CET 325

lifechanging University of Sunderland

The Top Layer: Layouts and UI Controls

Recap

- Android Architecture
- Android Components
- Event Handling

lifechanging University of Sunderland

Agenda

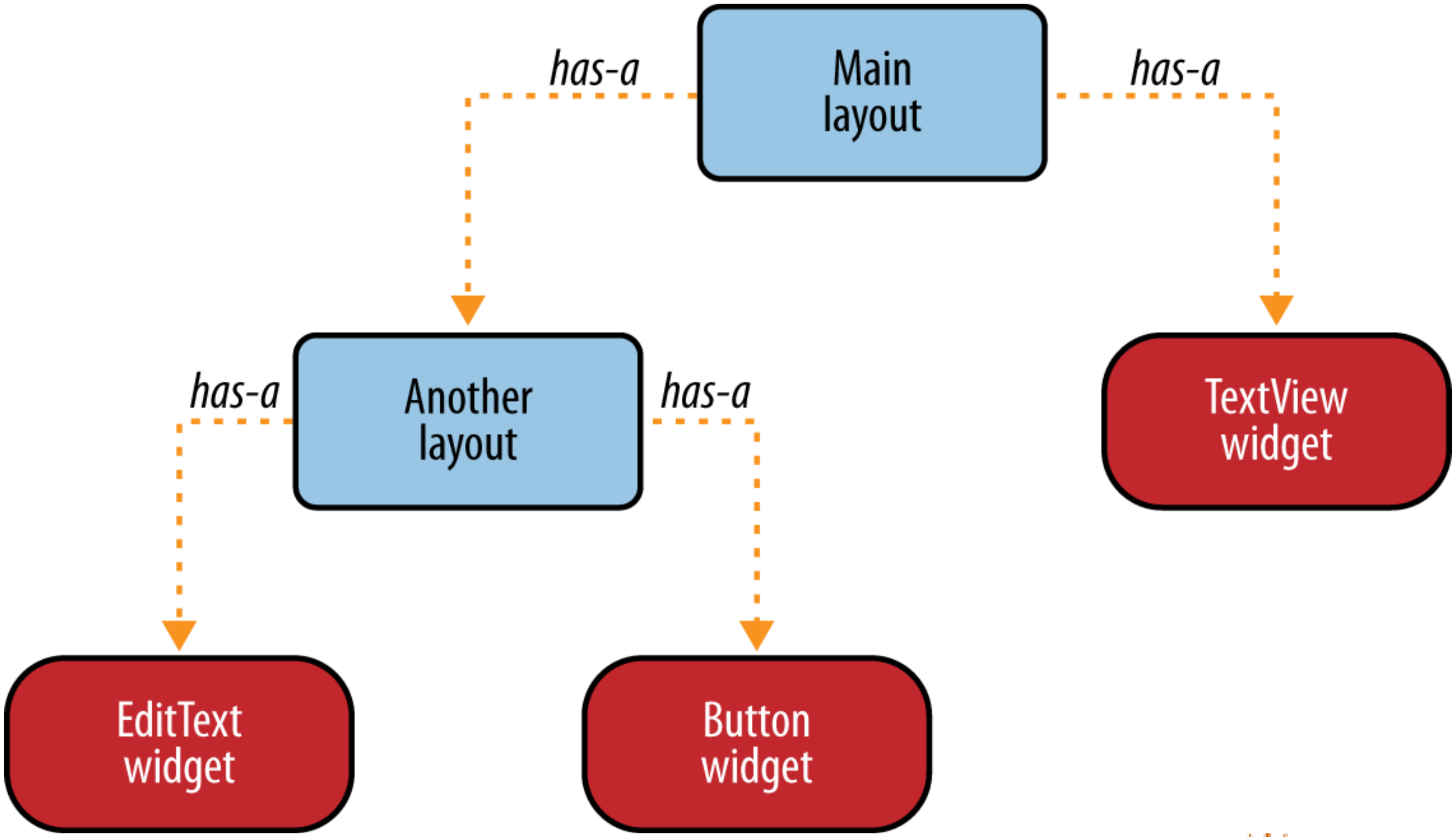
- Introduce Layouts for more sophisticated UI design

- Components: Check Boxes, Radio Buttons, Spinners, Toast, Pop up Dialog.
  - Theory by example

