

# Lesson 5 – User Interfaces Part 1

---

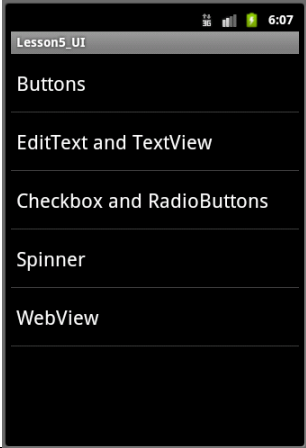
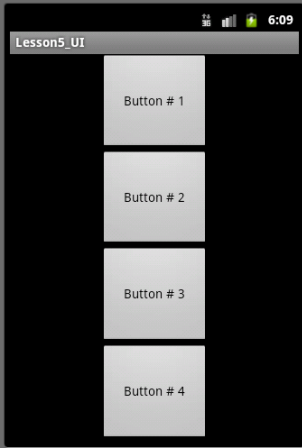
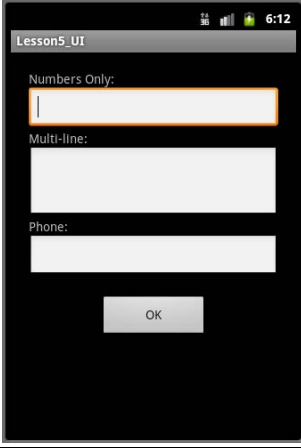
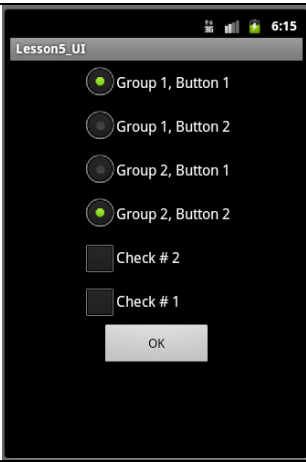
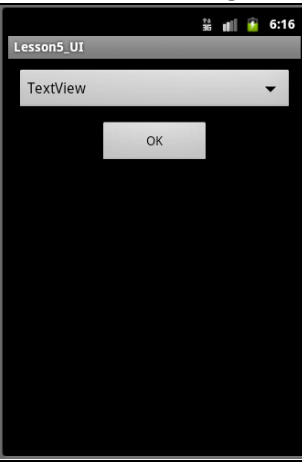
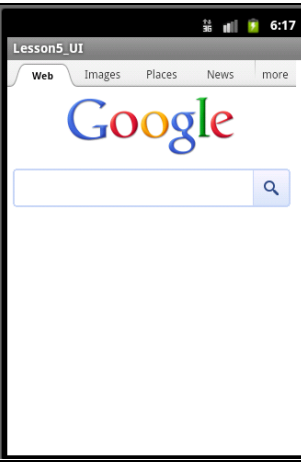
The application that we'll be building over the next several lessons consists of six different screens (see screenshots below), and will basically give us an example of the most popular GUI elements, and how they work within one single application.

In this part, we're going to tackle the initial setup of our application, as well as the building of a ListView control, and loading that control with data from an array.

We'll work on the following:

- Build new class files for the following screens:
  - Button Example
  - Checkbox And Radio Button Example
  - Edit Text Example
  - Spinner Example
  - WebView Example
- Modify the main.xml view to include a ListView control.
- Create and load an array of data into the ListView.
- Modify the AndroidManifest.xml file, and add the new classes to the list of allowed activities.
- Add the INTERNET access permission (also done in the AndroidManifest.xml file), needed by the WebView example.

## Screens

		
Main Screen – ListView used for navigation.	Buttons screen – Shows off the weight property, button click event, and toast messages.	EditText View, and different input types.
		
Checkboxes and Radio Buttons	Spinner (Dropdown list)	WebView

Create a new project in Eclipse, with the following settings:

Project name: **Lesson5\_UI**

Create new project in workspace should be selected.

Build Target: **Android 2.3.3**

Application Name: **Lesson5\_UI**

Package name: **lastname.firstname.Lesson5\_UI** (substitute lastname.firstname with your last name and first name – no spaces, apostrophes or dashes.)

Create Activity should be checked, and **Lesson5\_UI** should be the text for it.

Min SDK Version: **10**

Click **Finish**.

Start by opening the main.xml file. We're going to replace the current default layout with a ListView control. ListView control requires the id "@+id/android:list" in order to work.

```
<?xml version="1.0" encoding="utf-8"?>

<ListView xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/android:list" android:layout_width="fill_parent"
    android:layout_height="fill_parent">
</ListView>
```

Next, we'll need to add new classes for each of the screens that we're building in this application. In Eclipse, click on the new Class button in the toolbar:

