● Views
  ○ Rectangles that appear on the screen
  ○ All widgets, UI elements inherit from View
  ○ Handles drawing, event-handling
  ○ Examples: TextView, Button, Radio Buttons, Checkboxes, Date Pickers, . . .

● ViewGroup
  ○ a container for children views
  ○ Needed to display multiple views
  ○ Layouts
Checkboxes and Radio Buttons

- Checkboxes: define in xml and in your onclick method, check if box is checked
- Radio Buttons: define RadioGroup viewgroup, with RadioButton children, and in your onclick method, check to see which button was checked
XML FOR CHECKBOXES

```xml
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <CheckBox android:id="@+id/checkbox_meat"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/meat"
        android:onClick="onCheckboxClicked"/>
    <CheckBox android:id="@+id/checkbox_cheese"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/cheese"
        android:onClick="onCheckboxClicked"/>
</LinearLayout>
```

- Radio buttons are the same: to group, make them all have the same onClick method.
- For more info and starter code, see https://developer.android.com/guide/topics/ui/controls/checkbox
**Layouts**

- **Frame layout**: one element in each z dimension
  - multiple elements are stacked z-wise
- **Linear layout**: lays elements out in a line:
  - you’ve seen this before
  - Can specify weights
Relative Layout

- Allows elements to be positioned relative to each other
- It’s being replaced by the more powerful Constraint Layout
GridLayout

Invasion
View Width/Height

- width = wrap_content, height = wrap_content
- It's a view party

- width = match_parent, height = wrap_content
- It's a view party

- width = wrap_content, height = match_parent
- It's a view party

- width = match_parent, height = match_parent
- It's a view party
gravity: contents within element

layout_gravity: relative to parent
Padding vs Margin

padding = x

layout_margin = x

whitespace
View Visibility

- **visible**
  - [Cloud icon] Rain

- **invisible**
  - [Blank square] Rain

- **gone**
  - Rain
Quiz

Which will the layout render, 1, 2, or 3?
Constraint Layout

- Allows positioning of elements relative to each other by using constraints
  - Similar to RelativeLayout, but avoids nesting
- You must add at least one horizontal and one vertical constraint for the view to another view
CardViews in Your RecyclerView

- Views with consistent:
  - Elevation
  - Corner radius
  - Other attributes

- Need to add the dependency to gradle
Picasso: adding thumb images to recycler views

- Picasso is a library that handles image caching and other busiwork involved in putting images in listviews/recyclerviews. To use:
- Add the dependency to gradle,
- Get urls for the thumb images from your api (do the appropriate parsing of your json, make model class changes, store the image url in your db, etc
- Then use it like this:

```java
public void bind(int pos){
    cursor.moveToPosition(pos);
    title.setText(cursor.getString(cursor.getColumnIndex(Contract.TABLE_ARTICLES.COLUMN_NAME_TITLE)));
    abstr.setText(cursor.getString(cursor.getColumnIndex(Contract.TABLE_ARTICLES.COLUMN_NAME_PUBLISHED_DATE)));
    abstr.append(" ");
    abstr.append(cursor.getString(cursor.getColumnIndex(Contract.TABLE_ARTICLES.COLUMN_NAME_ABSTRACT)));
    String url = cursor.getString(cursor.getColumnIndex(Contract.TABLE_ARTICLES.COLUMN_NAME_THUMBURL));
    Log.d(TAG, url);
    if(url != null){
        Picasso.get()
            .load(url)
            .into(img);
    }
}
Constraint Layout Demo App: Last Year's News App

Let's look at it. Idea's for improvement?
**AVOID EXCESSIVE NESTING**

- Suggested Max:
- 80 views
- 10 nested viewgroups
- Which is more efficient, A or B?
Landscape Mode?

- Constraint Layout will handle some layout changes gracefully.
- But sometimes it's just better to make a separate layout for landscape mode.
- Do this by creating a new folder, layout-land, and put it in your res folder.
- Put a layout of the same name in this folder and adjust.
Including layouts

- Sometimes multiple layouts will share complex components
- The best way to handle these is make the components in a separate xml, and include them when needed

```xml
<include
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/textViewPassengerName"
    android:layout_marginTop="16dp"
    android:id="@+id/flight_info"
    layout="@layout/flight_info"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    app:layout_constraintRight_toRightOf="parent" />
```