Lecture 6: Layouts and Widgets

Jianjun Chen (Jianjun.Chen@xjtlu.edu.cn)

The res Folder and R.java

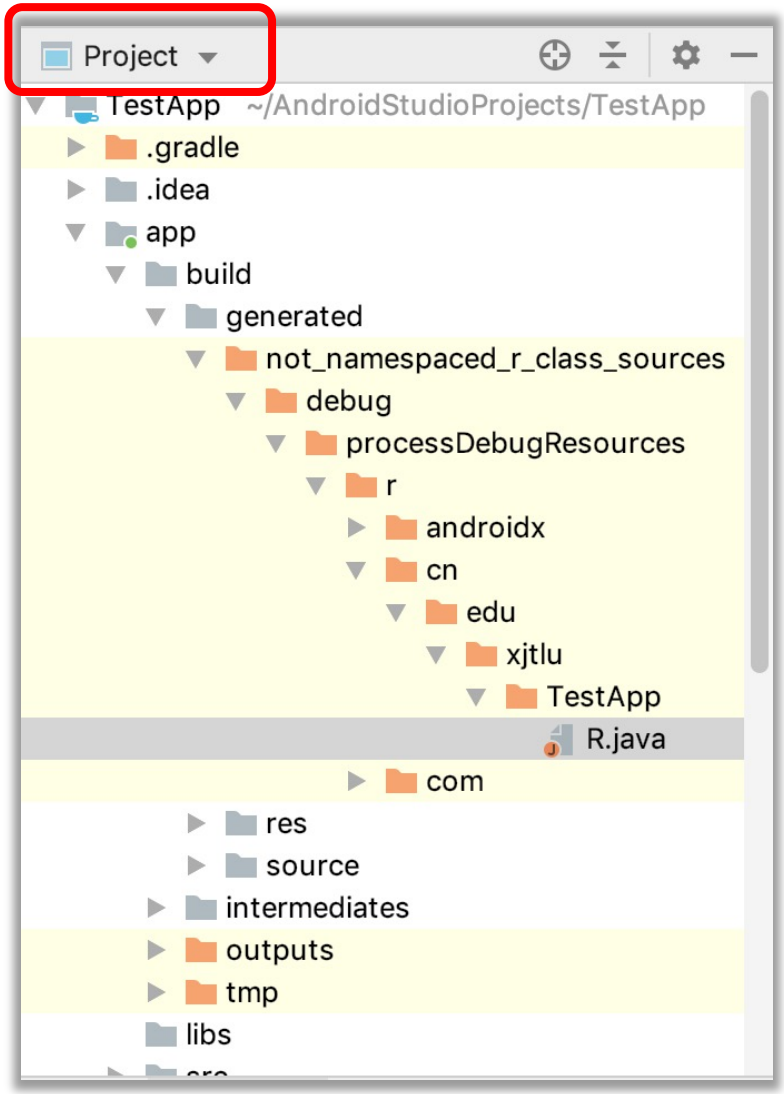
- Most resources in **res** folder will be assigned with a (`public static final int`) ID number by Android Asset Packaging Tool (AAPT) in `R.java`

- Android system use these ID numbers to access their corresponding resource elements.
  - The general format is:
    `R.<resource_type>.<resource_name>`

