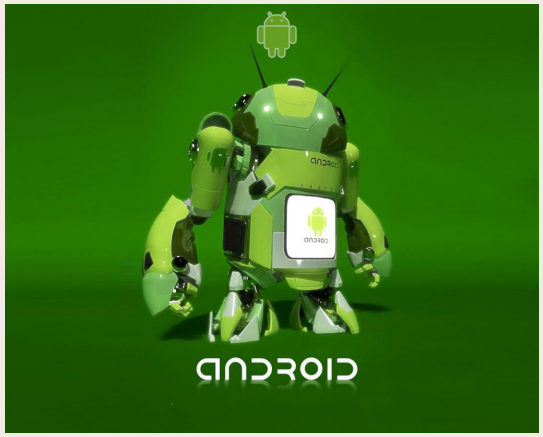
ACM Android Tutorial

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



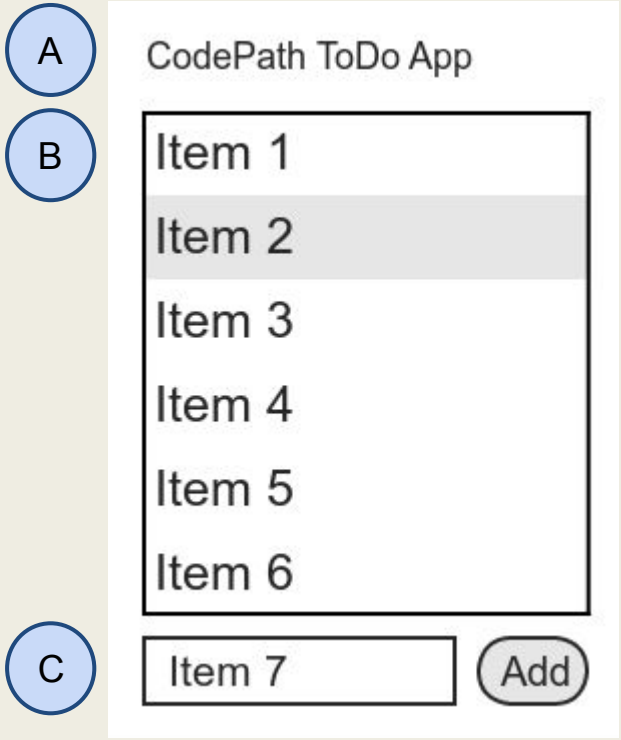
Scoping the Todo App

- **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:



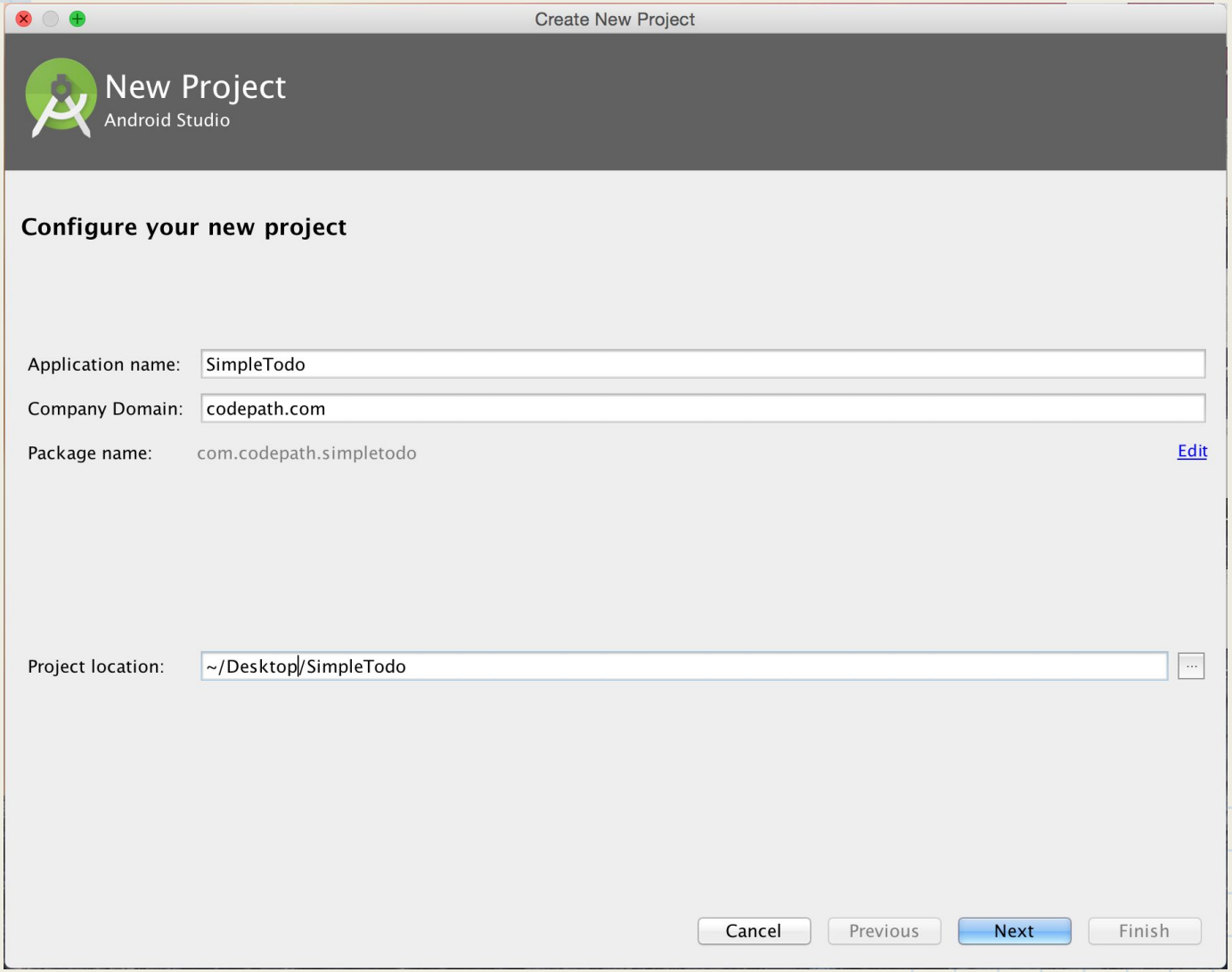
(A) Basic Label with App Name

(B) Basic List of Items

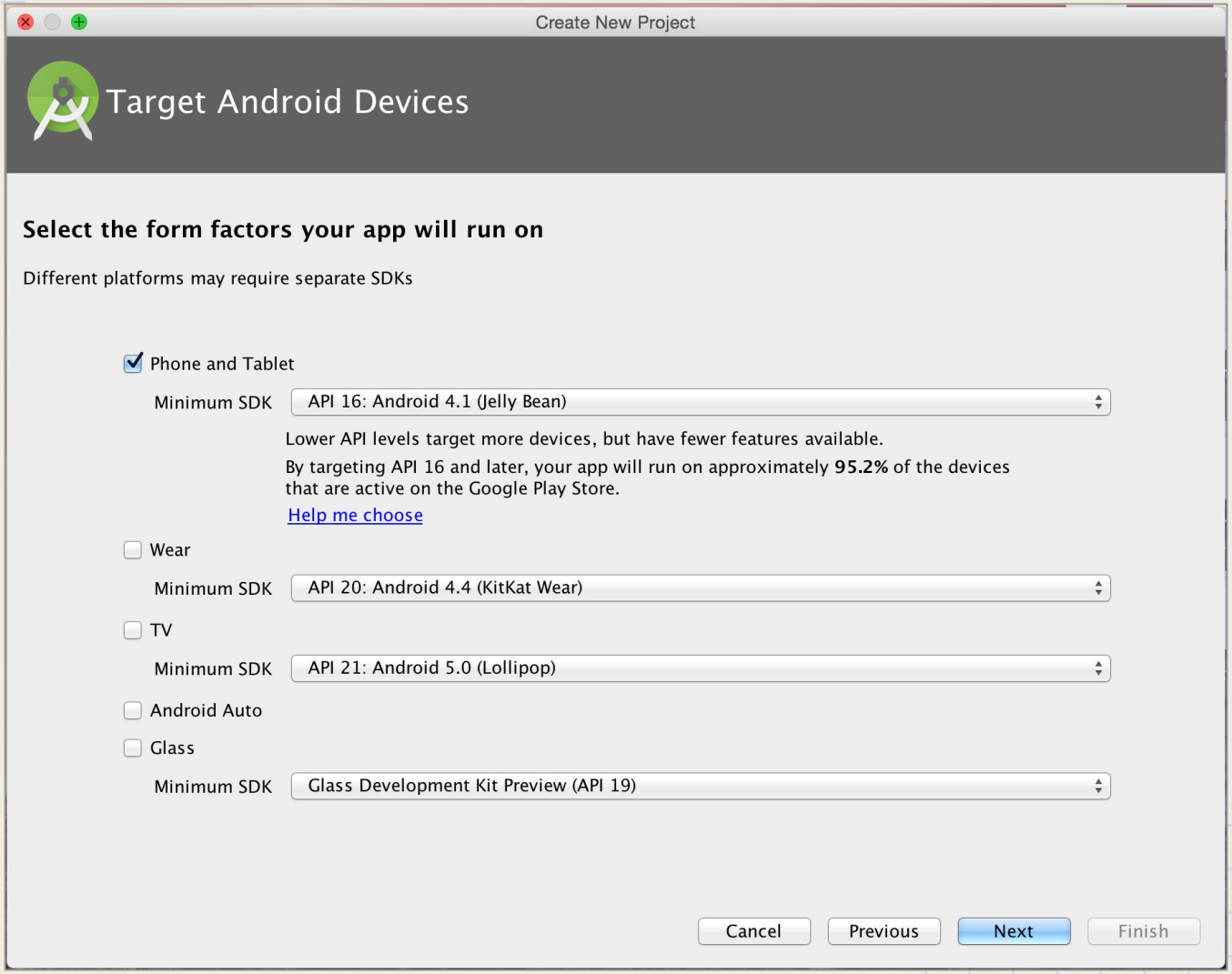
- Vertically Scrollable
- Hold Down to Remove Item

