Widgets and Layouts

Widgets and Input Handling

- Android widget tutorial
  - https://www.tutorialspoint.com/android/android_user_interface_controls.htm
- Event handling with listeners
  - https://developer.android.com/guide/topics/ui/ui-events

Family Map Client Widgets

TextView (text labels)
- text, textAppearance
- .setClickable(boolean) – make clickable
- .setOnClickListener(View.OnClickListener)

EditText (text fields)
- inputType, ems
- .addTextChangedListener(TextWatcher)

Space (blank space)
- set layout_width and layout_height to specific values (e.g., 30dp)

Button
- text
- .setOnClickListener(View.OnClickListener)

ImageView (display image)
- .setClickable(boolean) – make clickable
- .setImageDrawable(IconDrawable) – set icon to display
- .setOnClickListener(View.OnClickListener)

Switch (on/off)
- .setChecked(boolean) – set check state
- .setOnCheckedChangeListener(CompoundButton.OnCheckedChangeListener)

Spinner (dropdown list)
- .setAdapter(ArrayAdapter) – specify list values
- .setSelection(int) – specify selected item
- onItemSelectedListener(AdapterView.OnItemSelectedListener)

SearchView
- .setFocusable(boolean) – accept key focus
- .setIconified(boolean) – make always visible
- .requestFocusFromTouch() – request focus when touched
- .setOnQueryTextListener(SearchView.OnQueryTextListener)

ScrollView
- Wrap around any view to make it scrollable

RecyclerView
- Dynamic list of items

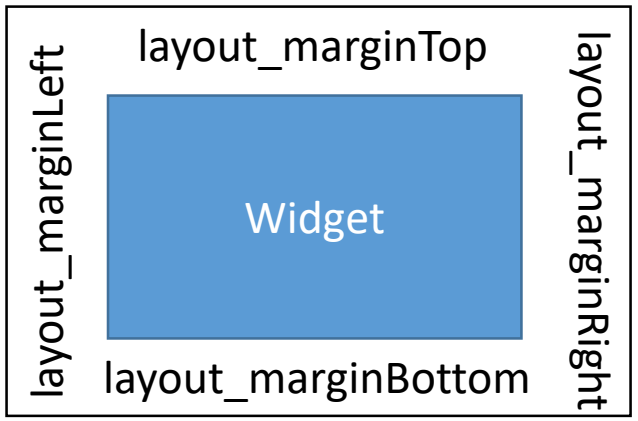
ExpandableListView
- Expandable list of items

The Layout Problem

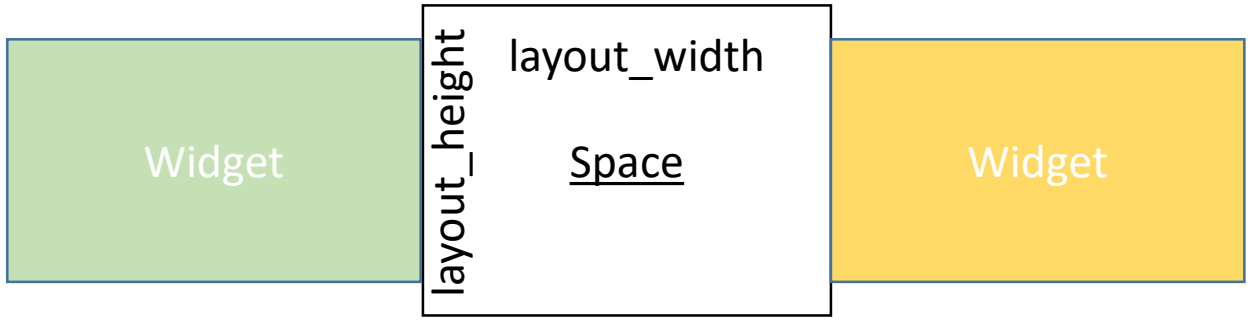
- Positioning widgets on the screen where you want them with the right sizes
- You don’t know exactly how big the device’s screen will be, so a layout needs to adjust dynamically to different screen sizes
  - You can have different layout XML files for different screen sizes, but you can’t have a different file for EVERY possible size, so a layout needs to adjust the best it can
- For a desktop UI, the analogous problem is adjusting the layout when the user expands and contracts the window
- How is extra space allocated (i.e., who gets bigger when the window expands, and who stays the same size?)

