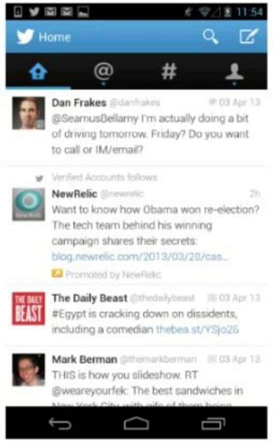
Dynamic UI

•••

Lecture 8

Generating UI at runtime

- Sometimes your app's UI cannot be fully specified in XML.
  - Example: You don't know how many widgets you will need until the user gives input or until a file is downloaded.
- In these cases, your app needs to be able to generate UI widgets dynamically in Java code.

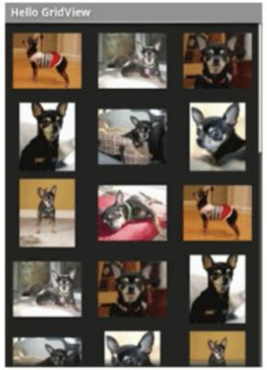


Dan Frakes @danfrakes 03 Apr 13
@SeamusBellamy I'm actually doing a bit of driving tomorrow. Friday? Do you want to call or IM/email?

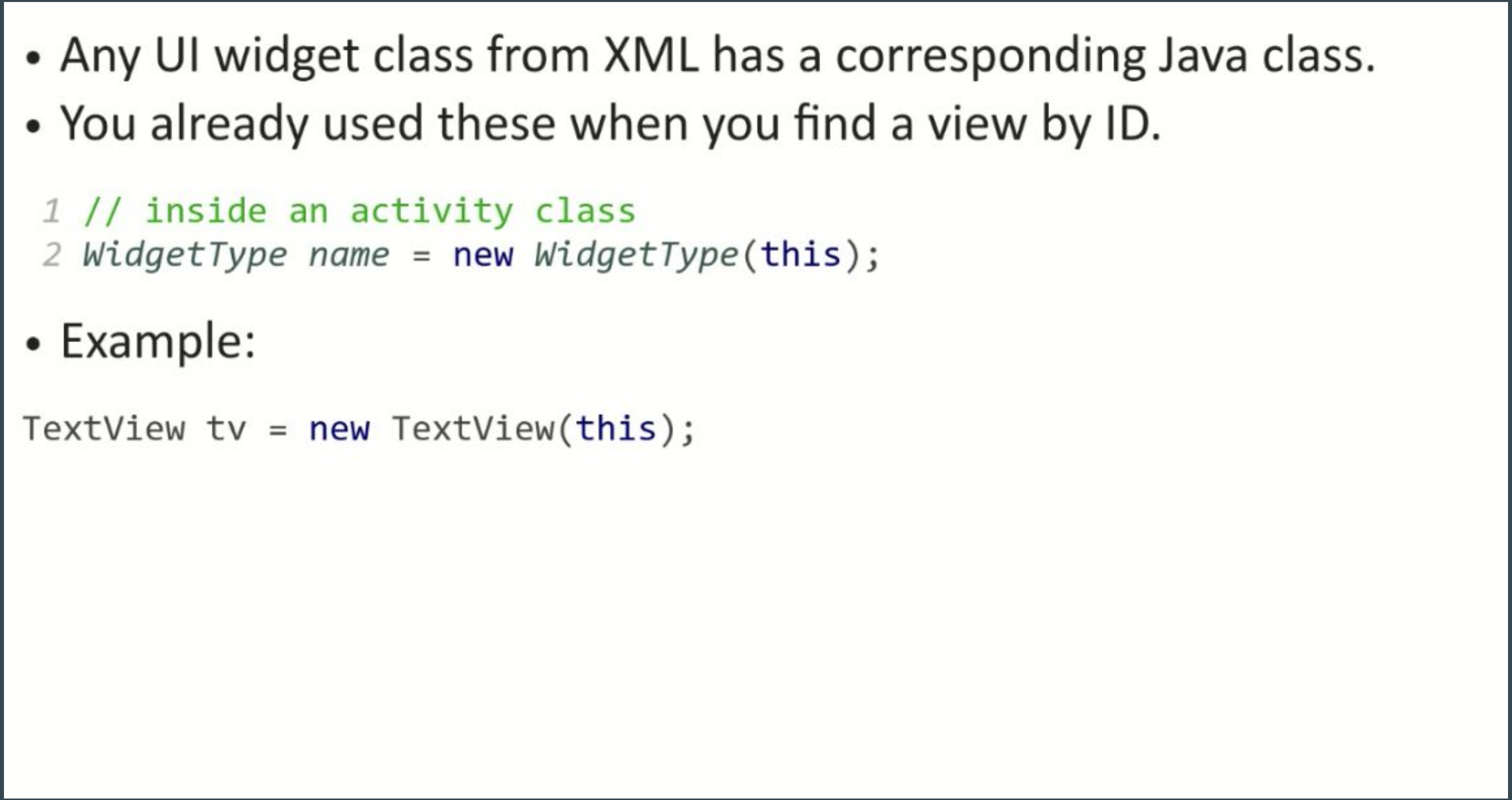
Verified Accounts follows
NewRelic @newrelic 2h
Want to know how Obama won re-election? The tech team behind his winning campaign shares their secrets: blog.newrelic.com/2013/03/28/cas...
Promoted by NewRelic

The Daily Beast @thedailybeast 03 Apr 13
#Egypt is cracking down on dissidents, including a comedian thebea.st/YSjo26

Mark Berman @themarkberman 03 Apr 13
THIS is how you slideshow. RT @weareyourfek: The best sandwiches in



UI Widget Objects



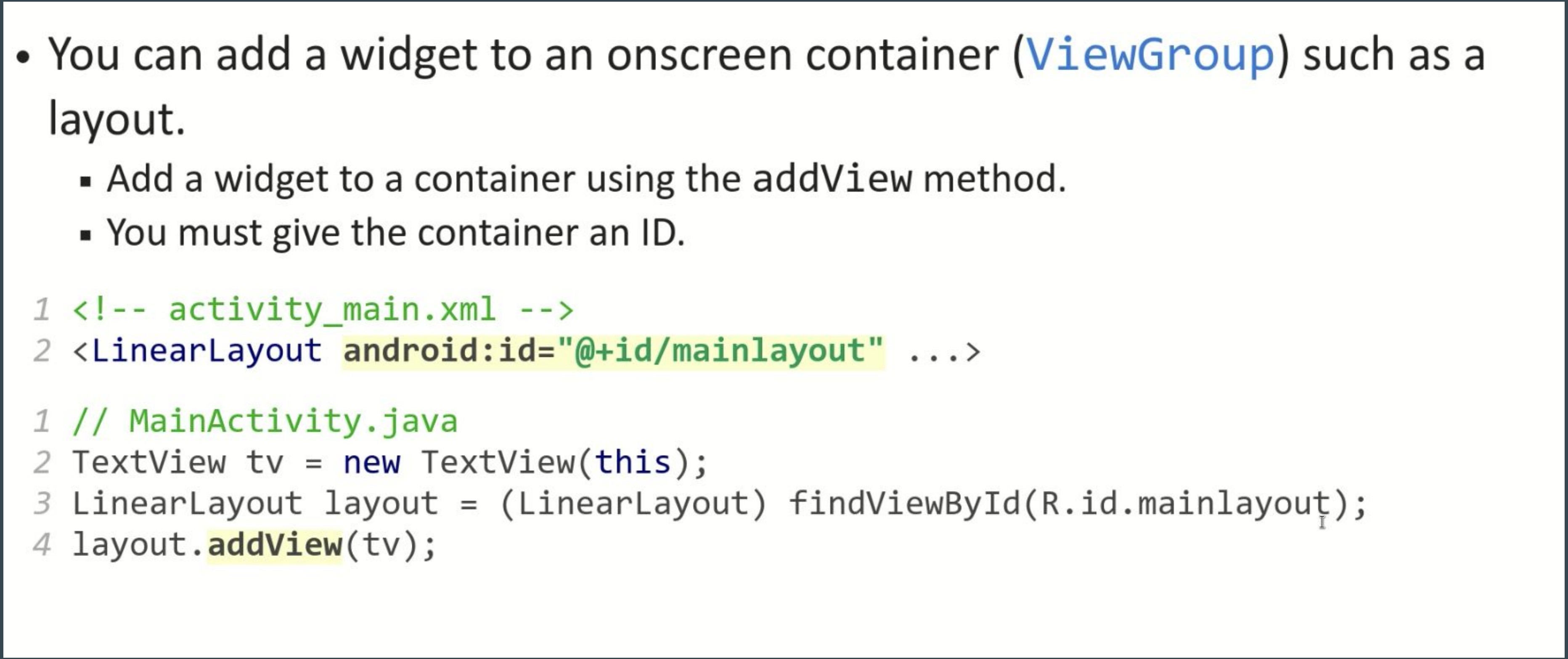
- Any UI widget class from XML has a corresponding Java class.
- You already used these when you find a view by ID.

```
// inside an activity class
WidgetType name = new WidgetType(this);
```

