

# PROGRAMMING HANDHELD SYSTEMS

ADAM PORTER

# USER INTERFACE CLASSES

# TODAY'S TOPICS

VIEWS & VIEW EVENTS

VIEW GROUPS, ADAPTERVIEWS & LAYOUTS

MENUS & ACTIONBAR

DIALOGS

# ANDROID USER INTERFACES

ACTIVITIES USUALLY DISPLAY A USER INTERFACE

ANDROID PROVIDES MANY CLASSES FOR CONSTRUCTING USER INTERFACES

# VIEW

KEY BUILDING BLOCK FOR UI  
COMPONENTS

OCCUPY A RECTANGULAR SPACE ON  
SCREEN

RESPONSIBLE FOR DRAWING  
THEMSELVES AND FOR HANDLING  
EVENTS

# SOME PREDEFINED VIEWS

BUTTON

TOGGLEBUTTON

CHECKBOX

RATINGBAR

AUTOCOMPLETETEXTVIEW

# BUTTON

VIEW THAT CAN BE CLICKED ON TO PERFORM  
AN ACTION

# UIButton



# UIButton

```
<Button  
    android:id="@+id/button"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:layout_alignParentBottom="true"  
    android:layout_marginLeft="10dip"  
    android:text="@string/press_me_string" >  
</Button>
```

```
// Get a reference to the Press Me Button  
final Button button = (Button) findViewById(R.id.button);  
  
// Set an OnClickListener on this Button  
// Called each time the user clicks the Button  
button.setOnClickListener(new OnClickListener() {  
    @Override  
    public void onClick(View v) {  
  
        // Maintain a count of user presses  
        // Display count as text on the Button  
        button.setText("Got Pressed:" + ++count);  
  
    }  
});
```

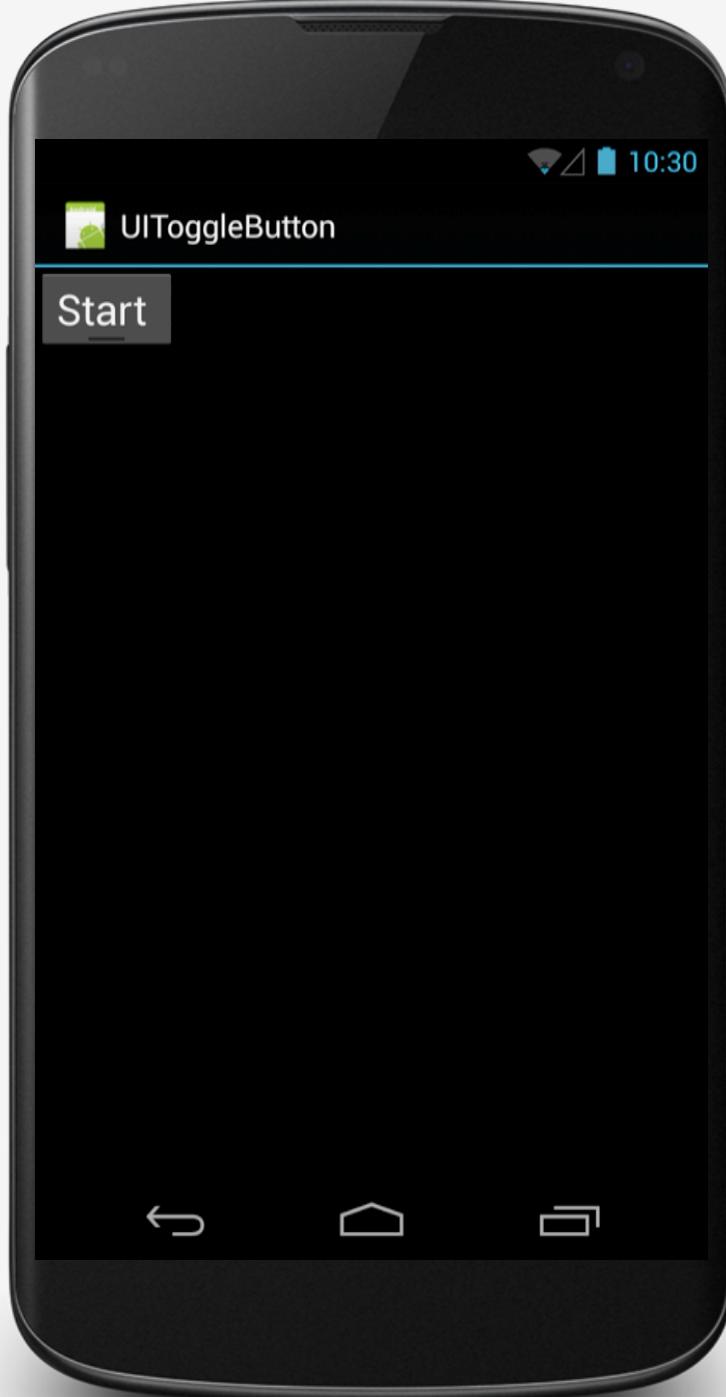
# TOGGLEBUTTON

A 2-STATE BUTTON

CHECKED/NOT CHECKED STATE

LIGHT INDICATOR SHOWING STATE

# UITOGGLEBUTTON



# UITOGGLEBUTTON

```
<ToggleButton  
    android:id="@+id/togglebutton"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:textOff="@string/start_string"  
    android:textOn="@string/stop_string"  
    android:textSize="24sp" />
```

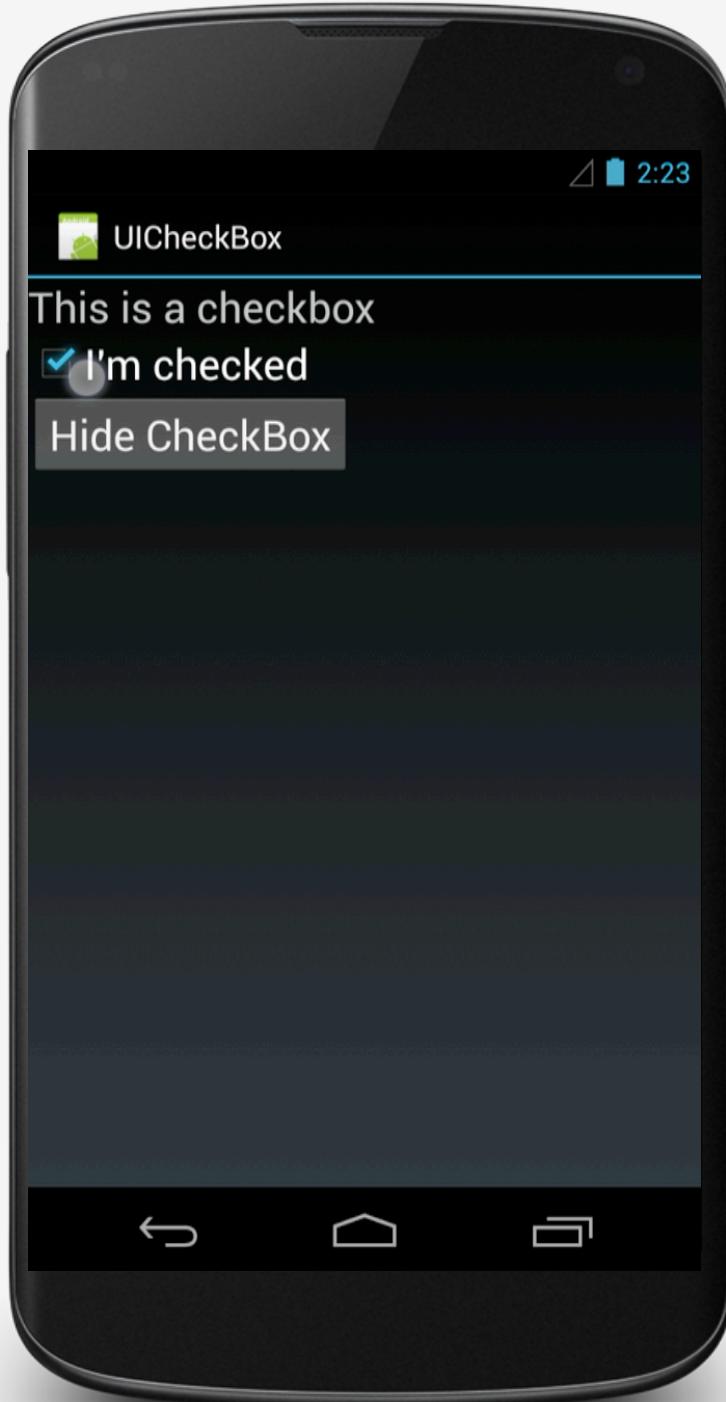
```
// Get a reference to the ToggleButton  
final ToggleButton button = (ToggleButton) findViewById(R.id.togglebutton);  
  
// Set an OnClickListener on the ToggleButton  
button.setOnClickListener(new OnClickListener() {  
    @Override  
    public void onClick(View v) {  
  
        // Toggle the Background color between a light and dark color  
        if (button.isChecked()) {  
            bg.setBackgroundColor(0xFFFF3F3F);  
        } else {  
            bg.setBackgroundColor(0xFF000000);  
        }  
    }  
});
```