UI Design

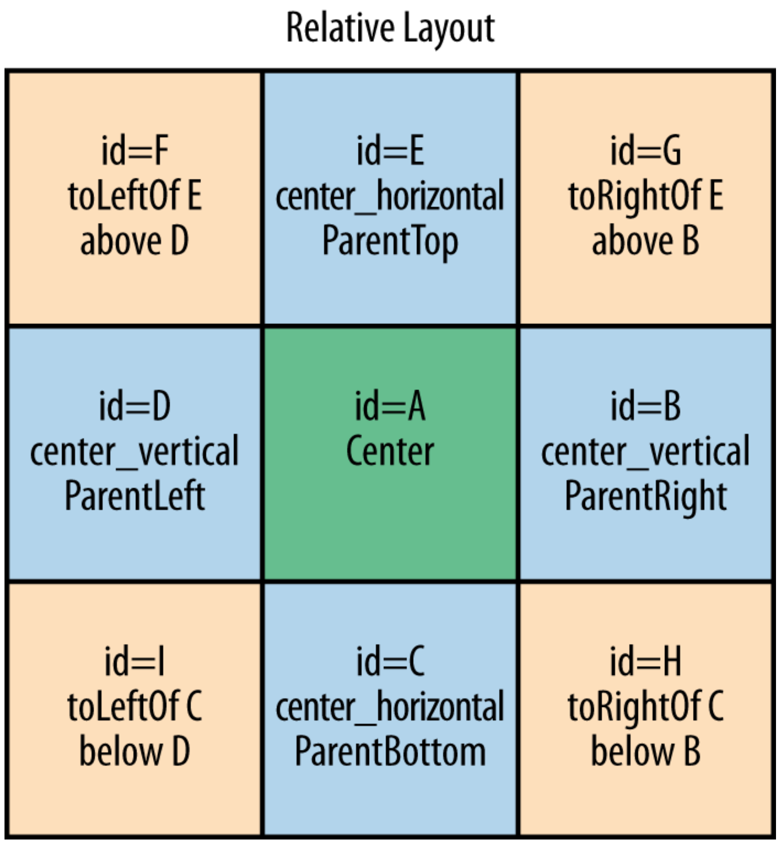
- Android organises UI elements into Views, ViewGroups and Layouts.
- View: Also known as widgets. Buttons, labels, text boxes etc.
- Layouts: Organise views, eg grouping together multiple elements.
  - Can contain other children, which may be other layouts.
  - Relative, Linear, Table, Frame
  - Grid Layout (API level 14 and above)

Layouts and Views



RelativeLayout

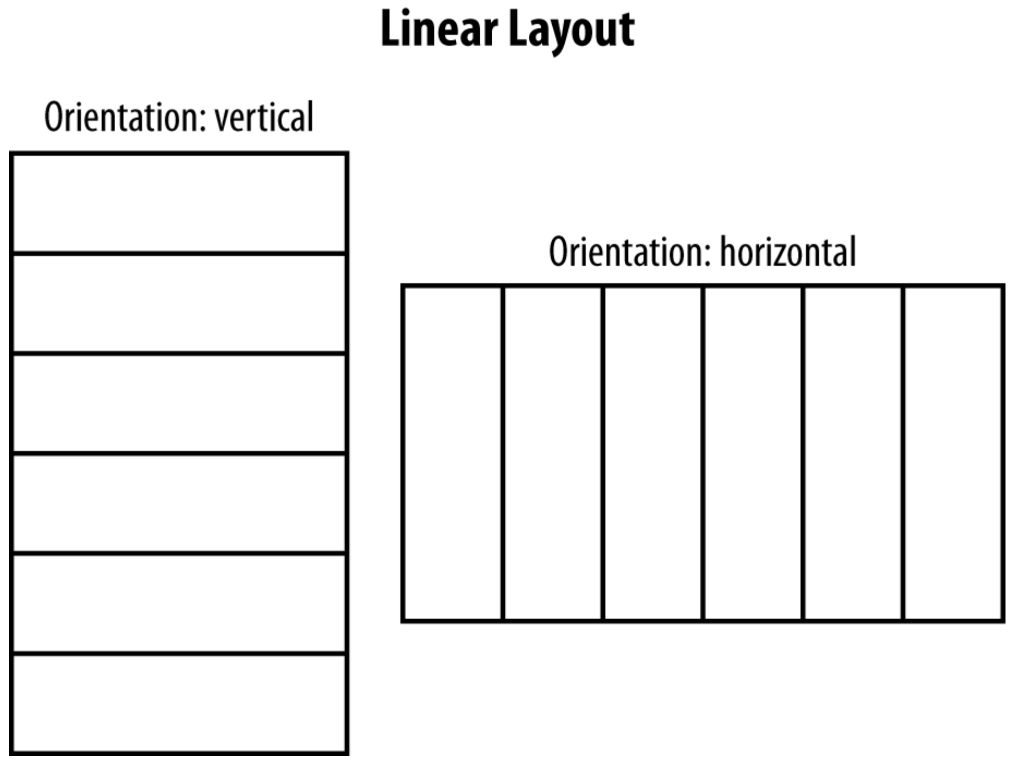
- Defines view object positions relative to each other.
- Doesn’t require you to nest layouts to achieve a certain look.
- Requires each child view to have an ID, which can add complexity.
- Versatile option, low overhead for simple view hierarchies.



