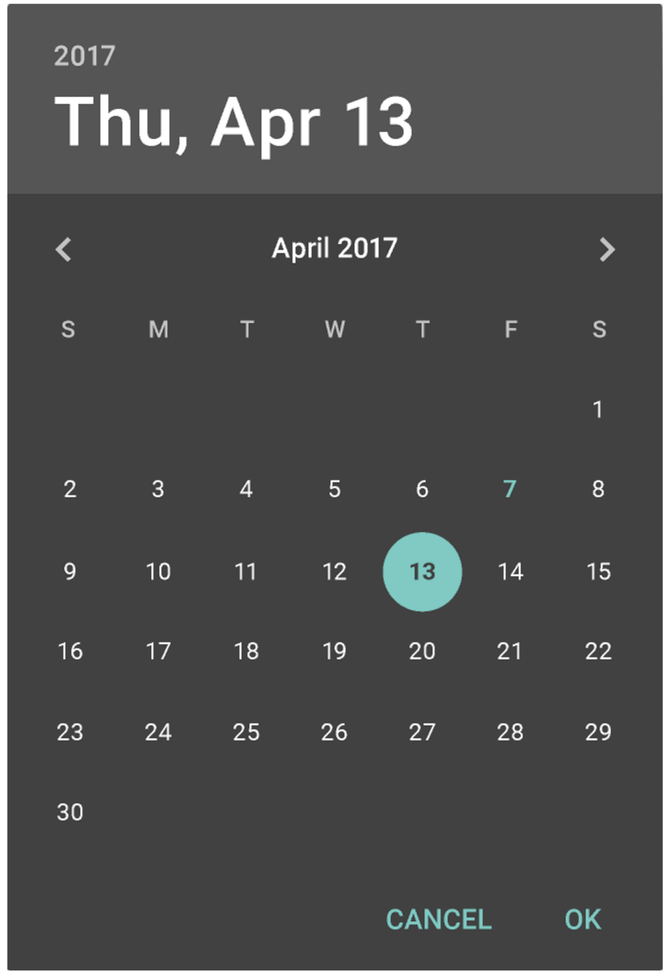


MOBILE APPLICATION DEVELOPMENT

ANDROID (2017)

LECTURE 05: CUSTOM CONTROLS

CUSTOM CONTROLS

- Subclasses of View, generally used to control values.
- Override the `onDraw()` method to customize display.
- Handle their own measurement with `onMeasure()`.
- Communicate value changes via listener interfaces.