- Example:

```
TextView tv = new TextView(this);
```

Adding widget to layout

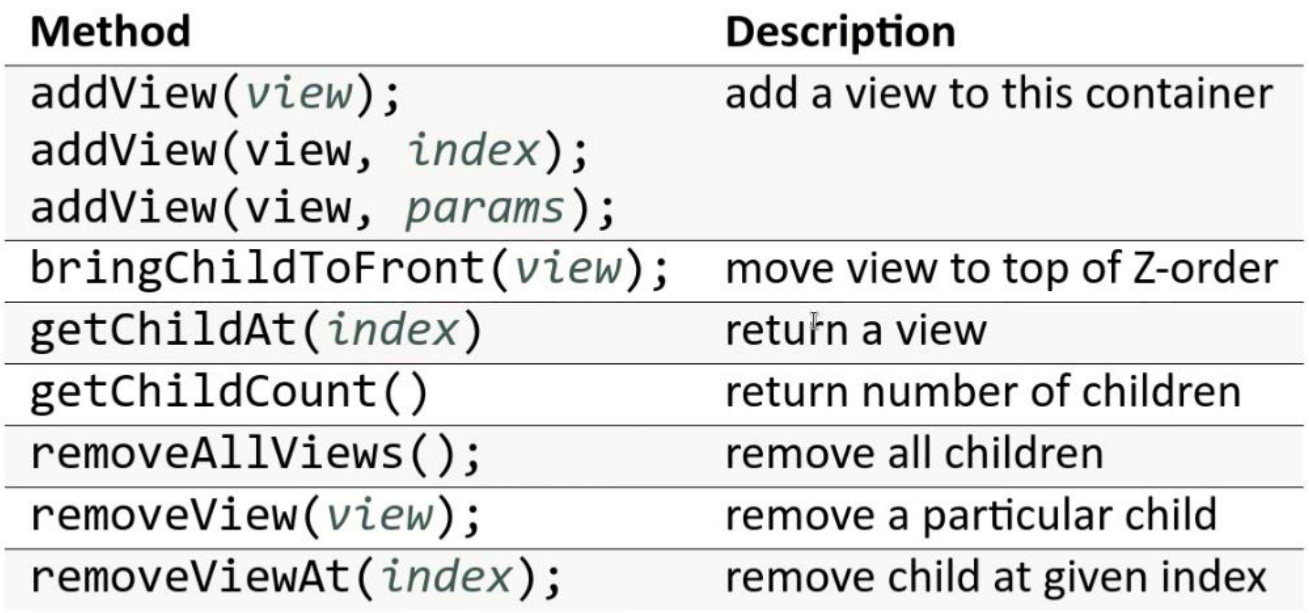


- You can add a widget to an onscreen container (ViewGroup) such as a layout.
  - Add a widget to a container using the addView method.
  - You must give the container an ID.

```
<!-- activity_main.xml -->
<LinearLayout android:id="@+id/mainlayout" ...>
```

```
// MainActivity.java
TextView tv = new TextView(this);
LinearLayout layout = (LinearLayout) findViewById(R.id.mainlayout);
layout.addView(tv);
```

ViewGroup methods

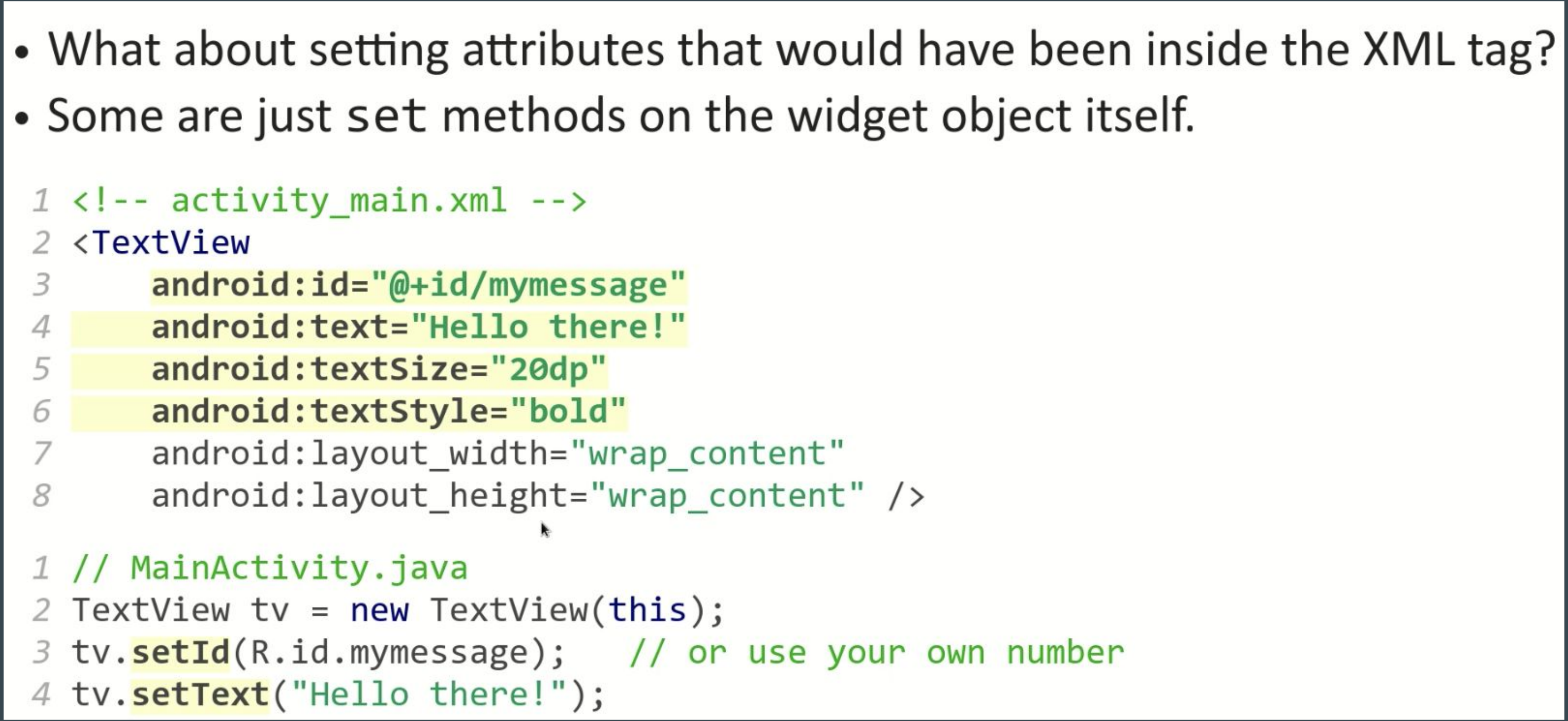


Method	Description

`addView(`*`view`*`);`
`addView(view, `*`index`*`);`
`addView(view, `*`params`*`);`
	add a view to this container

`bringChildToFront(`*`view`*`);`	move view to top of Z-order
`getChildAt(`*`index`*`)`	return a view
`getChildCount()`	return number of children
`removeAllViews();`	remove all children
`removeView(`*`view`*`);`	remove a particular child
`removeViewAt(`*`index`*`);`	remove child at given index

