Mobile Application Development
Aileen Pierce

ANDROID APP BASICS

Android App Development

- Chose your layout
- Add view objects to your user interface
- Define resources and IDs
- Wire the layout to the activity
- Add the logic to the activity

Views

- The **`View`** class is the building block for all user interface components
- **`View`** is the base class for all widgets
  - TextView
  - EditView
  - Button
  - ImageView
  - Check box
  - and many others

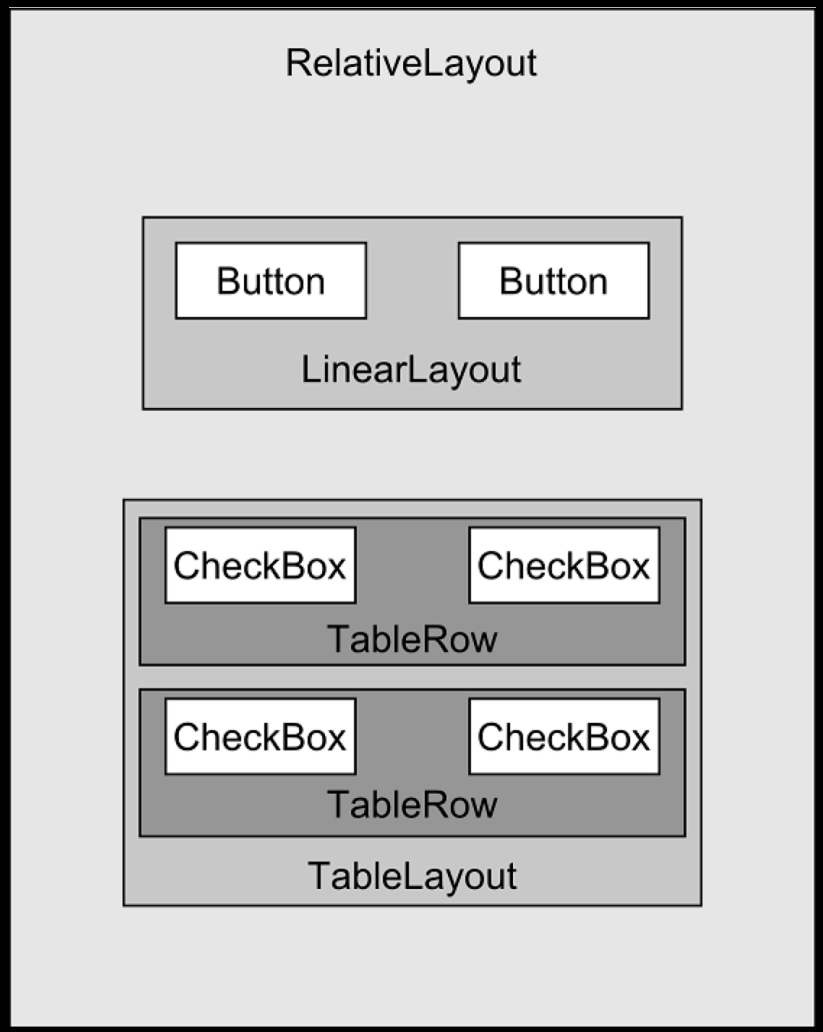
View Groups

- **`ViewGroup`** is a subclass of **`View`** that can contain other views
  - Menus
  - Lists
  - Radio groups
  - Web views
  - Spinner
  - Layouts
  - and many others

