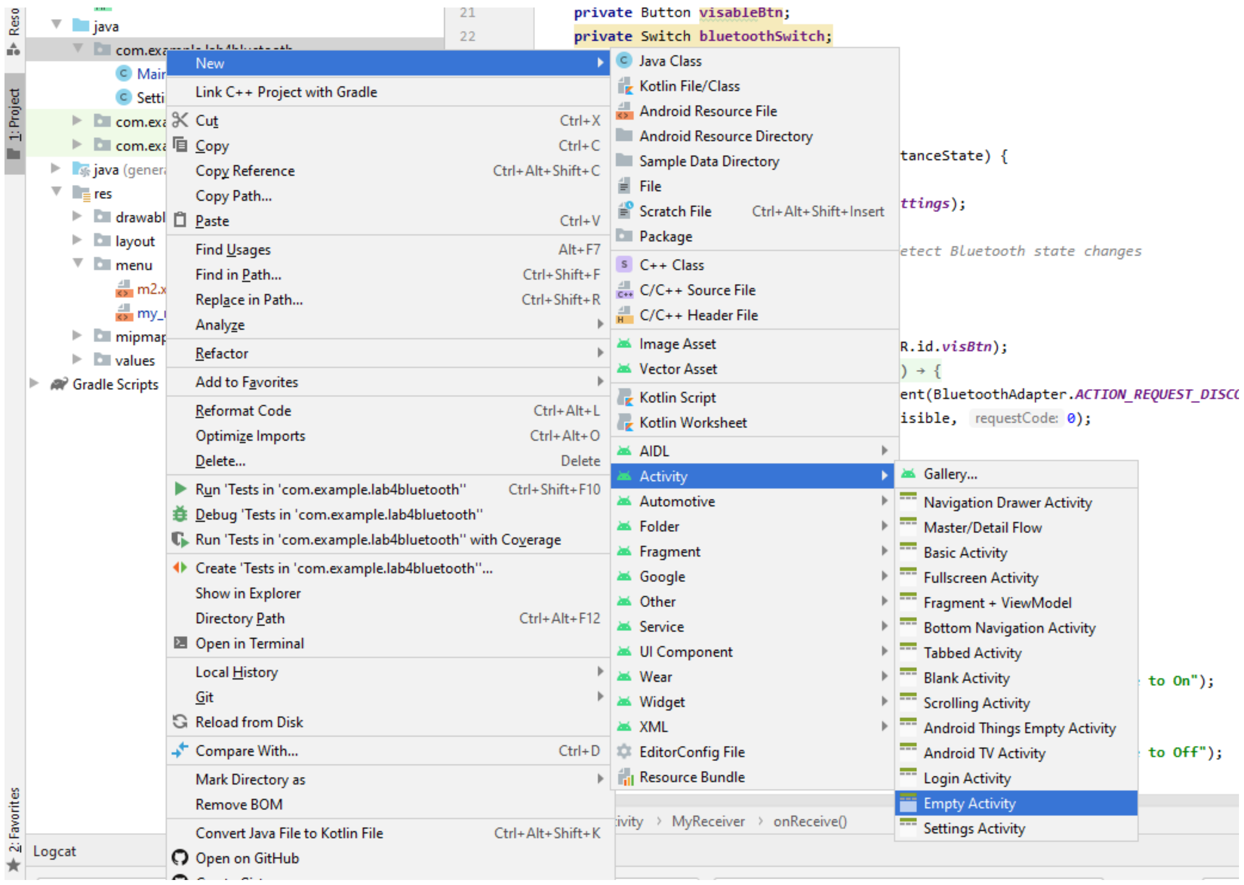
```
//menu onCreate
public boolean onCreateOptionsMenu(Menu menu) {
    MenuInflater inflater = getMenuInflater();
    inflater.inflate(R.menu.my_menu, menu);
    return true;
}
```

d) We now have a menu, but try to click on it, does anything happen? No .. ahh we must handle the Item selected Event. This again is in the MainActriviity.java file. I have used a switch statement here and set the setting Item to open a new Activity (which we haven't created yet) and the About Item opens a webpage.

```java
//handle menu clicks
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case R.id.settings:
            //open Settings Activity
            startActivity(new Intent(this, SettingsActivity.class));
            Log.d(TAG, "Opening settings");
            return true;
        case R.id.about:
            Log.d(TAG, "Opening website");
            //open website via intent
            String url = "https://www.latrobe.edu.au/";
            Intent i = new Intent(Intent.ACTION_VIEW);
            i.setData(Uri.parse(url));
            startActivity(i);
            return true;
        default:
            return super.onOptionsItemSelected(item);
    }
}
```

STEP 5 – THE SETTINGS ACTIVITY

a) First we need to create a new activity and name it SettingsActivity (so it matches our menu).



```
private Button visableBtn;
private Switch bluetoothSwitch;
tanceState) {
ttings);
etect Bluetooth state changes
R.id.visBtn);
) → {
ent(BluetoothAdapter.ACTION_REQUEST_DISC
isible,  requestCode: 0);
 to On");
 to Off");
```

b) Create the UI in the visual DESIGN view for the settings activity as in Figure 3. If you need a hint see the component tree in Figure 4 and layout code from previous labs.