CHECKBOX

ANOTHER KIND OF 2-STATE BUTTON

CHECKED/NOT CHECKED

# UICHECKBOX



# UICHECKBOX

```
<CheckBox  
    android:id="@+id/checkbox"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text="@string/im_not_checked_string"  
    android:textSize="24sp"/>
```

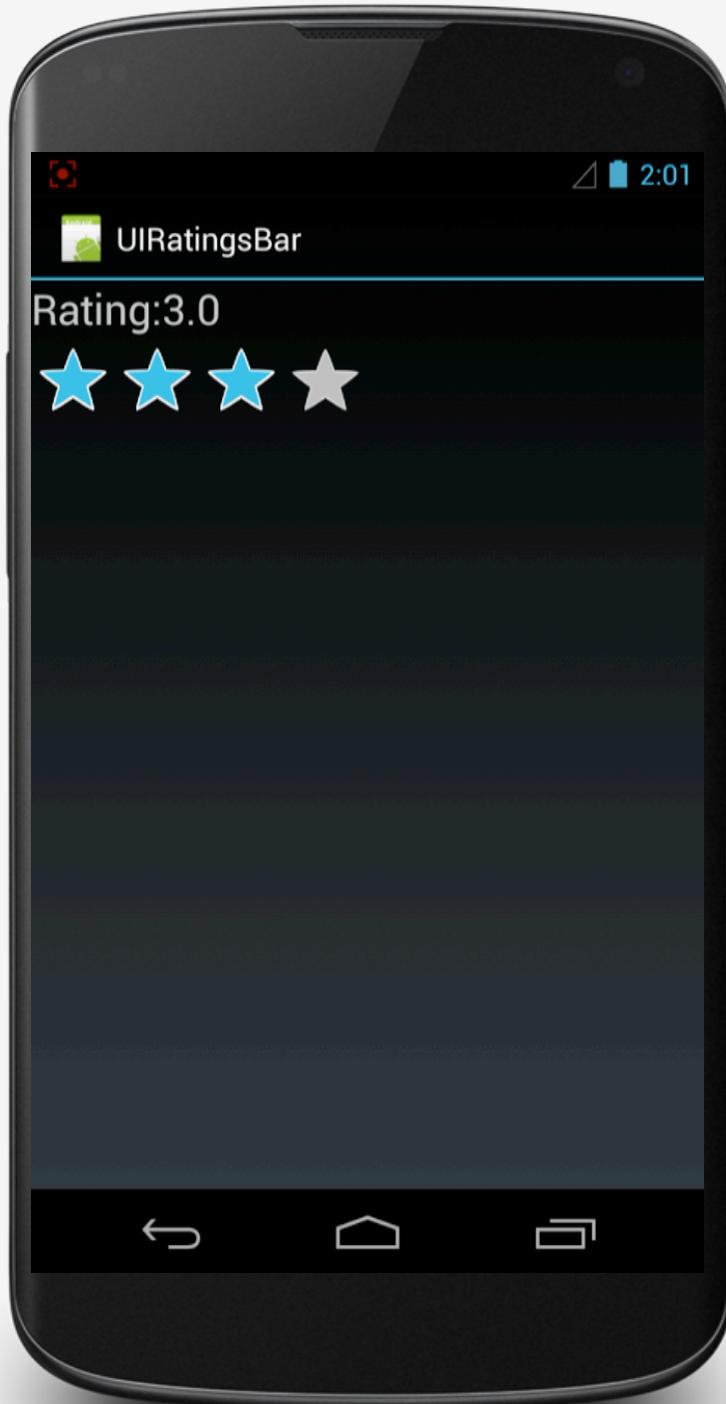
```
// Get a reference to the CheckBox  
final CheckBox checkbox = (CheckBox) findViewById(R.id.checkbox);  
  
// Set an OnClickListener on the CheckBox  
checkbox.setOnClickListener(new OnClickListener() {  
    @Override  
    public void onClick(View v) {  
  
        // Check whether CheckBox is currently checked  
        // Set CheckBox text accordingly  
        if (checkbox.isChecked()) {  
            checkbox.setText("I'm checked");  
        } else {  
            checkbox.setText("I'm not checked");  
        }  
    }  
});
```

