MEASURE.
LAYOUT.
DRAW.
REPEAT.

HUYEN TUE DAO
@QUEENCODEMONKEY

MEASURE.
LAYOUT.
DRAW.
REPEAT.

CUSTOM VIEWS.
WHY?

CUSTOM LOOK/BEHAVIOR

REUSABILITY/MODULARITY

PERFORMANCE

MEASURE.
LAYOUT.
DRAW.
REPEAT.

CUSTOM VIEWS.
WHY NOT?

TIME-CONSUMING

DIFFICULT

DRAWING/INTERACTIONS/
A11Y → ON YOUR OWN

MEASURE.
LAYOUT.
DRAW.
REPEAT.

HOW ANDROID DRAWS VIEWS

HOW TO GO CUSTOM

DRAW

MEASURE

VIEW GROUPS

LAYOUT

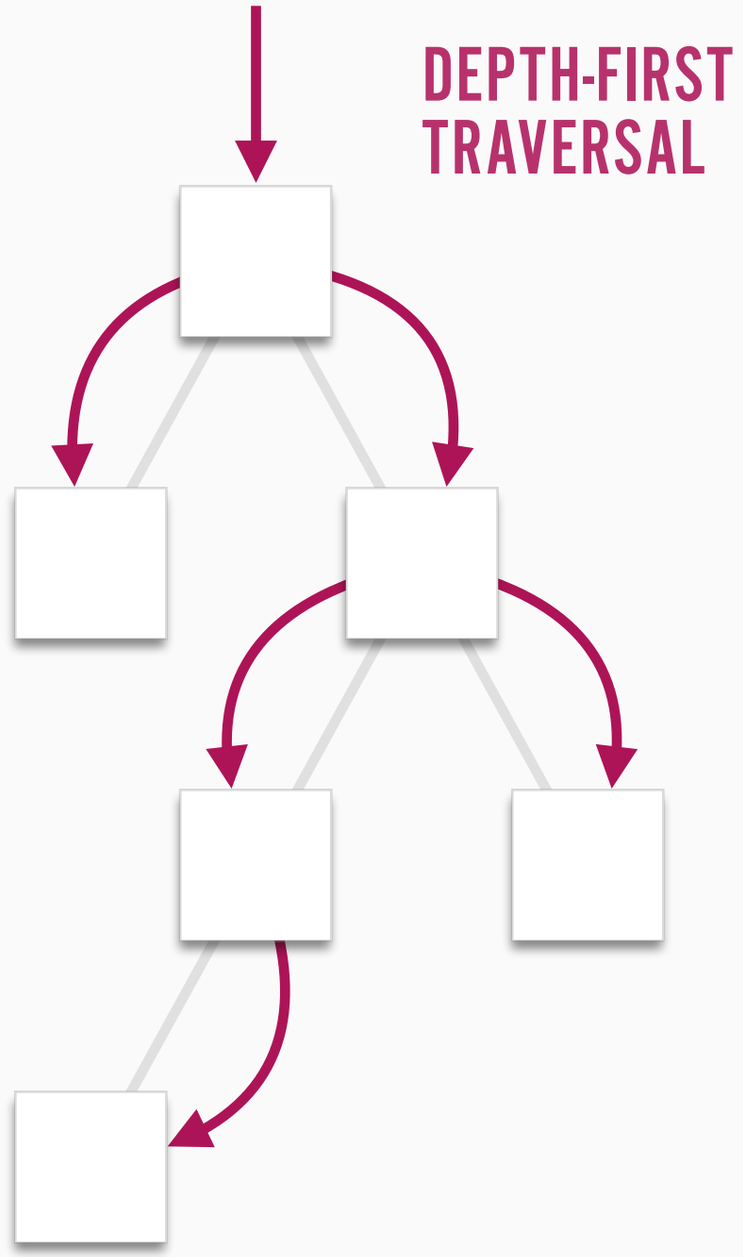
HOW ANDROID DRAWS VIEWS

INFLATION/INSTANTIATION

MEASURE
`onMeasure()`

LAYOUT
`onLayout()`

DRAW
`onDraw()`

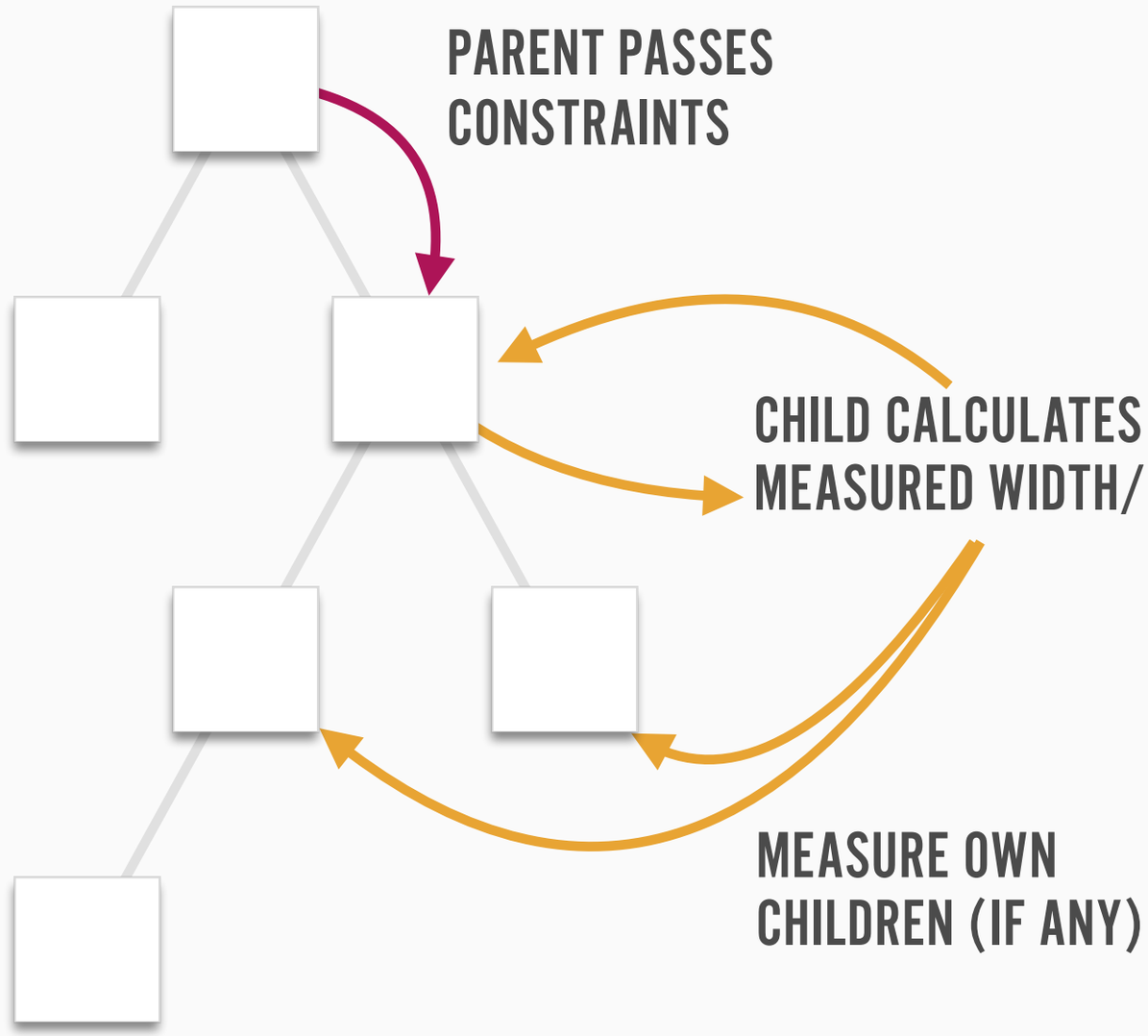


HOW ANDROID DRAWS VIEWS

MEASURE
`onMeasure()`

LAYOUT
`onLayout()`

DRAW
`onDraw()`

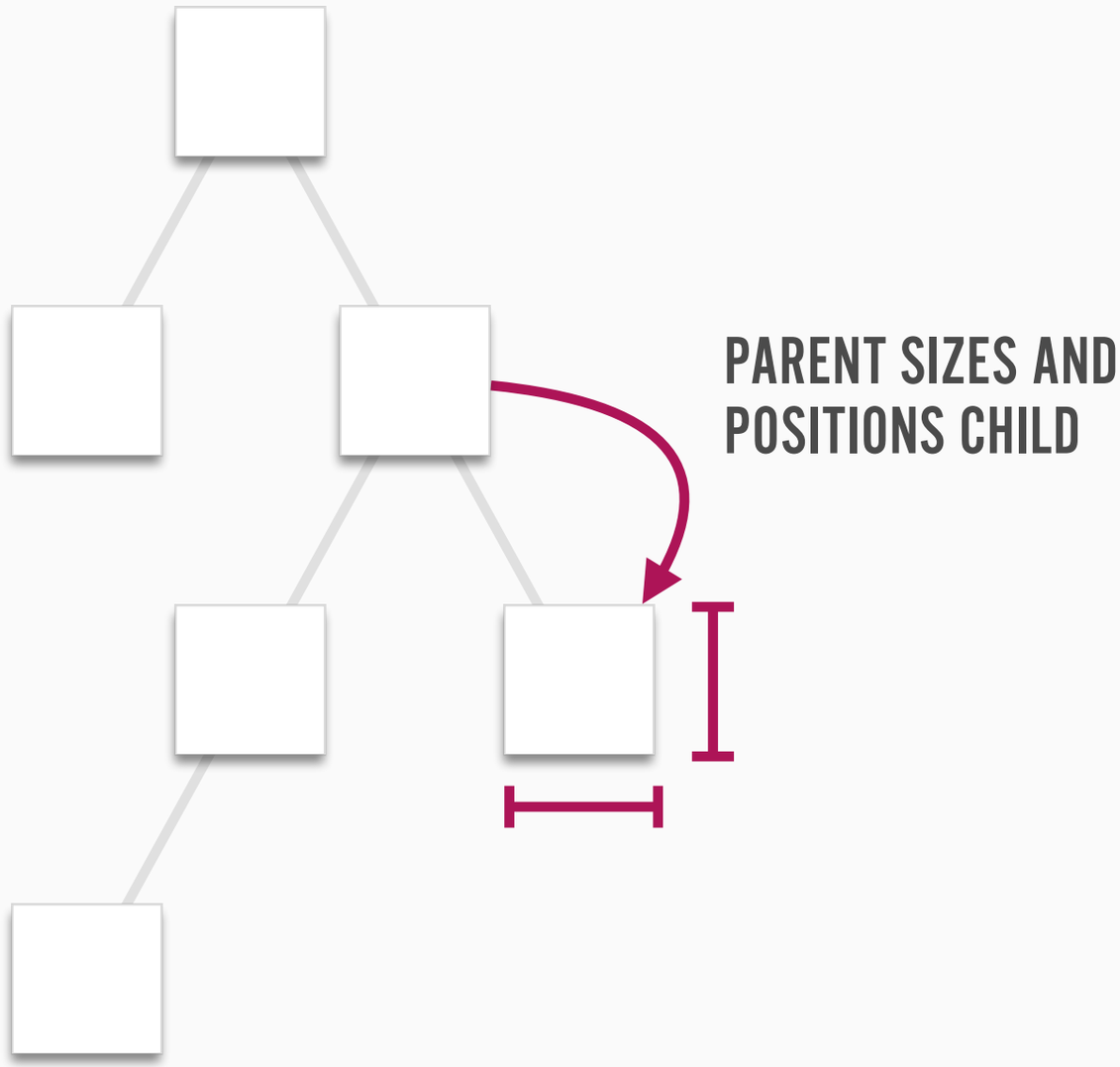


HOW ANDROID DRAWS VIEWS

MEASURE
onMeasure()

LAYOUT
onLayout()

DRAW
onDraw()