lifechanging University of Sunderland

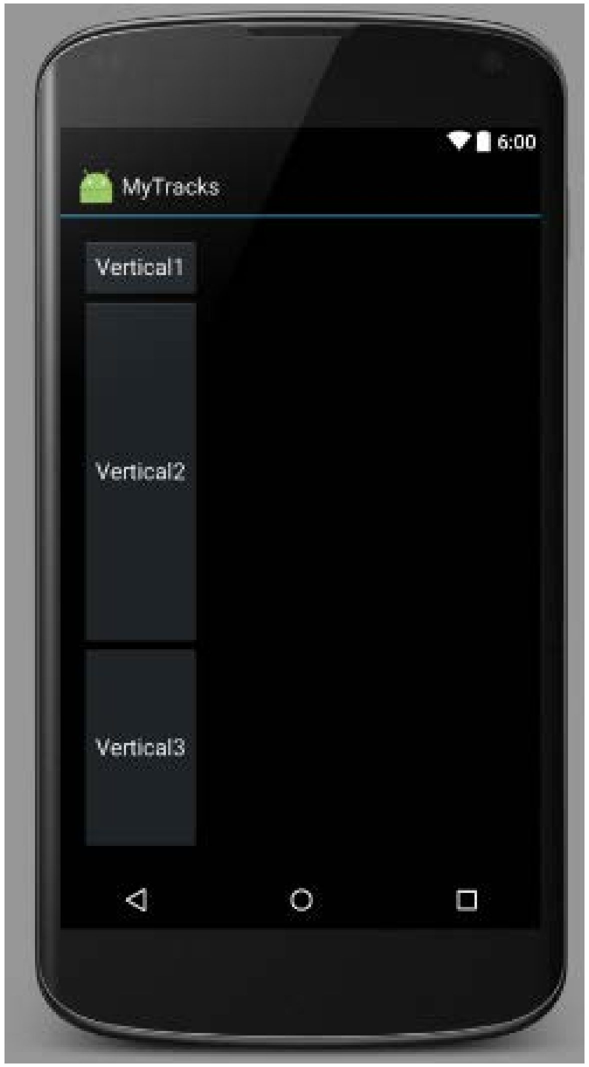
LinearLayout

- Lay out children next to the other, either horizontally or vertically (you define the orientation as a layout property in xml).
- The order of the children matters. If an older child requests more space, the others may not render appropriately.



```
android:orientation = "horizontal"
```

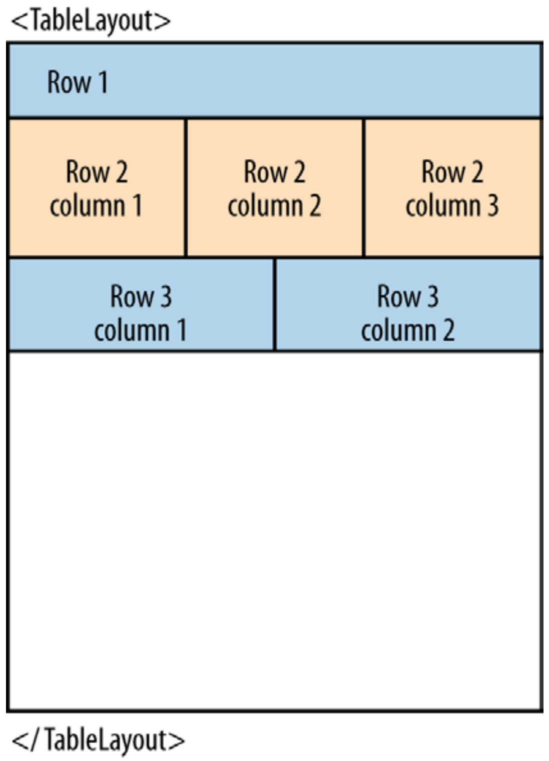
```xml
<LinearLayout xmlns:android=
                "http://schemas.android.com/apk/res/android"
                xmlns:tools="http://schemas.android.com/tools"
                android:orientation="vertical"
                tools:context=".MainActivity"
                >
    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Vertical1" />
    <Button
        android:id="@+id/button2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Vertical2"
        android:layout_weight="2"/>
    <Button
        android:id="@+id/button3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Vertical3"
        android:layout_weight="1"/>
</LinearLayout>
```