# RATINGBAR

A VIEW COMPRISING A ROW OF STARS

THE USER CAN CLICK OR DRAG THE STARS TO  
HIGHLIGHT SOME NUMBER OF THEM

# UIRATINGBAR



# UI RATING BAR

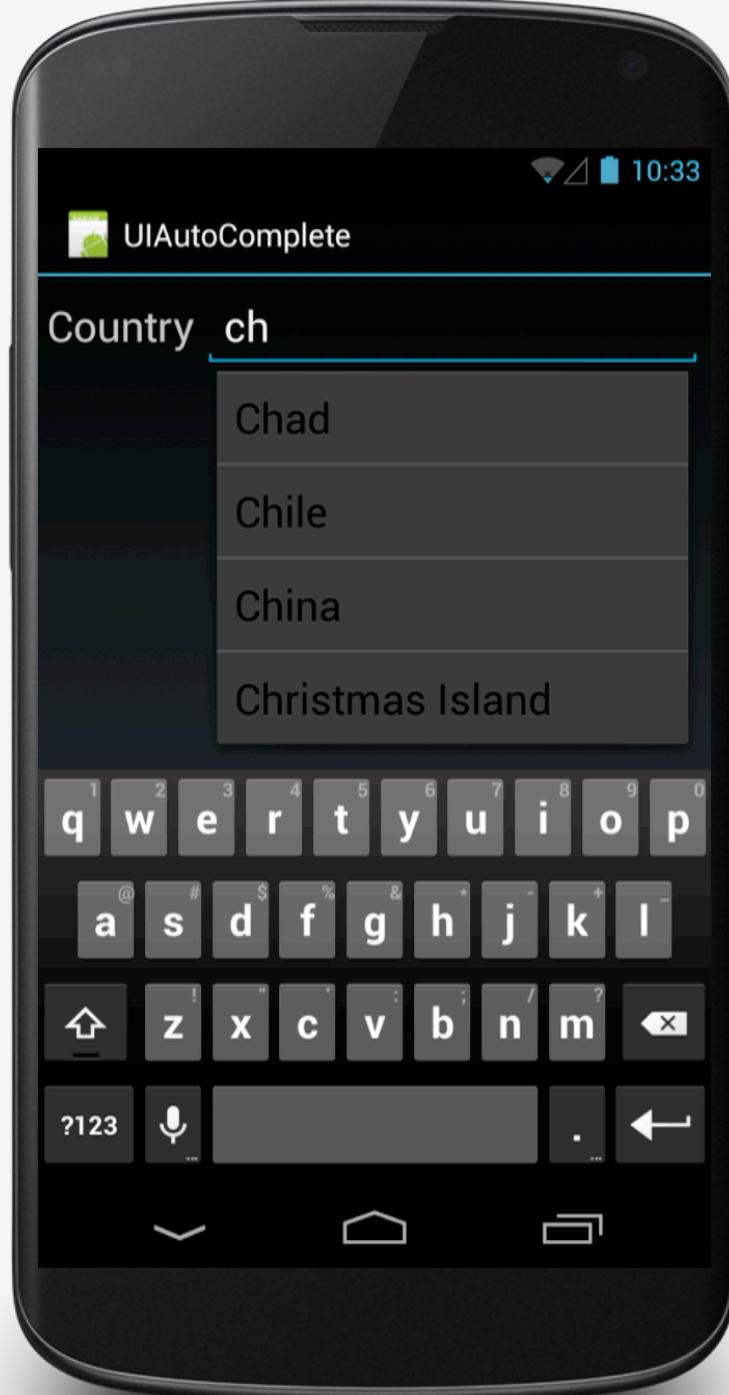
```
<RatingBar  
    android:id="@+id/ratingbar"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:numStars="4"  
    android:stepSize="1.0" >  
</RatingBar>
```

```
final RatingBar bar = (RatingBar) findViewById(R.id.ratingbar);  
  
bar.setOnRatingBarChangeListener(new OnRatingBarChangeListener() {  
  
    // Called when the user swipes the RatingBar  
    @Override  
    public void onRatingChanged(RatingBar ratingBar, float rating, boolean fromUser) {  
        tv.setText("Rating:" + rating);  
    }  
});
```

# AUTOCOMPLETETEXTVIEW

AN EDITABLE TEXT FIELD THAT PROVIDES  
COMPLETION SUGGESTIONS AS THE USER  
TYPES IN TEXT

# UIAutoComplete TEXTVIEW



# UIAUTOCOMPLETE

```
<AutoCompleteTextView  
    android:id="@+id/autocomplete_country"  
    android:layout_width="fill_parent"  
    android:layout_height="wrap_content"  
    android:layout_marginLeft="5dp"  
    android:textSize="24sp"/>
```

```
// Get a reference to the AutoCompleteTextView  
AutoCompleteTextView textView = (AutoCompleteTextView) findViewById(R.id.autocomplete_country);  
  
// Create an ArrayAdapter containing country names  
ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,  
    R.layout.list_item, COUNTRIES);  
  
// Set the adapter for the AutoCompleteTextView  
textView.setAdapter(adapter);
```

# COMMON VIEW OPERATIONS

SET VISIBILITY: SHOW OR HIDE VIEW

SET CHECKED STATE

SET LISTENERS: CODE THAT SHOULD BE  
EXECUTED WHEN SPECIFIC EVENTS OCCUR

SET PROPERTIES: OPACITY, BACKGROUND,  
ROTATION

MANAGE INPUT FOCUS: ALLOW VIEW TO  
TAKE FOCUS, REQUEST FOCUS

# VIEW EVENT SOURCES

USER INTERACTION

TOUCH

KEYBOARD/TRACKBALL/D-PAD

SYSTEM CONTROL

LIFECYCLE CHANGES

# HANDLING VIEW EVENTS

OFTEN HANDLE EVENTS WITH LISTENERS

NUMEROUS LISTENER INTERFACES DEFINED BY  
THE VIEW CLASS

# VIEW LISTENER INTERFACES

OnClickListener.onClick()

VIEW HAS BEEN CLICKED

OnLongClickListener.onLongClick()

VIEW HAS BEEN PRESSED & HELD

# VIEW LISTENER INTERFACES

OnFocusChangeListener.  
onFocusChange()

VIEW HAS RECEIVED OR LOST FOCUS

OnKeyListener.onKey()

VIEW ABOUT TO RECEIVE A HARDWARE  
KEY PRESS

# DISPLAYING VIEWS

VIEWS ARE ORGANIZED IN A TREE

DISPLAYING HAS MULTIPLE STEPS

MEASURE – GET DIMENSIONS OF EACH VIEW

LAYOUT – POSITION EACH VIEW

DRAW – DRAW EACH VIEW

# HANDLING VIEW EVENTS

CUSTOM VIEW SUBCLASSES CAN OVERRIDE  
VARIOUS VIEW METHODS

# HANDLING VIEW EVENTS

onMeasure()

DETERMINE THE SIZE OF THIS VIEW AND ITS CHILDREN

onLayout()

VIEW MUST ASSIGN A SIZE AND POSITION TO ALL ITS CHILDREN

onDraw()

VIEW SHOULD RENDER ITS CONTENT

# HANDLING VIEW EVENTS

onFocusChanged()

VIEW'S FOCUS STATE HAS CHANGED

onKeyUp(), onKeyDown()

A HARDWARE KEY EVENT HAS OCCURRED

onWindowVisibilityChanged()