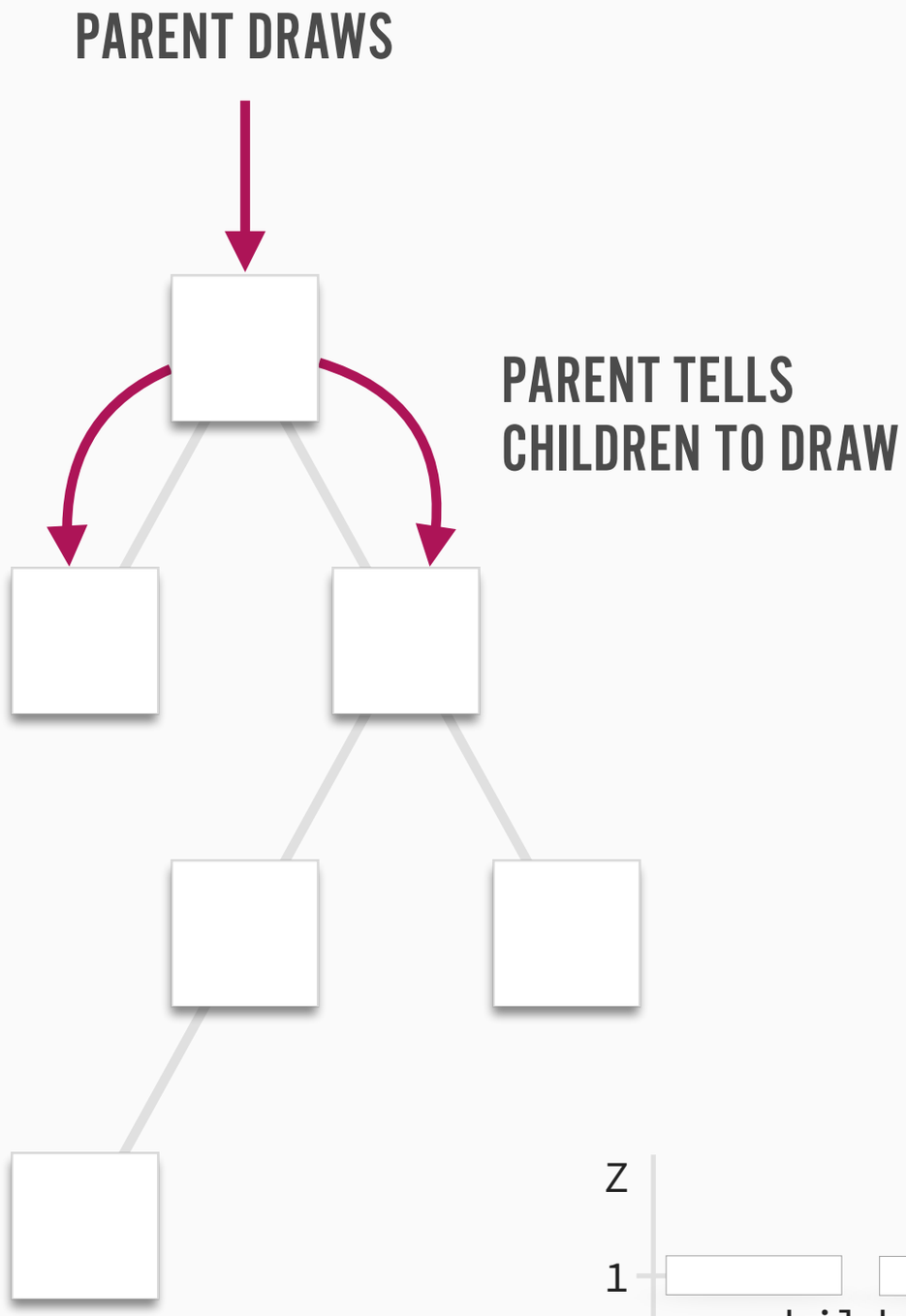


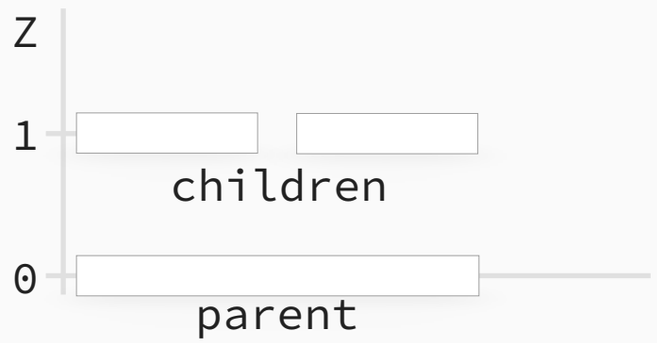
HOW ANDROID DRAWS VIEWS

MEASURE
onMeasure()

LAYOUT
onLayout()

DRAW
onDraw()





DIFFERENT WAYS TO GO GUSTOM

	VIEW	VIEWGROUP
EXTEND EXISTING	CUSTOM EXTENDED*	COMPOSITE
	TWEAK BEHAVIOR	MODULAR UI
	ARGUABLY SIMPLEST	LAYOUT REUSE
EXTEND BASE	FLAT CUSTOM	CUSTOM COMPOSITE
	CUSTOM DRAWING/BEHAVIOR	CUSTOM LAYOUT LOGIC
	PERFORMANCE	PERFORMANCE

WHAT DO YOU NEED TO IMPLEMENT?

VIEW

MEASURE `onMeasure()`	NO	BUT YOU SHOULD REUSABILITY/FLEXIBILITY
LAYOUT `onLayout()`	NO	VIEW HAS NO CHILDREN
DRAW `onDraw()`	YES	REQUIRED FOR VIEW TO APPEAR

WHAT WE WILL BUILD
GREAT, HUH?

```java
package randomlytyping.widget;

/**
 * An interface for a view implementing a tally counter.
 */
public interface TallyCounter {

    /**
     * Reset the counter.
     */
    void reset();

    /**
     * Increment the counter.
     */
    void increment();

    /**
     * @return The current count of the counter.
     */
    int getCount();

    /**
     * Set the counter value.
     */
    void setCount(int count);

}
```

VIEW CONSTRUCTORS

A DEEP DIVE INTO ANDROID VIEW CONSTRUCTORS
@DANLEW42

CREATE NEW FROM CODE

```
View(Context context)
```

CREATE FROM XML

```
View(Context context, AttributeSet attrs)
```

CREATE FROM XML WITH A STYLE FROM A THEME ATTRIBUTE

```
View(Context context, AttributeSet attrs, int defStyleAttr)
```

CREATE FROM XML WITH A STYLE FROM A THEME ATTRIBUTE OR STYLE RESOURCE

```
View(Context context, AttributeSet attrs, int defStyleAttr,
int defStyleRes)
```

```java
    /**
     * Simple constructor to use when creating a view using the {@code new} operator.
     *
     * @param context The Context the view is running in, through which it can
     *                access the current theme, resources, etc.
     */
    public TallyCounterView(Context context) { this(context, null); }

    /**
     * Constructor that is called when inflating a view from XML. This is called
     * when a view is being constructed from an XML file, supplying attributes
     * that were specified in the XML file.
     *
     * @param context The Context the view is running in, through which it can
     *                access the current theme, resources, etc.
     * @param attrs   The attributes of the XML tag that is inflating the view.
     */
    public TallyCounterView(Context context, AttributeSet attrs) {
        super(context, attrs);

        // Set up paints for canvas drawing.
        backgroundPaint = new Paint(Paint.ANTI_ALIAS_FLAG);
        backgroundPaint.setColor(ContextCompat.getColor(context, R.color.colorPrimary));
        linePaint = new Paint(Paint.ANTI_ALIAS_FLAG);
        linePaint.setColor(ContextCompat.getColor(context, R.color.colorAccent));
        linePaint.setStrokeWidth(1f);
        numberPaint = new TextPaint(Paint.ANTI_ALIAS_FLAG);
        numberPaint.setColor(ContextCompat.getColor(context, android.R.color.white));
        // Set the number text size to be 64sp.
        // Translate 64sp
        numberPaint.setTextSize(Math.round(64f * getResources().getDisplayMetrics().scaledDensity));

        // Allocate objects needed for canvas drawing here.
        backgroundRect = new RectF();

        // Initialize drawing measurements.
        cornerRadius = Math.round(2f * getResources().getDisplayMetrics().density);

        // Do initial count setup.
        setCount(0);
    }
```

```
    //
    // View overrides
    //

    @Override
    protected void onDraw(Canvas canvas) {

        // Grab canvas dimensions.
        final int canvasWidth = canvas.getWidth();
        final int canvasHeight = canvas.getHeight();

        // Calculate horizontal center.
        final float centerX = canvasWidth * 0.5f;

        // Draw the background.
        backgroundRect.set(0f, 0f, canvasWidth, canvasHeight);
        canvas.drawRoundRect(backgroundRect, cornerRadius, cornerRadius, backgroundPaint);

        // Draw baseline.
        final float baselineY = Math.round(canvasHeight * 0.6f);
        canvas.drawLine(0, baselineY, canvasWidth, baselineY, linePaint);

        // Draw text.

        // Measure the width of text to display.
        final float textWidth = numberPaint.measureText(displayedCount);
        // Figure out an x-coordinate that will center the text in the canvas.
        final float textX = Math.round(centerX - textWidth * 0.5f);
        // Draw.
        canvas.drawText(displayedCount, textX, baselineY, numberPaint);
    }
}
```