(C) Adding items with Textbox and Button

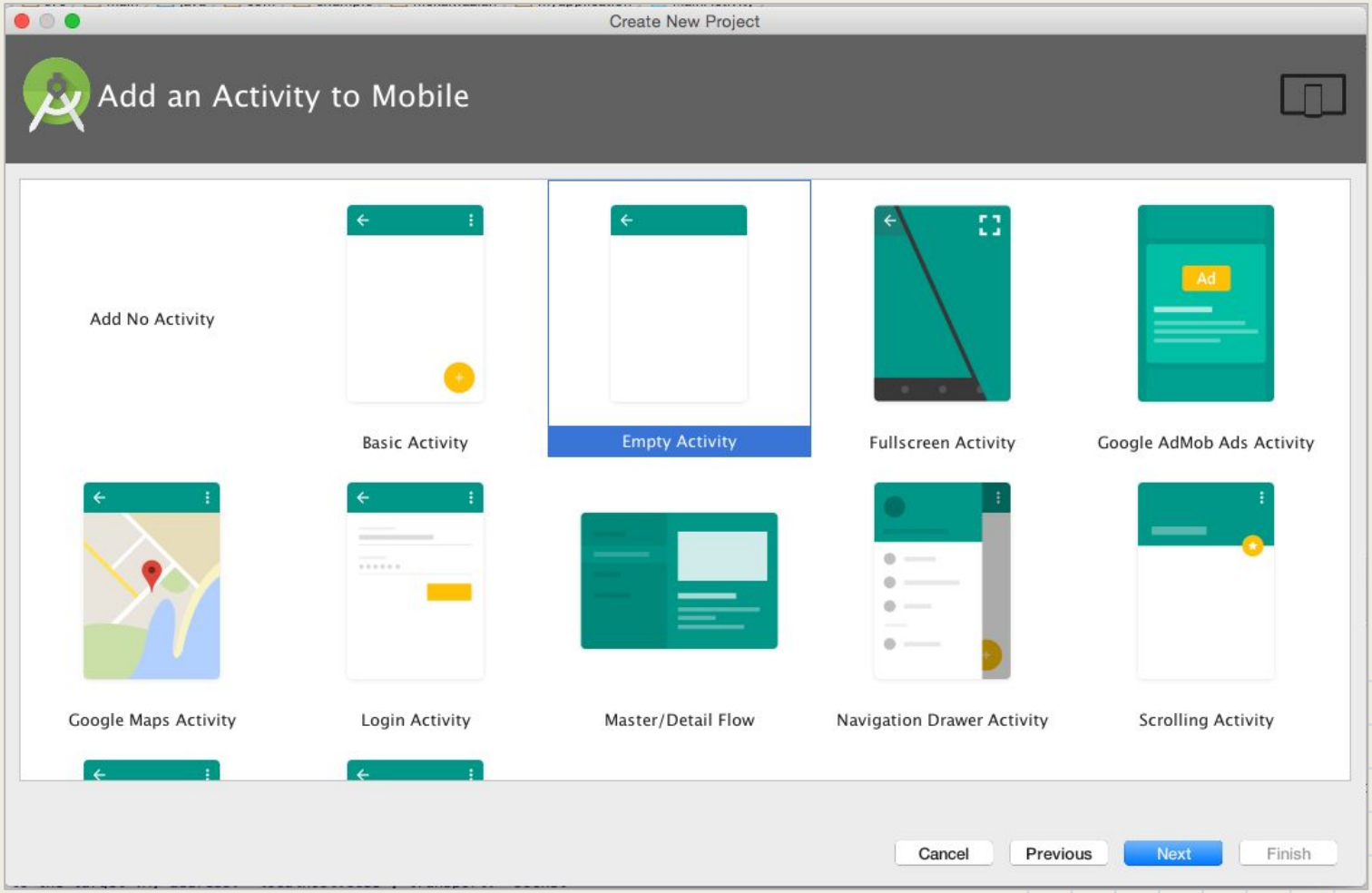
Create New Android Project



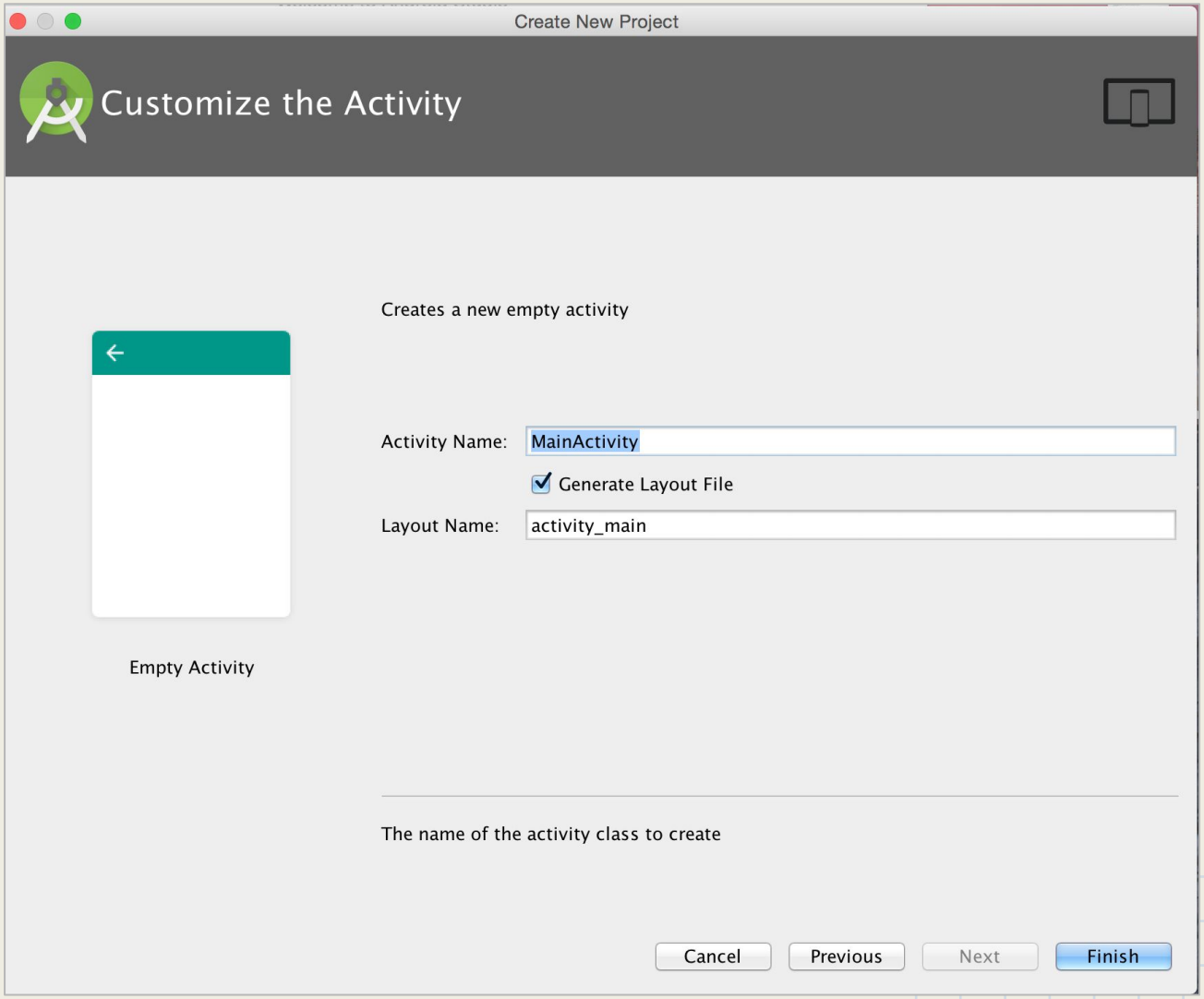
Select Platform and Version



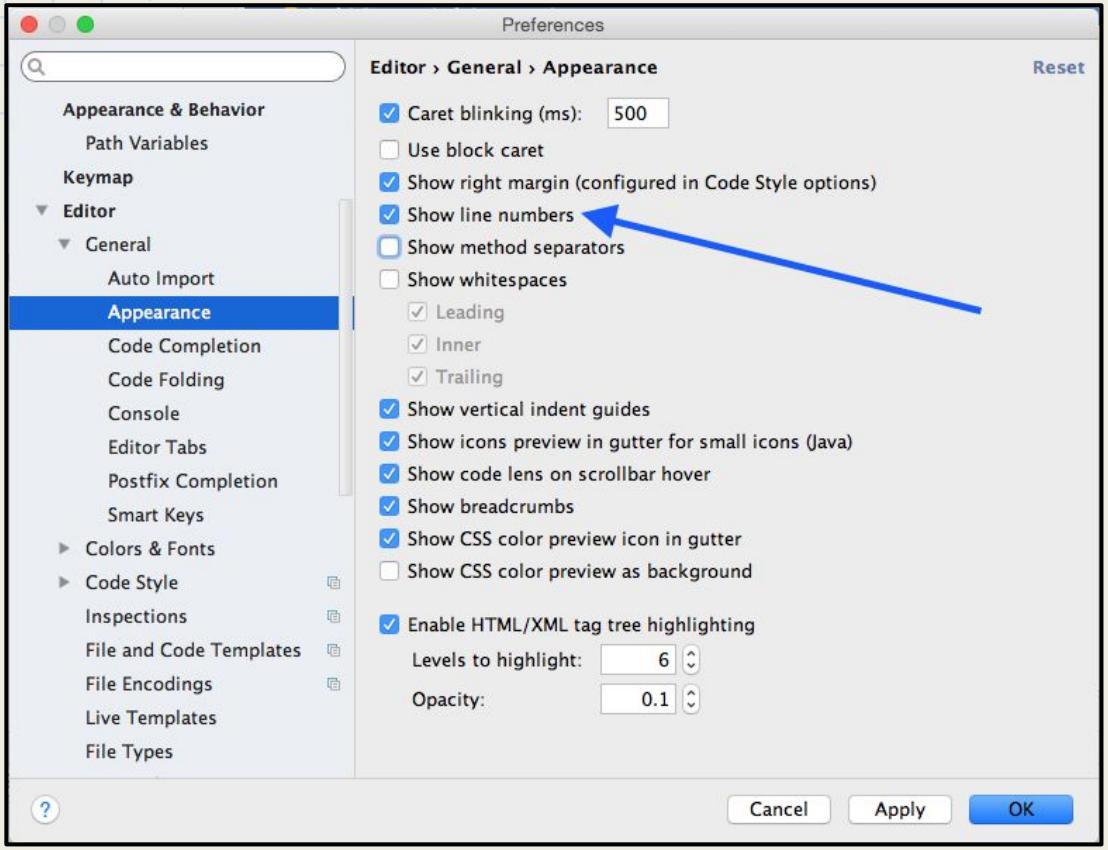
Select “Empty Activity”



Configure “MainActivity”