Layout – Blank Space Between Widgets

1. Margins



2. Space Widget

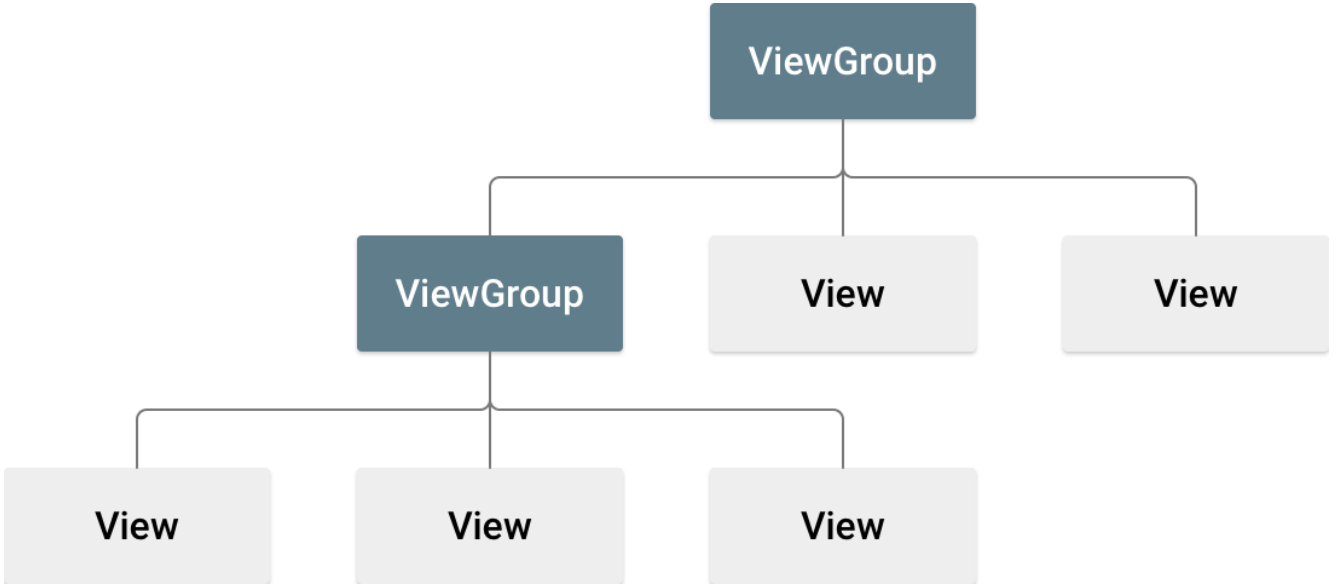


Layout - Scrolling

- Any activity that can become too big to fit on the screen should allow the user to scroll (by swiping)
- This is one way to handle the fact that you don’t really know how big the device screen will be
- To make a view scrollable, nest it in a ScrollView
- ScrollView lets the user scroll whenever the nested view is too big for the available space