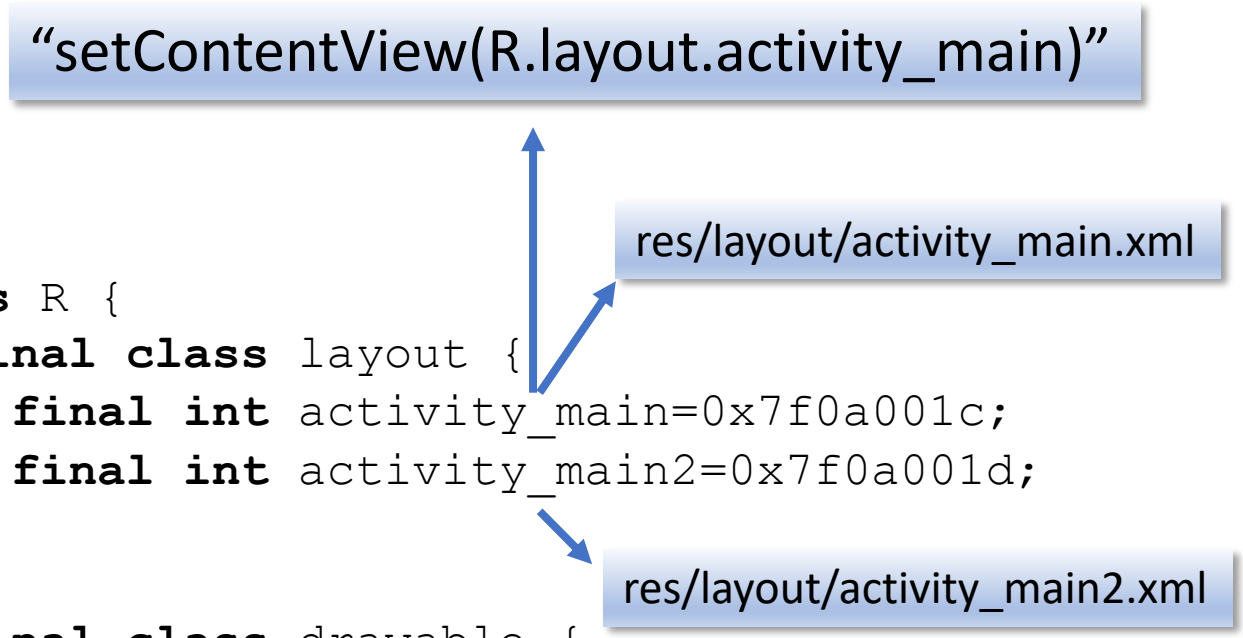
R.java

- This special class can be found under the “build” folder in “Project” view

- Each type of resource in res folder is associated with a static nested class.
  - Layouts
  - Mipmaps
  - Drawables
  - Strings
  - …



R.java



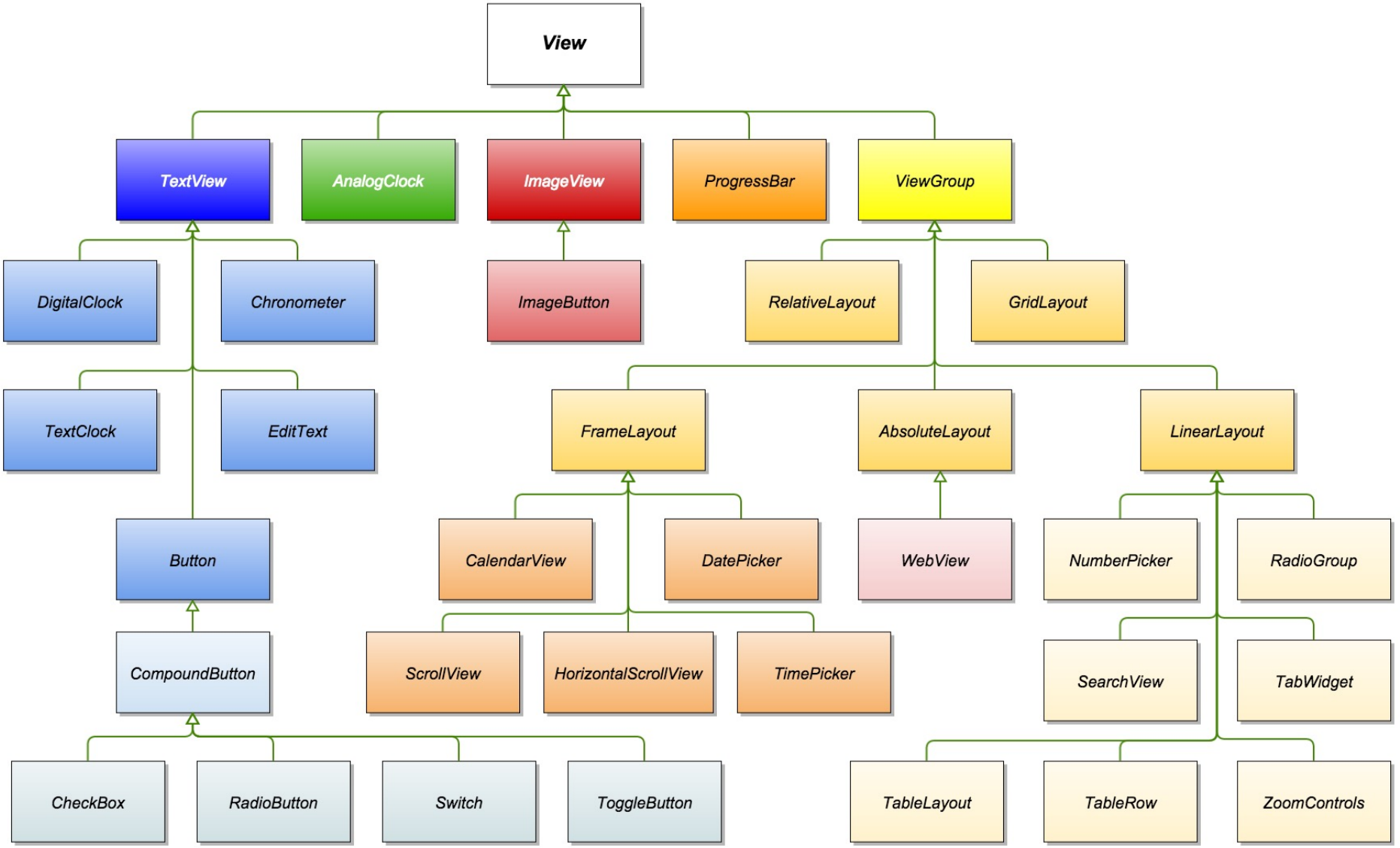

```
public final class R {
  public static final class layout {
    public static final int activity_main=0x7f0a001c;
    public static final int activity_main2=0x7f0a001d;
  }

  public static final class drawable {
    // IDs for the images in drawable folder
  }

  public static final class string {
    public static final int app_name=0x7f0c001b;
    // other string values in strings.xml
  }
}
```

*Final classes cannot be subclassed

android.view.View Class

The Base Classes of UI

The Android View Class



Base Classes of UI

- The base class of widgets is `View`.
  - `Button`, `List`, `EditText`, `Checkbox` …
- A View occupies a rectangle area on the screen.
  - It renders its visual elements in this area.
  - It also handles the events that take place in this area.