11:09
CLICK
0000

REFLECTING STATE CHANGES

MARK A VIEW AS DIRTY (NEEDS A RE-DRAW)

```
invalidate()
```

RE-DRAWN “AT SOME POINT IN THE FUTURE”

BE MORE PRECISE:

```
invalidate(int l, int t, int r, int b)
```

```
    //
    // TallyCounter interface
    //

    @Override
    public void reset() { setCount(0); }

    @Override
    public void increment() { setCount(count + 1); }

    @Override
    public void setCount(int count) {
        count = Math.min(count, MAX_COUNT);
        this.count = count;
        this.displayedCount = String.format(Locale.getDefault(), "%04d", count);
        invalidate();
    }

    @Override
    public int getCount() { return count; }

    //
    // View overrides
    //

    @Override
    protected void onDraw(Canvas canvas) {

        // Grab canvas dimensions.
        final int canvasWidth = canvas.getWidth();
        final int canvasHeight = canvas.getHeight();

        // Calculate horizontal center.
```


3G
11:11
CLICK
0000

THINGS TO REMEMBER WHEN DRAWING

NO ALLOCATIONS IN ONDRAW.
SERIOUSLY.

INVALIDATE ONLY WHEN YOU NEED.
INVALIDATE ONLY WHAT YOU NEED.

TEXT IS POSITIONED AT THE BASELINE

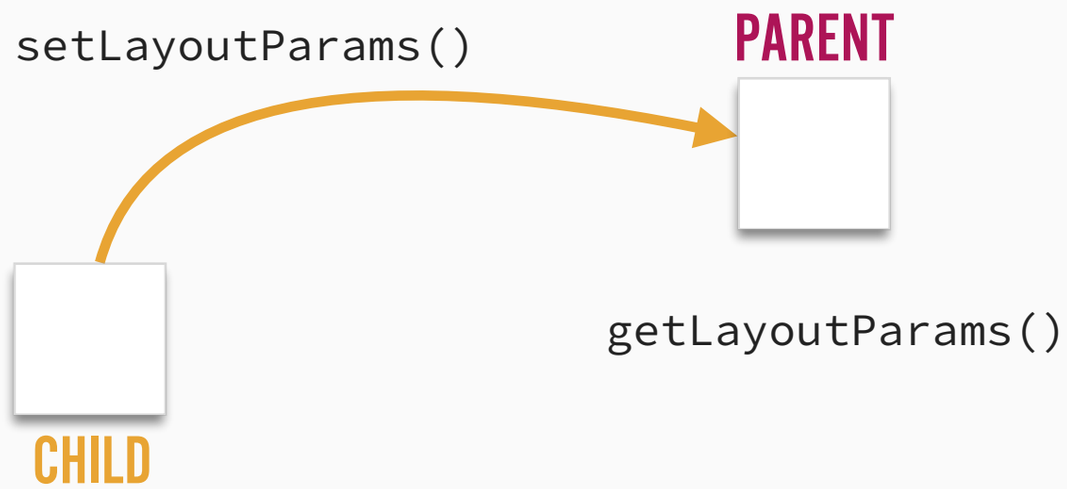
DRAW IN PX.
THINK IN DP AND SP.

WHERE'S THAT OTHER BUTTON?

```
<Button
    android:id="@+id/increment"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginEnd="64dp"
    android:layout_marginStart="64dp"
    android:layout_marginTop="48dp"
    android:text="Click" />

<randomlytyping.widget.TallyCounterView
    android:id="@+id/tally_counter"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginEnd="64dp"
    android:layout_marginStart="64dp"
    android:layout_marginTop="40dp" />

<Button
    android:id="@+id/reset"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginEnd="64dp"
    android:layout_marginStart="64dp"
    android:layout_marginTop="24dp"
    android:text="Reset" />
```

HOW MEASUREMENTS ARE MADE

1 Child defines `LayoutParams` in XML or Java



How a **child** tells its parent how it wants to be laid out.

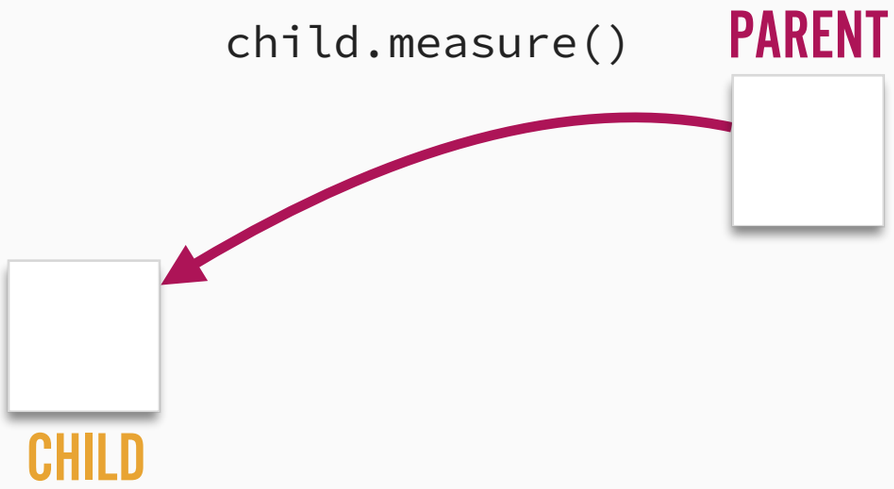
```
android:layout_width="match_parent"
android:layout_height="wrap_content"
```