WINDOW CONTAINING VIEW HAS CHANGED ITS  
VISIBILITY STATUS

# VIEWGROUP

AN INVISIBLE VIEW THAT CONTAINS  
OTHER VIEWS

USED FOR GROUPING & ORGANIZING A  
SET OF VIEWS

BASE CLASS FOR VIEW CONTAINERS &  
LAYOUTS

# SOME PREDEFINED VIEWGROUPS

RADIOGROUP

TIMEPICKER

DATEPICKER

WEBVIEW

MAPVIEW

GALLERY

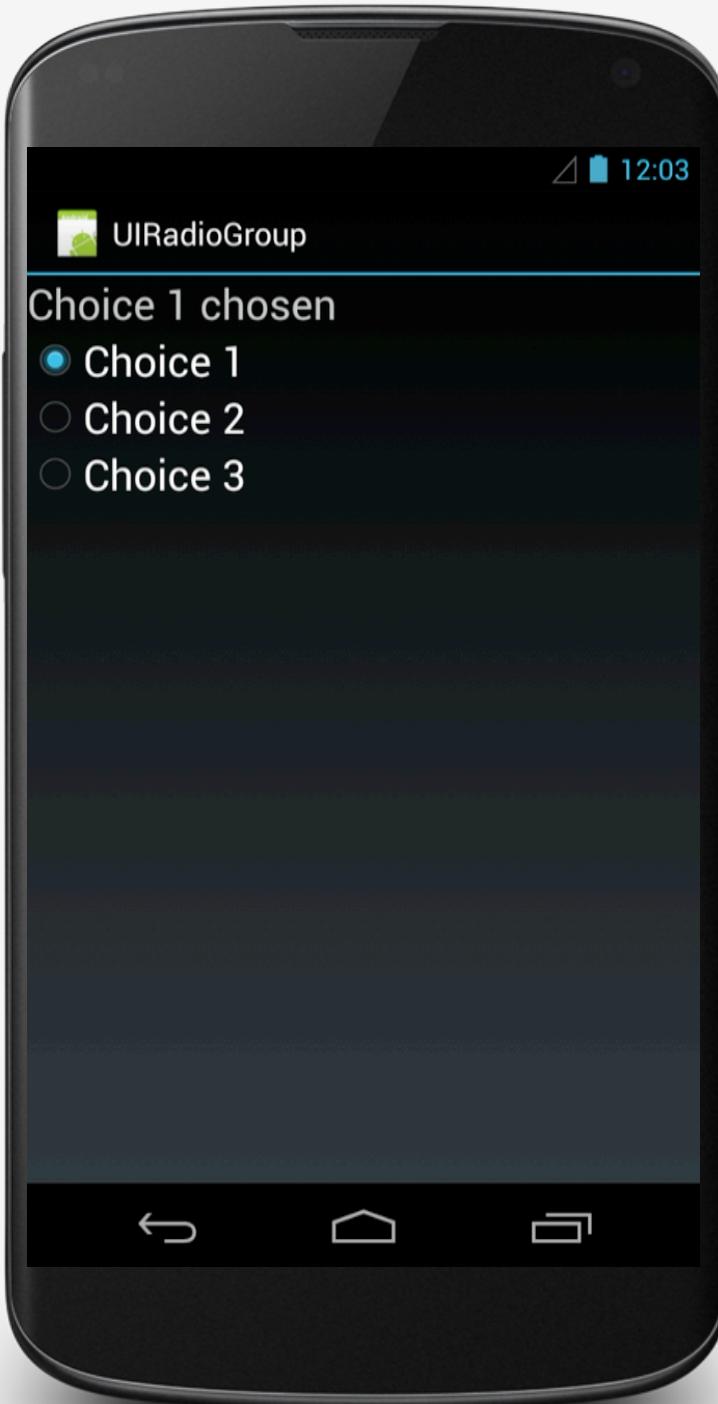
SPINNER

# RADIOGROUP

A VIEWGROUP CONTAINING A SET OF  
RADIO BUTTONS (CHECKBOXES)

ONLY ONE BUTTON CAN BE SELECTED AT  
ANY ONE INSTANT

# UIRADIOGROUP



# UIRADIOGROUP

```
<RadioGroup
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    >

    <RadioButton
        android:id="@+id/choice1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/choice_1_string"
        android:textSize="24sp"/>

    <RadioButton
        android:id="@+id/choice2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/choice_2_string"
        android:textSize="24sp"/>

    <RadioButton
        android:id="@+id/choice3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/choice_3_string"
        android:textSize="24sp" />
</RadioGroup>
```

# UIRADIOGROUP

```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    final TextView tv = (TextView) findViewById(R.id.textView);

    // Define a generic listener for all three RadioButtons in the RadioGroup
    final OnClickListener radioListener = new OnClickListener() {
        @Override
        public void onClick(View v) {
            RadioButton rb = (RadioButton) v;
            tv.setText(rb.getText() + " chosen");
        }
    };

    final RadioButton choice1 = (RadioButton) findViewById(R.id.choice1);
    // Called when RadioButton choice1 is clicked
    choice1.setOnClickListener(radioListener);

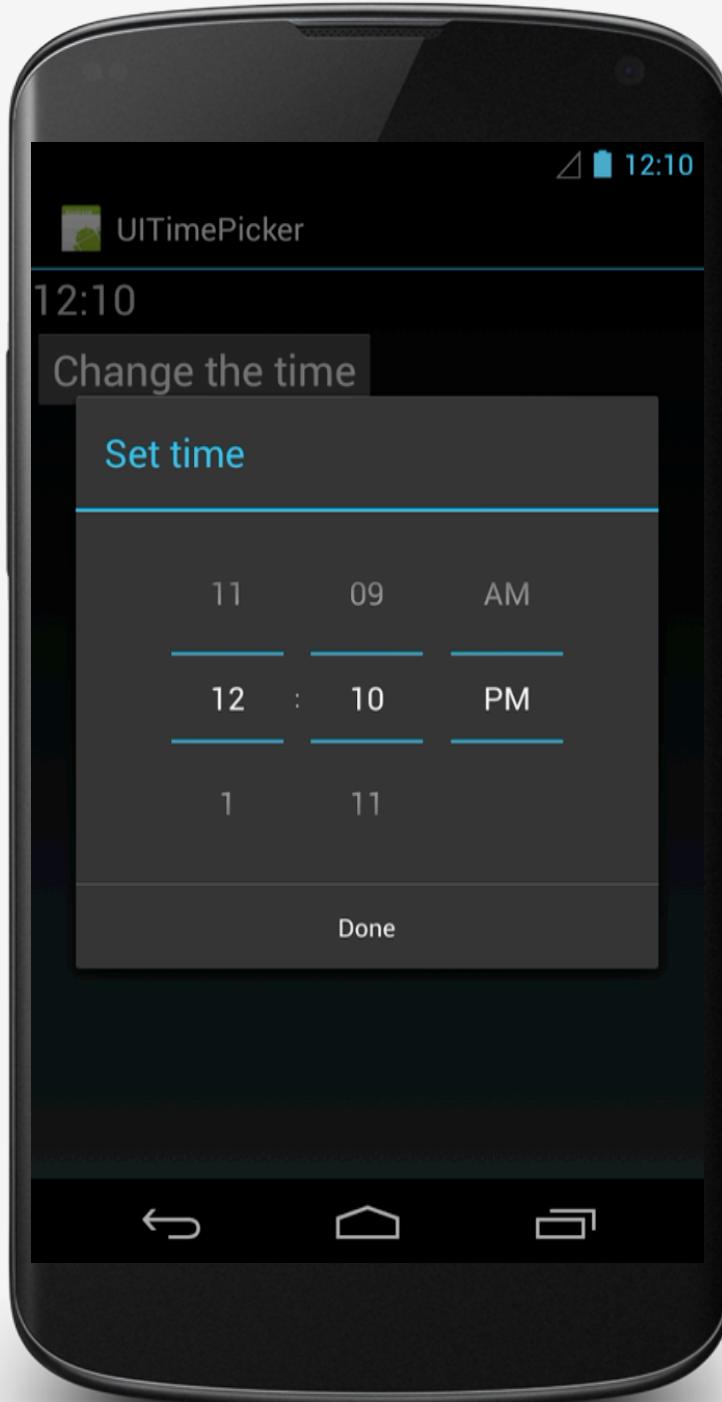
    final RadioButton choice2 = (RadioButton) findViewById(R.id.choice2);
    // Called when RadioButton choice2 is clicked
    choice2.setOnClickListener(radioListener);

    final RadioButton choice3 = (RadioButton) findViewById(R.id.choice3);
    // Called when RadioButton choice3 is clicked
    choice3.setOnClickListener(radioListener);
}
```