- The base class of layouts is `ViewGroup`.
  - `ViewGroup` itself is a subclass of `View`. It's a widget that contains other widgets.
  - `ViewGroup` automatically arranges positions of the `Views` that are added to it.
  - You can add another `ViewGroup` (layout) to a `ViewGroup`.

Getting View Objects From R.java

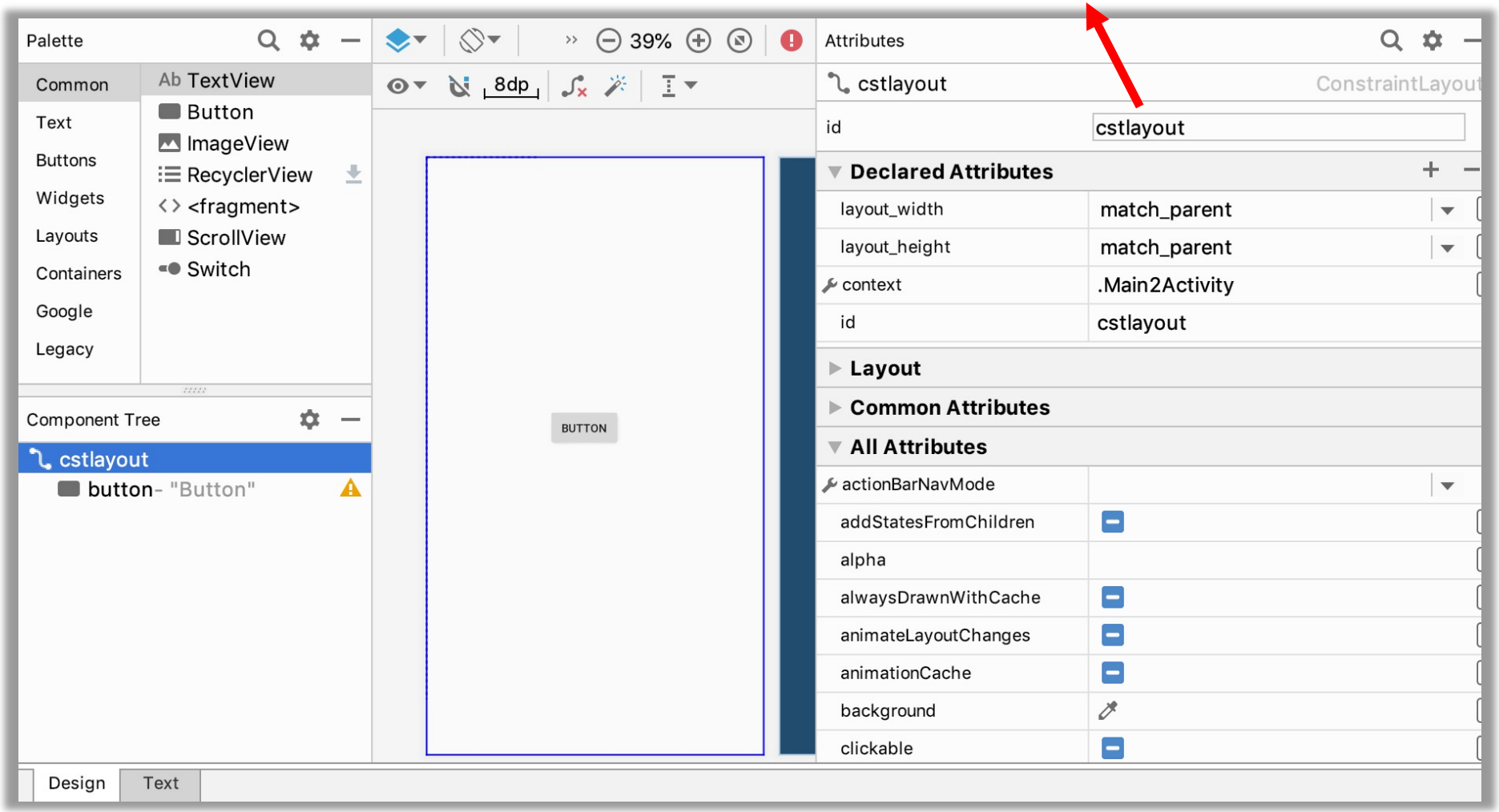
- To obtain a (e.g. TextView) widget called "textView1" from your layout XML file, use:

```
TextView view = (TextView)
        findViewByID(R.id.textView1);
```

You can use `findViewByID()` to get any `View` objects in your XML, including widgets and layouts

Where's the View ID?

```
findViewByID(R.id.cstlayout)
```



Where's the View ID in Layout XMLs?

```
<androidx.constraintlayout.widget.ConstraintLayout xmlns
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/cstlayout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".Main2Activity">

    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="149dp"
        android:layout_marginTop="307dp"
        android:layout_marginEnd="175dp"
        android:layout_marginBottom="376dp"
        android:text="Button"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
```