`MarginLayoutParams`
`LinearLayout.LayoutParams`

HOW MEASUREMENTS ARE MADE

1 Child defines `LayoutParams` in XML or Java

2 Parent calculates `MeasureSpecs` and passes to `child.measure()`



How a **parent** communicates constraints to children

`int = { mode | size }`

`AT_MOST X` → Be any size up to X

`EXACTLY X` → Be exactly X

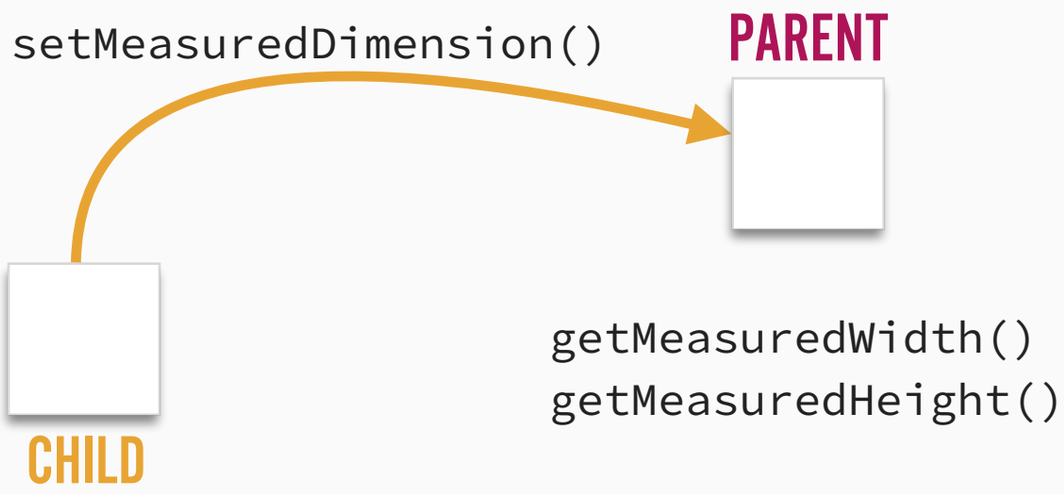
`UNSPECIFIED` → No constraints

HOW MEASUREMENTS ARE MADE

1 Child defines `LayoutParams` in XML or Java

2 Parent calculates `MeasureSpecs` and passes to `child.measure()`

3 Child calculates width/height; `setMeasuredDimension()`



Must call; otherwise → runtime exception.

```
resolveSize(int size, int measureSpec)
```

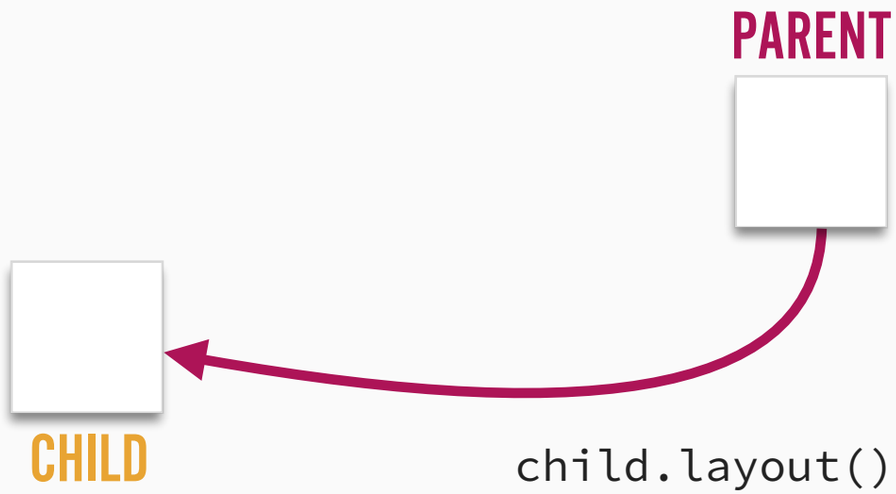
HOW MEASUREMENTS ARE MADE

1 Child defines `LayoutParams` in XML or Java

2 Parent calculates `MeasureSpecs` and passes to `child.measure()`

3 Child calculates width/height; `setMeasuredDimension()`

4 Parent calls `child.layout()`; final child size/position.



`getWidth()`
`getHeight()`

```java
    /**
     * Reconcile a desired size for the view contents with a {@link android.view.View.MeasureSpec}
     * constraint passed by the parent.
     *
     * This is a simplified version of {@link View#resolveSize(int, int)}
     *
     * @param contentSize Size of the view's contents.
     * @param measureSpec A {@link android.view.View.MeasureSpec} passed by the parent.
     * @return A size that best fits {@code contentSize} while respecting the parent's constraints.
     */
    private int reconcileSize(int contentSize, int measureSpec) {
        final int mode = MeasureSpec.getMode(measureSpec);
        final int specSize = MeasureSpec.getSize(measureSpec);
        switch (mode) {
            case MeasureSpec.EXACTLY:
                return specSize;
            case MeasureSpec.AT_MOST:
                if (contentSize < specSize) {
                    return contentSize;
                } else {
                    return specSize;
                }
            case MeasureSpec.UNSPECIFIED:
            default:
                return contentSize;
        }
    }

    //
    // View overrides
    //

    @Override
    protected void onMeasure(int widthMeasureSpec, int heightMeasureSpec) {
        final Paint.FontMetrics fontMetrics = numberPaint.getFontMetrics();
```