Widget parameters

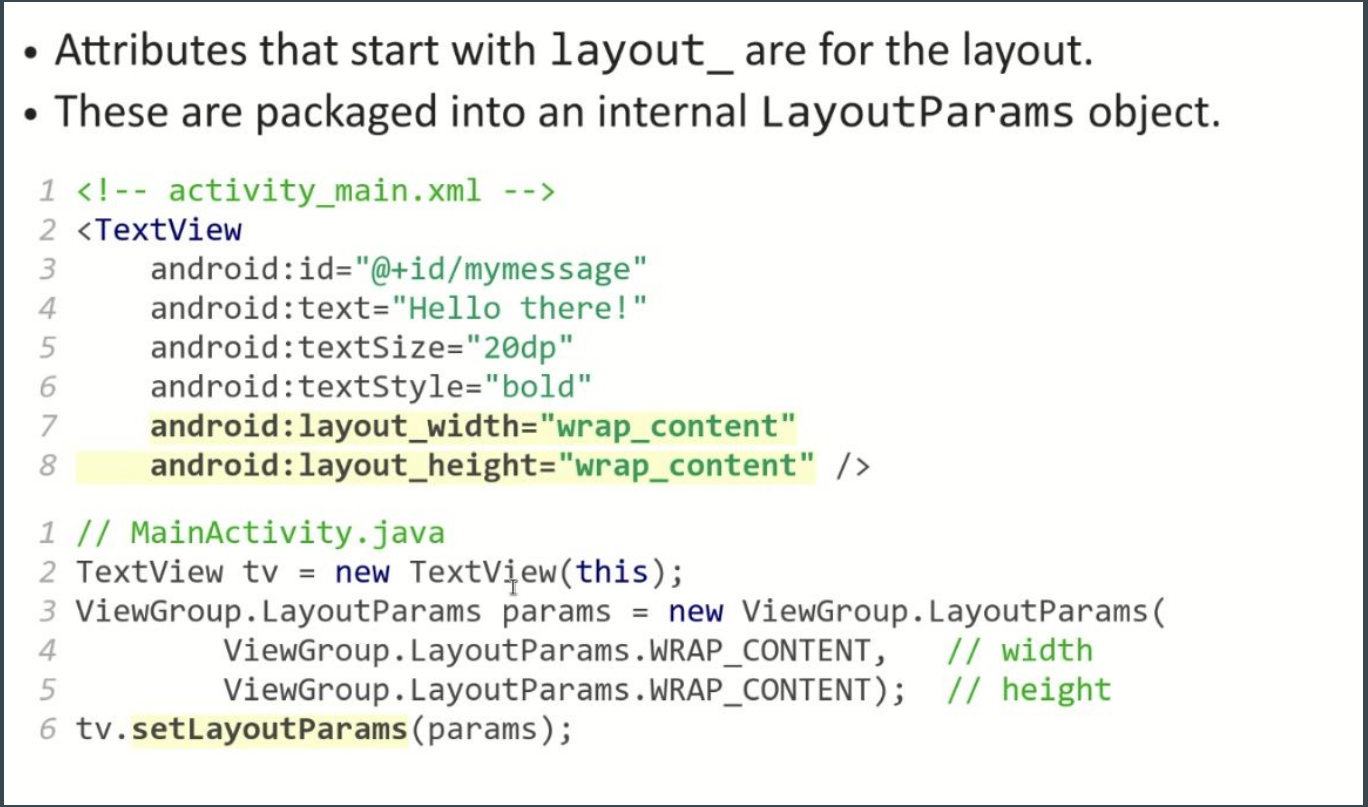


- What about setting attributes that would have been inside the XML tag?
- Some are just set methods on the widget object itself.

```xml
<!-- activity_main.xml -->
<TextView
    android:id="@+id/mymessage"
    android:text="Hello there!"
    android:textSize="20dp"
    android:textStyle="bold"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />
```

```java
// MainActivity.java
TextView tv = new TextView(this);
tv.setId(R.id.mymessage);   // or use your own number
tv.setText("Hello there!");
```

Layout parameters

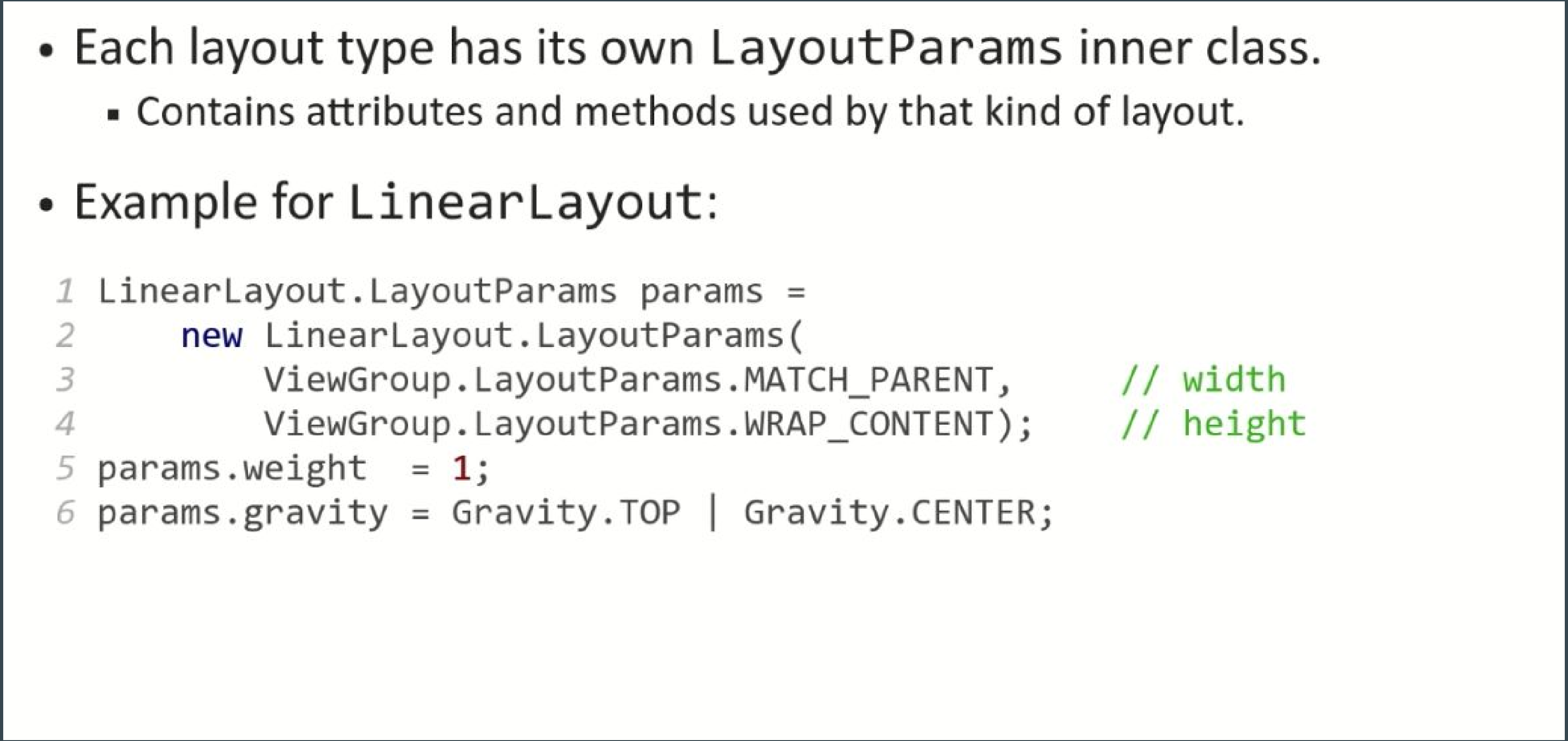


- Attributes that start with `layout_` are for the layout.
- These are packaged into an internal `LayoutParams` object.

```xml
<!-- activity_main.xml -->
<TextView
    android:id="@+id/mymessage"
    android:text="Hello there!"
    android:textSize="20dp"
    android:textStyle="bold"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />
```

```java
// MainActivity.java
TextView tv = new TextView(this);
ViewGroup.LayoutParams params = new ViewGroup.LayoutParams(
        ViewGroup.LayoutParams.WRAP_CONTENT,    // width
        ViewGroup.LayoutParams.WRAP_CONTENT);   // height
tv.setLayoutParams(params);
```