Widgets

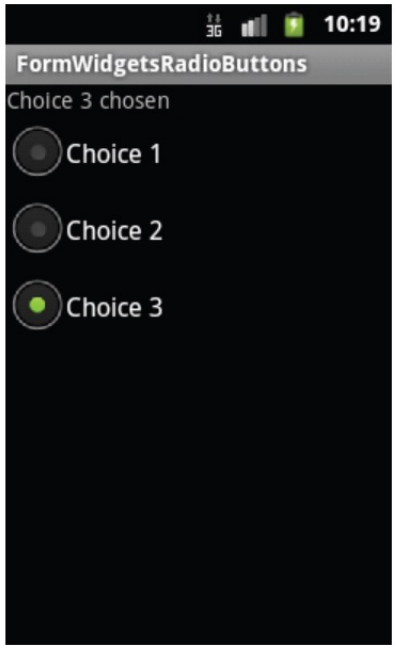
Widgets, Event handling

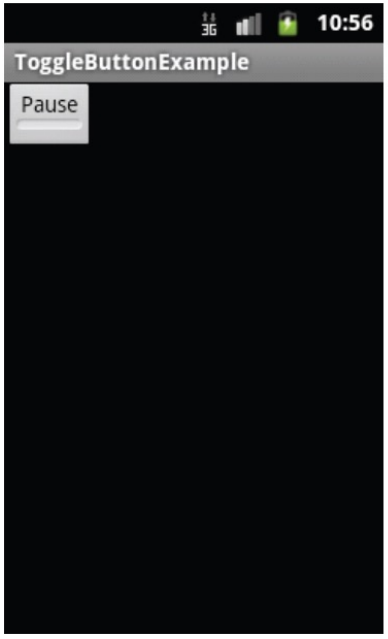
Widgets

- Buttons
- Text field
- Editable text field
- Check box
- Radio buttons
- Toggle Button
- Rating Bar
- Data Picker
- Spinner
- …


10:11
ButtonExample
Press Me!


10:14
FormWidgetsCheckBox
This is a checkbox
I'm not checked
Hide CheckBox


10:19
FormWidgetsRadioButtons
Choice 3 chosen
Choice 1
Choice 2
Choice 3


10:56
ToggleButtonExample
Pause


10:54
RatingsBarExample


Hello DatePicker
8-4-2008
change the date
Mon, 04 August, 2008
04
Aug
2008
Set
Cancel


Hello TimePicker
18:37
change the time
6:37 PM
6
37
PM
Set
Cancel

Event Listeners

- There are several listener interfaces defined inside View class:
  - `View.OnClickListener`
  - `View.OnDragListener`
  - `View.OnLayoutChangeListener`

- Each listener contains only one callback method.
  - The method will be called upon an event.
  - E.g. `onClick()` in `View.OnClickListener` will get called when the user clicks a widget.

Adding Listeners

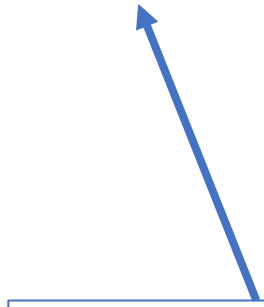
- First, create a class that implements the Listener interface.

- Then, register an object of this class by calling the `setXXXListener(Listener)` function of that `View` object.
  - View object can be a button, a rating bar etc..

- The callback method will be called automatically when a user interacts with the widget (view).

Event Listener Example

```
//----Create a TextView------
TextView textView = new TextView( context: this);
textView.setText("This TextView is dynamically created");
textView.setLayoutParams(params);

textView.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {

    }
});
```



Event Listener Example 2

```
public class Main2Activity extends AppCompatActivity implements View.OnClickListener {
    @Override
    public void onClick(View v) {

    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        Button b = (Button) this.findViewById(R.id.button);
        b.setOnClickListener(this);
    }
}
```

Widgets with Multiple Options

AdapterView and its subclasses

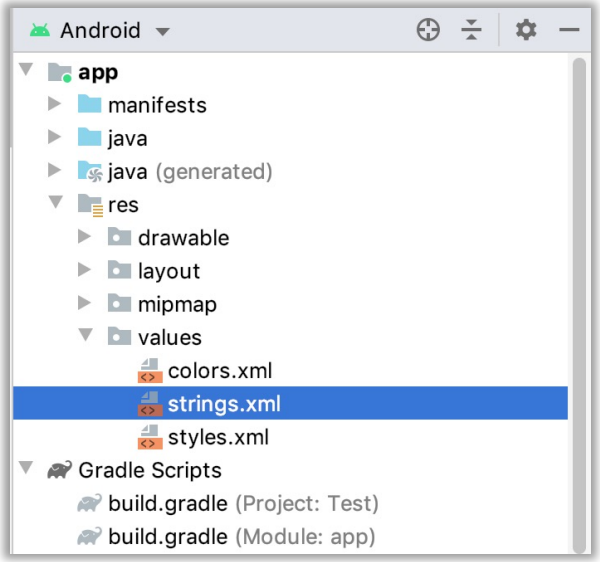
Widgets with Multiple Options

- Widgets like `Spinner` and `ListView` and `GridView` are capable of offering multiple choices to users.
  - Here, `Spinner` will be used as an example.