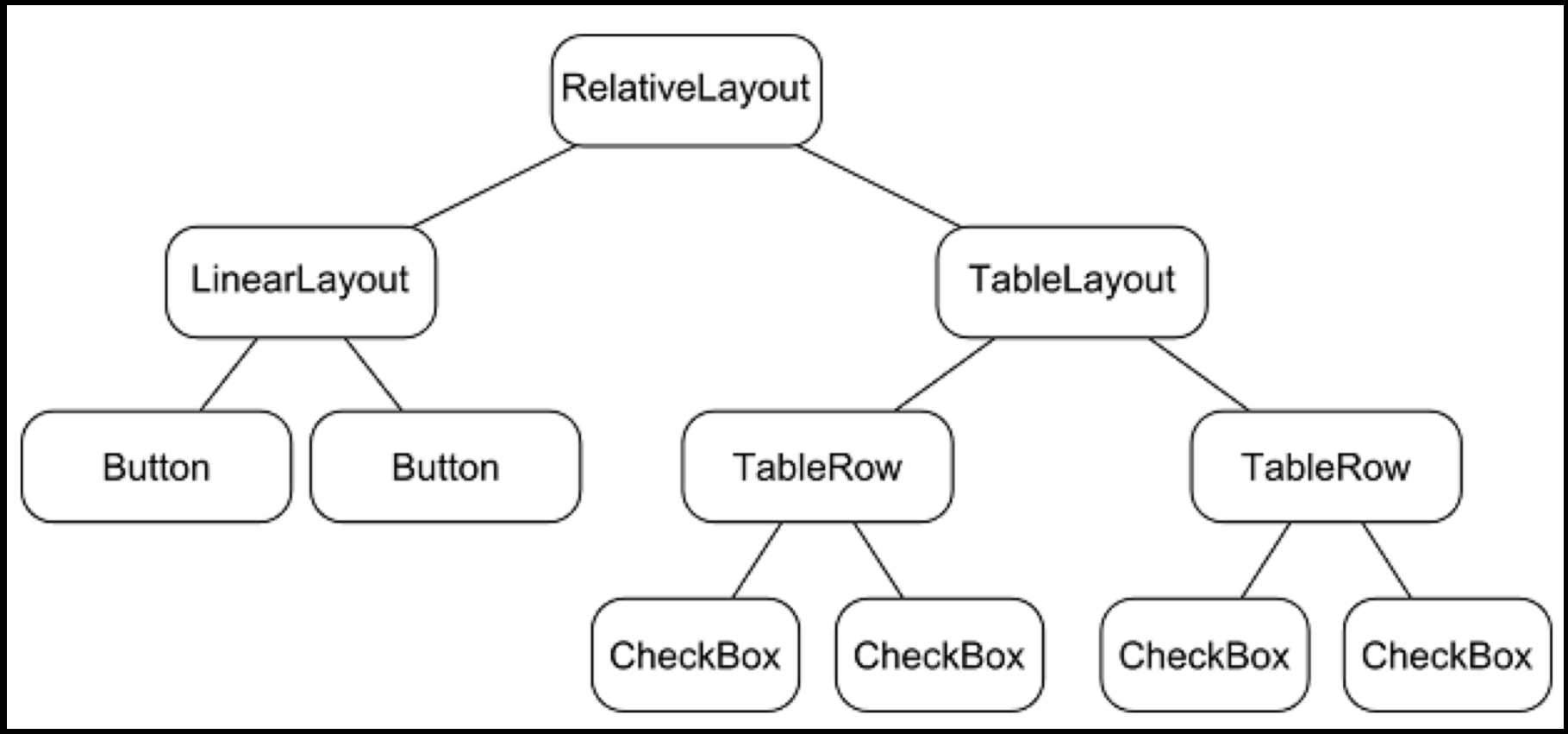
Layouts

- Linear Layout
  - Arranges views next to each other either horizontally or vertically
- Relative Layout
  - Arranges views relative to each other as well as to their parent view
- Table Layout
- Frame Layout
- Absolute Layout
- Grid Layout

Layouts


RelativeLayout
Button
Button
LinearLayout
CheckBox
CheckBox
TableRow
CheckBox
CheckBox
TableRow
TableLayout

Layouts


RelativeLayout
LinearLayout
TableLayout
Button
Button
TableRow
TableRow
CheckBox
CheckBox
CheckBox
CheckBox

Relative Layouts

- In relative layouts you can position views relative to a parent
  - layout_alignParent (Top, Bottom, Right, Left)
  - center (InParent, Horizontal, Vertical)
- You can also position views relative to other views
  - Above or below another view
  - Align (Top, Bottom, Right, Left)
  - toLeftOf, toRightOf

Attributes

- **`android:layout_width`** and **`android:layout_height`** are required attributes for almost all widgets
  - match_parent – view will be as big as its parent
  - wrap_content – view will be as big as its contents require

Attributes

- Padding attributes determine how much padding you want between each of its layout's sides and its parent.
  - padding (Top, Bottom, Right, Left)
- Margins add room between views
  - layout_margin (Top, Bottom, Right, Left)

Linear Layouts

- A linear layout displays its views next to each other, either vertically or horizontally.
- **`android:orientation`** determines which direction you want to arrange views in, horizontal or vertical
- **`layout:weight`** gives a view more room in the layout
- **`gravity`** specifies how you want to position the contents inside a view

Properties

- **`android:id`**
  - Gives the component a unique identifying name
  - Lets you access the widget in your code
  - Lets you refer to the widget in your layout
- **`android:text`**
  - the text displayed in that component

Resources

- A resource is a part of your app that is not code – images, audio, xml, etc
- You should use resources for strings instead of hard coding their values
  - Easier to make changes
  - Localization

`android:text="@string/heading"`

  - **`@string`** indicates it's a string in the strings.xml resource file
  - heading is the name of the string

Java

- All the logic for Android apps are written in Java
- Next week: Java!