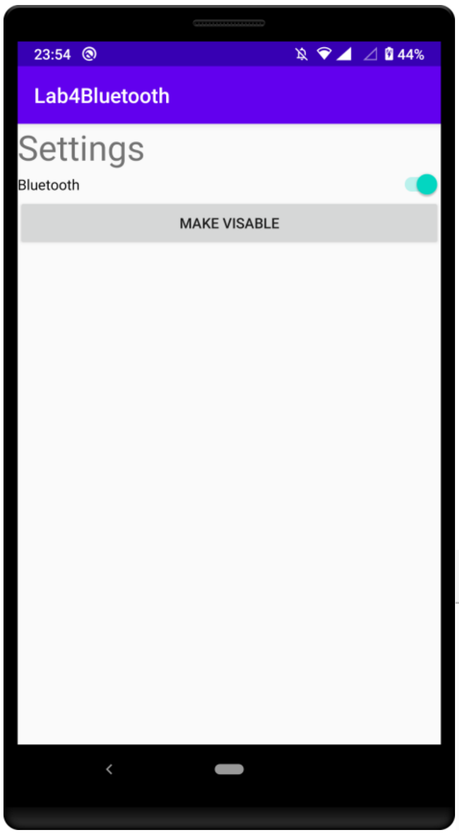


Fig 3

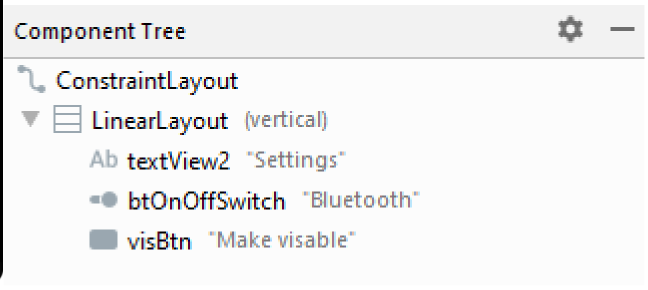


Fig 4

c) Now open SettingsActivity.java file and add in the button and switch member variables and initialise them in the onCreate().

```java
private Button visableBtn;
private Switch bluetoothSwitch;
private String TAG = "SettingsAct";

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_settings);

    //Button
    visableBtn = (Button) findViewById(R.id.visBtn);


    //Switch to set Bluetooth on/off
    bluetoothSwitch = (Switch) findViewById(R.id.btOnOffSwitch);
```

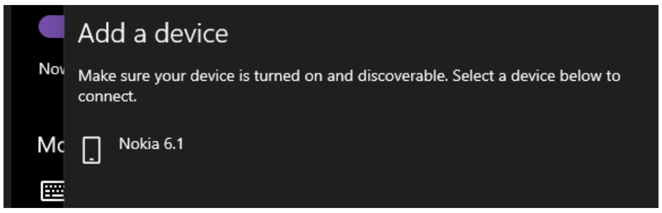
d) As we will be again dealing with Bluetooth we will need a Bluetooth Adapter object add the member and object initialisation as per the last activity.

```
private BluetoothAdapter BA;

//Bluetooth adapter
BA = BluetoothAdapter.getDefaultAdapter();
```

e) So what we want to do in this activity is let the user turn their bluetooth on/off and make their phone discoverable so other devices can connect to it.

So create a standard onClickListener for your button in the onCreate(), inside creating an intent that when passed to the startActivityForResult() method will enable device discovery.
https://developer.android.com/guide/topics/connectivity/bluetooth#EnableDiscoverability

```
 visableBtn.setOnClickListener(new View.OnClickListener() {
     @Override
     public void onClick(View view) {
         //Intent that will make your device discoverable
         Intent getVisible = new Intent(BluetoothAdapter.ACTION_REQUEST_DISCOVERABLE);
         startActivityForResult(getVisible, 0);
     }
 });
```


Add a device

Make sure your device is turned on and discoverable. Select a device below to connect.

Nokia 6.1

f) So now we have a switch but nothing happens when we change it, so we need a listener so we can respond to the switches change in state. We are again using an intent, this time to signal we require the bluetooth on, and the Bluetooth adapter method disable() to turn the bluetooth off.
https://developer.android.com/guide/topics/connectivity/bluetooth#SettingUp

```
//Respond to switch changes
bluetoothSwitch.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {
    public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
        if (isChecked) {
            Intent turnOn = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
            startActivityForResult(turnOn, 0);
            Log.d(TAG, "Attempting to turn on BT via intent");
        } else {
            BA.disable();
            Log.d(TAG, "Switch has been set to off, attempting to disable bluetooth");
        }
    }
});
```

So now we can turn the bluetooth on and off, and make the phone discoverable however what happens if your bluetooth is on and you open settings? Does it show the bluetooth is on.. or off?

g) Using the BluetoothAdapter method .isEnabled() (that returns a Boolean) set the switch to the correct position when the user opens the Activity.
https://developer.android.com/reference/android/bluetooth/BluetoothAdapter#isEnabled()

you can change the switch state with;

```
bluetoothSwitch.setChecked(true); or bluetoothSwitch.setChecked(false);
```

To work out where to put the code consider at which stage of the Activity life cycle do you want the switch to be checked if it is in the right position?

h) So now what happens if the user turns the bluetooth on in a dialog? Or somewhere else, the last task is to consider how your app could be informed of this operating system level change add how you could react to it. This is a bit tricky….maybe check the week 4 lectures..