public class

ImageView

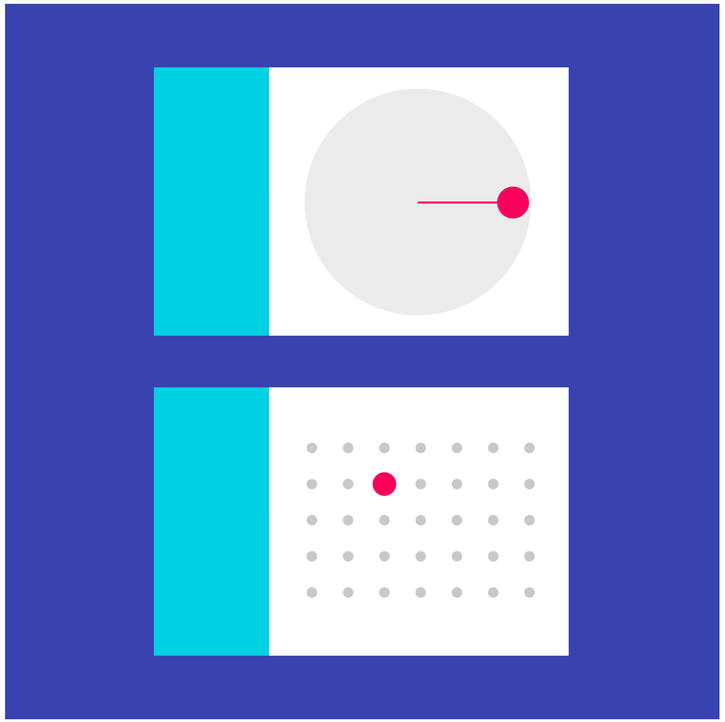
extends View

java.lang.Object
↳ android.view.View
↳ android.widget.ImageView

▸ Known Direct Subclasses
ImageButton, QuickContactBadge

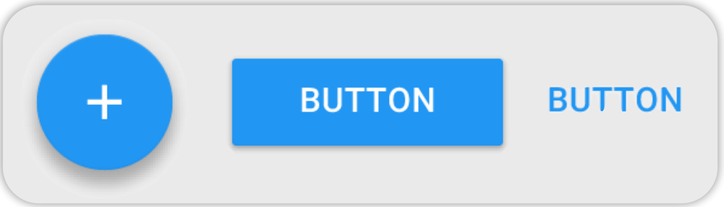
▸ Known Indirect Subclasses
ZoomButton

COMMON CONTROL ELEMENTS



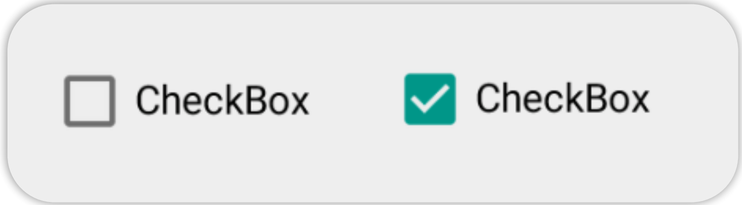
- Controls have most of the following properties:
  - Property: The property the control represents visually.
  - Principal Event: An event that triggers a value update.
  - Listener Setter: The way you attach a listener to the control.
  - Listener Event(s): The method(s) called on the listener when the control sends updates.

COMMON CONTROLS: BUTTON



- Buttons represent a title and respond to touch events:
  - Property: `title`
  - Principal Event: `onClick`
  - Listener Setter: `setOnClickListener()`
  - Listener Event(s): `onClick()`

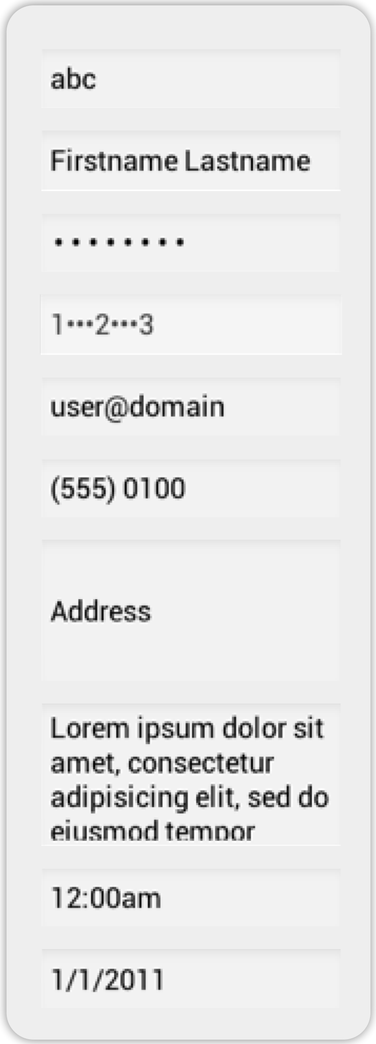
COMMON CONTROLS: CHECK BOXES



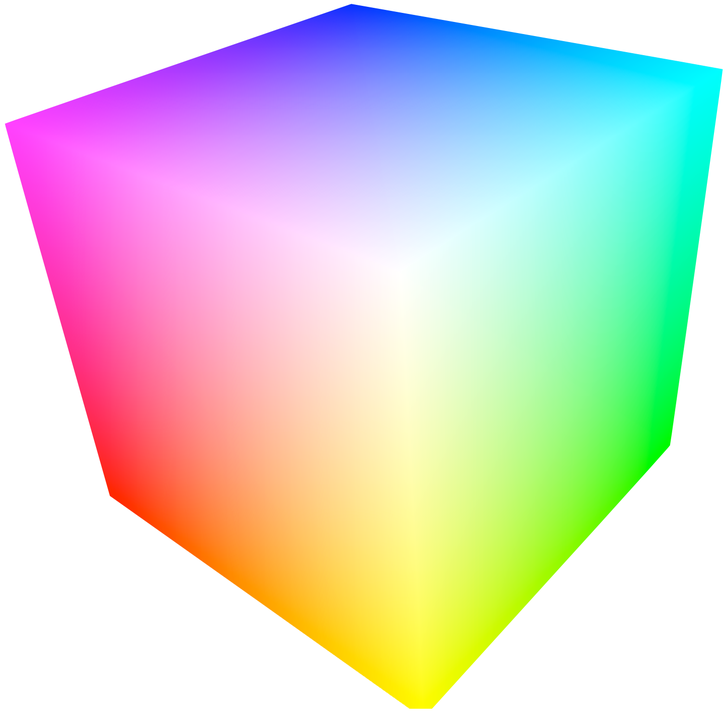
- CheckBoxes represent a check state and respond to touch events:
  - Property: `checked`
  - Principal Event: `onChecked`
  - Listener Setter: `setOnCheckChangedListener()`
  - Listener Event(s): `onCheckedChanged()`

COMMON CONTROLS: TEXT FIELDS

- Text fields (TextView/EditText/Etc.) represent text and handle editing events:
  - Property: `text`
  - Principal Event: `onKey`
  - Listener Setter: `setOnKeyListener()`
  - Listener Event(s): `onKey()`

