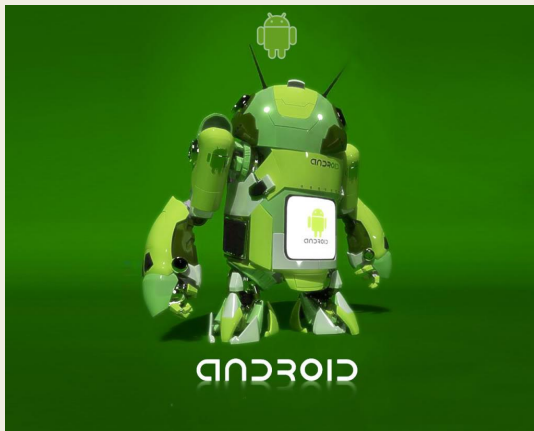


# ACM Android Tutorial

Building a Todo Application with Android Studio

# Todo App - Simple & Functional

- Best way to **learn** is to **build**!
- We'll look at the **essentials** for apps
- We'll **build** a simple **Todo** List



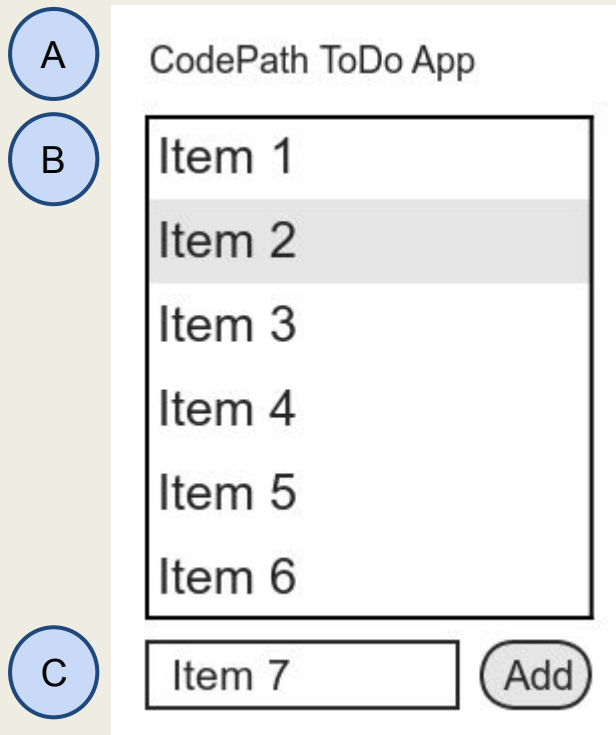
# Scoping the Todo App

- **View** a list of existing items
- **Add** a new item
- **Remove** an item

For this application we need just **one screen** which mean a single *Activity*.

# Wireframe the Todo App

## Basic interface:



(A) Basic Label with App Name


(B) Basic List of Items

- Vertically Scrollable
- Hold Down to Remove Item

(C) Adding items with Textbox and Button

# Create New Android Project

Create New Project

 **New Project**  
Android Studio

**Configure your new project**

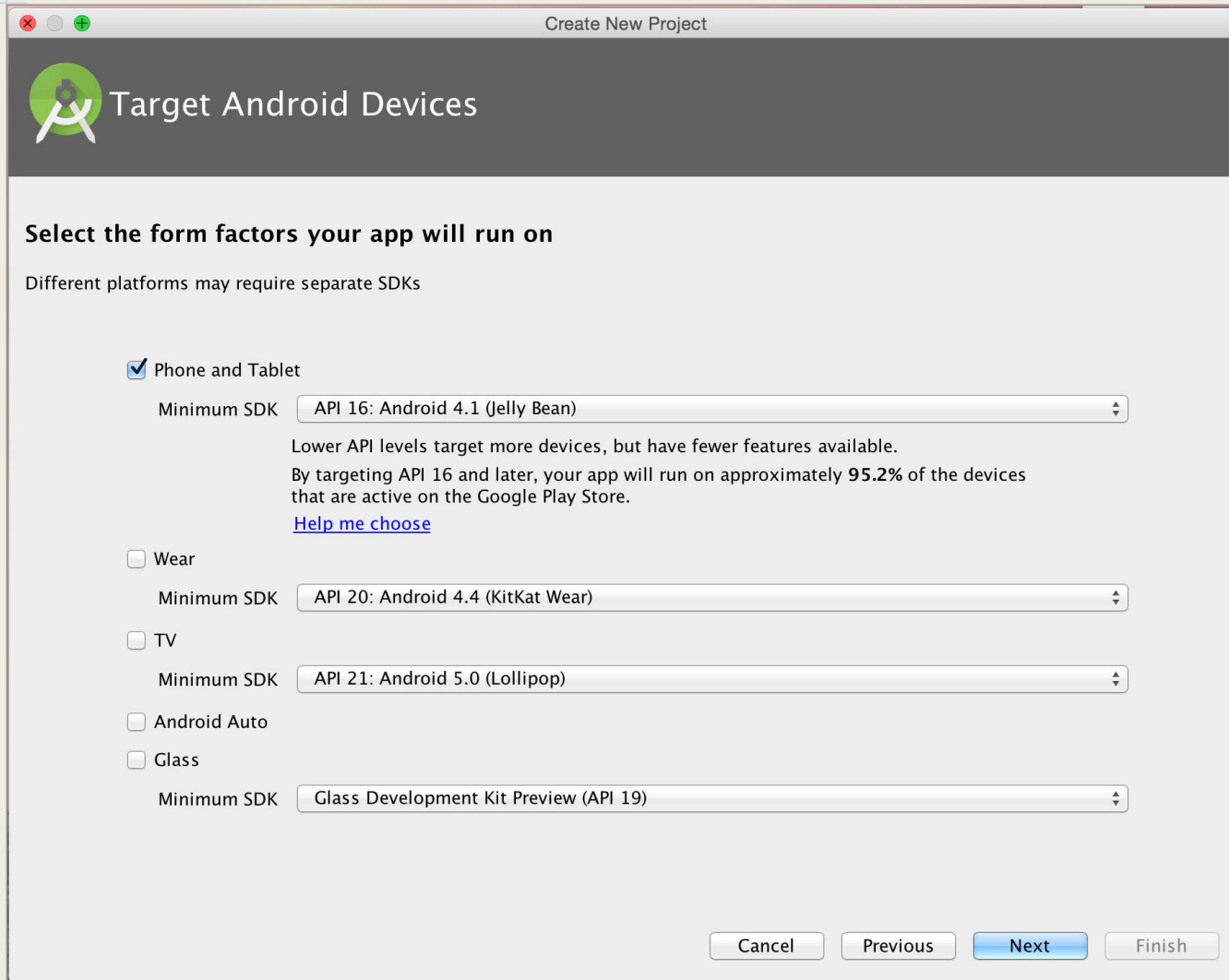
Application name:

Company Domain:

Package name:  [Edit](#)


Project location:

# Select Platform and Version



The screenshot shows the 'Create New Project' dialog box in Android Studio. The title bar reads 'Create New Project'. The main header area has the Android logo and the text 'Target Android Devices'. Below this, the section is titled 'Select the form factors your app will run on'. A subtitle states 'Different platforms may require separate SDKs'. There are five options, each with a checkbox and a 'Minimum SDK' dropdown menu. The 'Phone and Tablet' option is selected. Below it, there is explanatory text about API levels and a link to 'Help me choose'. The other options are 'Wear', 'TV', 'Android Auto', and 'Glass', each with its own 'Minimum SDK' dropdown. At the bottom, there are four buttons: 'Cancel', 'Previous', 'Next', and 'Finish'.