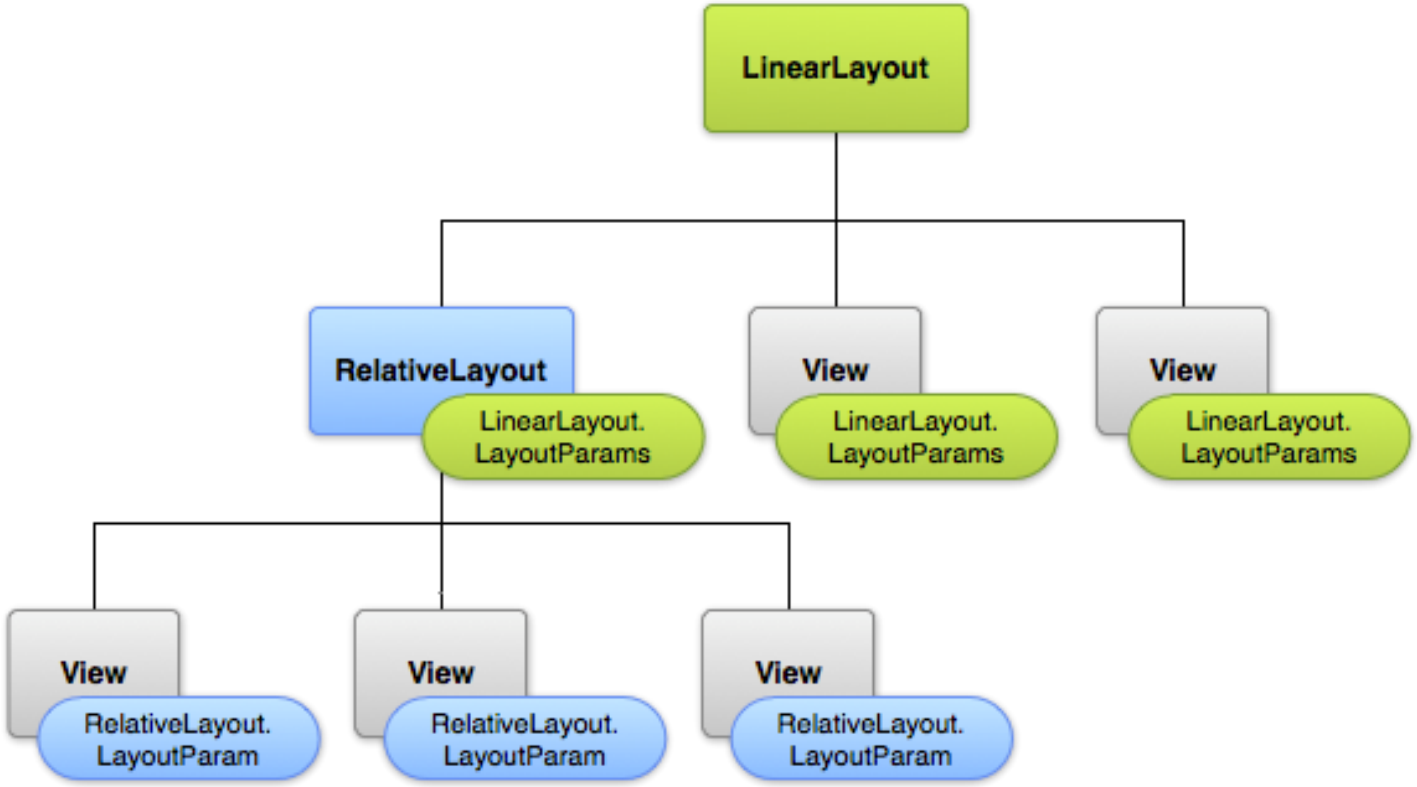
Layout Managers (ViewGroups)

- http://developer.android.com/guide/topics/ui/declaring-layout.html
- Layout manager arranges its child widgets on the screen
  - Sets each child's width, height, and location on the screen
- Already seen LinearLayout and FrameLayout



Layout Settings

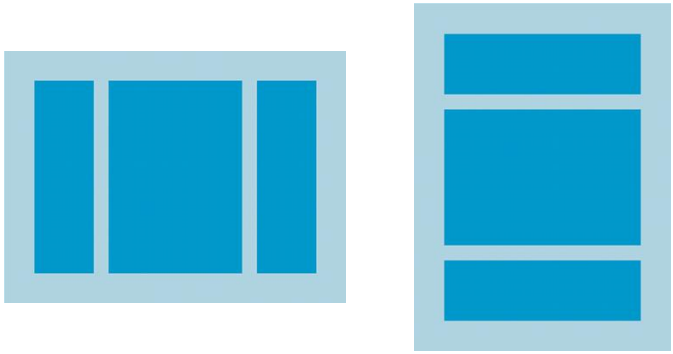
- `layout_*` attributes tell the parent layout manager how to lay out the widget



Universal Layout Settings (apply to all widgets)

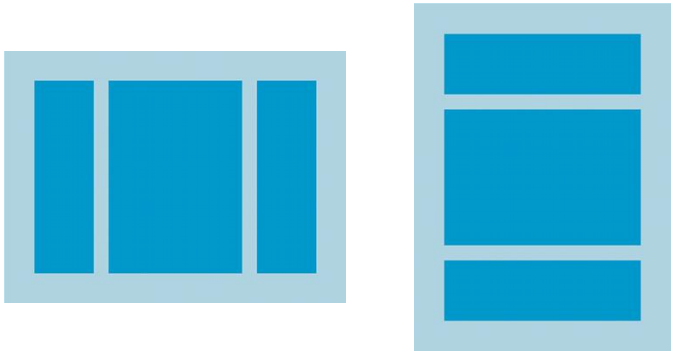
- `layout_width, layout_height`
  - match_parent
    - Make my size match my parent's size
  - wrap_content
    - Make me just big enough to display my contents
  - Hard-coded size (e.g., `30dp`)
    - px – pixels
    - in – inches
    - mm – millimeters
    - pt – points (1/72 of an inch)
    - dp – density-independent pixels, 1/160 of an inch)
    - sp – scale-independent pixels (similar to dp, but scaled by user's font size preference)