IRL USE `resolveSize`

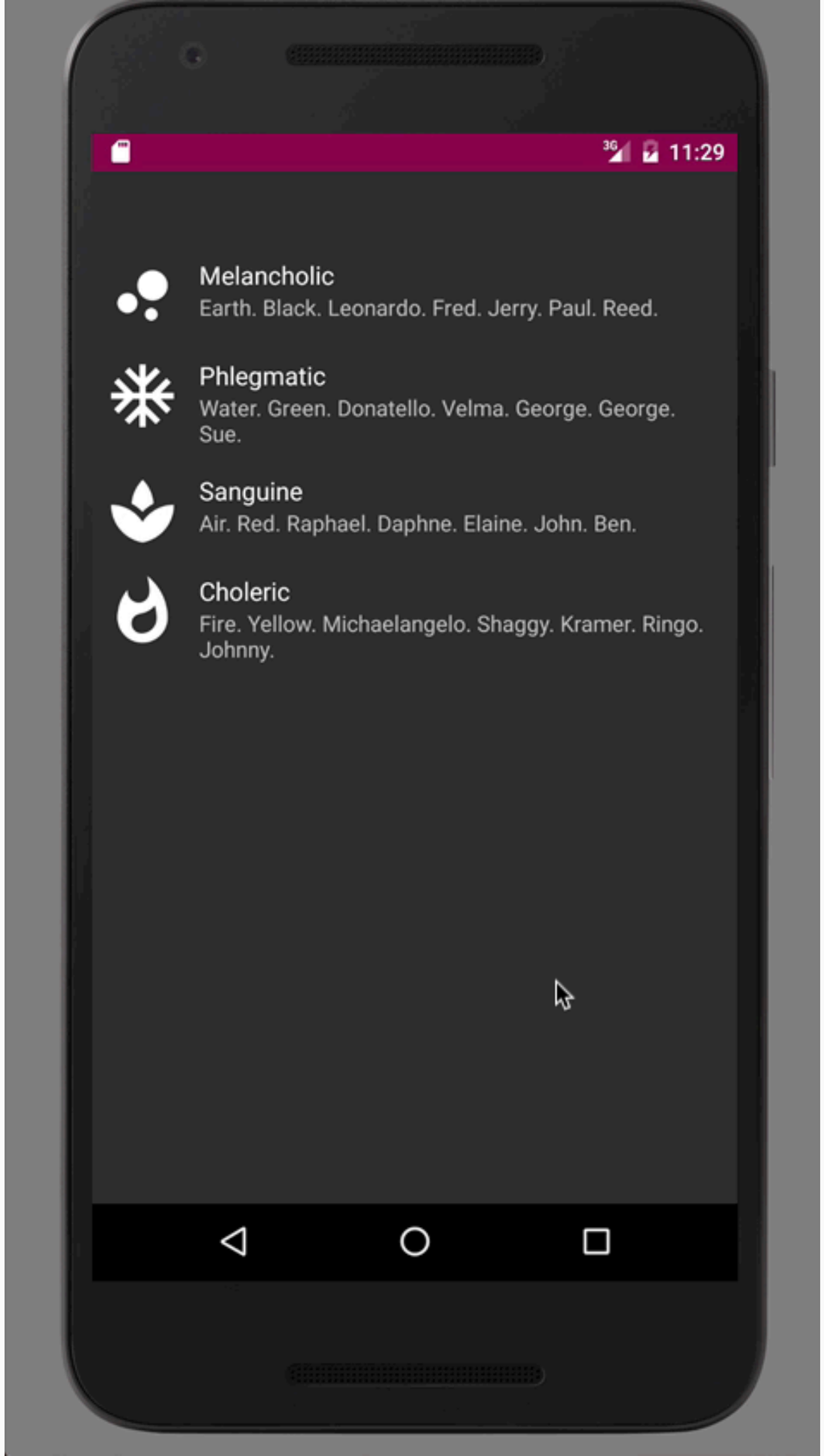
```
    //
    // View overrides
    //

    @Override
    protected void onMeasure(int widthMeasureSpec, int heightMeasureSpec) {
        final Paint.FontMetrics fontMetrics = numberPaint.getFontMetrics();

        // Measure maximum possible width of text.
        final float maxTextWidth = numberPaint.measureText(MAX_COUNT_STRING);
        // Estimate maximum possible height of text.
        final float maxTextHeight = -fontMetrics.top + fontMetrics.bottom;

        // Add padding to maximum width calculation.
        final int desiredWidth = Math.round(maxTextWidth + getPaddingLeft() + getPaddingRight());

        // Add padding to maximum height calculation.
        final int desiredHeight = Math.round(maxTextHeight * 2f + getPaddingTop() +
            getPaddingBottom());

        // Reconcile size that this view wants to be with the size the parent will let it be.
        final int measuredWidth = reconcileSize(desiredWidth, widthMeasureSpec);
        final int measuredHeight = reconcileSize(desiredHeight, heightMeasureSpec);

        // Store the final measured dimensions.
        setMeasuredDimension(measuredWidth, measuredHeight);
    }

    @Override
    protected void onDraw(Canvas canvas) {

        // Grab canvas dimensions.
        final int canvasWidth = canvas.getWidth();
        final int canvasHeight = canvas.getHeight();

        // Calculate horizontal center
```


3G
11:13
CLICK
0000
RESET

WHAT DO YOU NEED TO IMPLEMENT?

VIEWGROUP

MEASURE `onMeasure()`	YES	HAVE TO MEASURE EACH CHILD: `child.measure()`
LAYOUT `onLayout()`	YES	(ABSTRACT) HAVE TO LAYOUT EACH CHILD: `child.layout()`
DRAW `onDraw()`	NO	DOES NOT DRAW BY DEFAULT `setWillNotDraw`


11:29
Melancholic
Earth. Black. Leonardo. Fred. Jerry. Paul. Reed.
Phlegmatic
Water. Green. Donatello. Velma. George. George. Sue.
Sanguine
Air. Red. Raphael. Daphne. Elaine. John. Ben.
Choleric
Fire. Yellow. Michaelangelo. Shaggy. Kramer. Ringo. Johnny.

```
<randomlytyping.widget.SimpleListItem
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:paddingTop="@dimen/list_item_padding_vertical"
    android:paddingBottom="@dimen/list_item_padding_vertical"
    android:paddingStart="0dp"
    android:paddingEnd="@dimen/activity_horizontal_margin"
    android:background="?attr/selectableItemBackground"
    android:clickable="true">

    <ImageView
        android:id="@+id/icon"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="@dimen/list_item_icon_margin_start"
        android:layout_marginEnd="@dimen/list_item_icon_margin_end"
        android:contentDescription="@null" />

    <TextView
        android:id="@+id/title"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        style="@style/TextAppearance.AppCompat.Subhead" />

    <TextView
        android:id="@+id/subtitle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        style="@style/TextAppearance.AppCompat.Body1"
        android:textColor="?android:attr/textColorSecondary" />

</randomlytyping.widget.SimpleListItem>
```

```java
/**
 * Custom {@link ViewGroup} for displaying the example list items from the launch screen but
 * without the extra measure/layout pass from the {@link android.widget.RelativeLayout}.
 */
public class SimpleListItem extends ViewGroup {
    //
    // Fields
    //

    @BindView(R.id.icon)
    ImageView icon;

    @BindView(R.id.title)
    TextView titleView;

    @BindView(R.id.subtitle)
    TextView subtitleView;

    //
    // Constructors
    //

    /**
     * Constructor.
     *
     * @param context The current context.
     * @param attrs   The attributes of the XML tag that is inflating the view.
     */
    public SimpleListItem(Context context, AttributeSet attrs) { super(context, attrs); }

    @Override
    protected void onFinishInflate() {
        super.onFinishInflate();
        ButterKnife.bind(this);
    }
```

CUSTOM LAYOUT PARAMETERS

ViewGroup.LayoutParams
(width, height)

MarginLayoutParams
(width, height)
(left, top, right, bottom)

VALIDATE
`checkLayoutParams()`

VIEW **WITHOUT** LAYOUT PARAMETERS
`generateDefaultLayoutParams()`

VIEW WITH **INVALID** LAYOUT PARAMETERS
`generateLayoutParams(ViewGroup.LayoutParams p)`