# TIMEPICKER

A VIEWGROUP THAT ALLOWS THE USER TO  
SELECT A TIME

# UITIMEPICKER



# UITIMEPICKER

```
<TextView  
    android:id="@+id/timeDisplay"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text=""  
    android:textSize="24sp" />  
  
<Button  
    android:id="@+id/pickTime"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text="@string/change_the_time_string"  
    android:textSize="24sp" />
```

# UITIMEPICKER

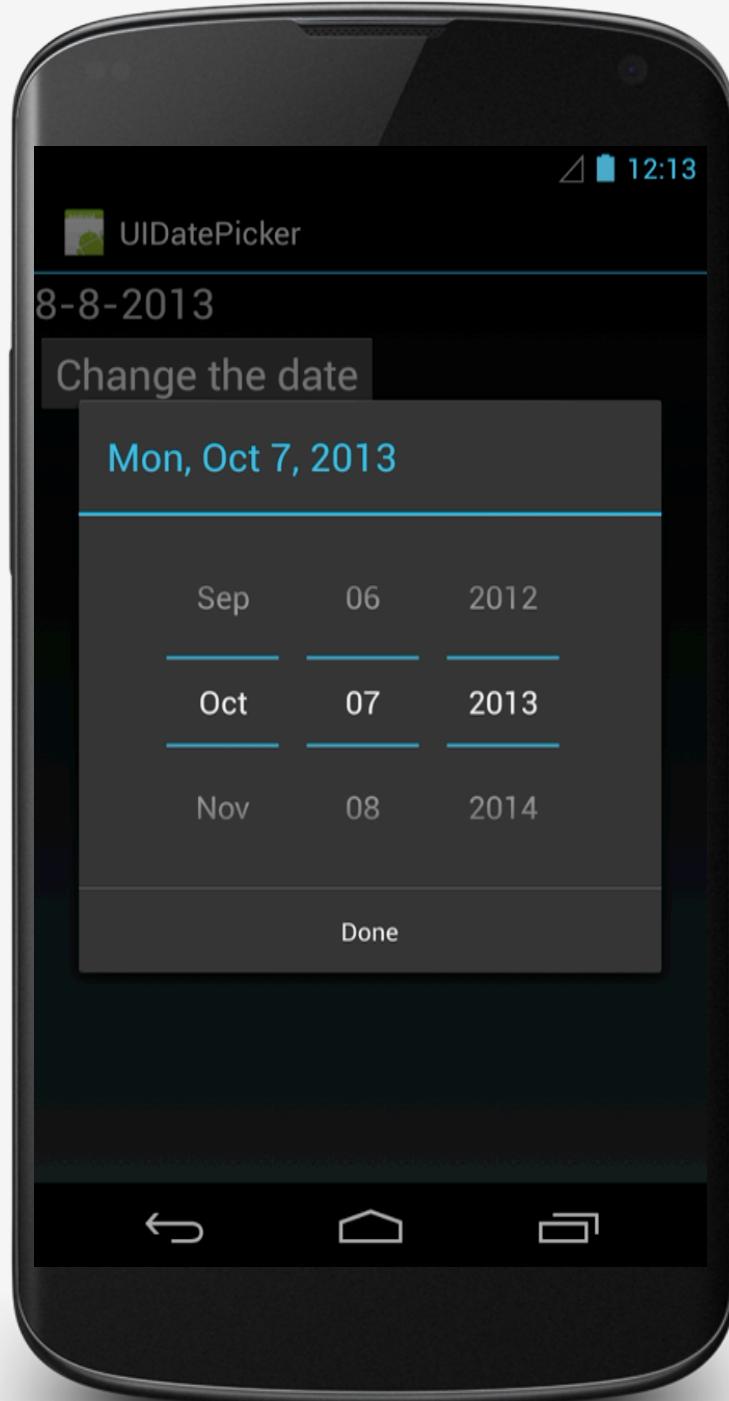
```
// The callback received when the user "sets" the time in the dialog
private TimePickerDialog.OnTimeSetListener mTimeSetListener = new TimePickerDialog.OnTimeSetListener() {
    public void onTimeSet(TimePicker view, int hourOfDay, int minute) {
        mHour = hourOfDay;
        mMinute = minute;
        updateDisplay();
    }
};
```

```
@Override
protected Dialog onCreateDialog(int id) {
    switch (id) {
        case TIME_DIALOG_ID:
            return new TimePickerDialog(this, mTimeSetListener, mHour, mMinute,
                false);
        }
    return null;
}
```

# DATEPICKER

A VIEWGROUP THAT ALLOWS THE USER TO  
SELECT A DATE

# UIDATEPICKER



# UIDATEPICKER

```
<TextView  
    android:id="@+id/dateDisplay"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text=""  
    android:textSize="24sp" />  
  
<Button  
    android:id="@+id/pickDate"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text="@string/change_the_date_string"  
    android:textSize="24sp" />
```

# UIDATEPICKER

```
// The callback received when the user "sets" the date in the Dialog
private DatePickerDialog.OnDateSetListener mDateSetListener = new DatePickerDialog.OnDateSetListener() {

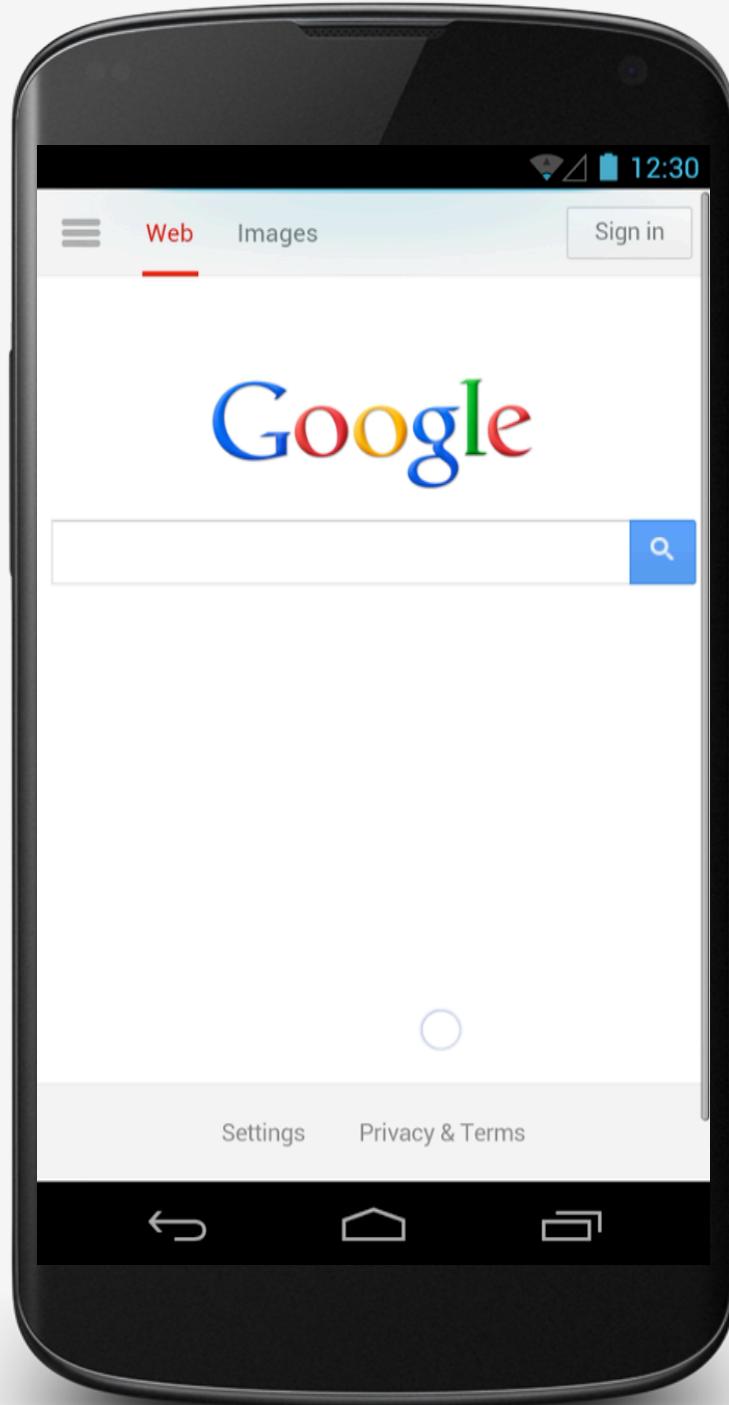
    public void onDateSet(DatePicker view, int year, int monthOfYear,
        int dayOfMonth) {
        mYear = year;
        mMonth = monthOfYear;
        mDay = dayOfMonth;
        updateDisplay();
    }
};
```

```
// Create and return DatePickerDialog
@Override
protected Dialog onCreateDialog(int id) {
    switch (id) {
        case DATE_DIALOG_ID:
            return new DatePickerDialog(this, mDateSetListener, mYear, mMonth,
                mDay);
    }
    return null;
}
```