- Assume we want to know the favourite activity of a user. There are two ways of creating a working spinner:
  - Using UI designer.
  - Using code.

Method 1: Using Designer

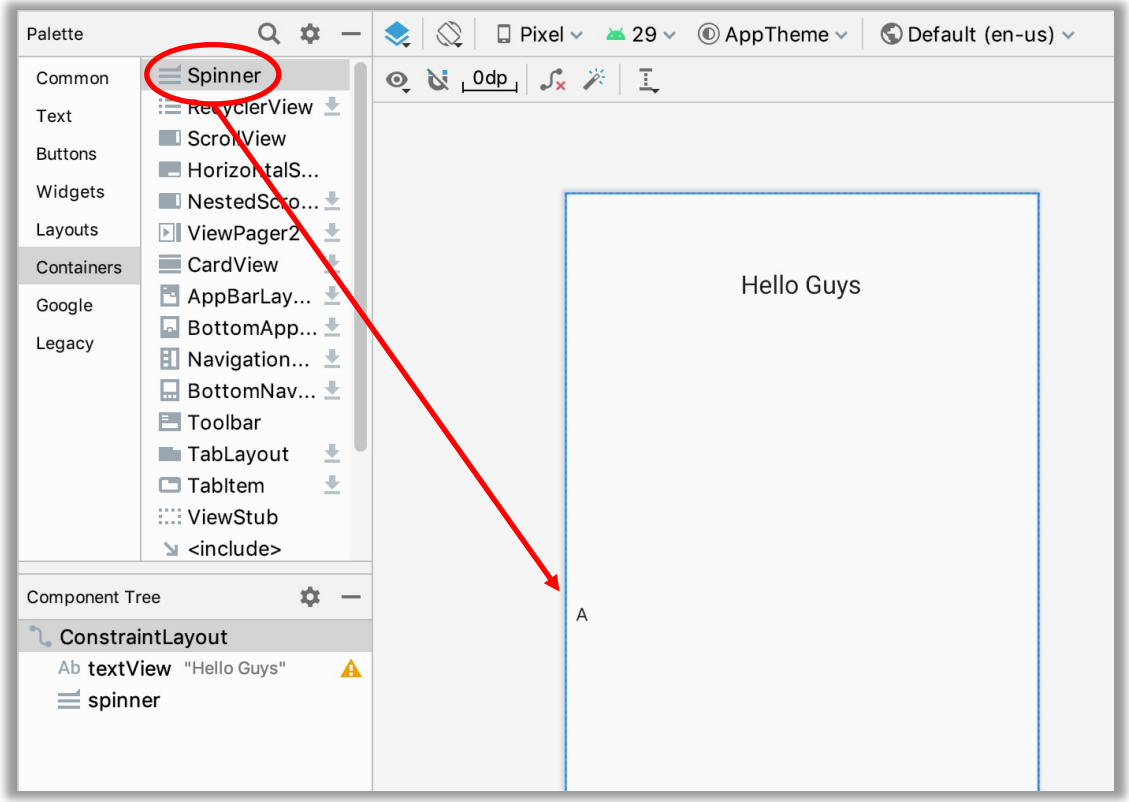
- Step 1: create spinner prompt and options in `strings.xml`




```
<string name="prompt">Choose a hobby</string>
<string-array name="hobbies">
    <item>A</item>
    <item>B</item>
    <item>C</item>
    <item>D</item>
</string-array>
```