VIEW WITH LAYOUT **XML ATTRIBUTES**
`generateLayoutParams(AttributeSet attrs)`

USEFUL MEASUREMENT METHODS

MEASURE SPEC + **PADDING**

`measureChild(…)`

`measureChildren(…)`

MEASURE SPEC + **PADDING + MARGINS**

`measureChildWithMargins(…)`

MEASURE SPEC + **CHILD LAYOUT PARAMETERS**

`getChildMeasureSpec(…)`

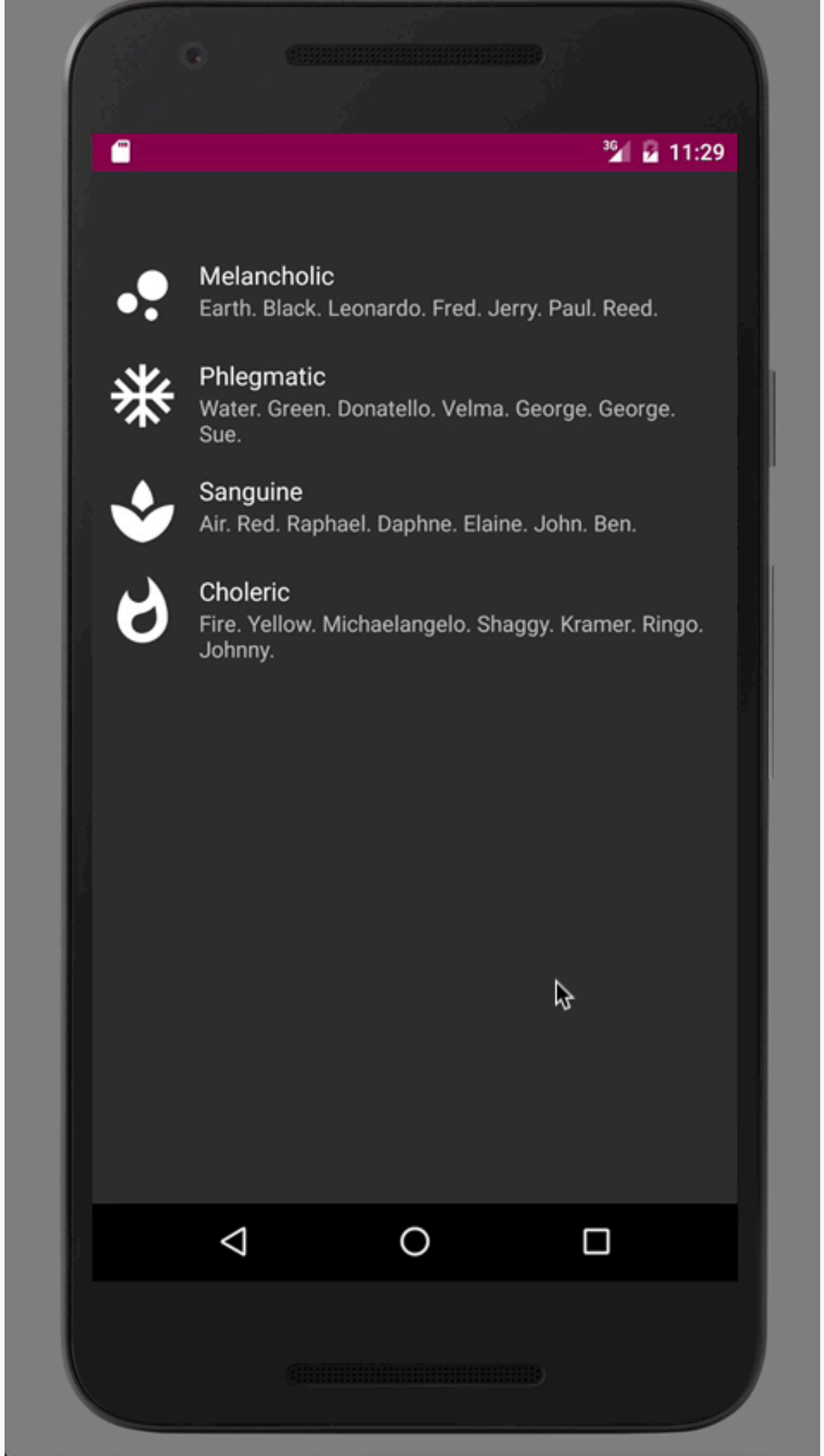
```java
    /**
     * Validates if a set of layout parameters is valid for a child of this ViewGroup.
     */
    @Override
    protected boolean checkLayoutParams(ViewGroup.LayoutParams p) {
        return p instanceof MarginLayoutParams;
    }


    /**
     * @return A set of default layout parameters when given a child with no layout parameters.
     */
    @Override
    protected LayoutParams generateDefaultLayoutParams() {
        return new MarginLayoutParams(LayoutParams.WRAP_CONTENT, LayoutParams.WRAP_CONTENT);
    }

    /**
     * @return A set of layout parameters created from attributes passed in XML.
     */
    @Override
    public LayoutParams generateLayoutParams(AttributeSet attrs) {
        return new MarginLayoutParams(getContext(), attrs);
    }

    /**
     * Called when {@link #checkLayoutParams(LayoutParams)} fails.
     *
     * @return A set of valid layout parameters for this ViewGroup that copies appropriate/valid
     * attributes from the supplied, not-so-good parameters.
     */
    @Override
    protected LayoutParams generateLayoutParams(ViewGroup.LayoutParams p) {
        return generateDefaultLayoutParams();
    }
```

```java
@Override
protected void onMeasure(int widthMeasureSpec, int heightMeasureSpec) {

    // Measure icon.
    measureChildWithMargins(icon, widthMeasureSpec, 0, heightMeasureSpec, 0);

    // Figure out how much width the icon used.
    MarginLayoutParams lp = (MarginLayoutParams) icon.getLayoutParams();
    int widthUsed = icon.getMeasuredWidth() + lp.leftMargin + lp.rightMargin;

    int heightUsed = 0;

    // Measure title
    measureChildWithMargins(
        titleView,
        // Pass width constraints and width already used.
        widthMeasureSpec, widthUsed,
        // Pass height constraints and height already used.
        heightMeasureSpec, heightUsed
    );

    // Measure the Subtitle.
    measureChildWithMargins(
        subtitleView,
        // Pass width constraints and width already used.
        widthMeasureSpec, widthUsed,
        // Pass height constraints and height already used.
        heightMeasureSpec, titleView.getMeasuredHeight());

    // Calculate this view's measured width and height.

    // Figure out how much total space the icon used.
    int iconWidth = icon.getMeasuredWidth() + lp.leftMargin + lp.rightMargin;
    int iconHeight = icon.getMeasuredHeight() + lp.topMargin + lp.bottomMargin;
    lp = (MarginLayoutParams) titleView.getLayoutParams();
```

```
            subtitleView,
            // Pass width constraints and width already used.
            widthMeasureSpec, widthUsed,
            // Pass height constraints and height already used.
            heightMeasureSpec, titleView.getMeasuredHeight());

    // Calculate this view's measured width and height.

    // Figure out how much total space the icon used.
    int iconWidth = icon.getMeasuredWidth() + lp.leftMargin + lp.rightMargin;
    int iconHeight = icon.getMeasuredHeight() + lp.topMargin + lp.bottomMargin;
    lp = (MarginLayoutParams) titleView.getLayoutParams();

    // Figure out how much total space the title used.
    int titleWidth = titleView.getMeasuredWidth() + lp.leftMargin + lp.rightMargin;
    int titleHeight = titleView.getMeasuredHeight() + lp.topMargin + lp.bottomMargin;
    lp = (MarginLayoutParams) subtitleView.getLayoutParams();

    // Figure out how much total space the subtitle used.
    int subtitleWidth = subtitleView.getMeasuredWidth() + lp.leftMargin + lp.rightMargin;
    int subtitleHeight = subtitleView.getMeasuredHeight() + lp.topMargin + lp.bottomMargin;

    // The width taken by the children + padding.
    int width = getPaddingTop() + getPaddingBottom() +
        iconWidth + Math.max(titleWidth, subtitleWidth);
    // The height taken by the children + padding.
    int height = getPaddingTop() + getPaddingBottom() +
        Math.max(iconHeight, titleHeight + subtitleHeight);

    // Reconcile the measured dimensions with the this view's constraints and
    // set the final measured width and height.
    setMeasuredDimension(
        resolveSize(width, widthMeasureSpec),
        resolveSize(height, heightMeasureSpec)
    );
}
```

```java
@Override
protected void onLayout(boolean changed, int left, int top, int right, int bottom) {
    MarginLayoutParams layoutParams = (MarginLayoutParams) icon.getLayoutParams();

    // Figure out the x-coordinate and y-coordinate of the icon.
    int x = getPaddingLeft() + layoutParams.leftMargin;
    int y = getPaddingTop() + layoutParams.topMargin;

    // Layout the icon.
    icon.layout(x, y, x + icon.getMeasuredWidth(), y + icon.getMeasuredHeight());

    // Calculate the x-coordinate of the title: icon's right coordinate +
    // the icon's right margin.
    x += icon.getMeasuredWidth() + layoutParams.rightMargin;

    // Add in the title's left margin.
    layoutParams = (MarginLayoutParams) titleView.getLayoutParams();
    x += layoutParams.leftMargin;

    // Calculate the y-coordinate of the title: this ViewGroup's top padding +
    // the title's top margin
    y = getPaddingTop() + layoutParams.topMargin;

    // Layout the title.
    titleView.layout(x, y, x + titleView.getMeasuredWidth(), y + titleView.getMeasuredHeight());

    // The subtitle has the same x-coordinate as the title. So no more calculating there.

    // Calculate the y-coordinate of the subtitle: the title's bottom coordinate +
    // the title's bottom margin.
    y += titleView.getMeasuredHeight() + layoutParams.bottomMargin;
    layoutParams = (MarginLayoutParams) subtitleView.getLayoutParams();

    // Add in the subtitle's top margin.
    y += layoutParams.topMargin;
```

```
        // Figure out the x-coordinate and y-coordinate of the icon.
        int x = getPaddingLeft() + layoutParams.leftMargin;
        int y = getPaddingTop() + layoutParams.topMargin;

        // Layout the icon.
        icon.layout(x, y, x + icon.getMeasuredWidth(), y + icon.getMeasuredHeight());

        // Calculate the x-coordinate of the title: icon's right coordinate +
        // the icon's right margin.
        x += icon.getMeasuredWidth() + layoutParams.rightMargin;

        // Add in the title's left margin.
        layoutParams = (MarginLayoutParams) titleView.getLayoutParams();
        x += layoutParams.leftMargin;

        // Calculate the y-coordinate of the title: this ViewGroup's top padding +
        // the title's top margin
        y = getPaddingTop() + layoutParams.topMargin;

        // Layout the title.
        titleView.layout(x, y, x + titleView.getMeasuredWidth(), y + titleView.getMeasuredHeight());

        // The subtitle has the same x-coordinate as the title. So no more calculating there.

        // Calculate the y-coordinate of the subtitle: the title's bottom coordinate +
        // the title's bottom margin.
        y += titleView.getMeasuredHeight() + layoutParams.bottomMargin;
        layoutParams = (MarginLayoutParams) subtitleView.getLayoutParams();

        // Add in the subtitle's top margin.
        y += layoutParams.topMargin;

        // Layout the subtitle.
        subtitleView.layout(x, y,
            x + subtitleView.getMeasuredWidth(), y + subtitleView.getMeasuredHeight());
    }
}
```