# WEBVIEW

A VIEWGROUP THAT DISPLAYS A WEB PAGE

# UIWEBVIEW



# UIWEBVIEW

```
<WebView  xmlns:android="http://schemas.android.com/apk/res/android"  
        android:id="@+id/webview"  
        android:layout_width="match_parent"  
        android:layout_height="match_parent"  
    />
```

# UIWEBVIEW

```
mWebView = (WebView) findViewById(R.id.webview);

// Set a kind of listener on the WebView so the WebView can intercept
// URL loading requests if it wants to

mWebView.setWebViewClient(new HelloWebViewClient());

mWebView.getSettings().setJavaScriptEnabled(true);
mWebView.loadUrl("http://www.google.com");

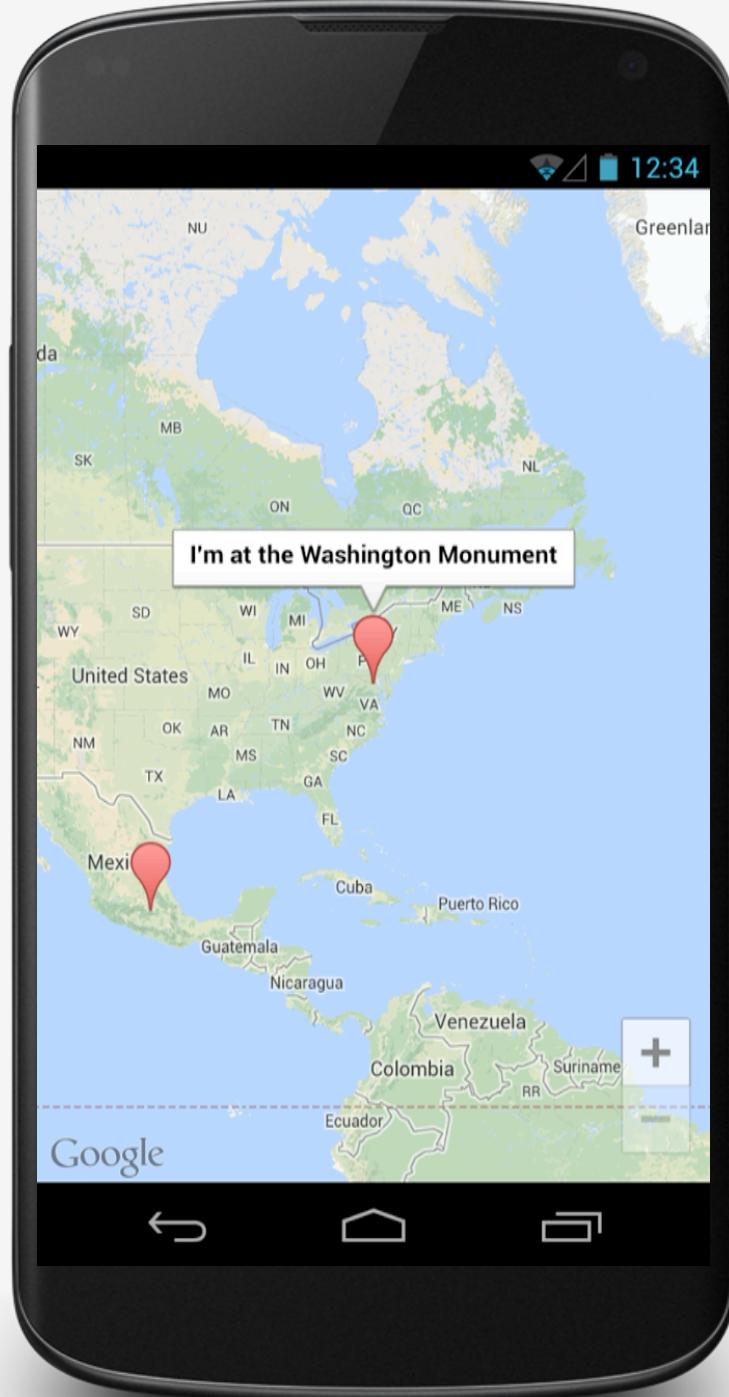
private class HelloWebViewClient extends WebViewClient {
    private static final String TAG = "HelloWebViewClient";

    // Give application a chance to catch additional URL loading requests
    @Override
    public boolean shouldOverrideUrlLoading(WebView view, String url) {
        Log.i(TAG, "About to load:" + url);
        view.loadUrl(url);
        return true;
    }
}
```

# MAPVIEW

A VIEWGROUP THAT DISPLAYS A MAP

# UIGOOGLEMAPS



# UI GOOGLE MAPS

```
<fragment xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/map"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:name="com.google.android.gms.maps.MapFragment"/>

// The GoogleMap instance underlying the GoogleMapFragment defined in main.xml
mMap = ((MapFragment) getSupportFragmentManager().findFragmentById(R.id.map))
    .getMap();

if (mMap != null) {

    // Set the map position
    mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(new LatLng(29,
        -88), 0));

    // Add a marker on Washington, DC, USA
    mMap.addMarker(new MarkerOptions().position(
        new LatLng(38.8895, -77.0352)).title(
        getString(R.string.in_washington_string)));

    // Add a marker on Mexico City, Mexico
    mMap.addMarker(new MarkerOptions().position(
        new LatLng(19.13, -99.4)).title(
        getString(R.string.in_mexico_string)));
}
```