Configure Line Numbers



1. Select Android Studio → Preferences in top menu

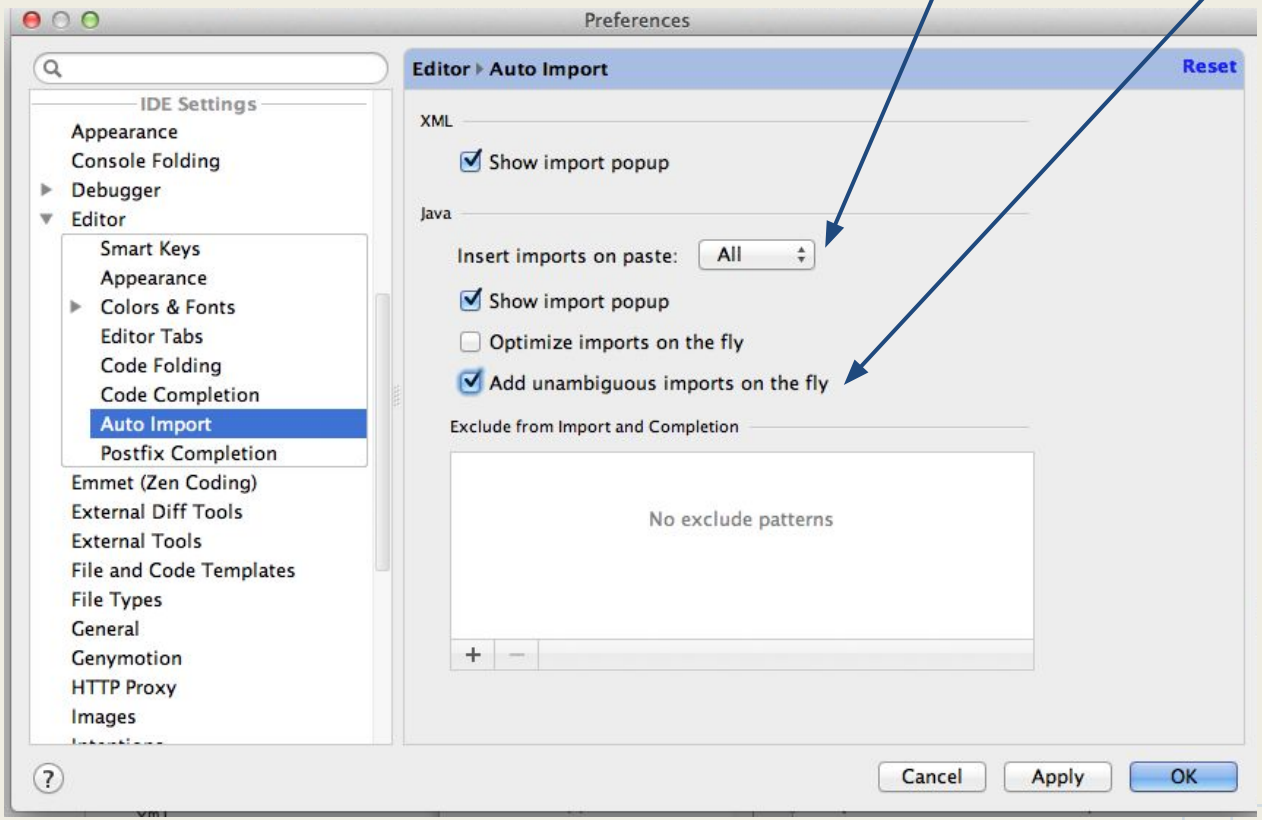
2. Find Editor → General → Appearance

3. Enable “**Show line numbers**”

Configure Imports

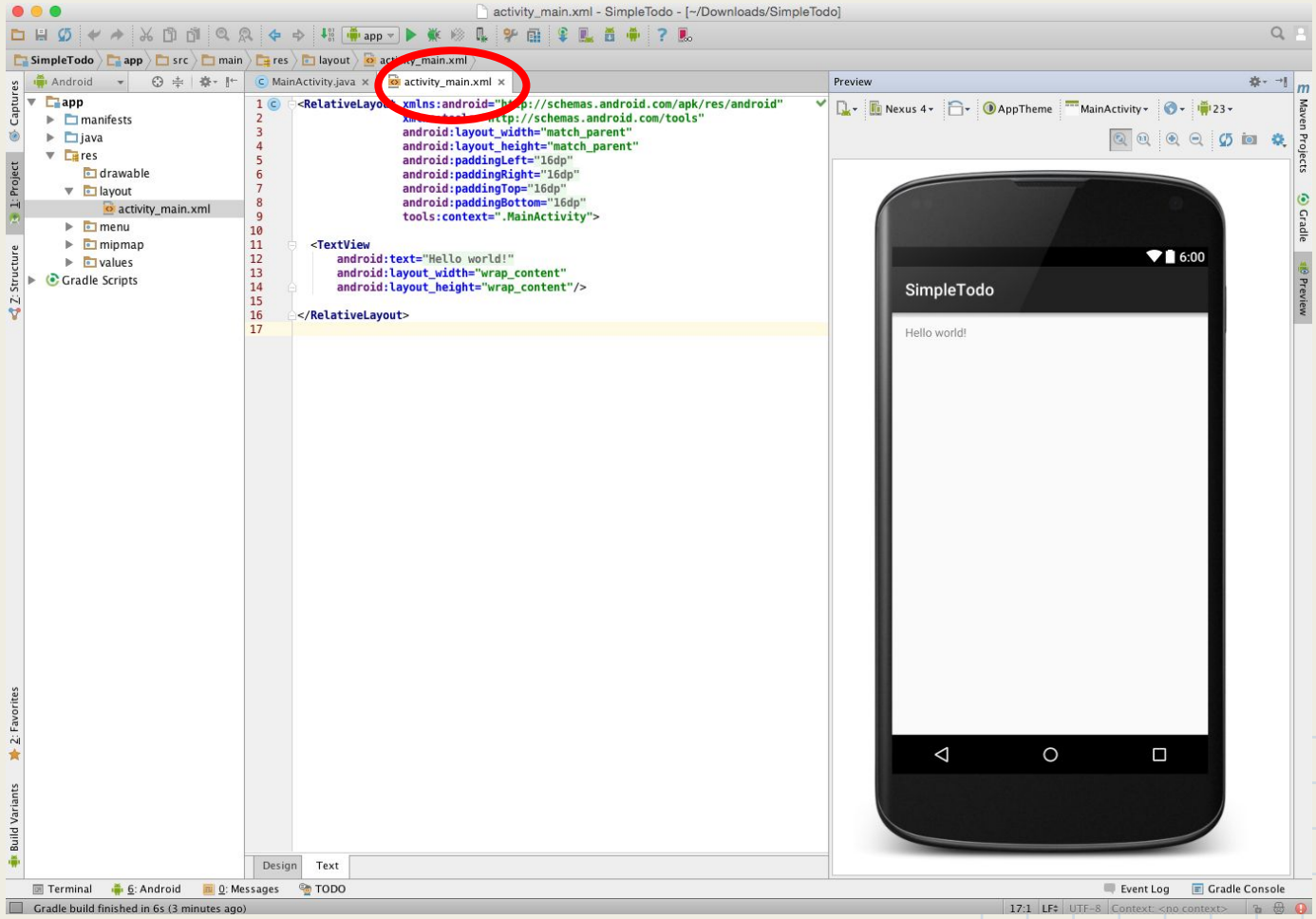
Set "**Imports on Paste**" to "**All**"

Check "**Add unambiguous imports on the fly**"