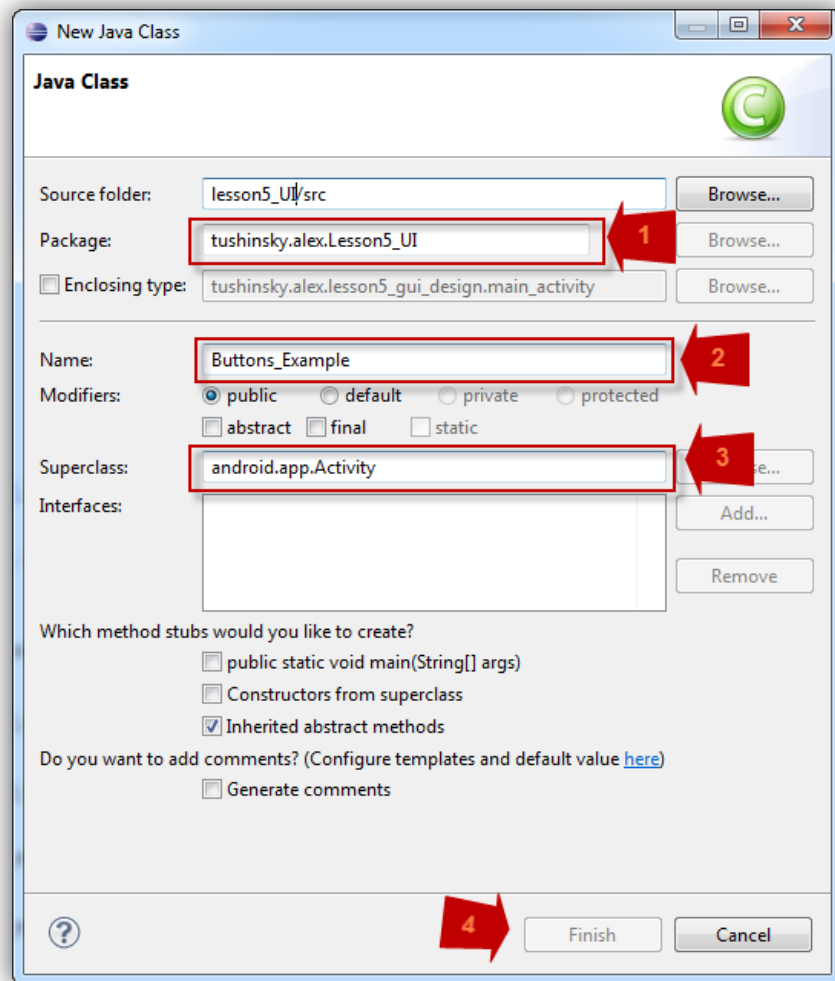


Fill in the form for each of the following class names (2):

- Button\_Example
- Checkbox\_Radio\_Example
- EditText\_Example
- Spinner\_Example
- WebView\_Example

All 5 classes need to be part of your default package (1), and will be extended from android.app.Activity (3).

When done, your package folder under SRC will contain a total of six java files.



Open each of the 5 newly created files, and add the default onCreate method. Right-click in a blank area of the code screen, select Source, then choose Override/Implement Methods... Select onCreate.

Next, we're going to setup our ListView screen. Open your Lesson5UI.java file, and add the following code:

```

public class main_activity extends ListActivity {

    String[] GUIExamples = { "Buttons", // 0
                             "EditText and TextView", // 1
                             "Checkbox and RadioButtons", // 2
                             "Spinner", // 3
                             "WebView" // 4
    };

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        setListAdapter(new ArrayAdapter<String>(this,
                                                android.R.layout.simple_list_item_1, GUIExamples));
    }

    public void onListItemClick(ListView parent, View v, int position, long id) {

        switch (position) {
            case 0: // Buttons
                startActivity(new Intent(this, Button_Example.class));
                break;
            case 1: // EditText / TextView
                startActivity(new Intent(this, EditText_Example.class));
                break;
            case 2: // Checkbox / Radiobutton
                startActivity(new Intent(this, Checkbox_Radio_Example.class));
                break;
            case 3: // Spinner
                startActivity(new Intent(this, Spinner_Example.class));
                break;
            case 4: // WebView
                startActivity(new Intent(this, WebView_Example.class));
                break;
        }
    }
}

```

We, first, change the class by extending it from ListActivity, instead of Activity. This allows it to inherit and work with the various list specific features. Next, we added an array of strings called GUIExamples, which will be the values we want our ListView to display. In the onCreate event, we call the setListAdapter, which loads our Array into the ListView, displaying it using a simple single item list template that is built into the OS. Alternatively, we could have defined our own look for the list's row by creating our own XML layout. Finally, the onListItemClick event is fired whenever we tap on something in the list. This will in turn call up one of our activities that we created using the new Class dialog.

Save the file.

Open the AndroidManifest.xml file, and add the following items, highlighted in yellow:

```
<uses-permission android:name="android.permission.INTERNET" />
<application android:icon="@drawable/icon" android:label="@string/app_name">
    <activity android:name=".Lesson5_UI"
        android:label="@string/app_name">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>

    <activity android:name="Button_Example"></activity>
    <activity android:name="Checkbox_Radio_Example"></activity>
    <activity android:name="EditText_Example"></activity>
    <activity android:name="Spinner_Example"></activity>
    <activity android:name="WebView_Example"></activity>
</application>
```

The first item, “uses-permission” will add the ability for our application to access the internet. This will be needed for the WebView example that we’ll work on later.

The second set of items tell Android that we have additional activities within our application. If we didn’t add them here to the manifest, we would get Force Close errors every time we opened one of the screens.

Save the manifest and run the application. Note that you’ll only be able to see and work with the first screen. The rest of the files are empty and won’t generate any output.