Create New Project

 Target Android Devices

**Select the form factors your app will run on**

Different platforms may require separate SDKs

☒ Phone and Tablet

Minimum SDK

Lower API levels target more devices, but have fewer features available.  
By targeting API 16 and later, your app will run on approximately **95.2%** of the devices that are active on the Google Play Store.  
[Help me choose](#)

☐ Wear

Minimum SDK

☐ TV

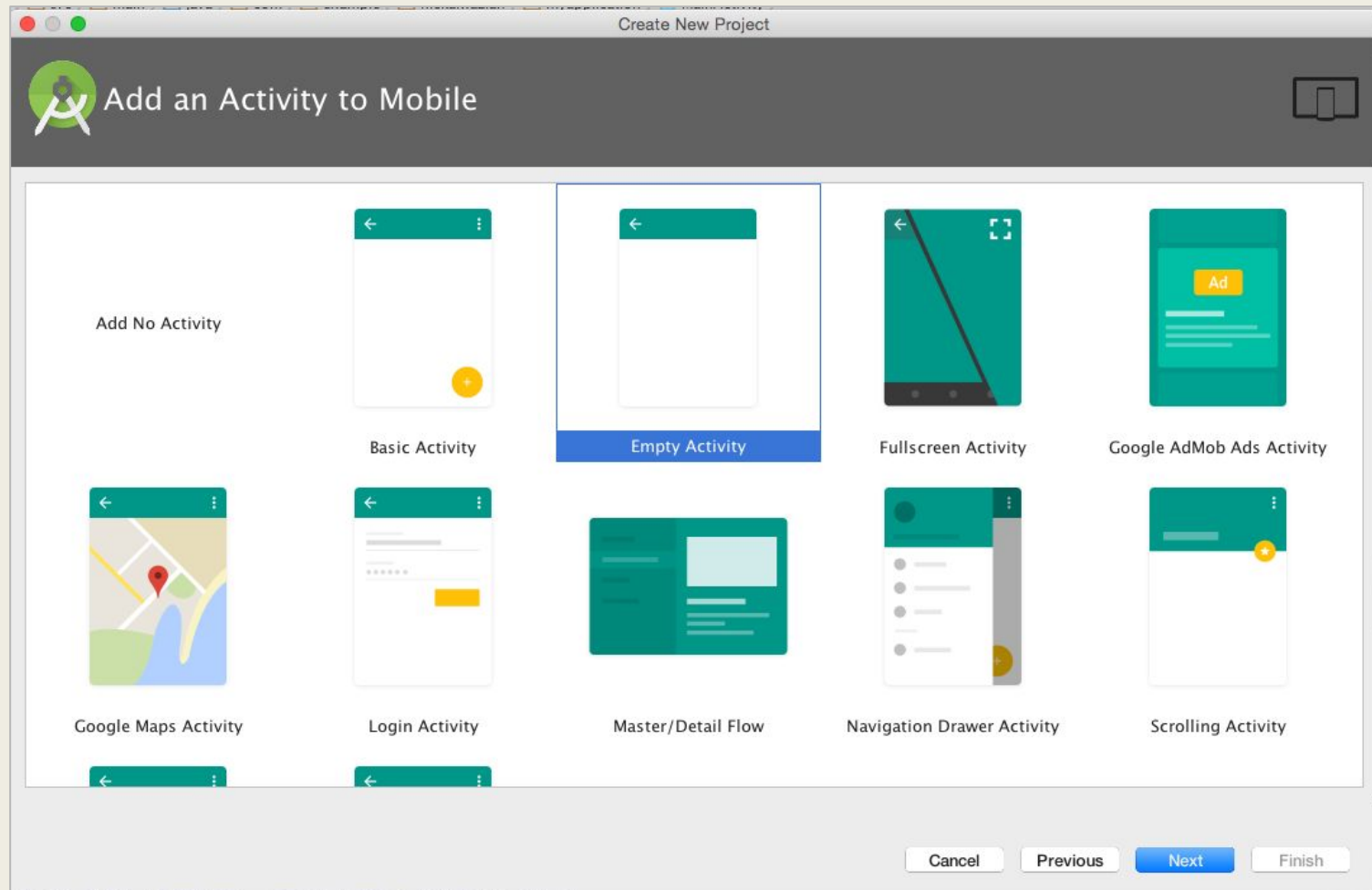
Minimum SDK

☐ Android Auto

☐ Glass



Minimum SDK

# Select “Empty Activity”

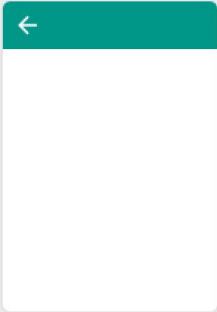


# Configure “MainActivity”

Create New Project

 Customize the Activity 

Creates a new empty activity



Empty Activity

Activity Name:

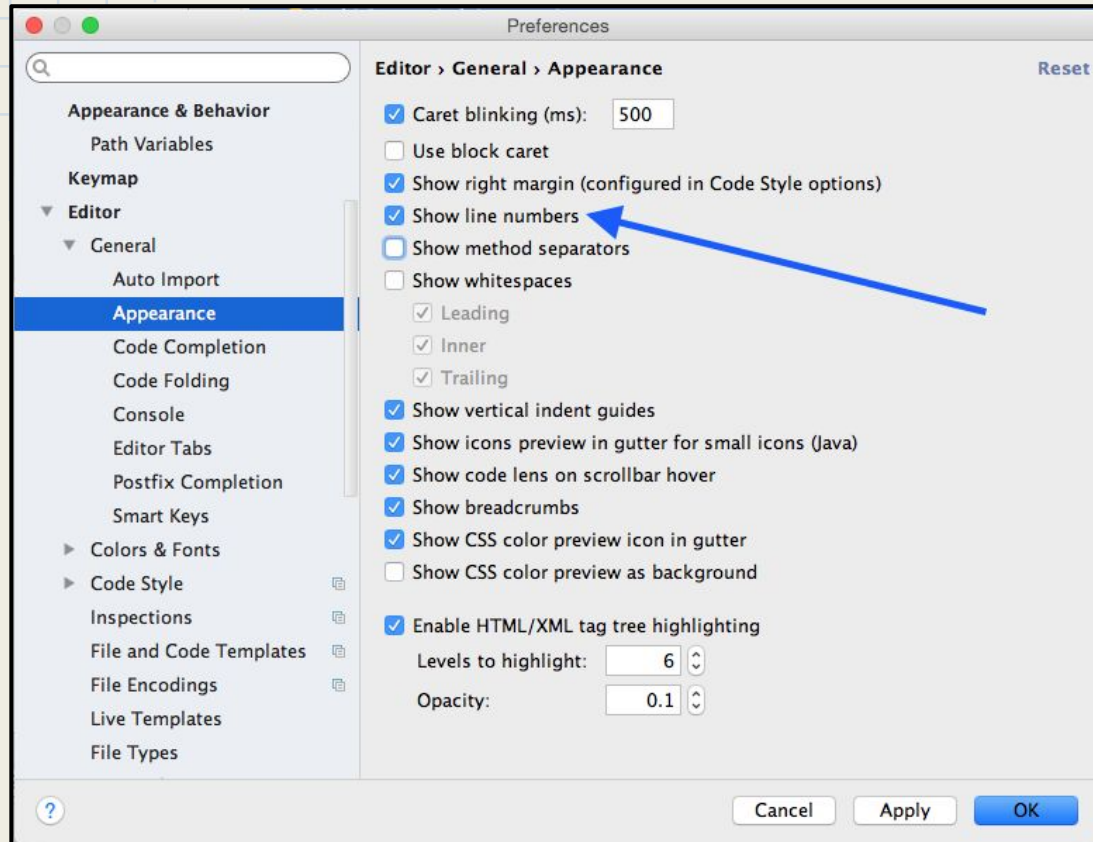
☒ Generate Layout File

Layout Name:

The name of the activity class to create



# Configure Line Numbers



1. Select Android Studio  
→ Preferences in top  
menu

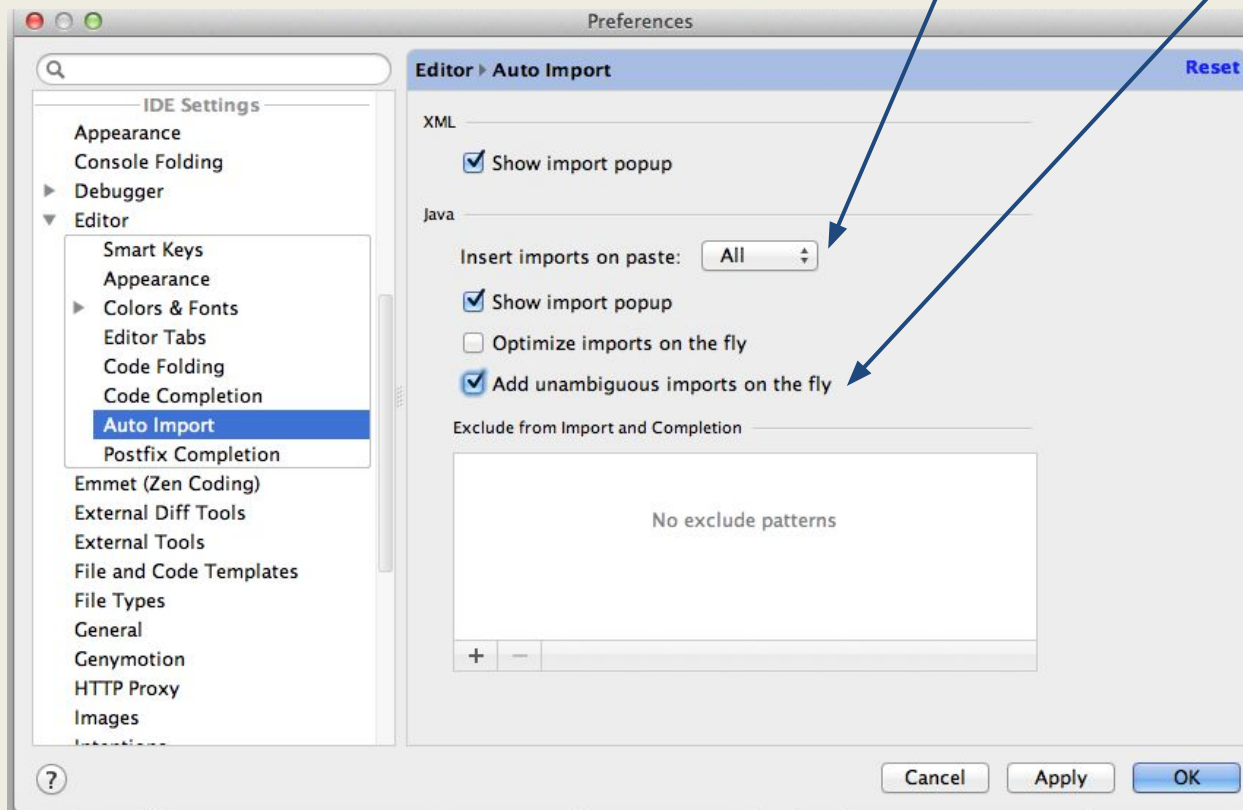
2. Find Editor →  
General → Appearance

3. Enable “**Show line  
numbers**”

# Configure Imports

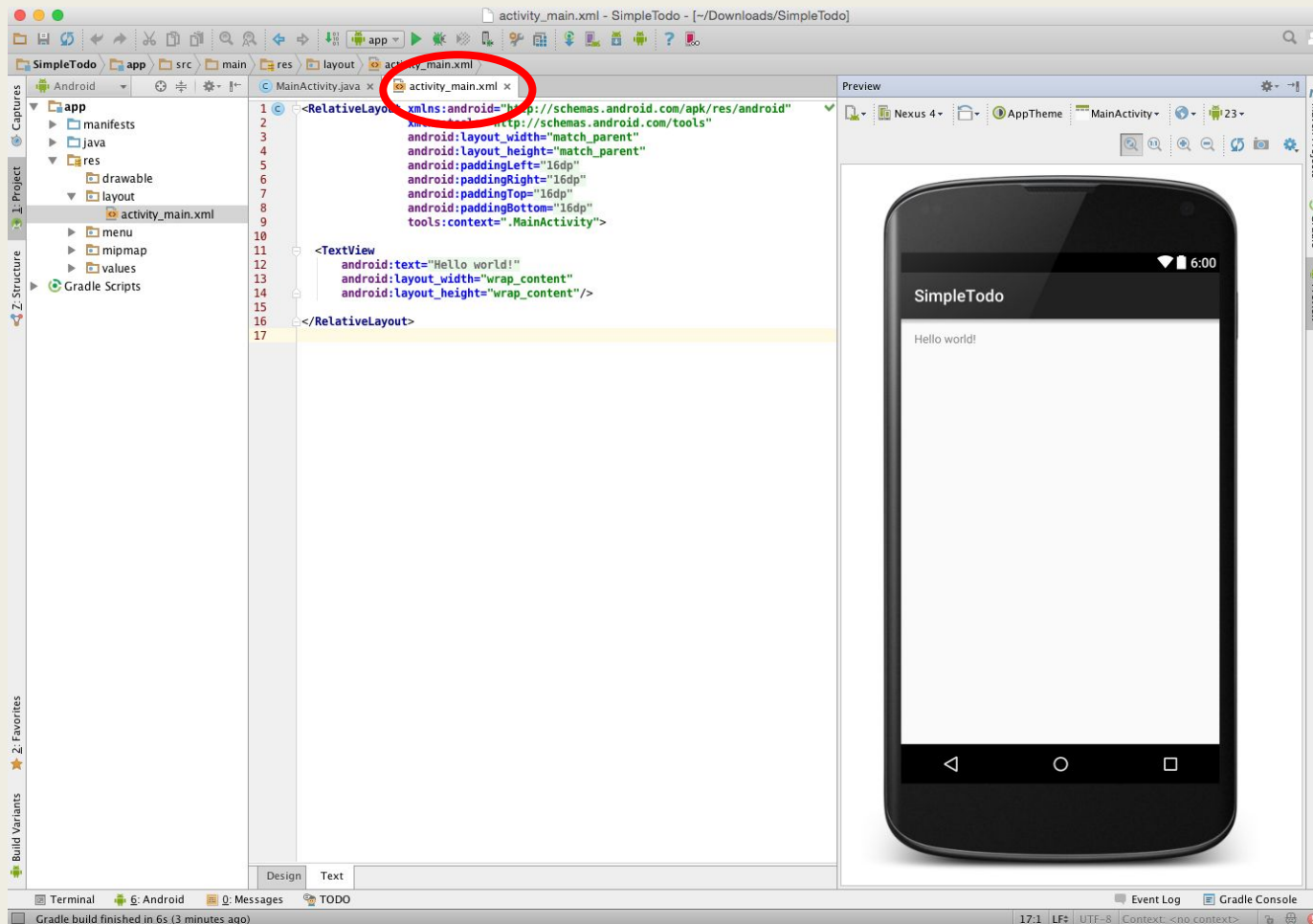
Set “Imports on Paste” to “All”

Check “Add unambiguous imports on the fly”