# ADAPTERS & ADAPTERVIEWS

ADAPTERVIEWS ARE VIEWS WHOSE CHILDREN AND DATA ARE MANAGED BY AN ADAPTER

ADAPTER MANAGES THE DATA AND PROVIDES DATA VIEWS TO ADAPTERVIEW

ADAPTERVIEW DISPLAYS THE DATA VIEWS

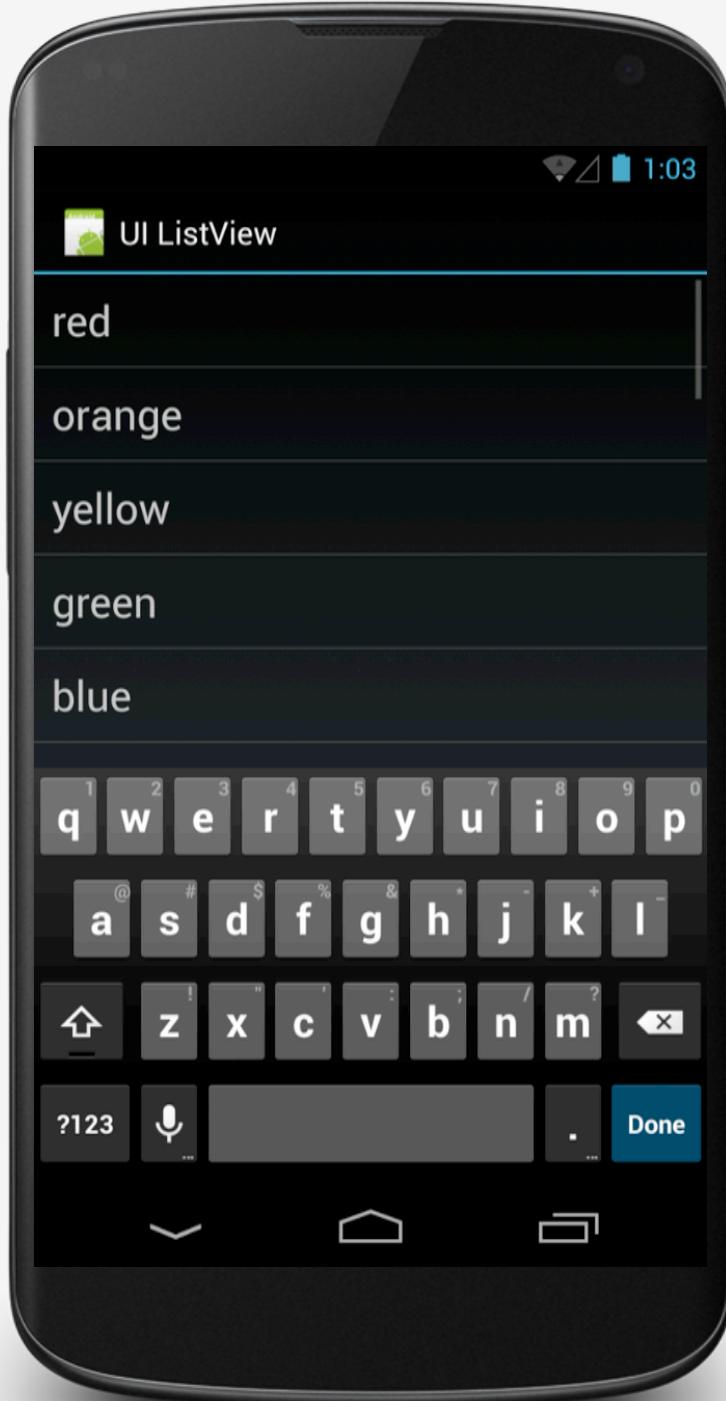
# LISTVIEW

ADAPTERVIEW DISPLAYING A  
SCROLLABLE LIST OF SELECTABLE ITEMS

ITEMS MANAGED BY A LISTADAPTER

LISTVIEW CAN FILTER THE LIST OF ITEMS  
BASED ON TEXT INPUT

# UILISTVIEW



# UILISTVIEW

```
// Create a new Adapter containing a list of colors
// Set the adapter on this ListActivity's built-in ListView
setListAdapter(new ArrayAdapter<String>(this, R.layout.list_item,
    getResources().getStringArray(R.array.colors)));

ListView lv = getListView();

// Enable filtering when the user types in the virtual keyboard
lv.setTextFilterEnabled(true);

// Set an setOnItemClickListener on the ListView
lv.setOnItemClickListener(new OnItemClickListener() {
    public void onItemClick(AdapterView<?> parent, View view,
        int position, long id) {

        // Display a Toast message indicating the selected item
        Toast.makeText(getApplicationContext(),
            ((TextView) view).getText(), Toast.LENGTH_SHORT).show();
    }
});
```

# SPINNER

AN ADAPTERVIEW THAT PROVIDES A  
SCROLLABLE LIST OF ITEMS

USER CAN SELECT ONE ITEM FROM THE LIST

ITEMS MANAGED BY A SPINNERADAPTER

# UI SPINNER



# UI SPINNER

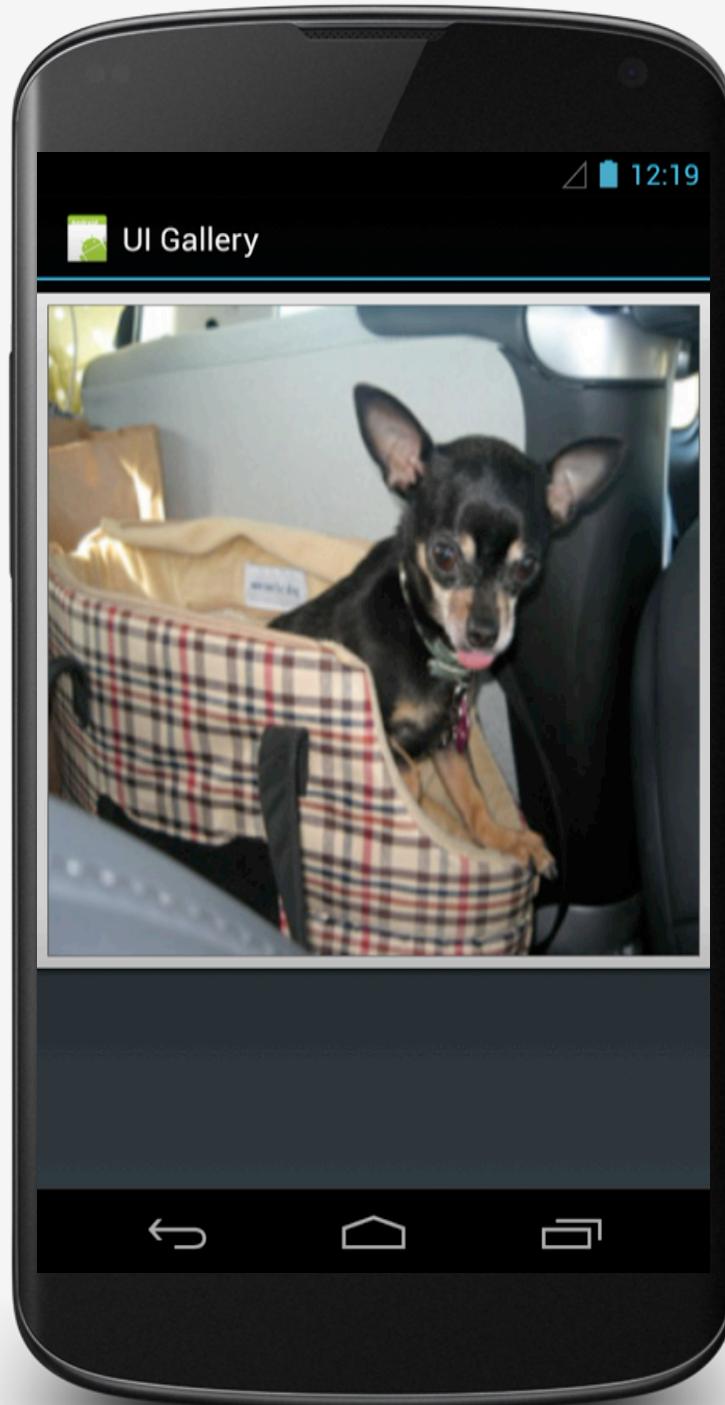
```
<Spinner  
    android:id="@+id/spinner"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:textSize="24sp" />  
  
// Get a reference to the Spinner  
Spinner spinner = (Spinner) findViewById(R.id.spinner);  
  
// Create an Adapter that holds a list of colors  
ArrayAdapter<CharSequence> adapter = ArrayAdapter.createFromResource(  
    this, R.array.colors, R.layout.dropdown_item);  
  
// Set the Adapter for the spinner  
spinner.setAdapter(adapter);  
  
// Set an setOnItemSelectedListener on the spinner  
spinner.setOnItemSelectedListener(new OnItemSelectedListener() {  
    public void onItemSelected(AdapterView<?> parent, View view,  
        int pos, long id) {
```

