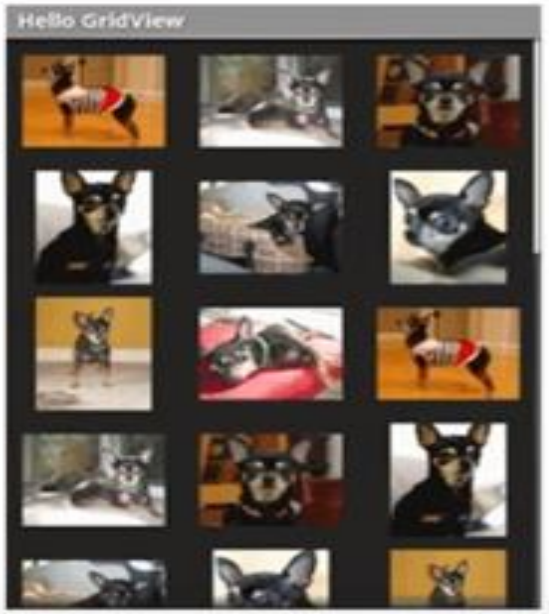
Generating a UI at runtime

- 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.



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.

```
1 // inside an activity class
2 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.

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

```
1 // MainActivity.java
2 TextView tv = new TextView(this);
3 LinearLayout layout = (LinearLayout) findViewById(R.id.mainlayout);
4 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.

```
1 <!-- activity_main.xml -->
2 <TextView
3     android:id="@+id/mymessage"
4     android:text="Hello there!"
5     android:textSize="20dp"
6     android:textStyle="bold"
7     android:layout_width="wrap_content"
8     android:layout_height="wrap_content" />
```

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`:

```
1 LinearLayout.LayoutParams params =
2     new LinearLayout.LayoutParams(
3         ViewGroup.LayoutParams.MATCH_PARENT,     // width
4         ViewGroup.LayoutParams.WRAP_CONTENT);    // height
5 params.weight  = 1;
6 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:

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

Creating the Example Project in Android Studio

Adding Views to an Activity

- The onCreate() method is currently designed to use a resource layout file for the user interface. Begin, therefore, by deleting this line from the method:

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

- The next modification to the onCreate() method is to write some Java code to add a RelativeLayout object with a single Button view child to the activity.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    Button myButton = new Button(this);
    RelativeLayout myLayout = new RelativeLayout(this);
    myLayout.addView(myButton);
    setContentView(myLayout);
}
```

Setting View Properties

- Set the background of the RelativeLayout view to be blue and the Button view to display text that reads “Press Me”.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    Button myButton = new Button(this);
    myButton.setText("Press Me");
    myButton.setBackgroundColor(Color.YELLOW);

    RelativeLayout myLayout = new RelativeLayout(this);
    myLayout.setBackgroundColor(Color.BLUE);

    myLayout.addView(myButton);
    setContentView(myLayout);
}
```

Adding Layout Parameters and Rules

- In order to instruct the layout view to place the button in a different location, in this case centered both horizontally and vertically, it will be necessary to create a LayoutParams object and initialize it with the appropriate values.

```
RelativeLayout.LayoutParams buttonParams =
                new RelativeLayout.LayoutParams(
                    RelativeLayout.LayoutParams.WRAP_CONTENT,
                    RelativeLayout.LayoutParams.WRAP_CONTENT);
```

- The next step is to add some additional rules to the parameters to instruct the layout parent to center the button vertically and horizontally.

```
buttonParams.addRule(RelativeLayout.CENTER_HORIZONTAL);
buttonParams.addRule(RelativeLayout.CENTER_VERTICAL);
```

- Last step, need to pass the LayoutParams object through as an argument when the child view is added to the parent.

```
myLayout.addView(myButton, buttonParams);
```

or:

```
myButton.setLayoutParams(buttonParams);
```

Complete code

```
@Override
protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        Button myButton = new Button(this);
        myButton.setText("Press me");
        myButton.setBackgroundColor(Color.YELLOW);

      RelativeLayout myLayout = new RelativeLayout(this);
      myLayout.setBackgroundColor(Color.BLUE);

      RelativeLayout.LayoutParams buttonParams =
                new RelativeLayout.LayoutParams(
                    RelativeLayout.LayoutParams.WRAP_CONTENT,
                    RelativeLayout.LayoutParams.WRAP_CONTENT);

      buttonParams.addRule(RelativeLayout.CENTER_HORIZONTAL);
      buttonParams.addRule(RelativeLayout.CENTER_VERTICAL);

      myLayout.addView(myButton, buttonParams);
      setContentView(myLayout);
}
```

Output