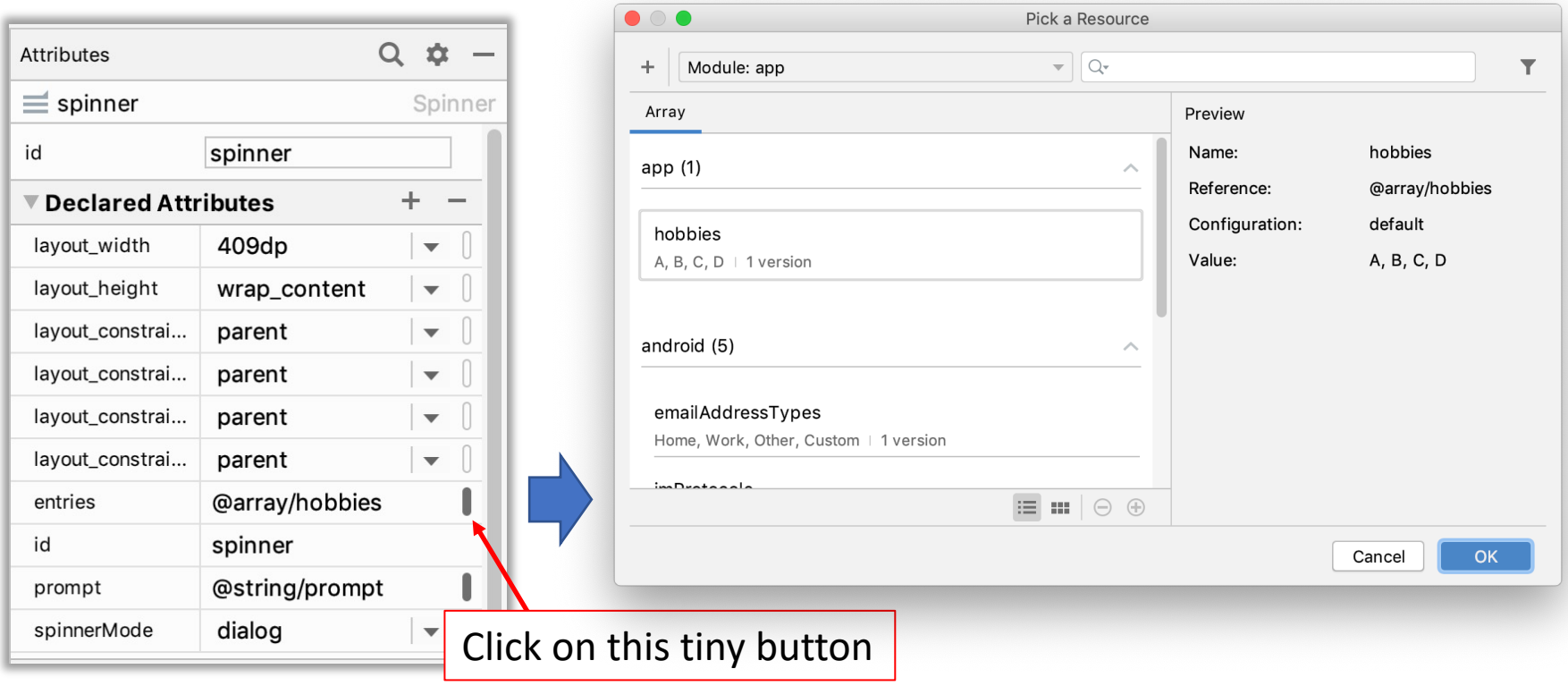
Method 1: Using Designer

- Step 2: Add the `Spinner` into the layout.
  - Open the layout, and add spinner into the layout.



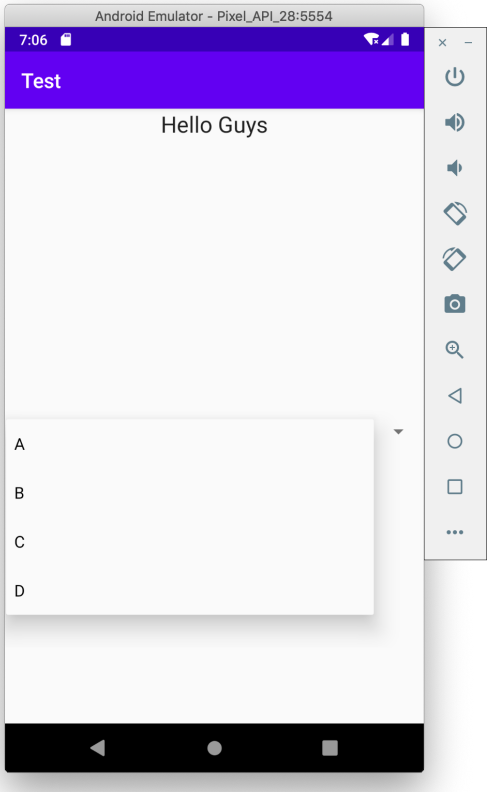
Method 1: Using Designer

- Step 3: Add the strings you defined in step 1.
  - In “declared attributes”, choose the prompt as well as the string array.