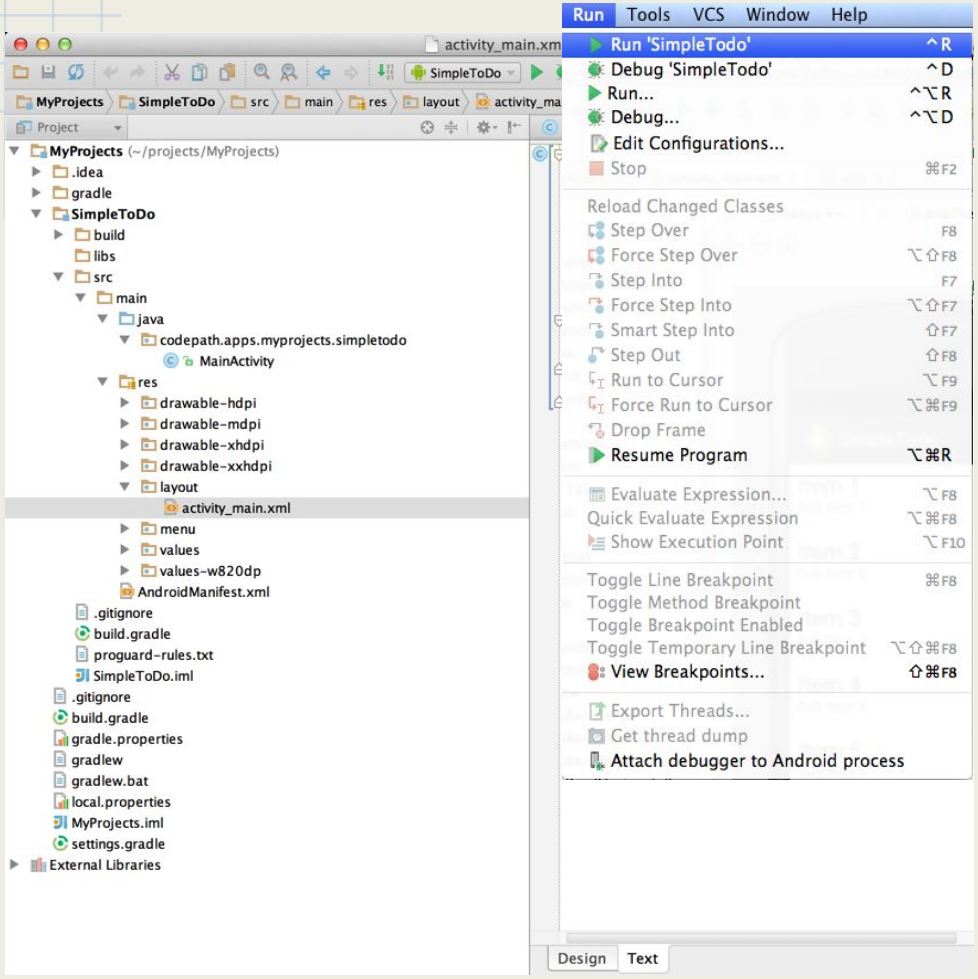


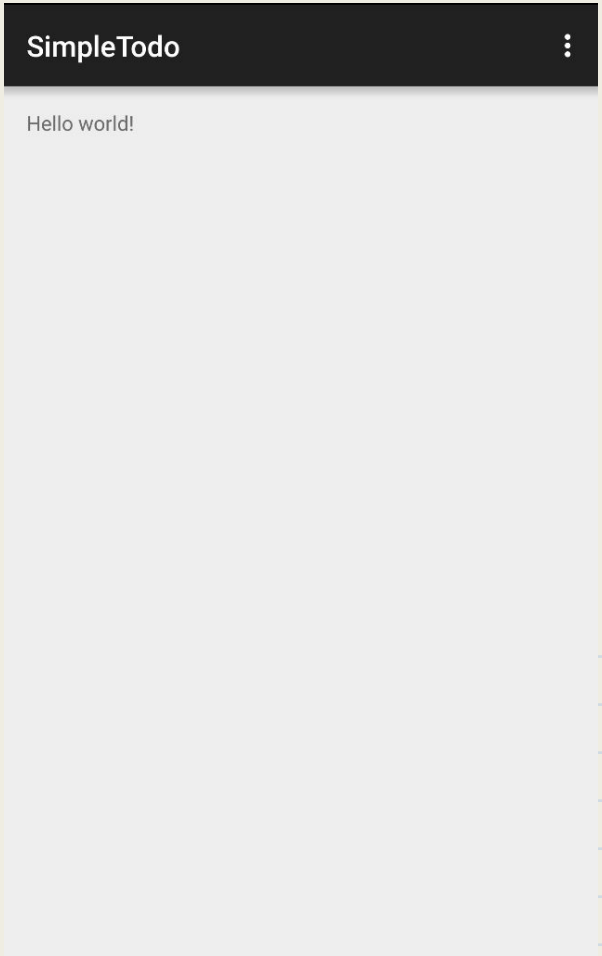
Open activity_main.xml

Click on the second tab to reveal the XML file



Give it a Run on a Device or in the Emulator

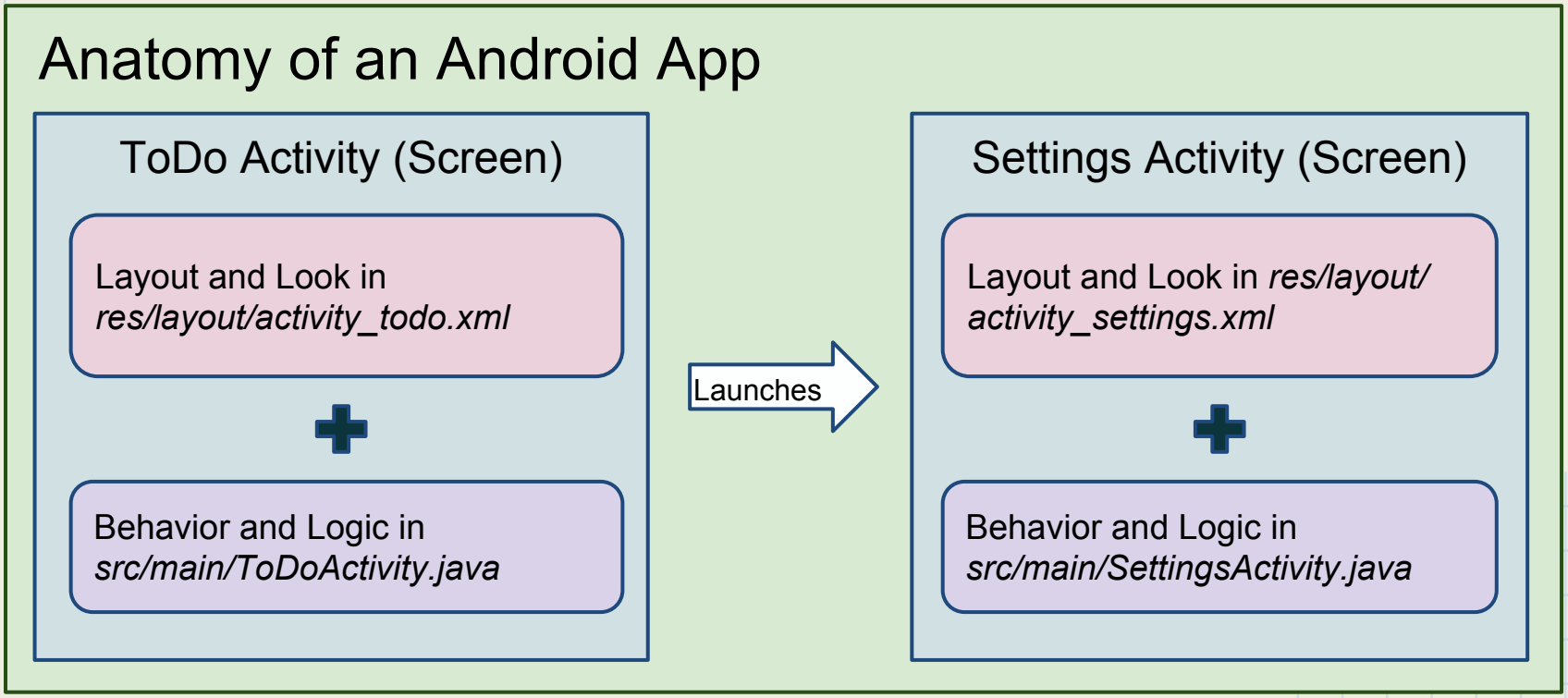




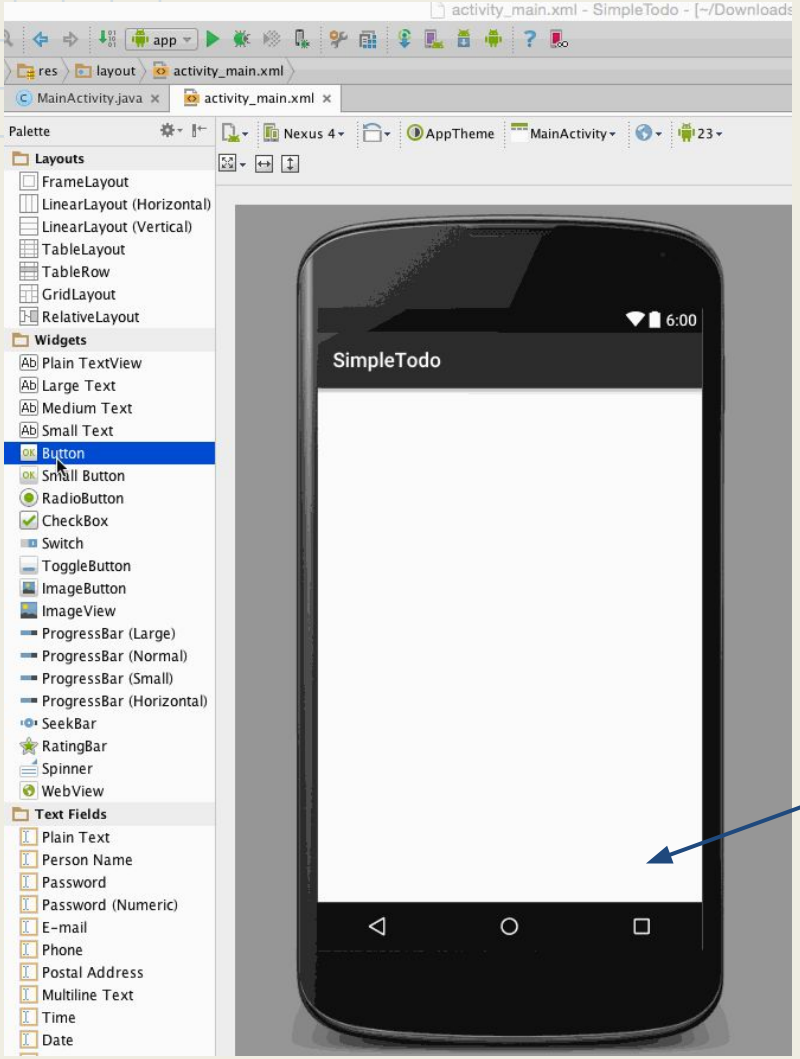
Android Framework Orientation

Activity	**src/res/**	**src/main/**
==	**layout**	***.java**
1 UI	**==**	**==**
Screen	Look	Behavior