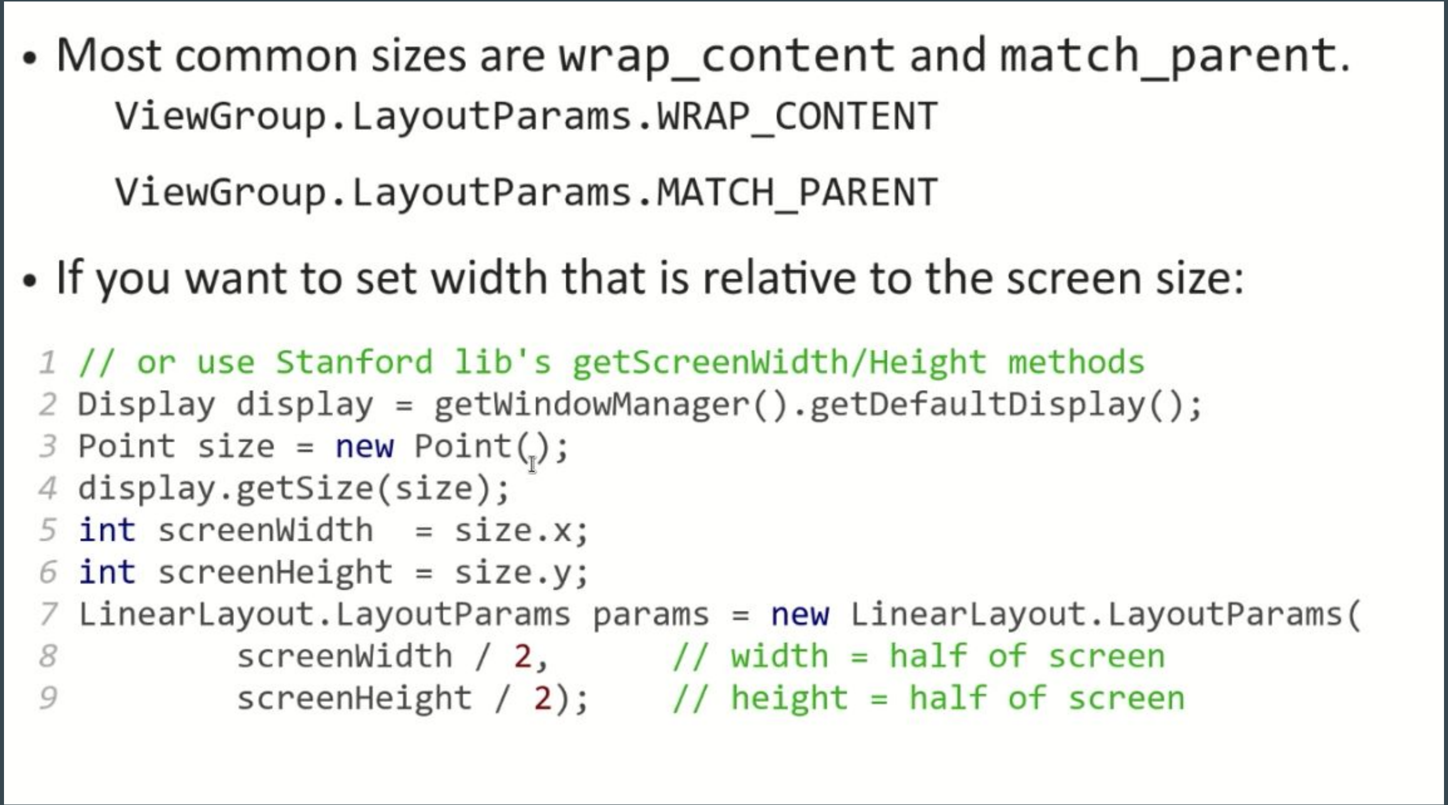
Layout-specific params



- Each layout type has its own `LayoutParams` inner class.
  - Contains attributes and methods used by that kind of layout.
- Example for `LinearLayout`:

```
LinearLayout.LayoutParams params =
    new LinearLayout.LayoutParams(
        ViewGroup.LayoutParams.MATCH_PARENT,     // width
        ViewGroup.LayoutParams.WRAP_CONTENT);    // height
params.weight  = 1;
params.gravity = Gravity.TOP | Gravity.CENTER;
```

Setting Widget Size



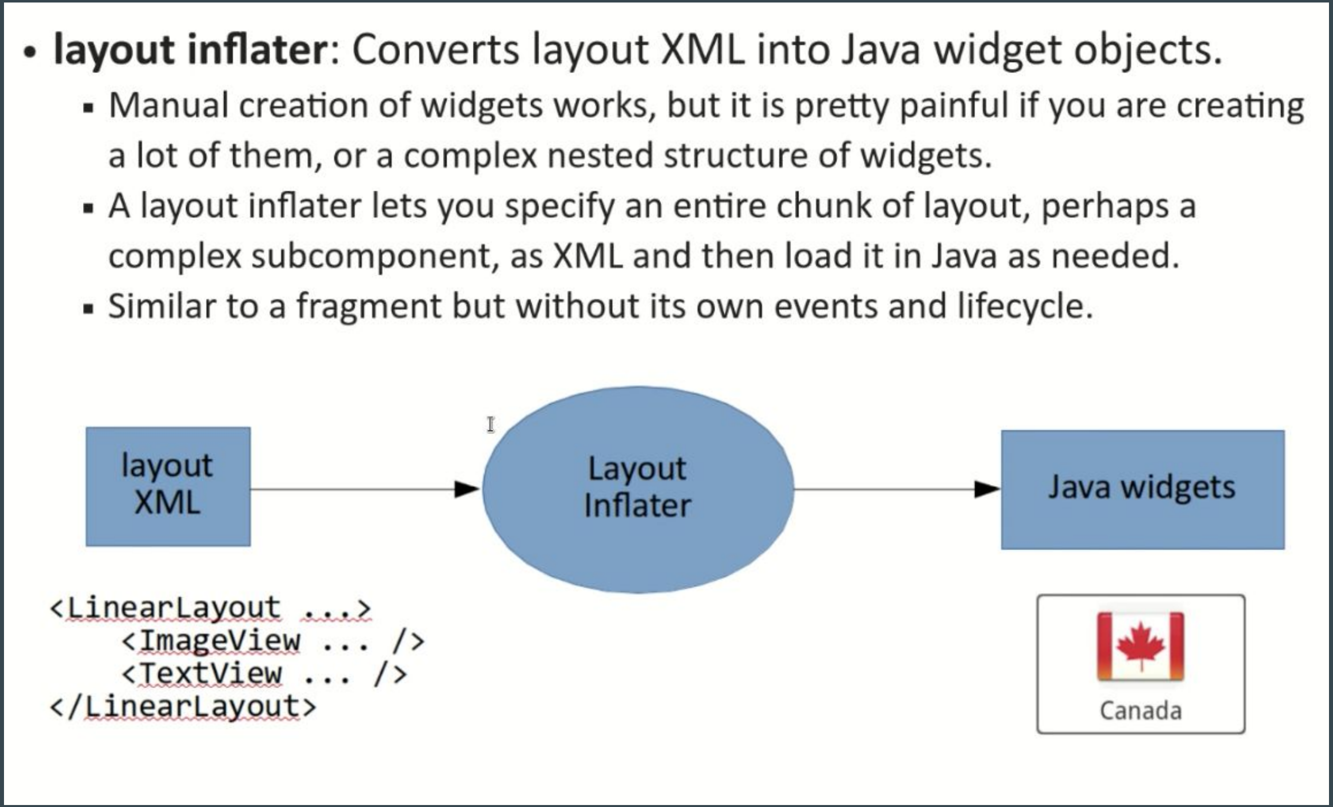
- Most common sizes are `wrap_content` and `match_parent`.

  ```
  ViewGroup.LayoutParams.WRAP_CONTENT

  ViewGroup.LayoutParams.MATCH_PARENT
  ```

- If you want to set width that is relative to the screen size:

```
// or use Stanford lib's getScreenWidth/Height methods
Display display = getWindowManager().getDefaultDisplay();
Point size = new Point();
display.getSize(size);
int screenWidth  = size.x;
int screenHeight = size.y;
LinearLayout.LayoutParams params = new LinearLayout.LayoutParams(
        screenWidth / 2,      // width = half of screen
        screenHeight / 2);    // height = half of screen
```