11:29
Melancholic
Earth. Black. Leonardo. Fred. Jerry. Paul. Reed.
Phlegmatic
Water. Green. Donatello. Velma. George. George. Sue.
Sanguine
Air. Red. Raphael. Daphne. Elaine. John. Ben.
Choleric
Fire. Yellow. Michaelangelo. Shaggy. Kramer. Ringo. Johnny.

WHERE TO GO FROM HERE

ACCESSIBILITY. A11Y FTW. ADD IT.
@KELLYSHUSTER

TOUCH. GESTURES.
@PBREAULT

CUSTOM DRAWABLES. CUSTOM DRAWABLE STATES.
@RHARTER @MARCOSPAULOSD

ALL THE DRAWING OPERATIONS. PATHS. FILTERS. SHADERS. OH MY!
@ROMAINGUY @CHIUKI

HARDWARE ACCELERATION. LEARN ABOUT LAYERS + DISPLAY LISTS.
@WORKINGKILLS

GO CUSTOM.

randomlytyping.com

youtube.com/androiddialogs

caster.io

speakerdeck.com/queencodemonkey

THANK YOU!

HUYEN TUE DAO

@QUEENCODEMONKEY

CUSTOM VIEWS/VIEWGROUPS

Custom Views and ViewGroups (Part 1) https://caster.io/episodes/custom-views-and-viewgroups-part-1/

Creating Custom Views https://developer.android.com/training/custom-views/index.html

A deep dive into Android View constructors http://blog.danlew.net/2016/07/19/a-deep-dive-into-android-view-constructors/

Google I/O 2013 - Writing Custom Views for Android https://youtu.be/NYtB6mlu7vA

Enhancing Android UI with Custom Views https://newcircle.com/s/post/1663/tutorial_enhancing_android_ui_with_custom_views_dave_smith_video

CUSTOM VIEW/VIEWGROUPS

Custom Android Components http://chiuki.github.io/android-custom-components

Custom Views and Performance https://youtu.be/zK2i7ivzK7M

Taming Android UIs with Eric Burke of Square https://youtu.be/jF6Ad4GYjRU

Custom Layouts on Android http://lucasr.org/2014/05/12/custom-layouts-on-android/

Layout traversals on Android https://youtu.be/sdkcuvZCh1U

Custom View Groups https://sriramramani.wordpress.com/2015/05/06/custom-viewgroups/

TOUCH/GESTURES

Making Sense of the Touch System https://speakerdeck.com/pbreault/making-sense-of-the-touch-system | https://youtu.be/usBaTHZdXSI

Making the View Interactive https://developer.android.com/training/custom-views/making-interactive.html

TEXT + DRAWING

Typography on Android https://youtu.be/OgiZa22Zpus

Android Textual Layout https://youtu.be/GZ0eKqvzJa8

Hinting Around: Text Demystified https://www.youtube.com/watch?v=VS7co3TrgKE | https://speakerdeck.com/rock3r/hinting-around-android-text-demystified

Calligraphy. Custom fonts in Android the easy way… https://github.com/chrisjenx/Calligraphy

DRAWING + DRAWABLES

Fun with Android Shaders and Filters https://chiuki.github.io/android-shaders-filters

Don’t Fear the Canvas https://youtu.be/KH8Ldp39TUk

Custom drawable states in Android http://charlesharley.com/2012/programming/custom-drawable-states-in-android

Custom Drawables http://ryanharter.com/blog/2015/04/03/custom-drawables/

The Magic World of Drawables https://youtu.be/1YjB1uUfxgE

Hardware Acceleration https://developer.android.com/guide/topics/graphics/hardware-accel.html