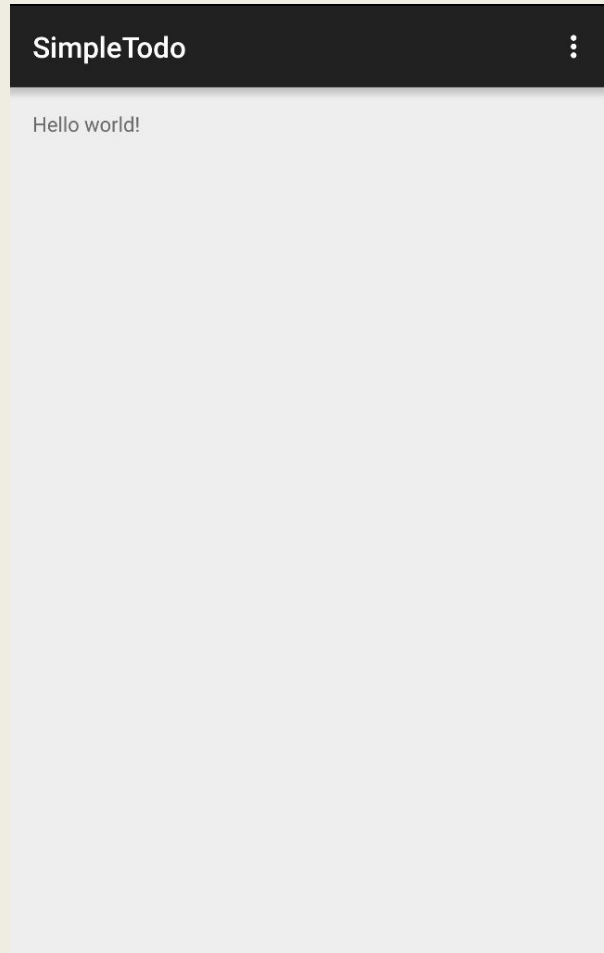
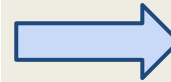
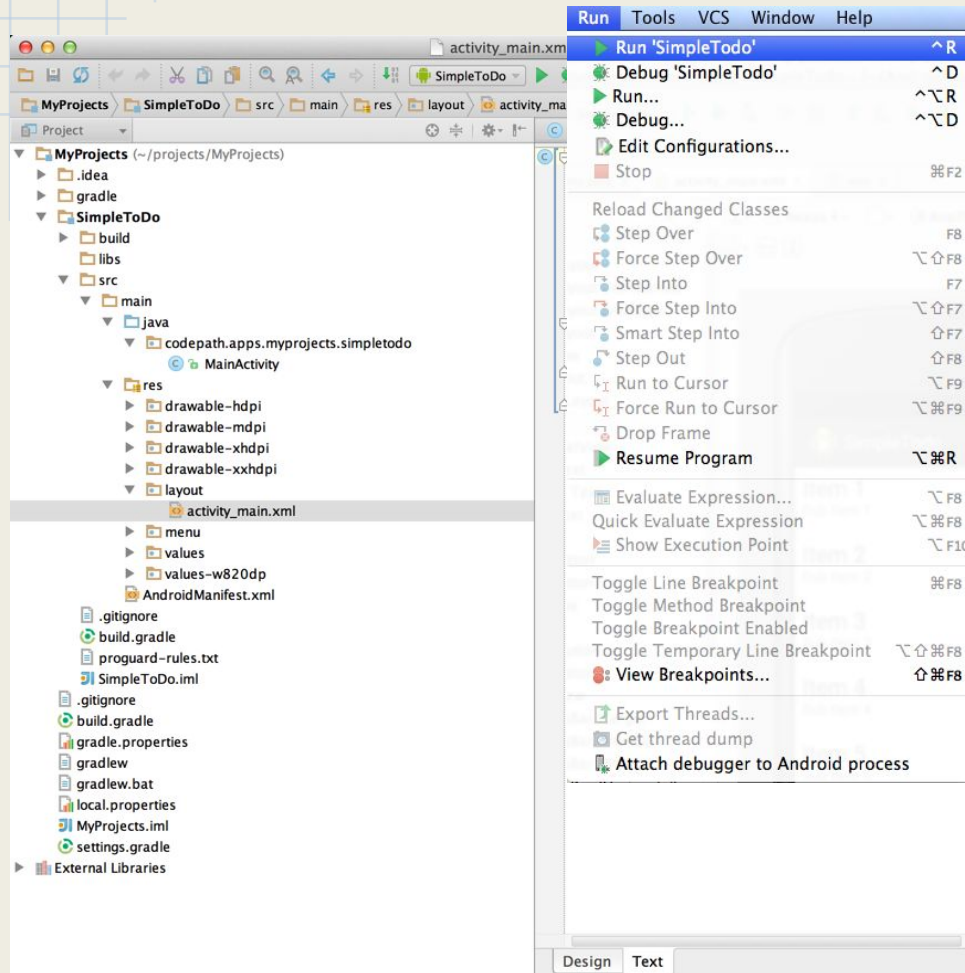


# Open activity\_main.xml

Click on the second tab to reveal the XML file



# Give it a Run on a Device or in the Emulator



# Android Framework Orientation

**Activity**

**==**

**1 UI**

**Screen**

**src/res/  
layout**

**==**

**Look**

**src/main/  
\*.java**

**==**

**Behavior**

# Anatomy of Android App

## Anatomy of an Android App

### ToDo Activity (Screen)

Layout and Look in  
*res/layout/activity\_todo.xml*



Behavior and Logic in  
*src/main/ToDoActivity.java*

Launches

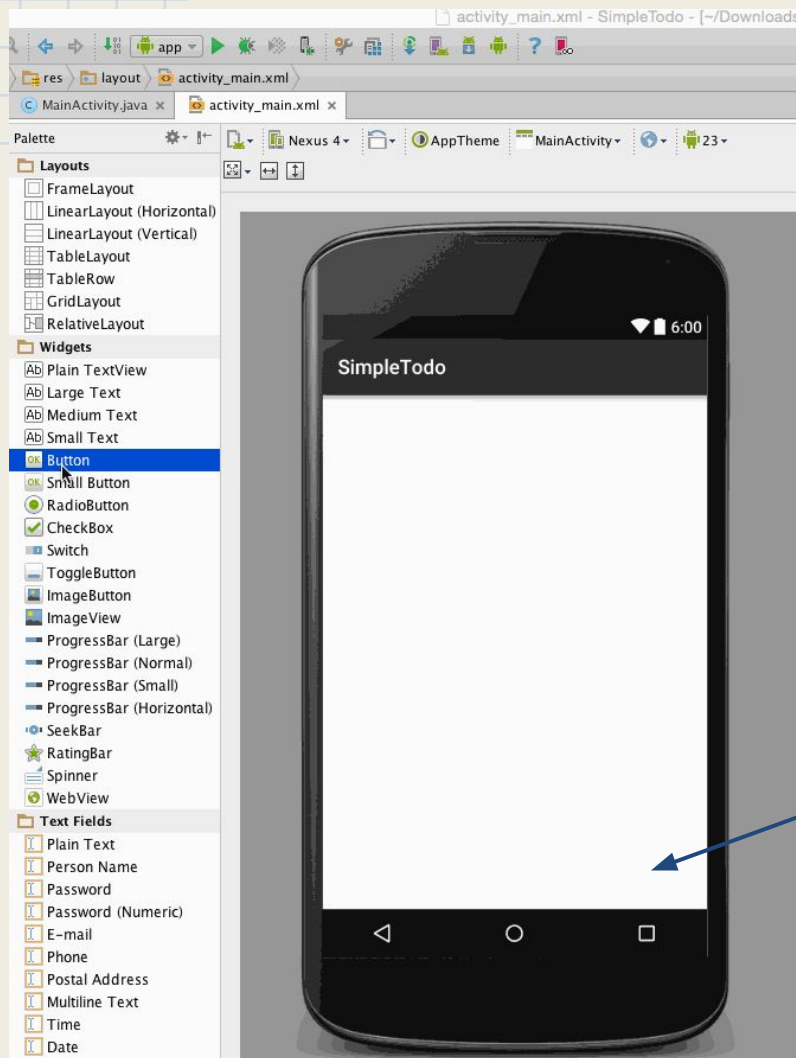
### Settings Activity (Screen)