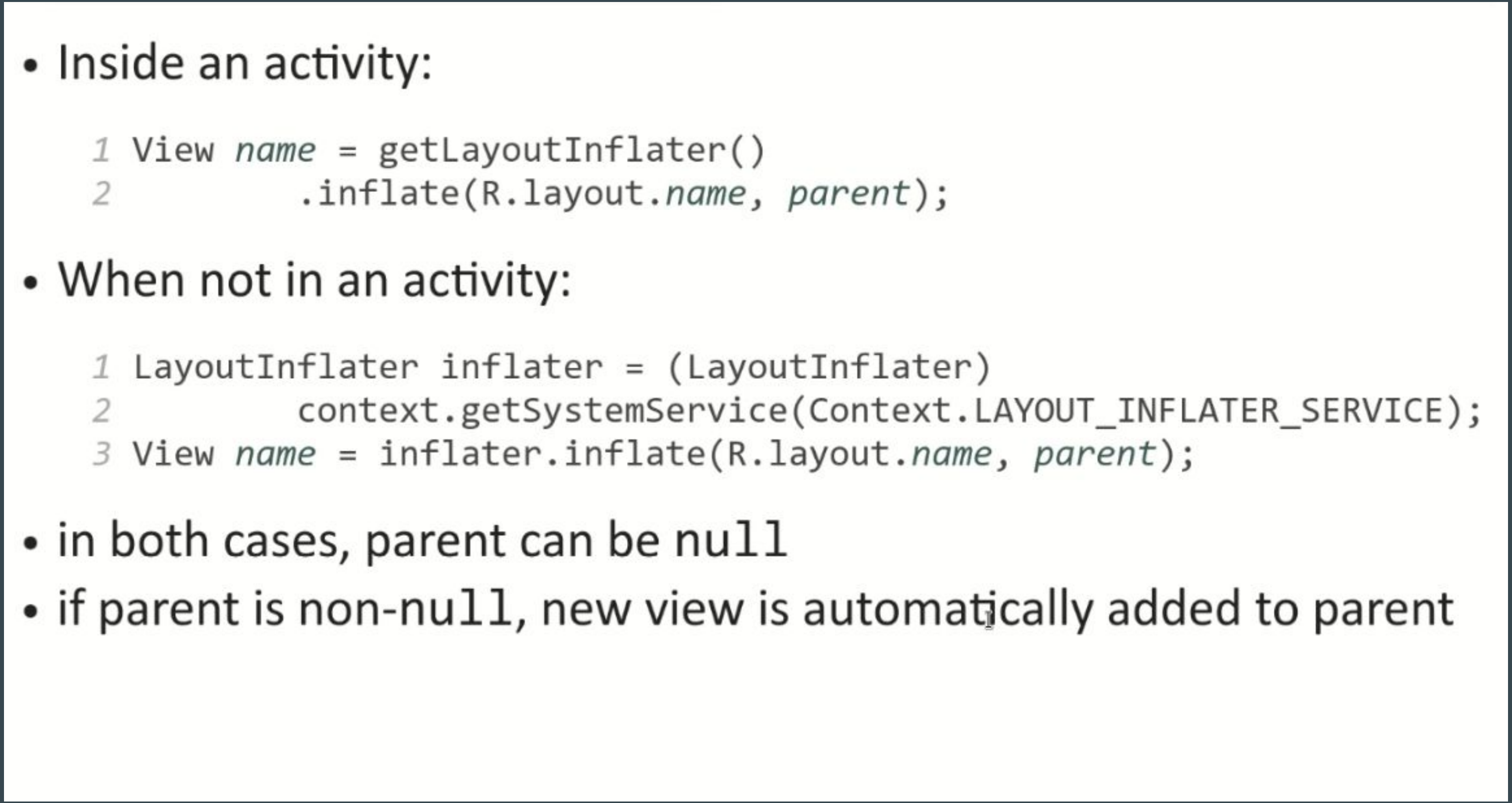
Layout Inflater



- **layout inflater**: Converts layout XML into Java widget objects.
  - Manual creation of widgets works, but it is pretty painful if you are creating a lot of them, or a complex nested structure of widgets.
  - A layout inflater lets you specify an entire chunk of layout, perhaps a complex subcomponent, as XML and then load it in Java as needed.
  - Similar to a fragment but without its own events and lifecycle.

```
<LinearLayout ...>
    <ImageView ... />
    <TextView ... />
</LinearLayout>
```

Using the Layout Inflater



- Inside an activity:

```java
View name = getLayoutInflater()
        .inflate(R.layout.name, parent);
```

- When not in an activity:

```java
LayoutInflater inflater = (LayoutInflater)
        context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
View name = inflater.inflate(R.layout.name, parent);
```

- in both cases, parent can be `null`
- if parent is non-`null`, new view is automatically added to parent

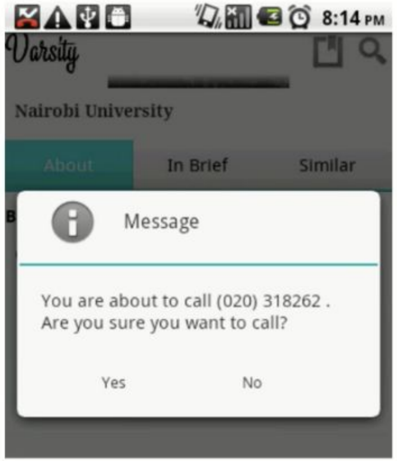
Demo

Demo App (Flags?)

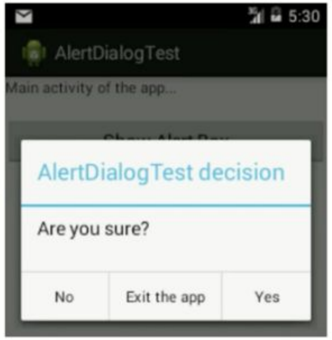
Dialogs

Motivation

- **dialog**: A pop-up UI that interrupts your activity.
  - not a different activity itself; sits on top of the activity
  - meant to briefly display information or ask for a bit of input
  - once the user is done interacting with the dialog, it closes, and app resumes activity it was on before



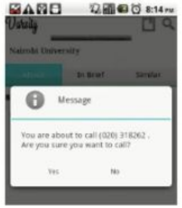
You are about to call (020) 318262 .
Are you sure you want to call?



Are you sure?

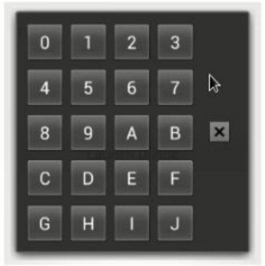
Types of Dialogs

- Android has a Dialog class with subclasses including:

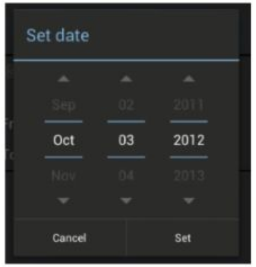


You are about to call (020) 318262 .
Are you sure you want to call?