lifechanging University of Sunderland

TableLayout

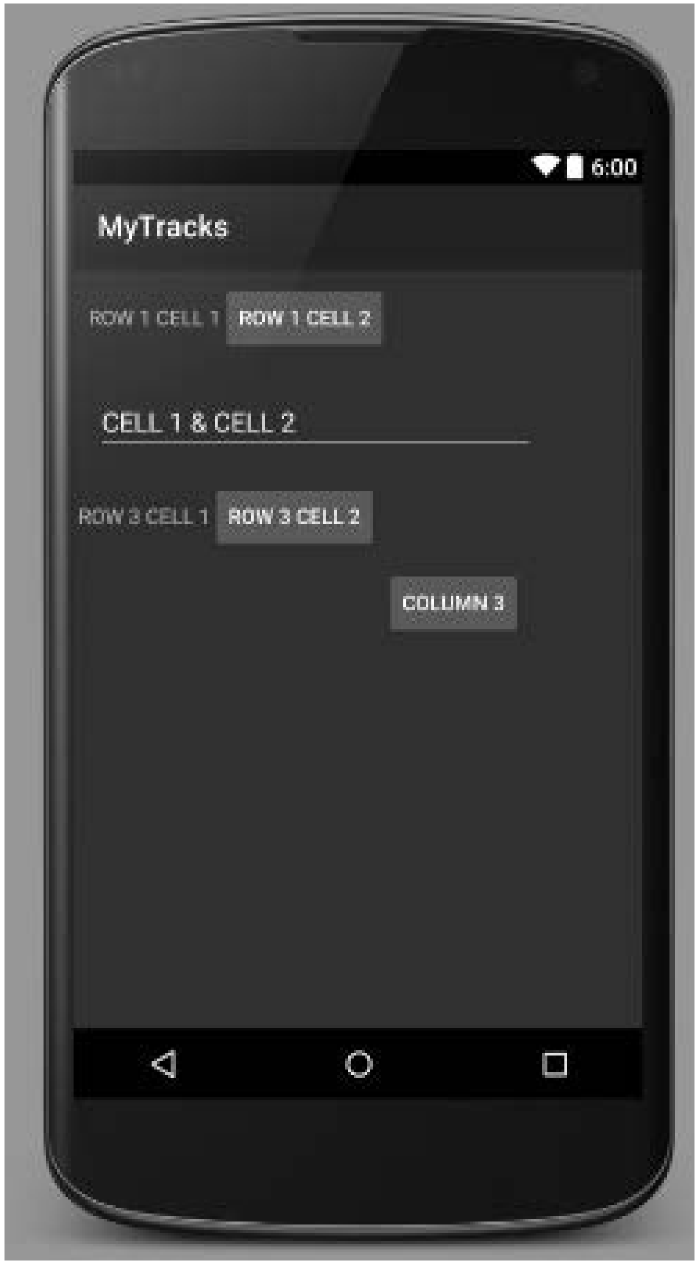
- Lays child views out in a table.
- The views it contains are TableRow widgets.
- Each TableRow represents a row in a table and can contain other UI widgets.
- The property stretch_columns can be used to stretch a column of the table. You can also use * to stretch all columns



```
<TableLayout ......(header details omitted)
    <TableRow
        android:id="@+id/Row1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:padding="10dip" >
        <TextView
            android:id="@+id/View1"
            android:text="ROW 1 CELL 1"
            />
        <Button
            android:id="@+id/bttn1"
            android:text="ROW 1 CELL 2" />
    </TableRow>

    <!-- 2nd ROW -->
    <TableRow
        android:id="@+id/Row2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:padding="15dip" >
        <EditText
            android:id="@+id/Text1"
            android:layout_span="4"
            android:text="CELL 1 &amp; CELL 2" />
    </TableRow>
```

```
<TableRow
        android:id="@+id/Row3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:padding="3dip" >
        <TextView
            android:id="@+id/View2"
            android:text="ROW 3 CELL 1"/>
        <Button
            android:id="@+id/bttn2"
            android:text="ROW 3 CELL 2" />
    </TableRow>

    <TableRow
        android:id="@+id/Row4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:padding="7dip" >

        <Button
            android:id="@+id/bttn4"
            android:layout_column="2"
            android:text="Column 3" />
    </TableRow>
</TableLayout>
```