Layout and Look in *res/layout/activity\_settings.xml*



Behavior and Logic in  
*src/main/SettingsActivity.java*

# Build the Interface, Pt 1

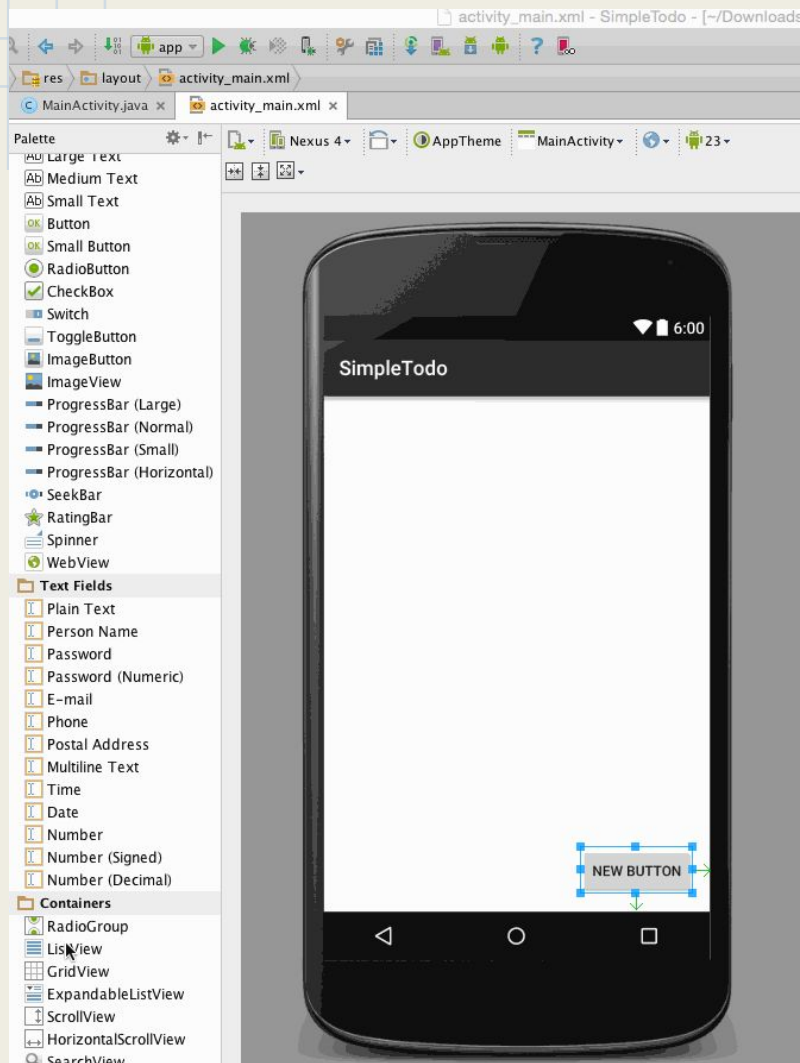


1. Click on “Hello World” label and press the backspace key to remove.

2. Drag a Button from left-hand side (Widgets) to Bottom-Right of Layout

**Note:** When dragging views onto layout, ensure **green lines** confirm location before releasing.

# Build the Interface, Pt. 2



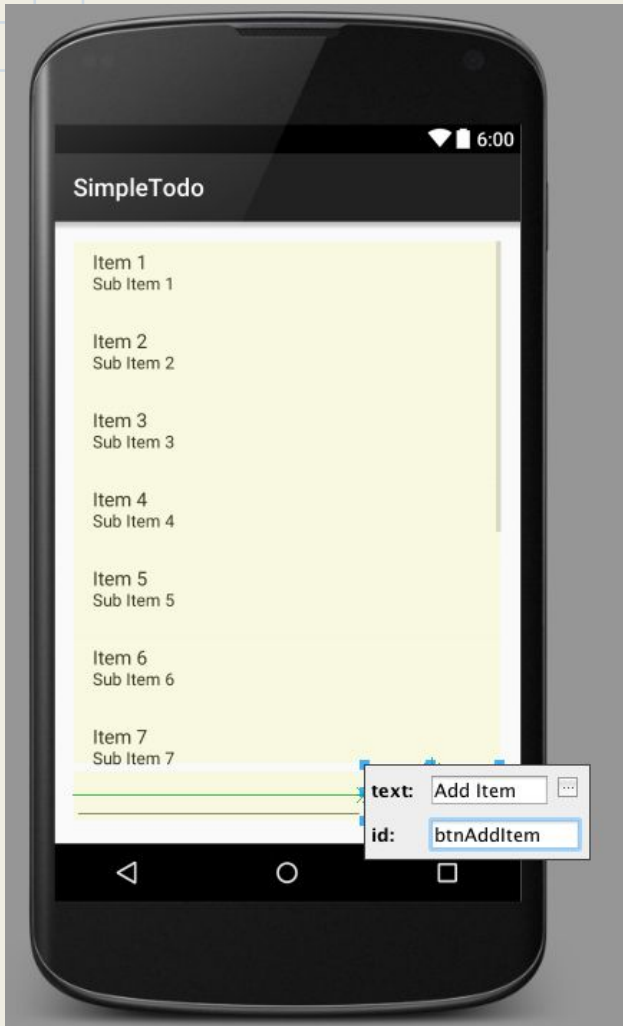
## Drag More Views onto Layout

- 1. Text Fields → **Plain Text**
  - Drag to Bottom-Left
  - Resize Right Side to Button
- 2. Containers → **ListView**
  - Drag to Top Left Corner
  - Resize Bottom Above Button

Note: When dragging new views to layout, ensure **green lines** confirm location before releasing.

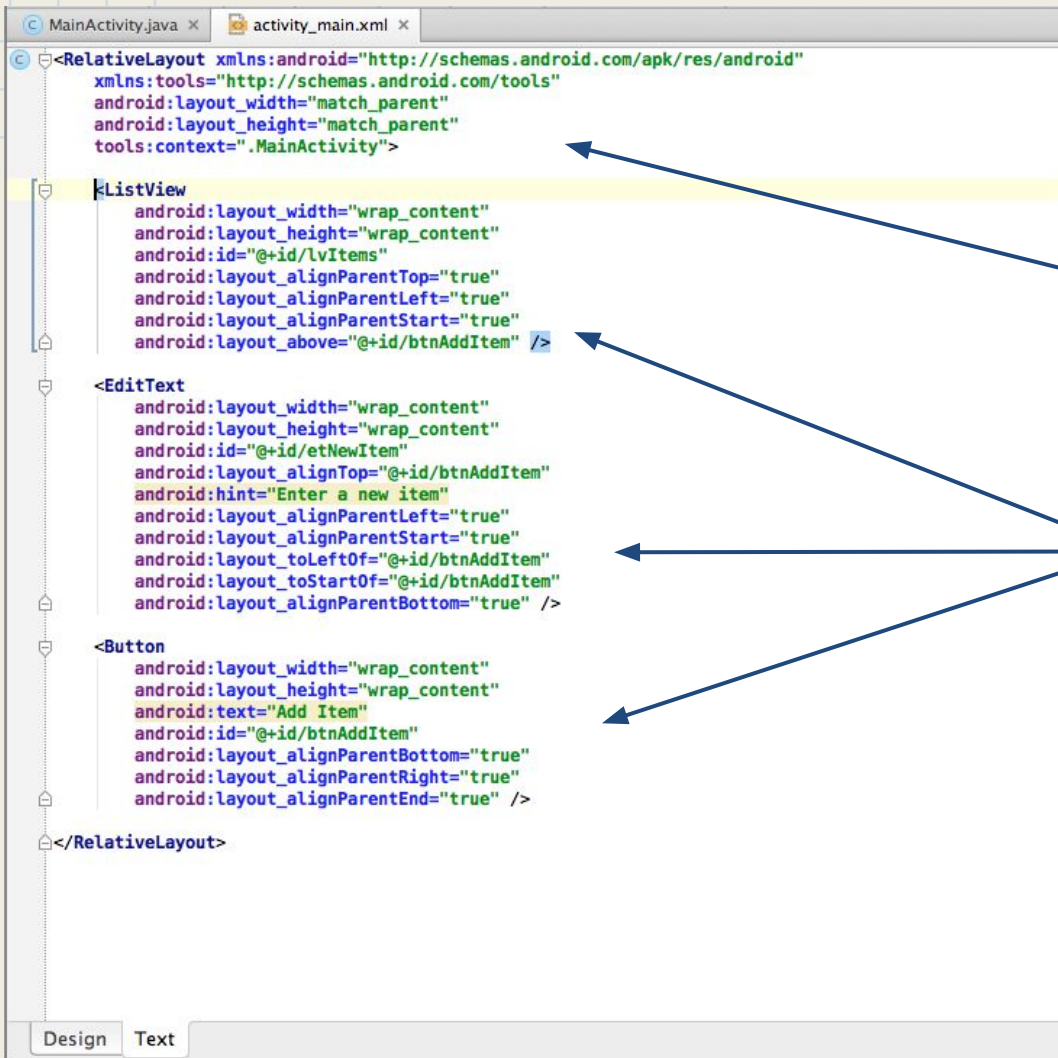


# Build the Interface



- Assign Hint to EditText
- Double-click on view to assign ID
  - ListView = lvItems
  - EditText = etNewItem
  - Button = btnAddItem