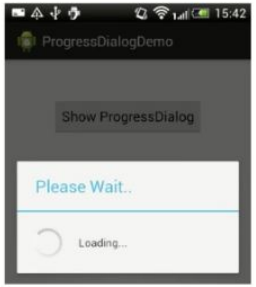
AlertDialog



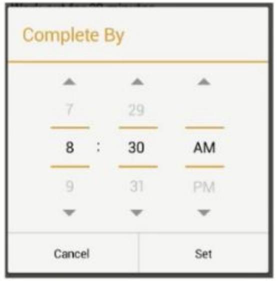
CharacterPickerDialog



DatePickerDialog

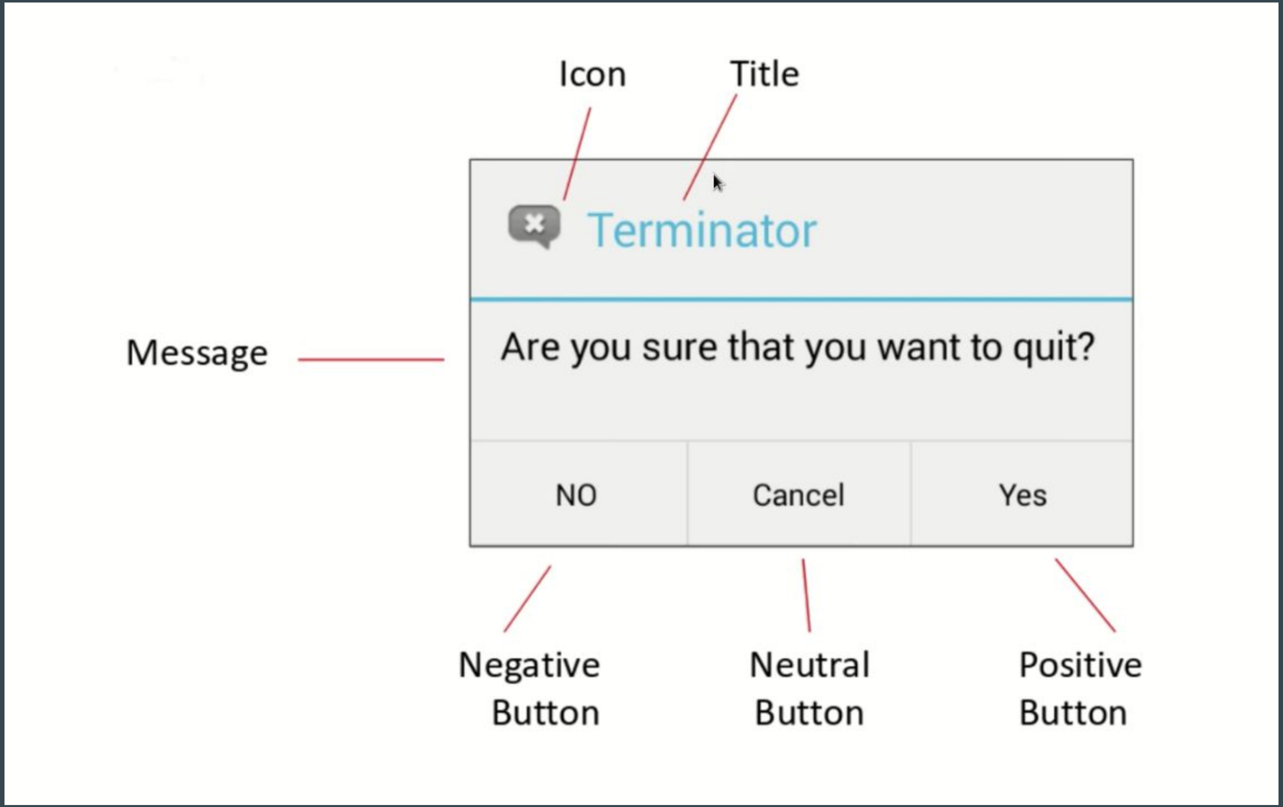


ProgressDialog

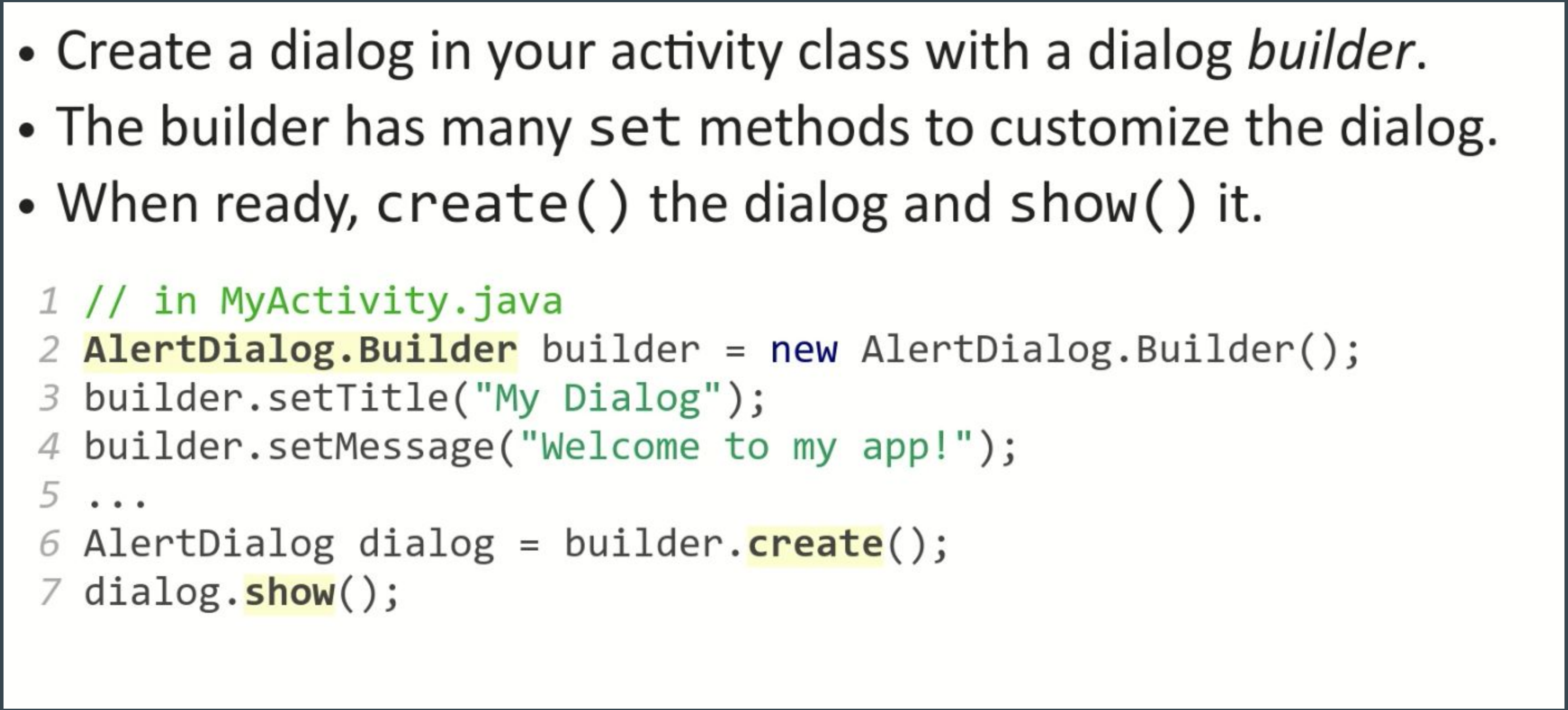


TimePickerDialog

Parts of a Dialog



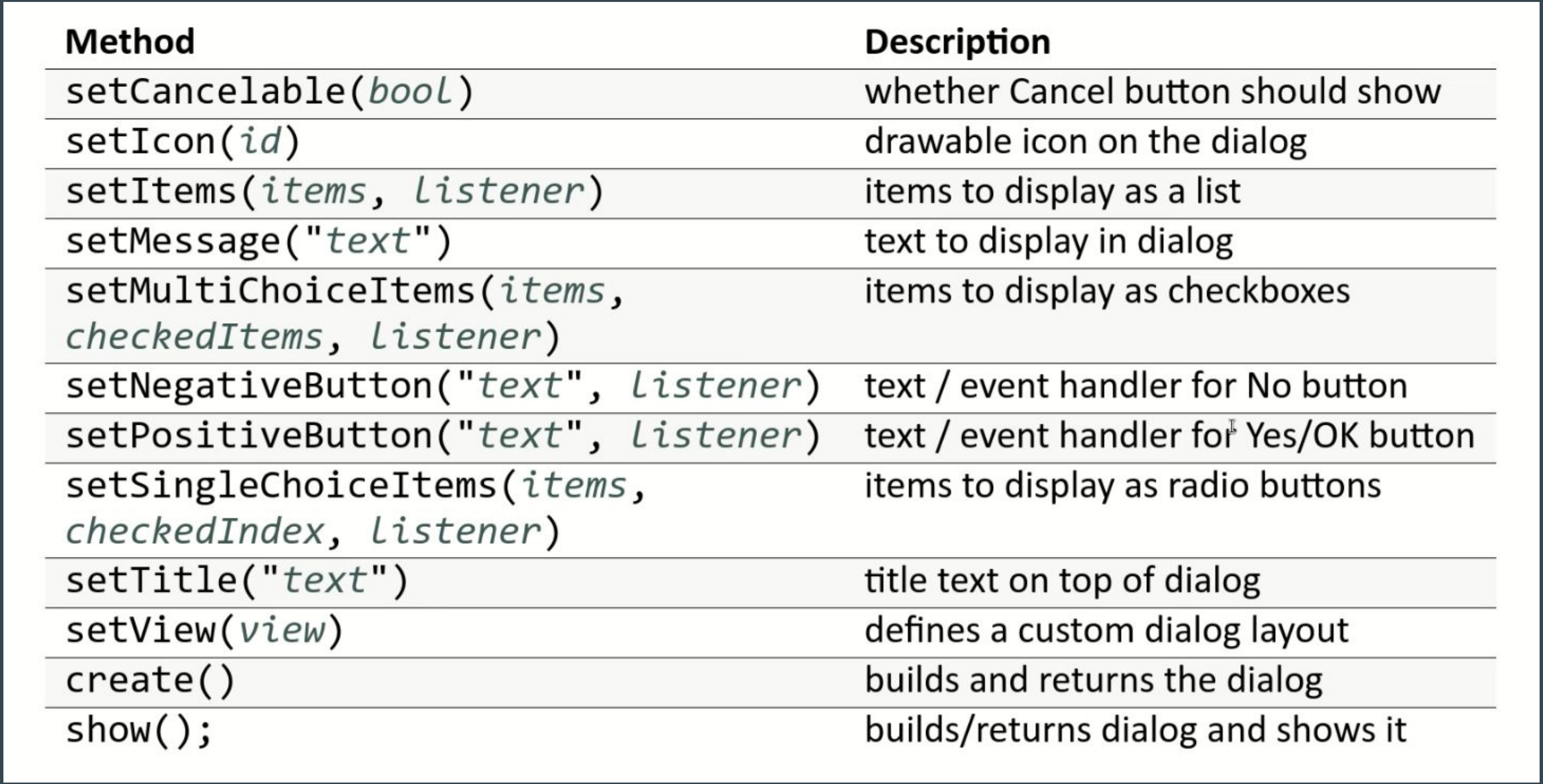
Building a Dialog



- Create a dialog in your activity class with a dialog *builder*.
- The builder has many `set` methods to customize the dialog.
- When ready, `create()` the dialog and `show()` it.