lifechanging University of Sunderland



Example Layout Summary

4 rows

3 columns

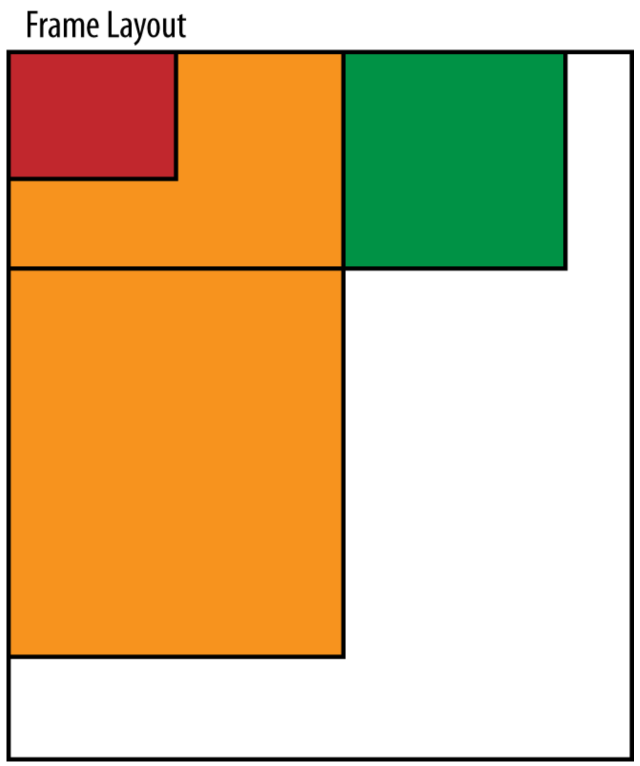
Combination of widgets

Widgets spanning multiple rows

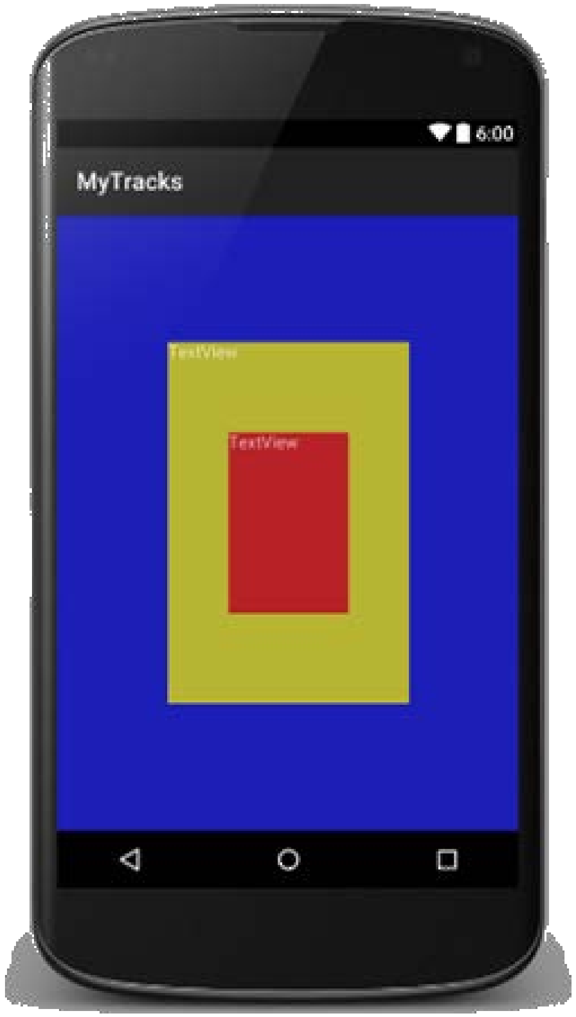
lifechanging University of Sunderland

FrameLayout

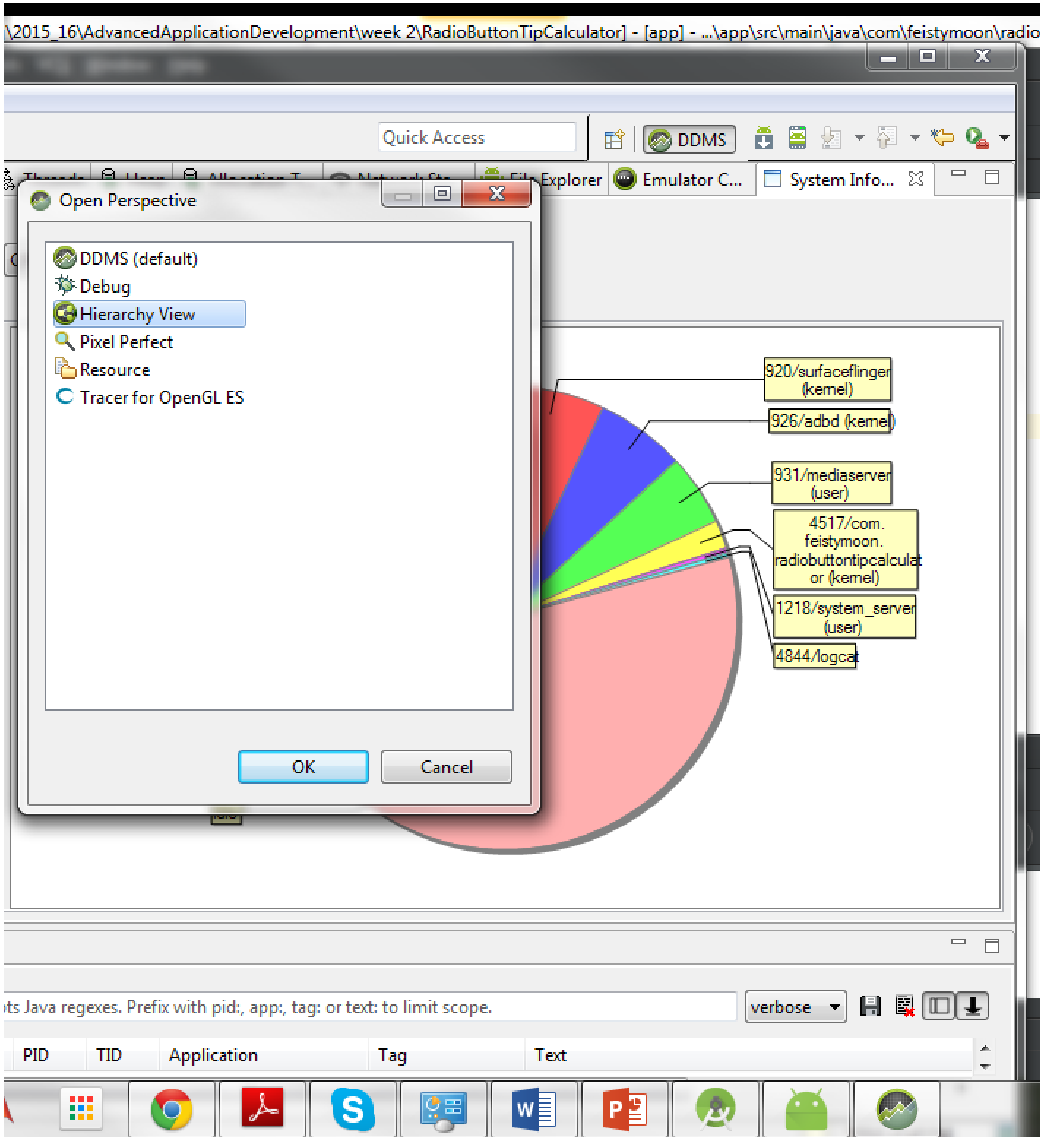
- Places children on top of each other
- Latest child covers the previous one
- Can be useful mechanism for implementing tabs, or for creating placeholders for widgets which will be added programmatically at a later stage.



```xml
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
             android:layout_width="fill_parent"
             android:layout_height="fill_parent"
             android:id="@+id/framelayout"
             android:background="#1c1eb7">

    <TextView
        android:id="@+id/frameImage"
        android:layout_width="200dp"
        android:layout_height="300dp"
        android:layout_gravity="center"
        android:background="#b7b432"/>

    <TextView
        android:id="@+id/frameImage2"
        android:layout_width="100dp"
        android:layout_height="150dp"
        android:layout_gravity="center"
        android:background="#b72126"/>

    <TextView
        android:id="@+id/frameText"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center" />

</FrameLayout>
```



Inspecting Your Layout



its Java regexes. Prefix with pid:, app:, tag: or text: to limit scope.