# Visual UI Builder Automatically Generates XML



```
MainActivity.java x activity_main.xml x
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <ListView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/lvItems"
        android:layout_alignParentTop="true"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"
        android:layout_above="@+id/btnAddItem" />

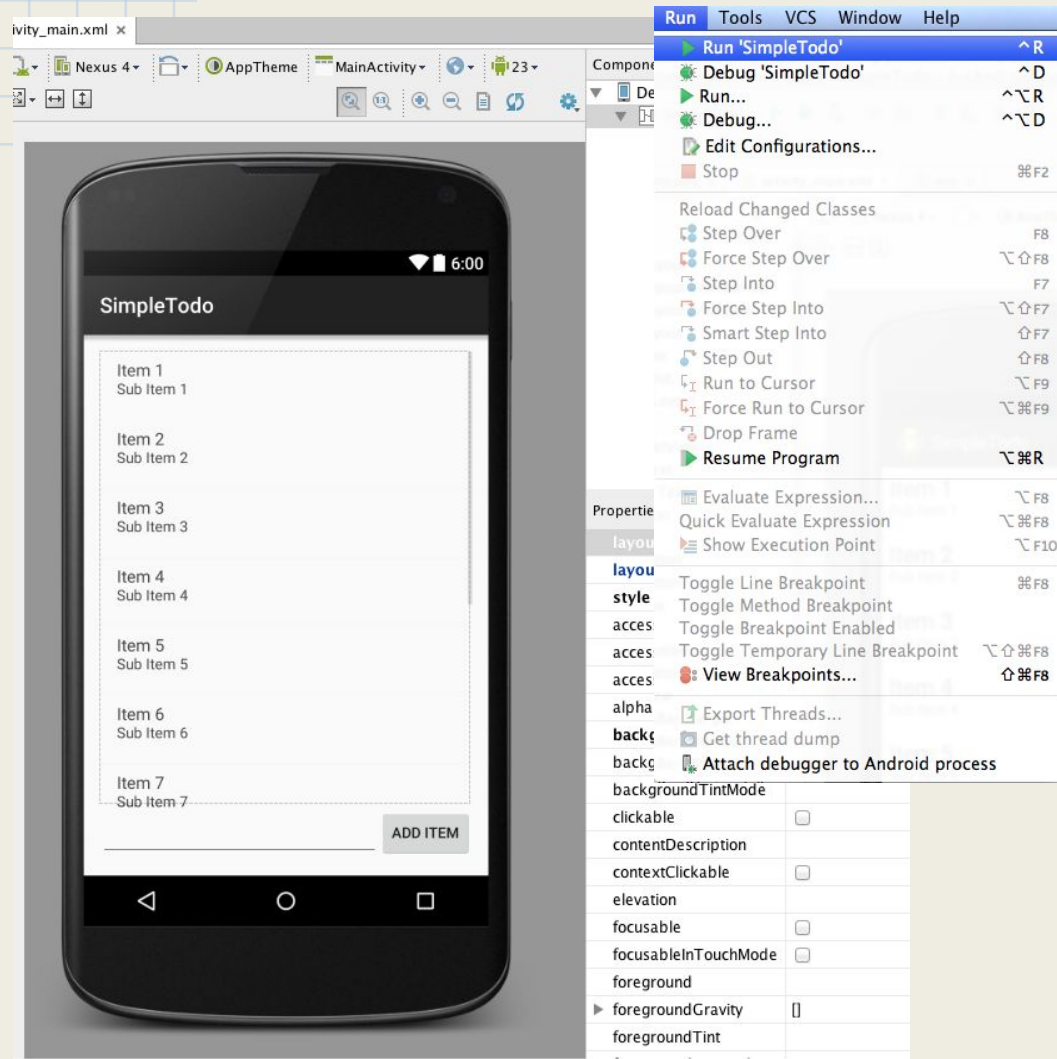
    <EditText
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/etNewItem"
        android:layout_alignTop="@+id/btnAddItem"
        android:hint="Enter a new item"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"
        android:layout_toLeftOf="@+id/btnAddItem"
        android:layout_toStartOf="@+id/btnAddItem"
        android:layout_alignParentBottom="true" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Add Item"
        android:id="@+id/btnAddItem"
        android:layout_alignParentBottom="true"
        android:layout_alignParentRight="true"
        android:layout_alignParentEnd="true" />

</RelativeLayout>
```

- Every action was translated into XML
- All three views are wrapped in a "Layout"
- All three views have listed properties (id, height, width, etc.)

# Give it a Run on a Device or in the Emulator

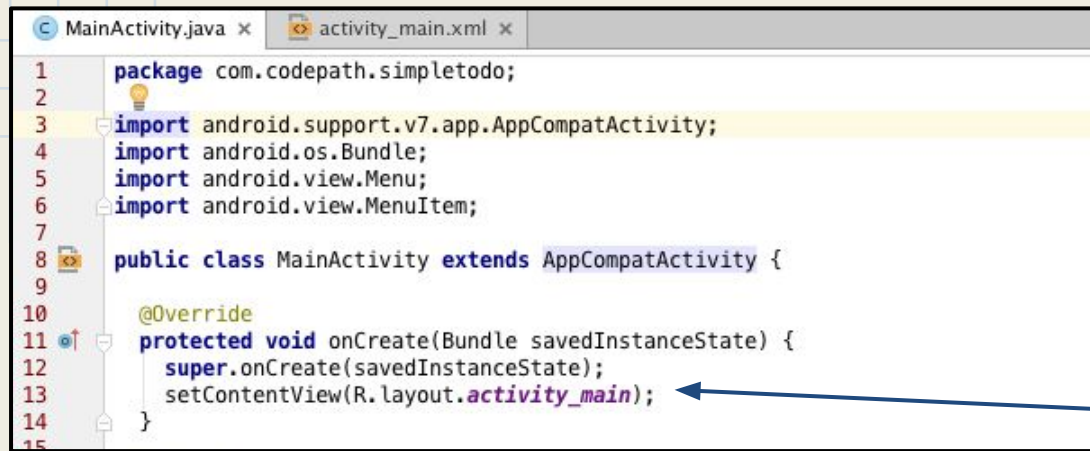


SimpleToDo

Enter a new item

ADD ITEM

# Code Basic Behavior



```
1 package com.codepath.simpletodo;
2
3 import android.support.v7.app.AppCompatActivity;
4 import android.os.Bundle;
5 import android.view.Menu;
6 import android.view.MenuItem;
7
8 public class MainActivity extends AppCompatActivity {
9
10     @Override
11     protected void onCreate(Bundle savedInstanceState) {
12         super.onCreate(savedInstanceState);
13         setContentView(R.layout.activity_main);
14     }
15 }
```

- The XML layout for this activity is applied in `onCreate()`
- Every Activity extends from the same base class
- This is where we add our application logic