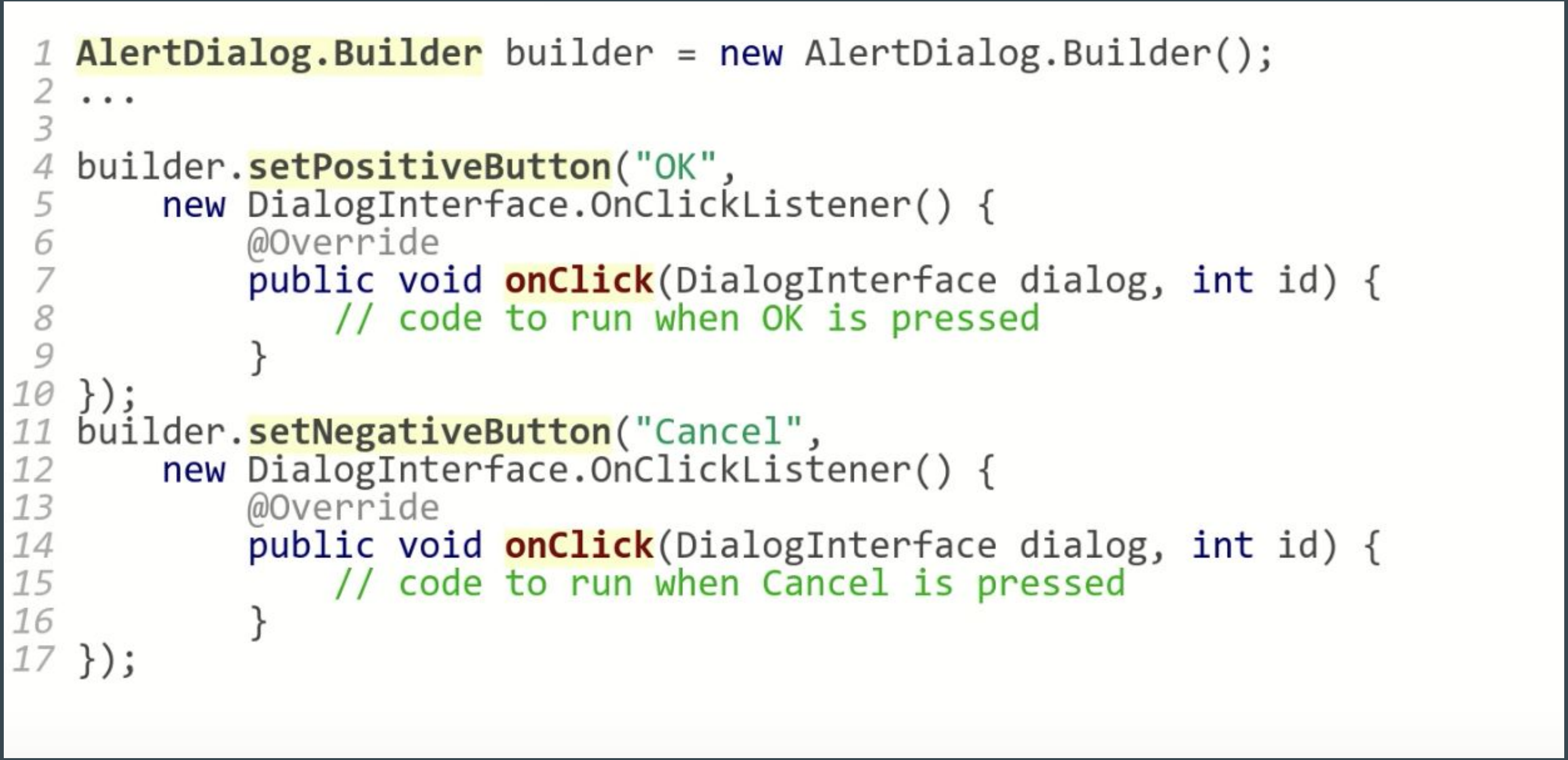
```java
// in MyActivity.java
AlertDialog.Builder builder = new AlertDialog.Builder();
builder.setTitle("My Dialog");
builder.setMessage("Welcome to my app!");
...
AlertDialog dialog = builder.create();
dialog.show();
```

Dialog builder methods



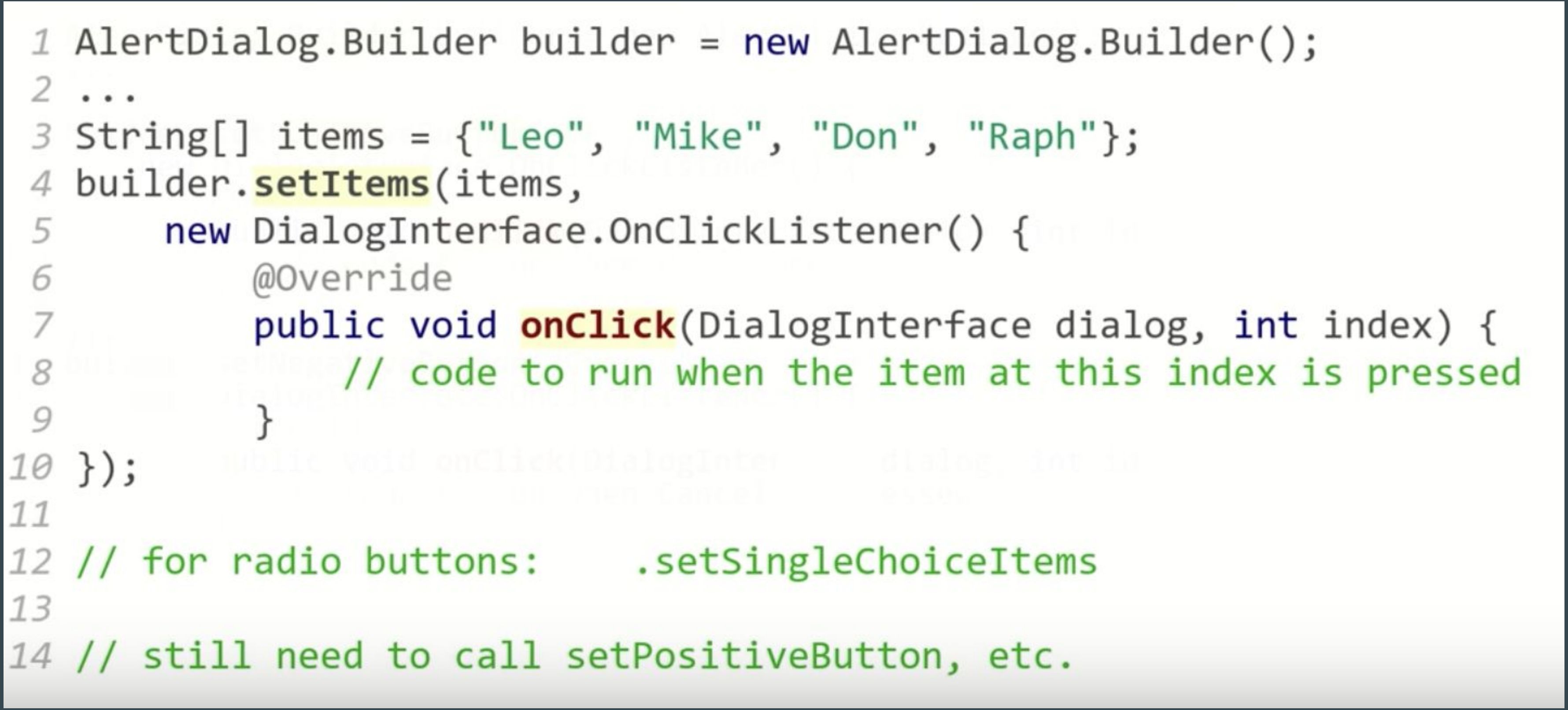
Method	Description
`setCancelable(bool)`	whether Cancel button should show
`setIcon(id)`	drawable icon on the dialog
`setItems(items, listener)`	items to display as a list
`setMessage("text")`	text to display in dialog
`setMultiChoiceItems(items, checkedItems, listener)`	items to display as checkboxes
`setNegativeButton("text", listener)`	text / event handler for No button
`setPositiveButton("text", listener)`	text / event handler for Yes/OK button
`setSingleChoiceItems(items, checkedIndex, listener)`	items to display as radio buttons
`setTitle("text")`	title text on top of dialog
`setView(view)`	defines a custom dialog layout
`create()`	builds and returns the dialog
`show();`	builds/returns dialog and shows it

Attaching Listeners



```java
AlertDialog.Builder builder = new AlertDialog.Builder();
...

builder.setPositiveButton("OK",
    new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int id) {
            // code to run when OK is pressed
        }
});
builder.setNegativeButton("Cancel",
    new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int id) {
            // code to run when Cancel is pressed
        }
});
```