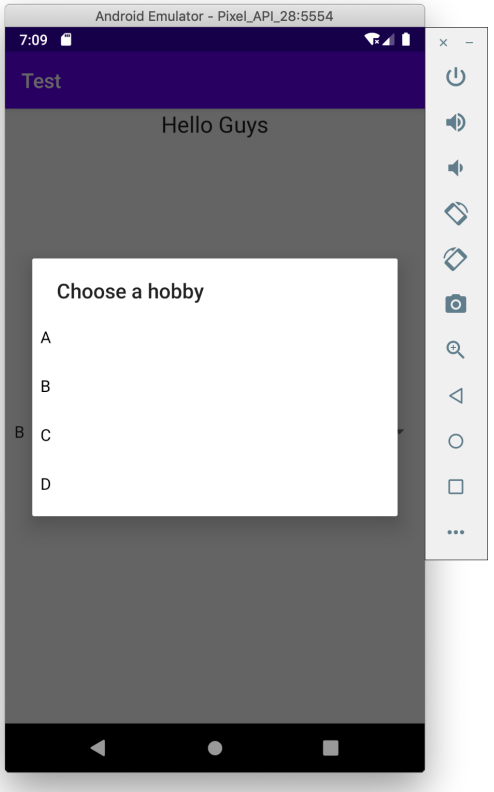


Method 1: Using Designer

- Step 4: Choose spinner mode: dialog or dropdown



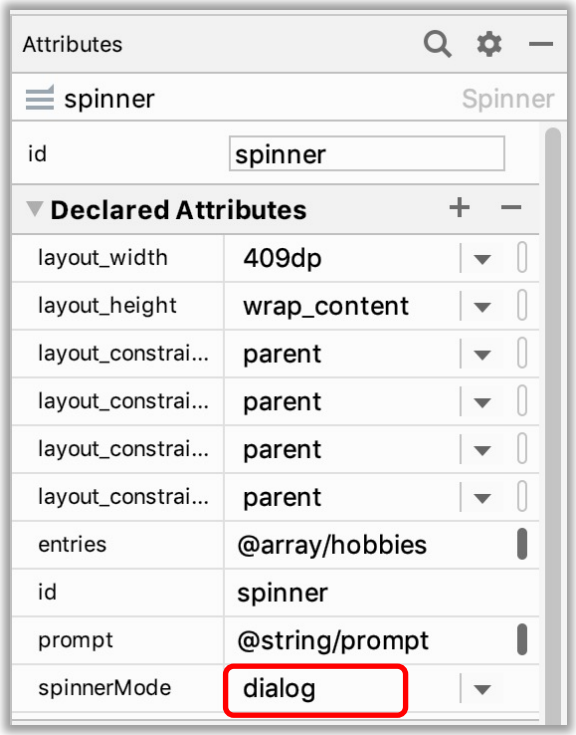
Dropdown: No prompt



Dialog: shows the prompt

Method 1: Using Designer

- Step 4: Choose spinner mode: dialog or dropdown




```
<Spinner
   android:id="@+id/spinner"
   ……
   android:spinnerMode="dropdown"
   android:prompt="@string/prompt"
   android:entries="@array/hobbies" />
```

```
<Spinner
   android:id="@+id/spinner"
   ……
   android:spinnerMode="dialog"
   android:prompt="@string/prompt"
   android:entries="@array/hobbies" />
```

Method 1: Using Designer

- Step 5: Implement `OnItemSelectedListener`.

```
final Spinner s = findViewById(R.id.spinner);
s.setOnItemSelectedListener(new Spinner.OnItemSelectedListener() {
    @Override
    public void onItemSelected(AdapterView<?> parent,
                               View view, int position, long id) {
        System.out.println(parent.getSelectedItem());
    }
    @Override
    public void onNothingSelected(AdapterView<?> parent) {

    }
});
```

https://stackoverflow.com/questions/3009745/what-does-the-question-mark-in-java-generics-type-parameter-mean

onItemSelected()

Parameters:

- parent – The AdapterView where the selection happened
- view – The view within the AdapterView that was clicked
- position – The position of the view in the adapter
- id – The row id of the item that is selected

https://stackoverflow.com/questions/12965817/practical-difference-between-position-and-row-id-in-onlistitemclick

Method 2: Using code

- Step 1: create spinner in UI

```
<Spinner
   android:id="@+id/spinner2"
   android:layout_width="match_parent"
   android:layout_height="wrap_content" />
```

More flexible, allows you to dynamically change the content

Method 2: Dynamic Creation

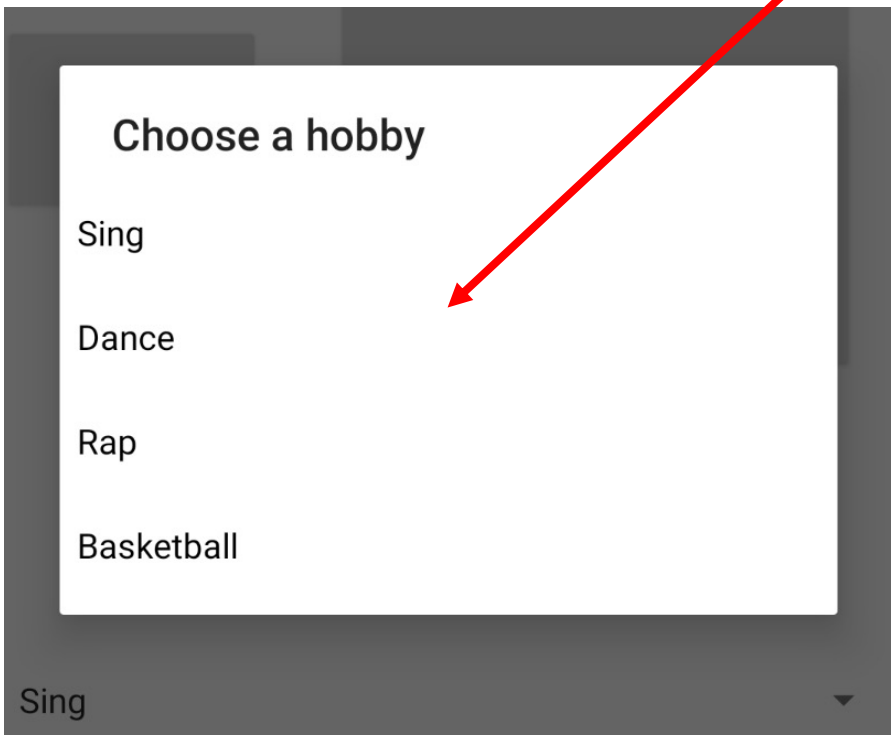
- Step 2: Create items and Adapter.

```
Spinner spinner2 = (Spinner) findViewById(R.id.spinner2);
List<String> list = new ArrayList<String>();
list.add("Sing");
list.add("Dance");
list.add("Rap");
list.add("Basketball");
ArrayAdapter<String> dataAdapter = new ArrayAdapter<String> (
                                    this, android.R.layout.simple_spinner_item, list);
spinner2.setAdapter(dataAdapter);
```

- Step 3: Set up listener like method 1.