# Create List of Items and Display in ListView

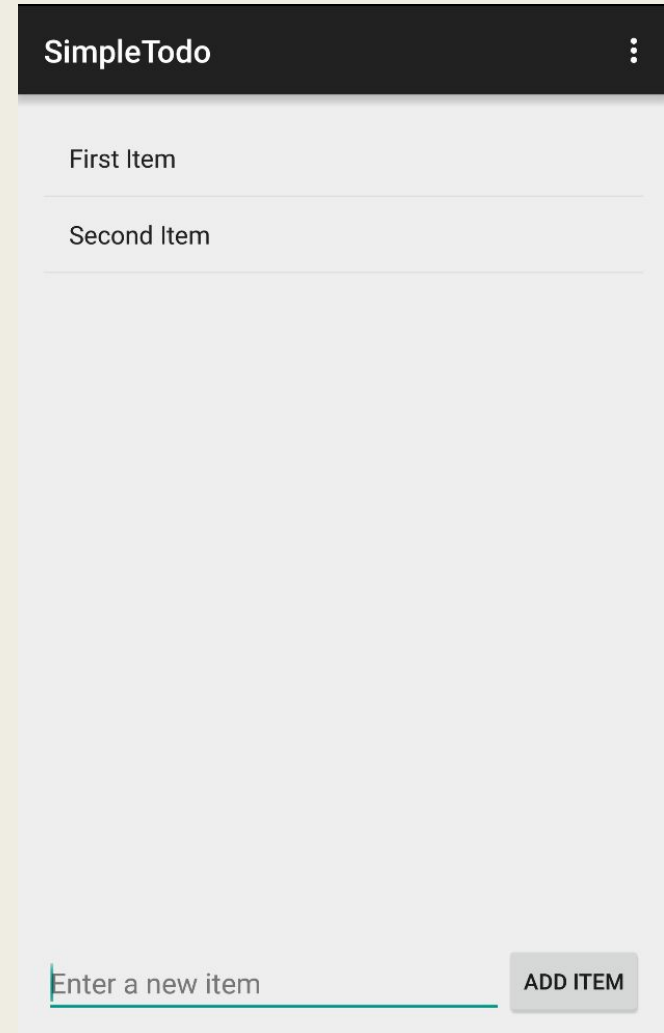
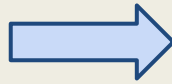
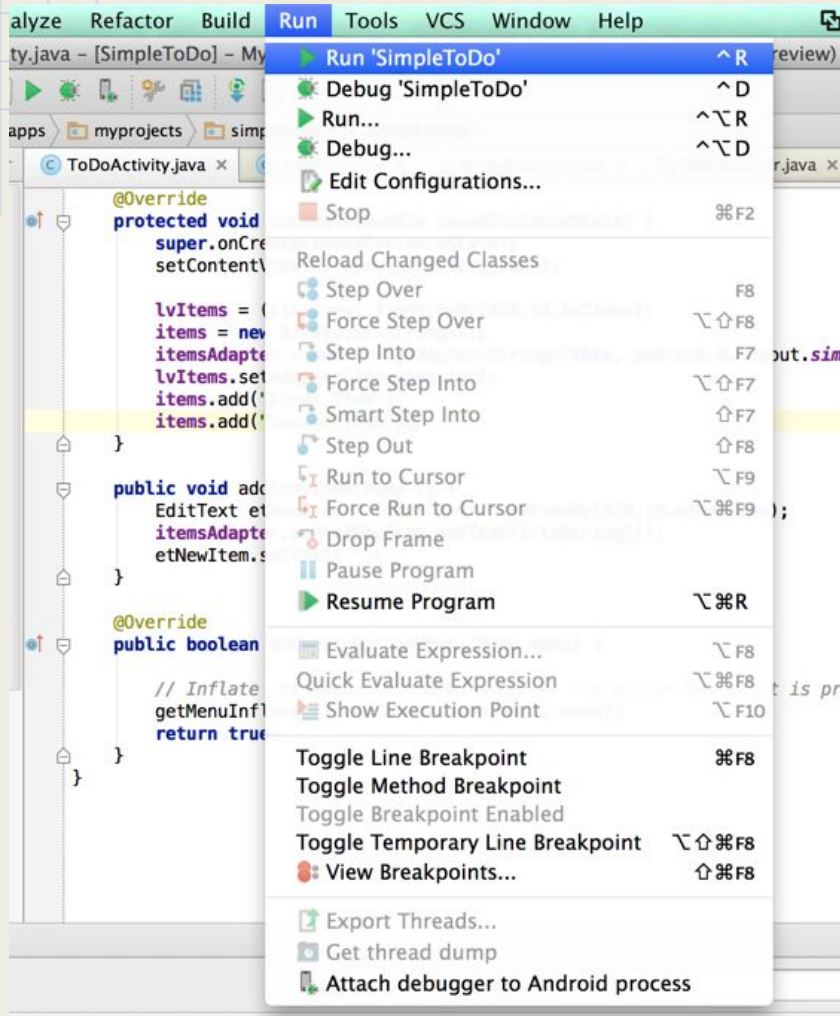
```
public class MainActivity extends AppCompatActivity {
    ArrayList<String> items;
    ArrayAdapter<String> itemsAdapter;
    ListView lvItems;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        lvItems = (ListView) findViewById(R.id.lvItems);
        items = new ArrayList<>();
        itemsAdapter = new ArrayAdapter<>(this,
            android.R.layout.simple_list_item_1, items);
        lvItems.setAdapter(itemsAdapter);
        items.add("First Item");
        items.add("Second Item");
    }
}
```

1. Create an ArrayList
2. Create an ArrayAdapter
3. Get a handle to ListView
4. Attach adapter to ListView

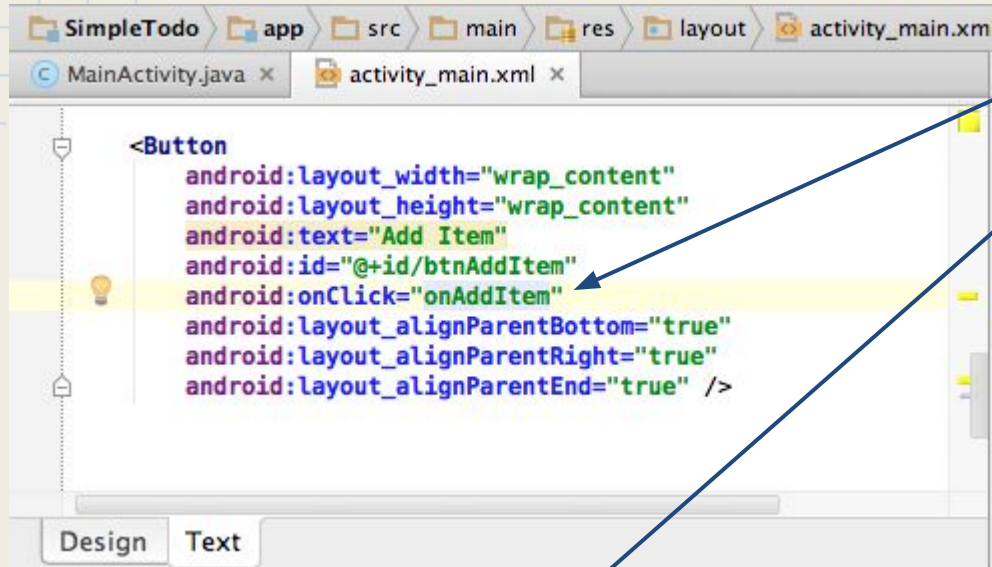
An **adapter** allows us to easily display the contents of an ArrayList within a ListView.