Anatomy of Android App



Build the Interface, Pt 1

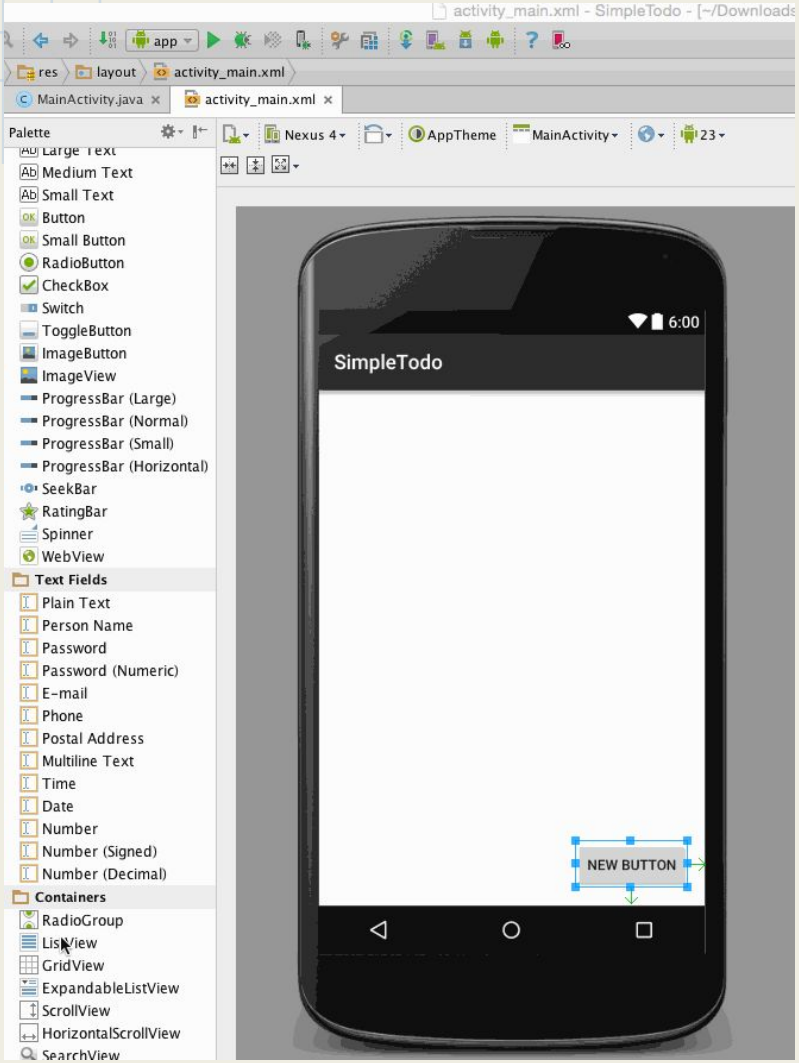


1. Click on “Hello World” label and press the backspace key to remove.

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

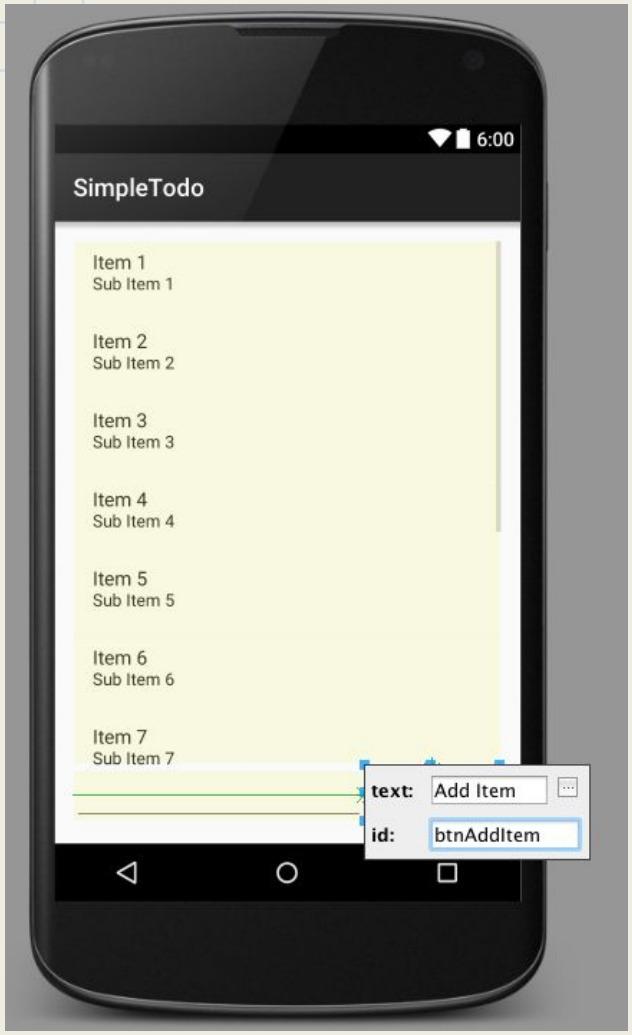


Drag More Views onto Layout

- 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

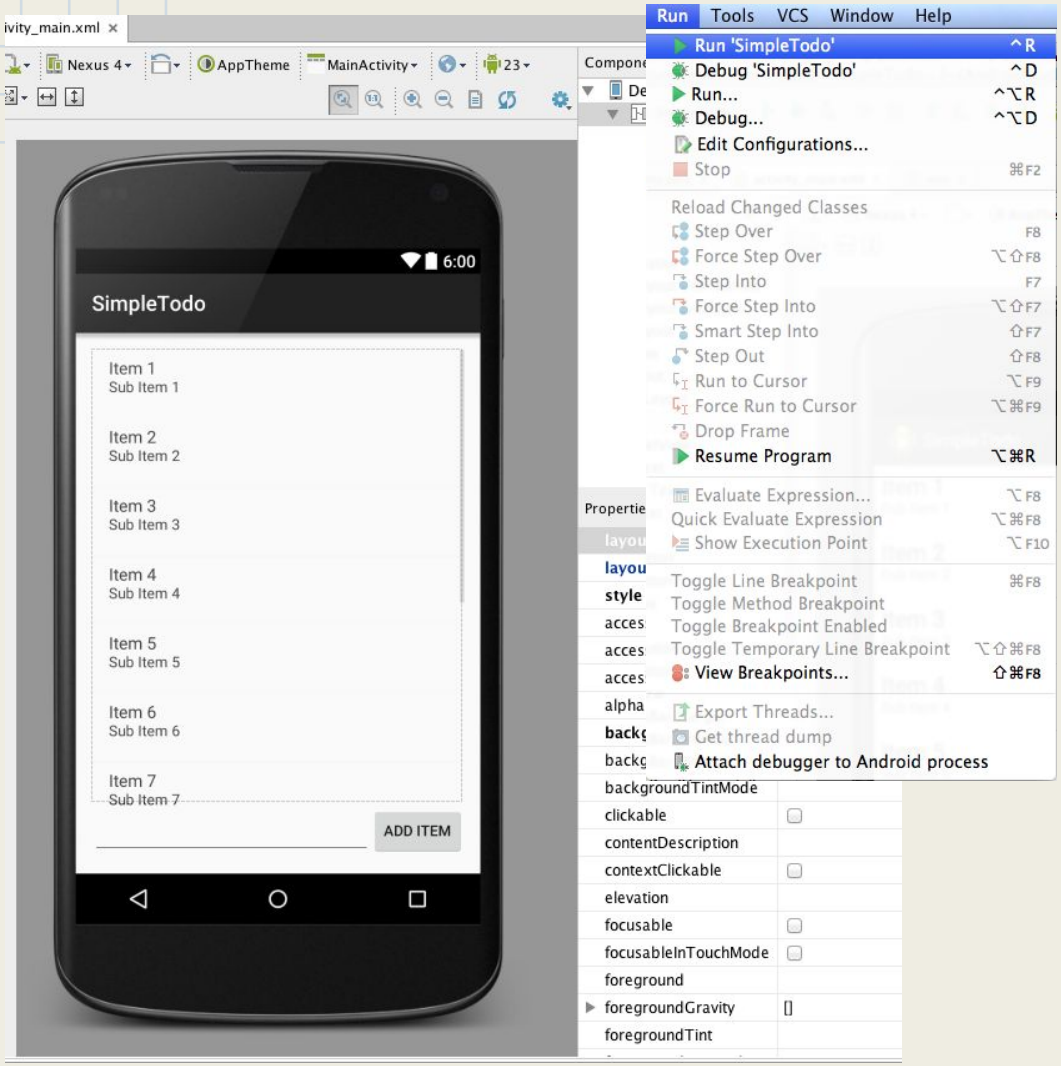
MainActivity.java × | activity_main.xml ×

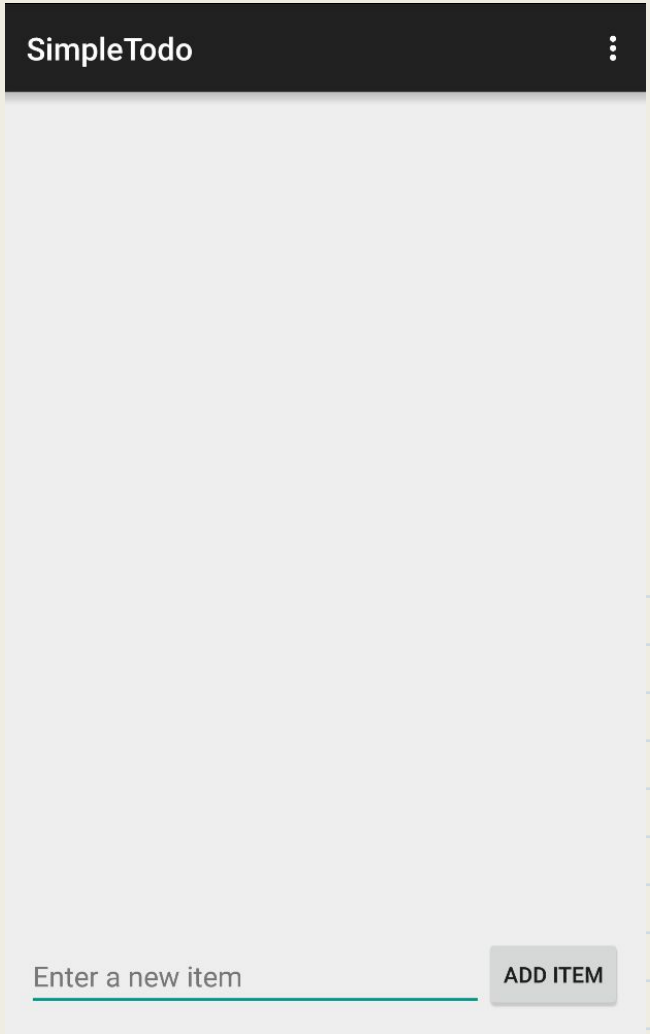
```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <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>
```

Design | Text