Adapter & AdapterView

An `AdapterView` is a view whose children are determined by an `Adapter`.



Adapter & Adapter View

- The `ArrayAdapter` we saw is a `BaseAdapter` backed by an array of arbitrary objects.
  - It's a subclass of `BaseAdapter`
- A `BaseAdapter` is an implementation of `Adapter` interface that can support `ListView` and `Spinner`.
- An `Adapter` object acts as a bridge between an `AdapterView` and the underlying data for that view.
- An `AdapterView` is a `View` whose children are determined by an `Adapter`.
  - `Spinner`, `ListView`, `GridView` are different `AdapterViews`.

Layout Basics

Creating Layouts, Layout XML attributes, (Some) Layouts

Creating Layouts

- Using java code only:
  - Instantiate layout elements at runtime.
  - Then use `this.setContentView()` to assign a `View` object to the current activity.
- Create layouts XMLs using android studio UI designer and call `setContentView()`
  - You may still modify UI elements at runtime later.

```
@Override
public void setContentView(@LayoutRes int layoutResID) {
    getDelegate().setContentView(layoutResID);
}
```

These two methods are defined inside Activity

```
@Override
public void setContentView(View view) { getDelegate().setContentView(view); }
```

UI Design Using Code

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    //Create params for views---------------
    LinearLayout.LayoutParams params =
            new LinearLayout.LayoutParams(LinearLayout.LayoutParams.FILL_PARENT,
                    LinearLayout.LayoutParams.WRAP_CONTENT);
    //Create a layout----------------
    LinearLayout linearLayout = new LinearLayout( context: this);
    linearLayout.setOrientation(LinearLayout.VERTICAL);

    //----Create a TextView------
    TextView textView = new TextView( context: this);
    textView.setText("This TextView is dynamically created");
    textView.setLayoutParams(params);

    //---Add all elements to the layout
    linearLayout.addView(textView);

    //---Create a layout param for the layout-----------------
    LinearLayout.LayoutParams layoutParams =
            new LinearLayout.LayoutParams(ActionBar.LayoutParams.FILL_PARENT,
                    ActionBar.LayoutParams.WRAP_CONTENT);
    this.addContentView(linearLayout, layoutParams);
}
```

IDs in layout XMLs are similar to object names

UI Design Using XMLs

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="ht
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:text="@+string/the_message"
        android:textSize="36sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

Some XML attributes

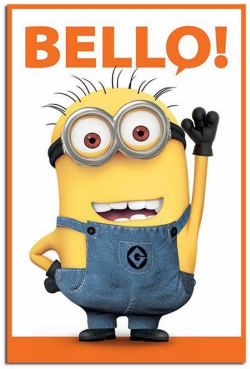
XML attributes controls the properties of Views

- **`android:id="@+id/records"`**
  - This is the ID of this widget.
  - Android Studio will create a nested class called `id` in `R`, with a variable called `records`.
  - `id` class contains IDs for all kinds of views.

  -> `(TextView) findViewById(R.id.records)`

- **`android:layout_width="0dp"`**
  `android:layout_height="wrap_content"`
  - Width and height of this view

Some XML attributes



- **`android:text="@string/the_message"`**

Inside res/values/strings.xml, you need:

```
<resources>
    <string name="the_message">Bello!</string>
</resources>
```

- Alternatively: `android:text="Bello!"`

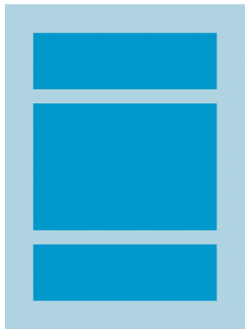
- The first way is recommended as strings.xml allows the user to use translations in different language settings.

Layout XMLs and Java Code

"In general, the XML vocabulary for declaring UI elements closely follows the structure and naming of the classes and methods, where element names correspond to class names and attribute names correspond to methods."

- There are exceptions though.
- Check the official Android documentation:
  - https://developer.android.com/guide/topics/ui

Common Layouts



Linear layout: vertical/horizontal
A scroll bar will be automatically added if the total height/width exceeds the length of the screen.

Relative/Constraint layout:
specify the location of child objects relative to each other (child A to the left of child B) or to the parent (aligned to the top of the parent).

Web view:
you can create your own web browser based on it

Other Layouts

Details are available in the API reference.

- `FrameLayout`
- `TableLayout`
- `AbsoluteLayout`
- `GridLayout`
- …
- You can start by following the API reference for `ViewGroup`, as all these layouts are it's subclasses
  - Check "Known (In)Direct Subclasses" section

Lab Session:

- Implement a simple calculator with + - * / operators.

- Implement a password text input, which displays * when entering the password. If a user clicks “show password” button, the password will be shown instead.