# Give it a Run on a Device or in the Emulator



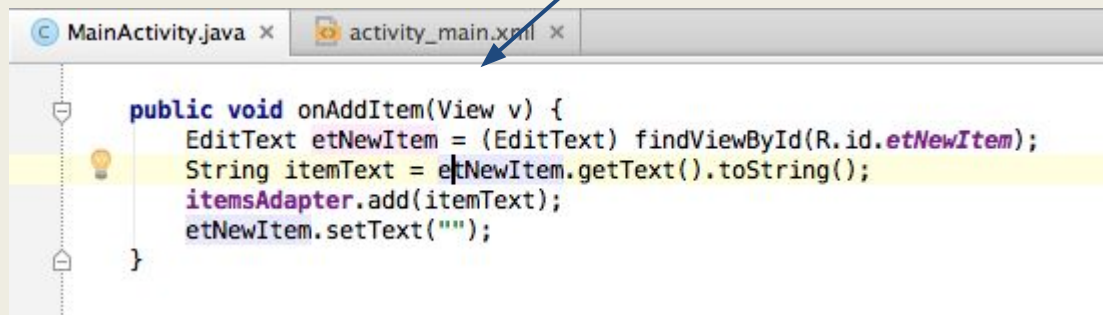


# Add Items to the List



```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Add Item"
    android:id="@+id/btnAddItem"
    android:onClick="onAddItem"
    android:layout_alignParentBottom="true"
    android:layout_alignParentRight="true"
    android:layout_alignParentEnd="true" />
```

1. Add "onClick" property to Button in XML
2. Create onAddItem() method to MainActivity.java which adds input item to the list



```
public void onAddItem(View v) {
    EditText etNewItem = (EditText) findViewById(R.id.etNewItem);
    String itemText = etNewItem.getText().toString();
    itemsAdapter.add(itemText);
    etNewItem.setText("");
}
```

# Remove Items from List

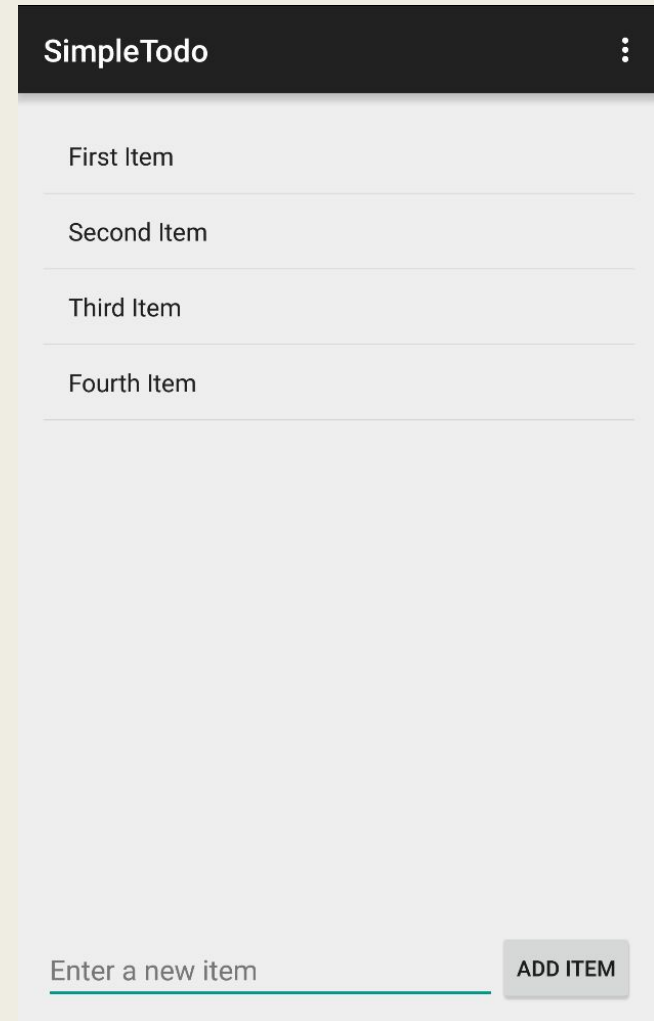
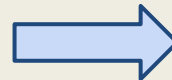
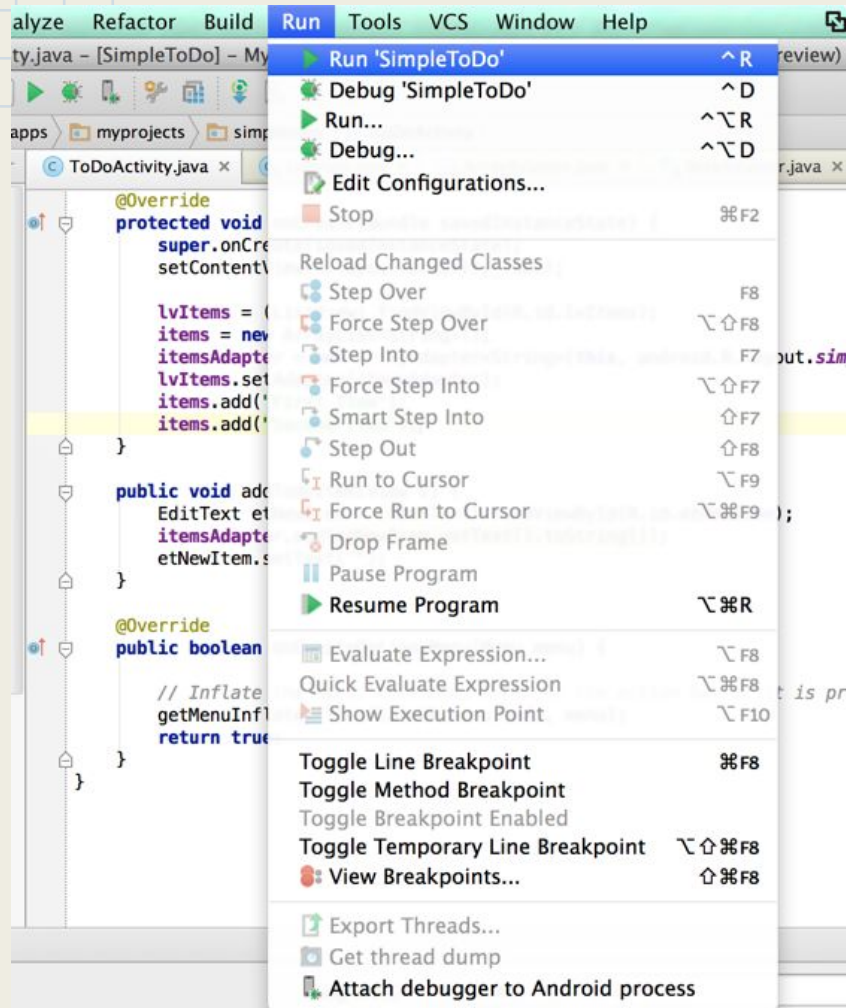
```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    lvItems = (ListView) findViewById(R.id.lvItems);
    items = new ArrayList<String>();
    itemsAdapter = new ArrayAdapter<String>(this,
        android.R.layout.simple_list_item_1, items);
    lvItems.setAdapter(itemsAdapter);
    items.add("First Item");
    items.add("Second Item");
    setupListViewListener();
}

private void setupListViewListener() {
    lvItems.setOnItemLongClickListener(
        new AdapterView.OnItemClickListener() {
            @Override
            public boolean onItemLongClick(AdapterView<?> adapter,
                View item, int pos, long id) {
                items.remove(pos);
                itemsAdapter.notifyDataSetChanged();
                return true;
            }
        }
    );
}
```

1. Create method for setting up the listener
2. Invoke listener from onCreate()
3. Attach a LongClickListener to each Item for ListView that:
  - a. Removes that item
  - b. Refreshes the adapter



# Run the Final Todo App



# ToDo App Summary

Congratulations! You have built your very first functioning application using many essential concepts:

- Activity XML (Layouts and Views)
- Activity Source (Java Code for App Logic)
- View IDs and Properties
- ListViews, EditText and Button View Types
- List Adapters for Displaying List Items
- Click Handling for Buttons and List Items
- Testing Applications with the Emulator

What you would add to the ToDo List next?