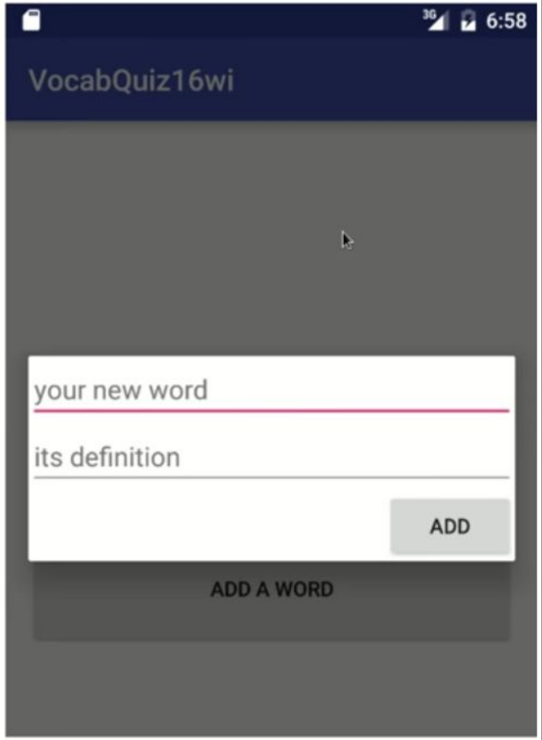
Listening to a list of items



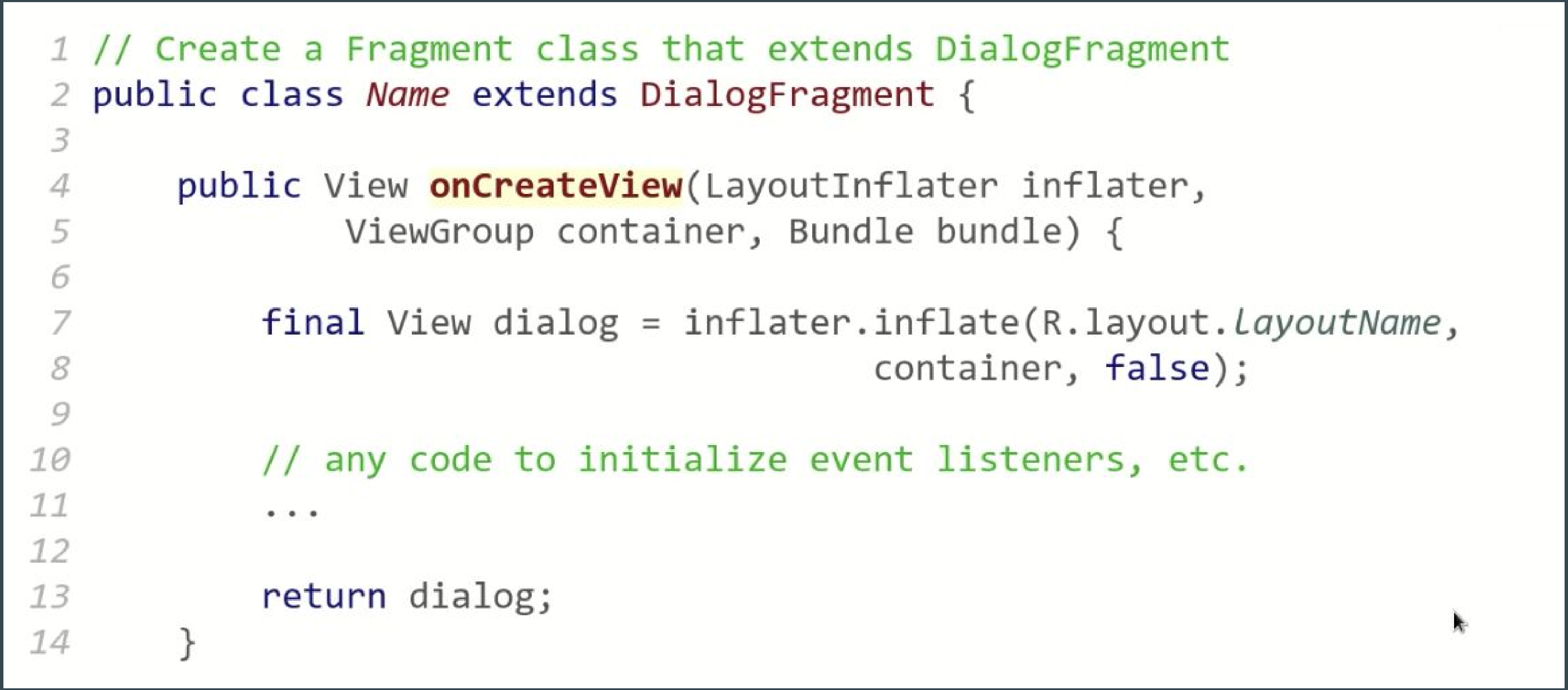
```java
AlertDialog.Builder builder = new AlertDialog.Builder();
...
String[] items = {"Leo", "Mike", "Don", "Raph"};
builder.setItems(items,
    new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int index) {
            // code to run when the item at this index is pressed
        }
});

// for radio buttons:    .setSingleChoiceItems

// still need to call setPositiveButton, etc.
```

Custom Dialogs

- `AlertDialog` is useful but very limited.
- To make your own custom dialog with its own widgets, layout, and behavior:
  1. create a new **fragment** that extends DialogFragment (**.java**)
  2. create a **layout** for it (**.xml**)
  3. write the Java/XML code to create the fragment's UI and handle its **events**
  4. write the Java code in your **activity** to launch the dialog

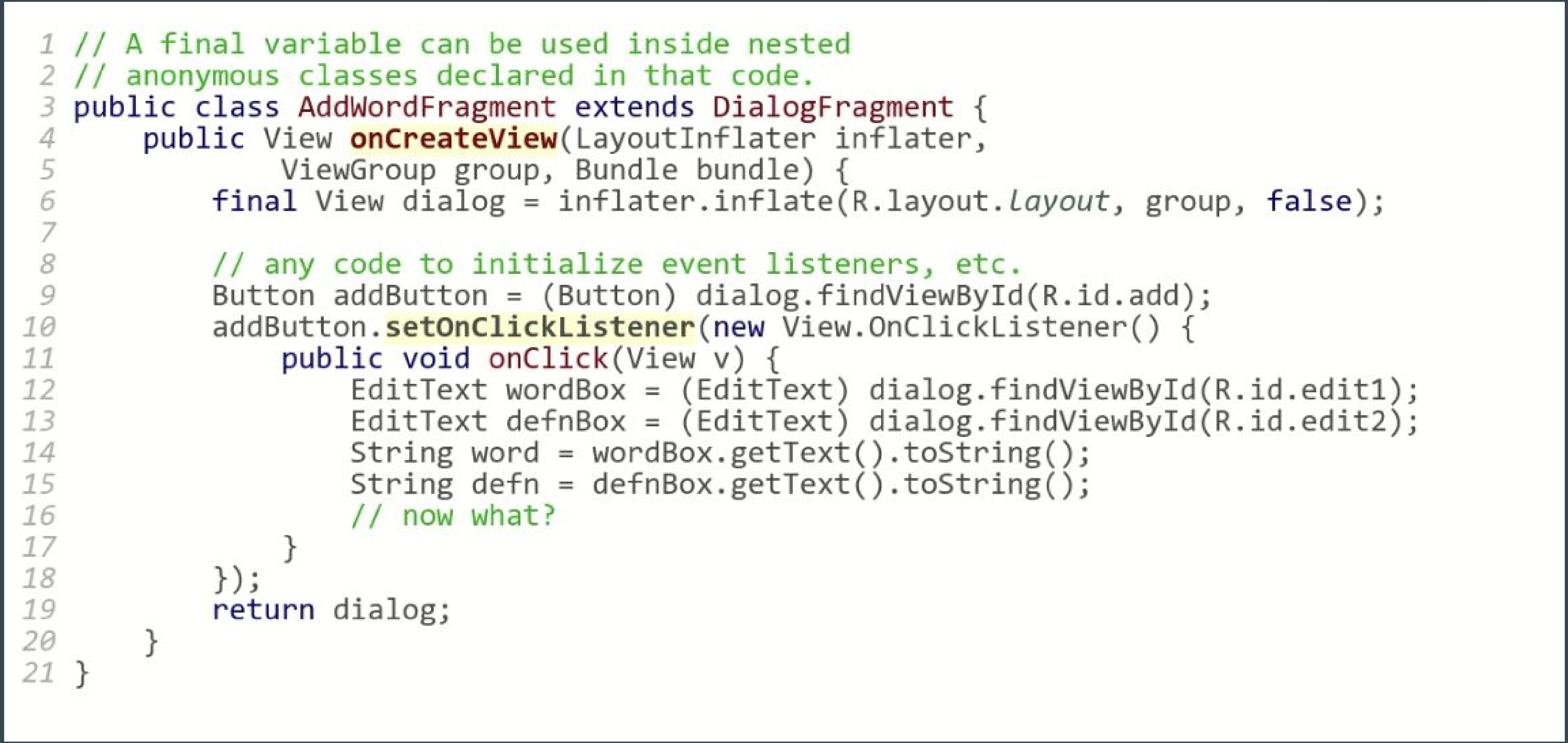


Dialog Fragment



```java
// Create a Fragment class that extends DialogFragment
public class Name extends DialogFragment {

    public View onCreateView(LayoutInflater inflater,
            ViewGroup container, Bundle bundle) {

        final View dialog = inflater.inflate(R.layout.LayoutName,
                                             container, false);

        // any code to initialize event listeners, etc.
        ...

        return dialog;
    }
```

Why final?



```java
// A final variable can be used inside nested
// anonymous classes declared in that code.
public class AddWordFragment extends DialogFragment {
    public View onCreateView(LayoutInflater inflater,
            ViewGroup group, Bundle bundle) {
        final View dialog = inflater.inflate(R.layout.layout, group, false);

        // any code to initialize event listeners, etc.
        Button addButton = (Button) dialog.findViewById(R.id.add);
        addButton.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                EditText wordBox = (EditText) dialog.findViewById(R.id.edit1);
                EditText defnBox = (EditText) dialog.findViewById(R.id.edit2);
                String word = wordBox.getText().toString();
                String defn = defnBox.getText().toString();
                // now what?
            }
        });
        return dialog;
    }
}
```