Building custom components

http://javat.in

Building custom components

- Custom component can be built by Subclassing View. This can be useful even if few alterations to a specific View or Widget are needed.
- Here is an overview for creating your own View
  1. Extend an existing View class or subclass with your own class.
  2. Override some of the methods from the superclass.
  3. Use your new extension class.

Fully Customized Components

- To create a fully customized component:
  1. Extend the most generic view View.
  2. Supply a constructor which can take attributes and parameters from the XML
  3. Create your own event listeners, property accessors and modifiers, and possibly more sophisticated behavior in your component class as well.
  4. Override onMeasure() and onDraw() if you want the component to show something.
     While both have default behavior, the default onDraw() will do nothing, and the default onMeasure() will always set a size of 100x100 .
  5. Other on... methods may also be overridden as required.

Extend onDraw() and onMeasure()

protected void onDraw (Canvas canvas)

protected void onMeasure (int widthMeasureSpec,
    int heightMeasureSpec)

- The onDraw() method delivers you a Canvas upon which you can implement anything you want: 2D graphics.
- onMeasure() is a critical piece of the rendering contract between your component and its container. onMeasure() should be overridden to efficiently and accurately report the measurements of its contained parts.
- call the setMeasuredDimension() method with the measured width and height once they have been calculated.
  If you fail to call this method from an overridden onMeasure() method, the result will be an exception at measurement time.

Extend onDraw() and onMeasure()

- If onMeasure() method is overridden, it is the subclass's responsibility to make sure the measured height and width are at least the view's minimum height and width (getSuggestedMinimumHeight() and getSuggestedMinimumWidth()).

Compound Controls

- Compound controls brings together a number of more atomic controls (or views) into a logical group of items that can be treated as a single thing.
- *Compound controls are atomic, reusable widgets that contain multiple child controls laid out and wired* together.
- To create a compound component:
  - Create a class that extends a Layout. Just like with an Activity, either use the declarative (XML-based) approach to creating the contained components, or nest them programmatically from code.
  - In the constructor for the new class, take whatever parameters the superclass expects, and pass them through to the superclass constructor first. Then you can set up the other views to use within your new component.

Compound Controls

- You can also create listeners for events that your contained views might generate.
- You might also create your own properties with accessors and modifiers.
- In the case of extending a Layout, you don't need to override the onDraw() and onMeasure() methods since the layout will have default behavior that will likely work just fine. However, you can still override them if you need to.
- You might override other on... methods, like onKeyDown(), to perhaps choose certain default values from the popup list of a combo box when a certain key is pressed.

Modifying an Existing View Type

- If there is a component that is already very similar to what you want, you can simply extend that component and just override the behavior that you want to change.