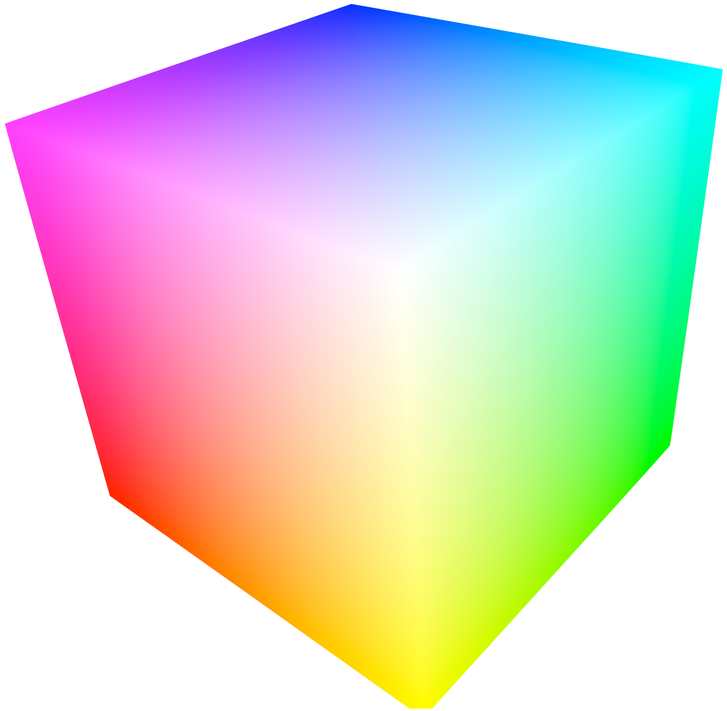


ONDRAW METHOD



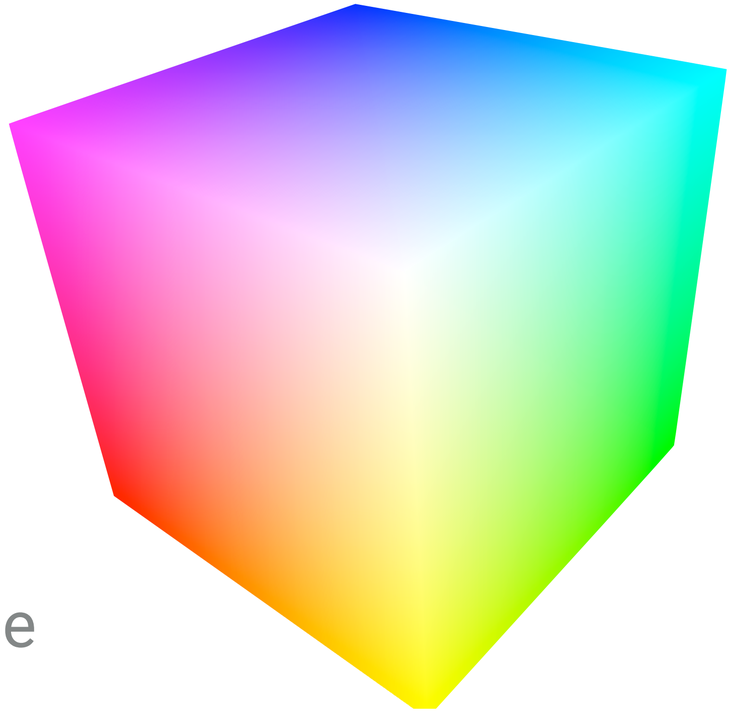
- Provides a Canvas object which is used for drawing.
- Information is cached by the system after being drawn.
  - If content doesn't change, no need to re-draw.
  - Allows cheap compositing with other Views/controls.
- Should be implemented to respect size, padding, and gravity of the View.
  - Don't (usually) draw inside padding.
  - If extra space is present, use gravity to determine where to draw content.

DRAW COMMANDS



- Within `onDraw()` there are a number of ways to draw.
  - Use `drawARGB()` to fill the Canvas with color.
  - Use `drawRect()` to draw rectangles.
  - Use `drawOval() / drawArc()` to draw ovals or arcs.
  - Use `drawPoint() / drawPoints()` to draw points.
  - Use `drawLine() / drawLines() / drawPath()` to draw lines or poly-lines.
  - Use `drawBitmap() / drawPicture()` to draw images within rectangles.
  - Use `drawText() / drawTextOnPath()` to draw text, linearly or on a specified path.

CANVAS OBJECT



- Supports advanced drawing features.
- Allows clipping of the draw area using a path.
- Allows the drawing coordinate system to be altered.
- Provides a stack of matrix and clip settings, which may be saved and restored between draw commands.

EXAMPLE: ACQUIRE