Tools → Android Device Manager

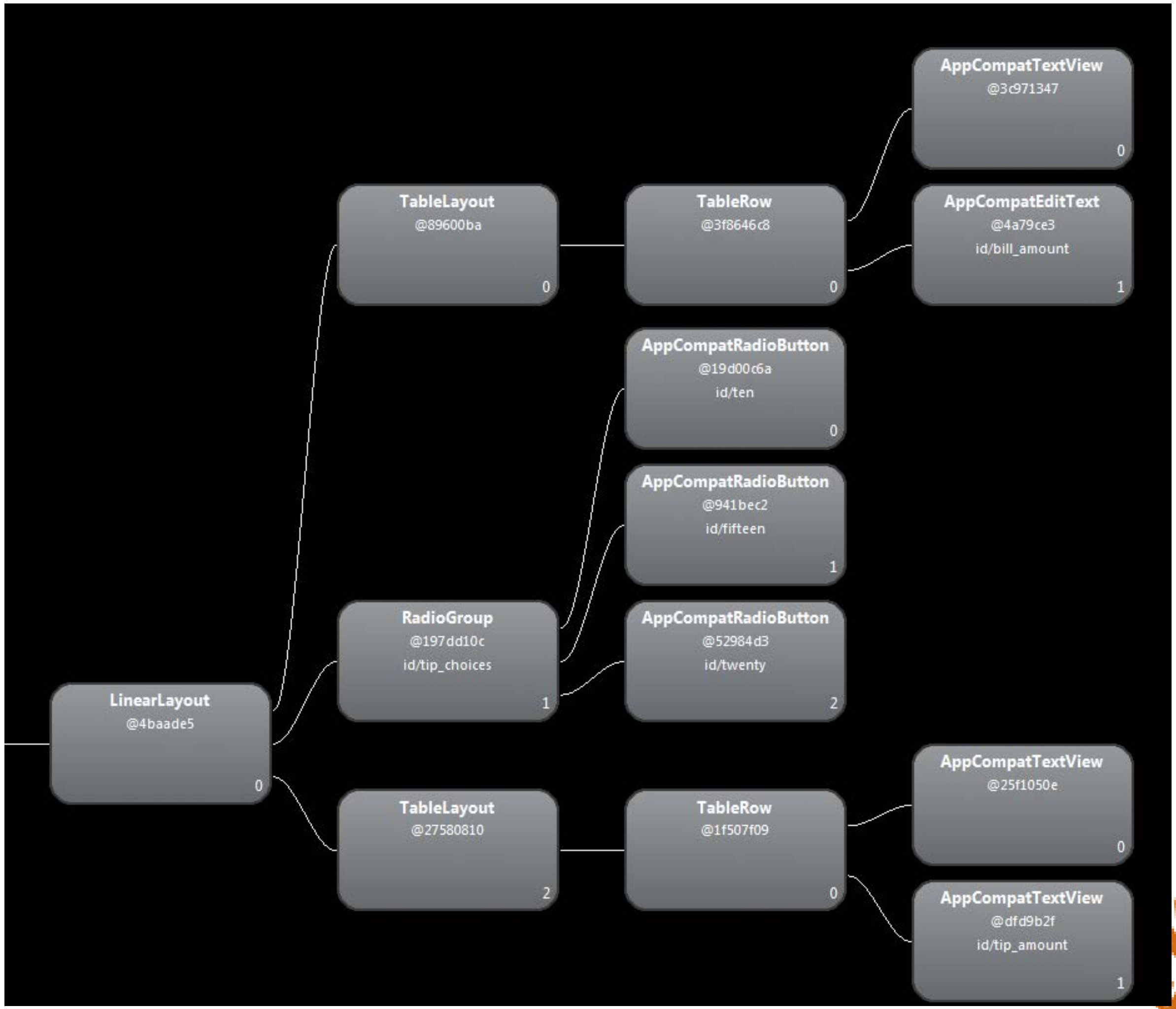
Click on Open Perspective Icon

Click on Hierarchy View

lifechanging University of Sunderland

Inspecting Your Layout

- Click on the running activity and you hierarchy is shown



University of Sunderland

UI Controls

- Check Boxes
- Radio Buttons
- Radio Group
- Spinner
- Date Picker
- … many more!

UI Controls

- Theory by Example
  - CheckBoxes: Task List Application
  - Radio Buttons: Tip Calculator
  - Spinner: Tip Calculator

Check Boxes

Class: CheckBox

Package: android.widget

Extends: android.widget.CompoundButton

Description: Similar to a Button, but has only two states: checked and unchecked.

Example Methods:

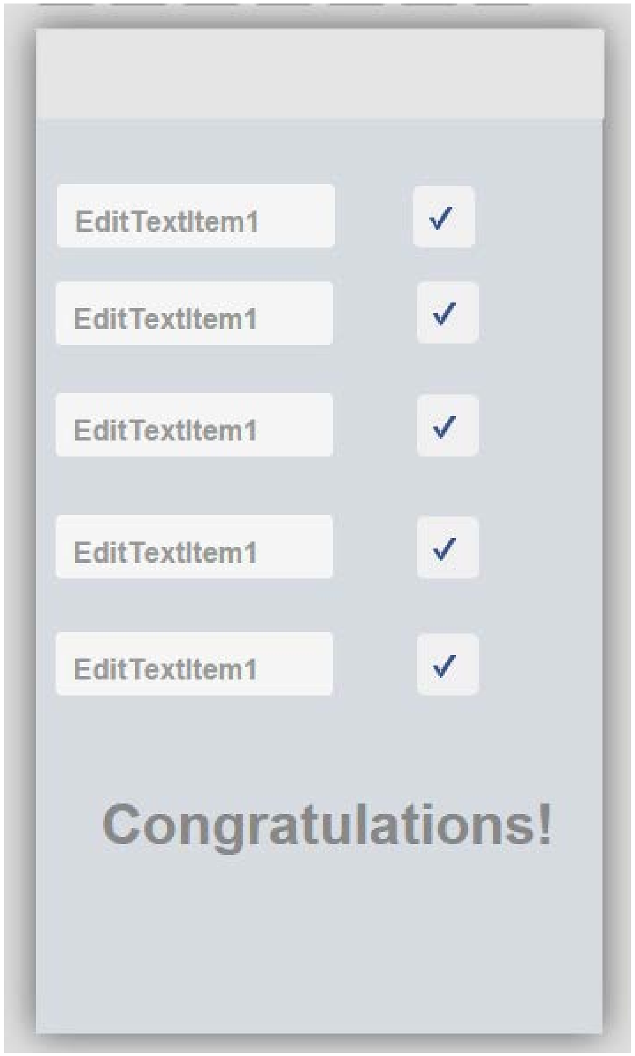
void setChecked(Boolean checked). *Inherited from CompoundButton. This method changes the state of the CheckBox.*

boolean isChecked(). *This method returns the state of the CheckBox.*

void setOnClickListener(View.OnClickListener listener*). Inherited from View. Registers a listener and process to be invoked when the control is clicked.*

API: http://developer.android.com/reference/android/widget/CheckBox.html

Example – 'Tasks List' Application



'Task List' application should lets users:

- Write a maximum of five tasks they want to do
- Check off tasks when they are complete
- Congratulate the user when all tasks are finished.

lifechanging University of Sunderland

TaskList

- Create new project
- Look at XML for your activity – delete the Text and change RelativeLayout to TableLayout.
- Set Layout orientation to Vertical

```
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
                xmlns:tools="http://schemas.android.com/tools"
                android:layout_width="match_parent"
                android:layout_height="match_parent"
               tools:context=".MainActivity"
                android:orientation = "vertical">
</TableLayout>
```

Add TableRows

```
<TableRow
    android:layout_width="fill_parent"
    android:layout_height="wrap_content">

    <EditText
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:minWidth="250dp"
        android:id="@+id/editTextTask1"/>

    <CheckBox
        android:id="@+id/checkBoxTask1"/>
</TableRow>
```

- Each set of EditText and CheckBox widgets can be added on a separate row in your layout.

Add Text Widget

```
<TableRow
    android:layout_width="fill_parent"
    android:layout_height="wrap_content">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:id="@+id/done"
        android:minWidth="250dp"
        />
</TableRow>
```

Application Logic

- Adjust the Class header to reflect that we will be using an OnClickListener (or use Anonymous method, or update onClick arrtribute in XML)
- Declare variables to represent each EditText and CheckBox component
- When onCreate is instantiate variables by pass the relevant id into findViewById()
- When a checkbox is clicked, check to see if they are all selected.

Application Logic

- Adjust the Class header to reflect that we will be using an OnClickListener

```
public class MainActivity
       extends ActionBarActivity implements OnClickListener {
```

- We will also need to implement onClick()

Application Logic

- Declare member variables to represent each EditText and CheckBox component

```
EditText editTextTask1 = null;
CheckBox checkBoxTask1 = null;
...
EditText editTextTask5 = null;
CheckBox checkBoxTask5 = null;
```

lifechanging University of Sunderland

Application Logic

- When onCreate is called instantiate the member variables and set them to unchecked.

```
editTextTask1 = (EditText)findViewById(R.id.editTextTask1);
checkBoxTask1 = (CheckBox)findViewById(R.id.checkBoxTask1);
checkBoxTask1.setOnClickListener(this);
checkBoxTask1.setChecked(false);
```

Application Logic

- When a checkbox is clicked, check to see if they are all checked.

```
public void onClick(View v){
    if(checkBoxTask1.isChecked() &
            checkBoxTask2.isChecked() &
            checkBoxTask3.isChecked() &
            checkBoxTask4.isChecked() &
            checkBoxTask5.isChecked()){
        TextView done = (TextView)findViewById(R.id.done);
        //Update Text using String Reference
        done.setText(R.string.done);
    }
}
```