LinearLayout



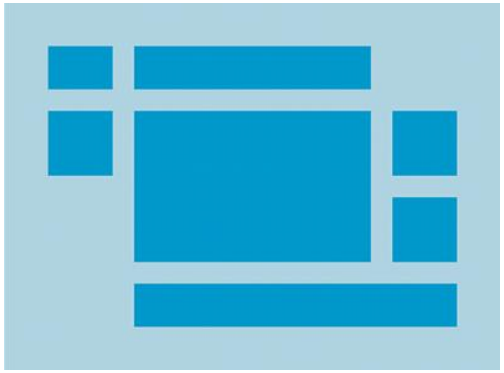
- Arranges children in either vertical column or horizontal row
- Important attributes
  - `orientation` – vertical or horizontal
  - `gravity` – if the children don't take up all the space inside the LinearLayout, where should they be placed (center, top, bottom, left, right, etc.)
- How should available empty space inside the LinearLayout be allocated to the children?
  - Put `layout_weight` attributes on the children indicating what proportion of the empty space each child would like to have
  - `layout_weight` values are numbers. Absolute values don't matter, just the relative sizes of the weights
  - `layout_weight="0"` means the child does not want any empty space
- `layout_height="0dp"` or `layout_width="0dp"` means let that dimension be controlled entirely by the weight

LinearLayout



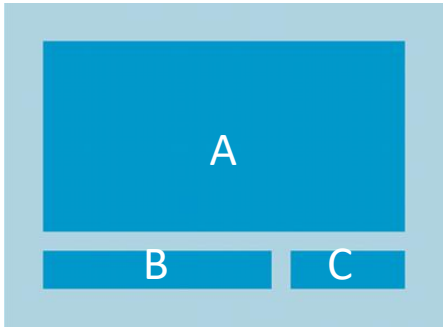
- Suppose there are three children
- Scenario #1
  - If the weights are 0, 0, 0, nobody will get any extra space, and the children will be positioned according to the LinearLayout's `gravity` attribute
- Scenario #2
  - If the weights are 1, 1, 1, the empty space will be allocated to the children equally
- Scenario #3
  - If the weights are 0, 1, 0, then the middle child will get all the empty space and the first and third children won't get any

GridLayout



- Arranges children in a row/column grid
- Important attributes
  - rowCount, columnCount
- Child layout attributes
  - layout_row, layout_column – location of child in the grid
  - layout_rowSpan, layout_columnSpan – the number of rows and columns occupied by the child (can span more than one row and one column)
  - layout_gravity – if the child doesn't take up all the space in its group of cells where should it be placed? (center, top-left, bottom-middle, etc.)
  - layout_rowWeight, layout_columnWeight – numbers indicating what proportion of available empty space should be allocated to this child (in both vertical and horizontal dimensions)

RelativeLayout



- Arranges children relative to the sides of the parent and/or relative to each other
- For example, to build the layout above:
  - A
    - layout_alignParentTop, layout_alignParentLeft, layout_alignParentRight
  - B
    - layout_alignParentLeft, layout_alignParentBottom, layout_below="A", layout_toLeftOf="C"
  - C
    - layout_alignParentRight, layout_alignParentBottom, layout_below="A", layout_toRightOf="B"

FrameLayout

- Contains a single child
- Useful for allocating space for widgets that are dynamically added or removed at runtime (like fragments, but could be any widget)

- If a fragment does not dynamically appear and disappear, you do not need to embed it in a FrameLayout. Instead, you can just hard-code the fragment in the XML layout file using a <fragment> element
  - Example: MapFragment contains static Google Map fragment that can be embedded in a <fragment> element

RadioGroup

- Subclass of LinearLayout
  - Can be horizontal or vertical
- RadioButtons nested in RadioGroup are mutually exclusive (only one can be selected at a time)

Layout Exercises

- Login Fragment
- Map Fragment
- Person Activity
- Settings Activity
- What do the layouts for MainActivity and EventActivity have in common?

Layout Exercises

- Login Fragment
- Map Fragment
- Person Activity
- Settings Activity
- What do the layouts for MainActivity and EventActivity have in common?
  - They both contain a single fragment
  - MainActivity is dynamic, so its layout should contain a <FrameLayout>
  - EventActivity is not dynamic, so its layout could contain a <FrameLayout> for <fragment>