- 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





Code Basic Behavior

MainActivity.java × activity_main.xml ×

```java
package com.codepath.simpletodo;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;

public class MainActivity extends AppCompatActivity {

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

- 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

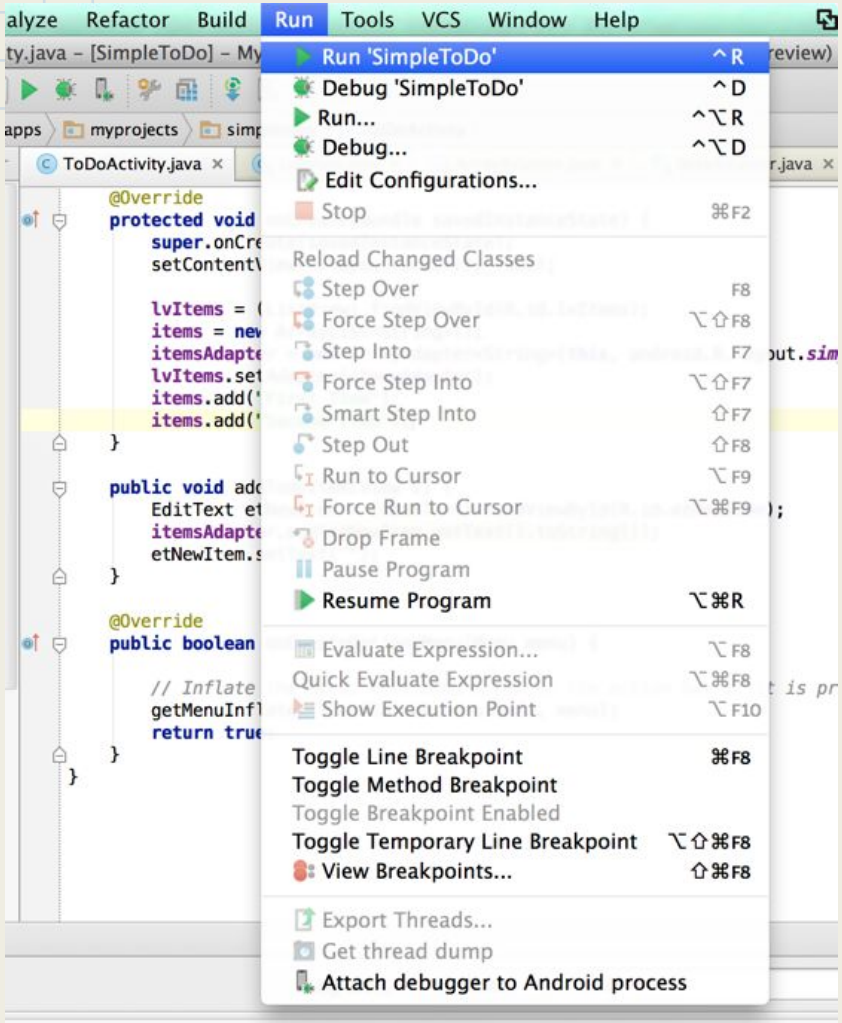
Create List of Items and Display in ListView

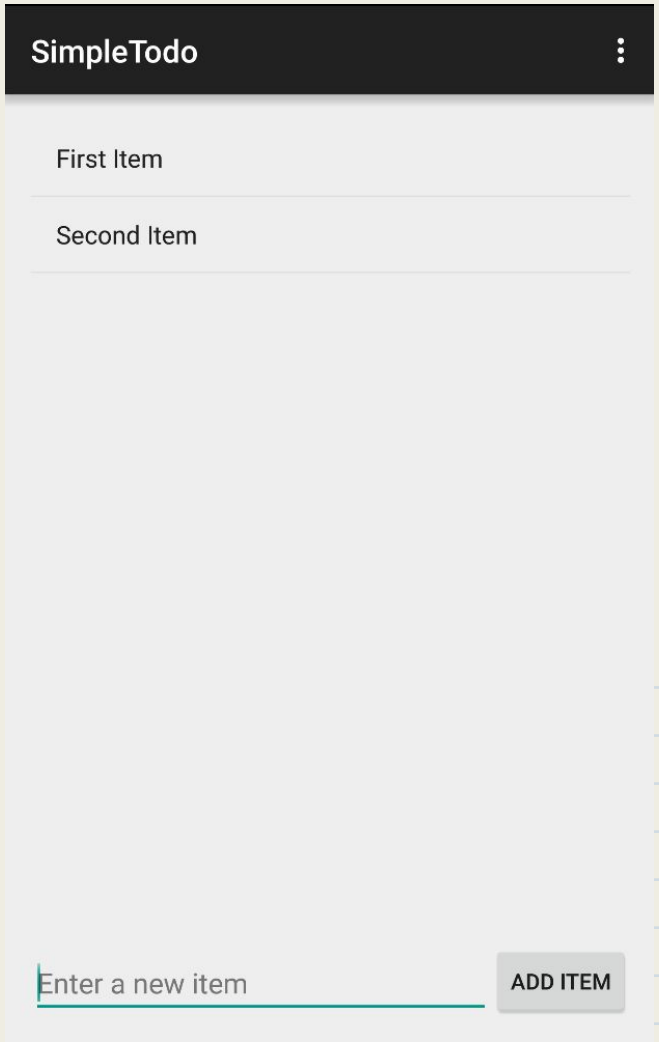
```java
public class MainActivity extends AppCompatActivity {
  ArrayList<String> items;
  ArrayAdapter<String> itemsAdapter;
  ListView lvItems;

  @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

SimpleTodo > app > src > main > res > layout > activity_main.xm

MainActivity.java × | activity_main.xml ×

```
<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" />
```

Design | Text

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

MainActivity.java × | activity_main.xml ×

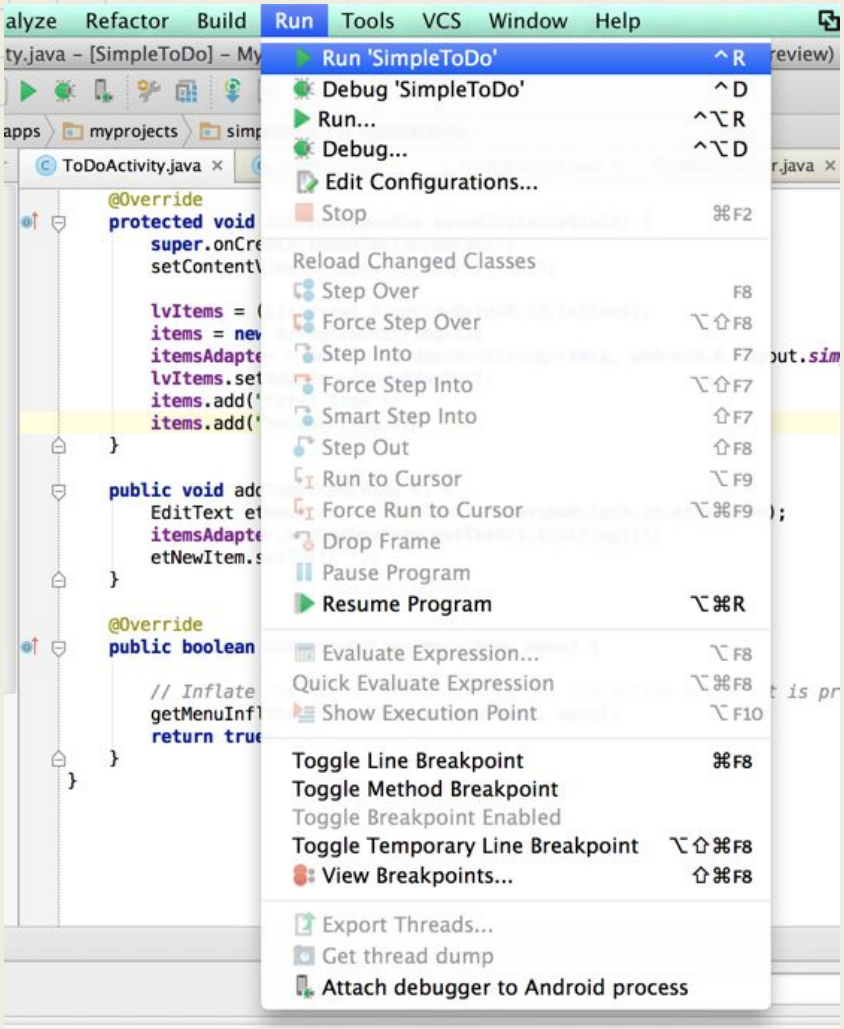
```
public void onAddItem(View v) {
    EditText etNewItem = (EditText) findViewById(R.id.etNewItem);
    String itemText = etNewItem.getText().toString();
    itemsAdapter.add(itemText);
    etNewItem.setText("");
}
```

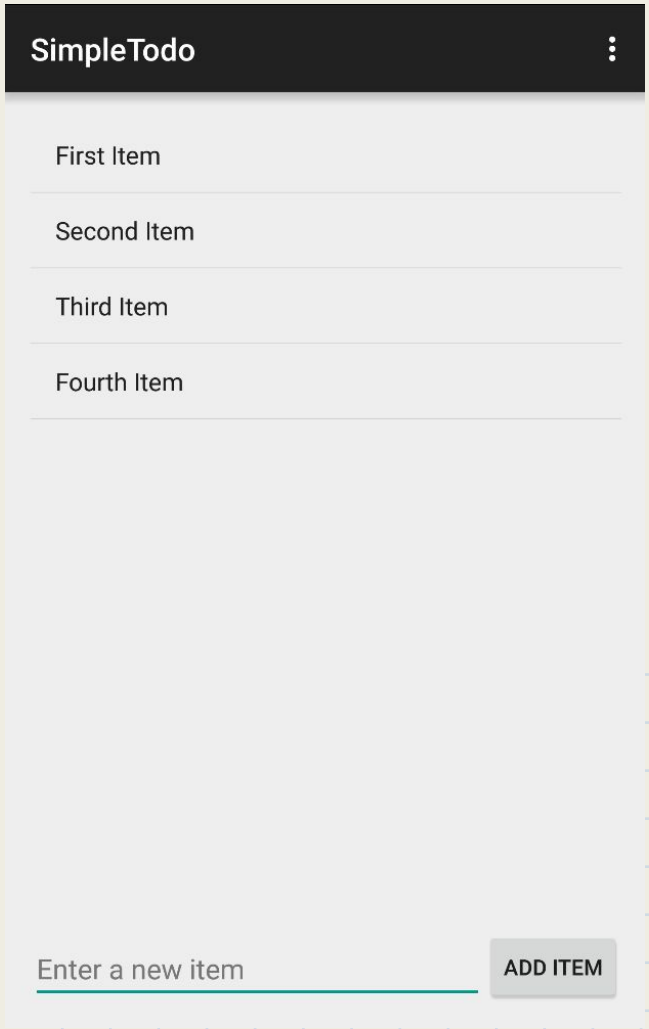
Remove Items from List

```java
@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.OnItemLongClickListener() {
        @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?