# GALLERY

A VIEWGROUP SHOWING A HORIZONTALLY SCROLLING LIST

ITEMS MANAGED BY A SPINNERADAPTER

# UI GALLERY



# UI GALLERY

```
<Gallery xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/gallery"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
/>
Gallery g = (Gallery) findViewById(R.id.gallery);

// Create a new ImageAdapter and set in as the Adapter for the Gallery
g.setAdapter(new ImageAdapter(this));

// Set an setOnItemClickListener on the Gallery
g.setOnItemClickListener(new OnItemClickListener() {
    public void onItemClick(AdapterView<?> parent, View v,
        int position, long id) {
        // Display a Toast message indicate the selected item
        Toast.makeText(GalleryActivity.this, "" + position,
            Toast.LENGTH_SHORT).show();
    }
});
```

# UI GALLERY

```
// The Adapter class used with the Gallery
public class ImageAdapter extends BaseAdapter {

    private static final int IMAGE_DIM = 800;

    private int mGalleryItemBackground;
    private Context mContext;

    // List of IDs corresponding to the images
    private Integer[] mImageIds = { R.drawable.sample_1,
        R.drawable.sample_2, R.drawable.sample_3, R.drawable.sample_4,
        R.drawable.sample_5, R.drawable.sample_6, R.drawable.sample_7 };

    public ImageAdapter(Context c) {
        mContext = c;
        TypedArray a = obtainStyledAttributes(R.styleable.GalleryActivity);
        mGalleryItemBackground = a.getResourceId(
            R.styleable.GalleryActivity_android_galleryItemBackground,
            0);
        a.recycle();
    }

    public int getCount() {
        return mImageIds.length;
    }

    public Object getItem(int position) {
        return mImageIds[position];
    }

    public long getItemId(int position) {
        return position;
    }
}
```

# UI GALLERY

```
public View getView(int position, View convertView, ViewGroup parent) {  
    ImageView imageView = (ImageView) convertView;  
  
    // If convertView is not recycled set it up now  
    if (null == imageView) {  
        imageView = new ImageView(mContext);  
  
        imageView.setLayoutParams(new Gallery.LayoutParams(IMAGE_DIM,  
                IMAGE_DIM));  
        imageView.setScaleType(ImageView.ScaleType.FIT_XY);  
        imageView.setBackgroundResource(mGalleryItemBackground);  
    }  
  
    // Set the image for the imageView  
    imageView.setImageResource(mImageIds[position]);  
  
    return imageView;  
}
```

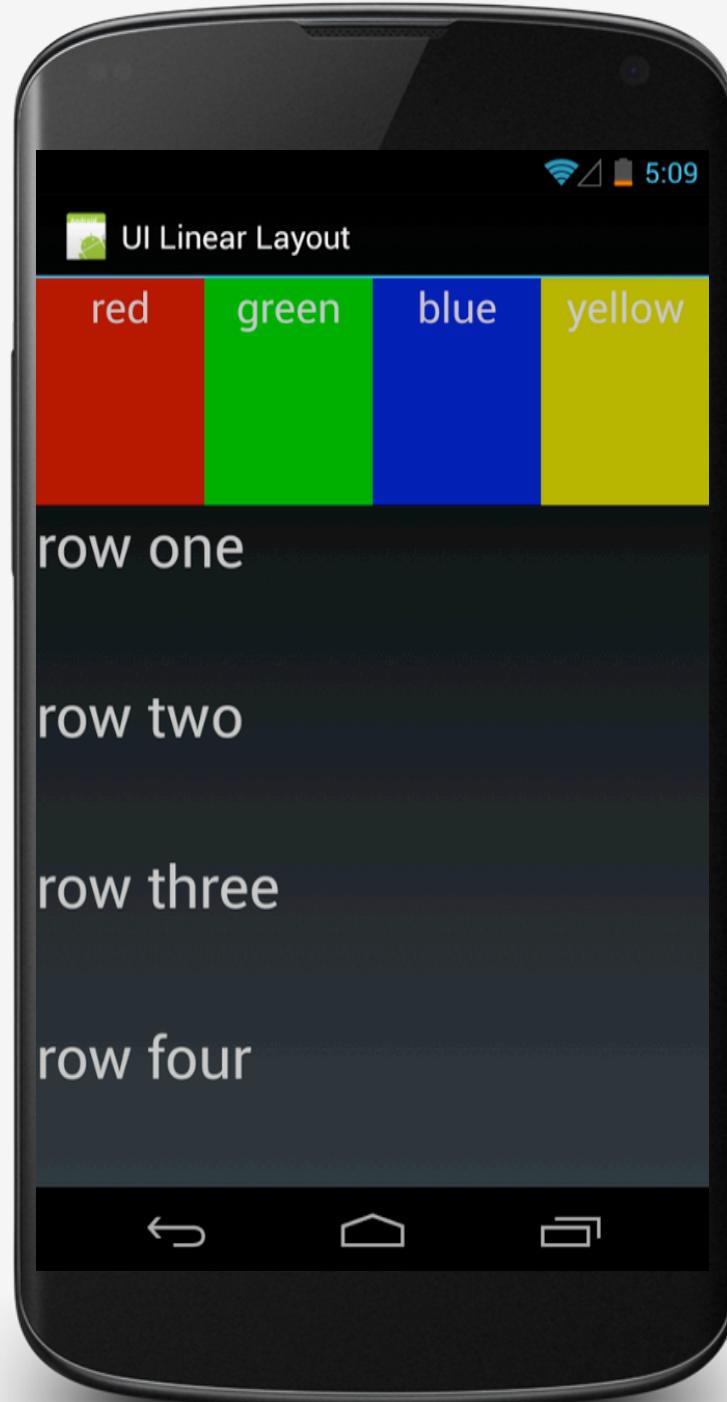
# LAYOUTS

A GENERIC VIEWGROUP THAT DEFINES A  
STRUCTURE FOR THE VIEWS IT CONTAINS

# LINEAR LAYOUT

CHILD VIEWS ARRANGED IN A SINGLE  
HORIZONTAL OR VERTICAL ROW

# UILINEARLAYOUT



# UILINEAR LAYOUT

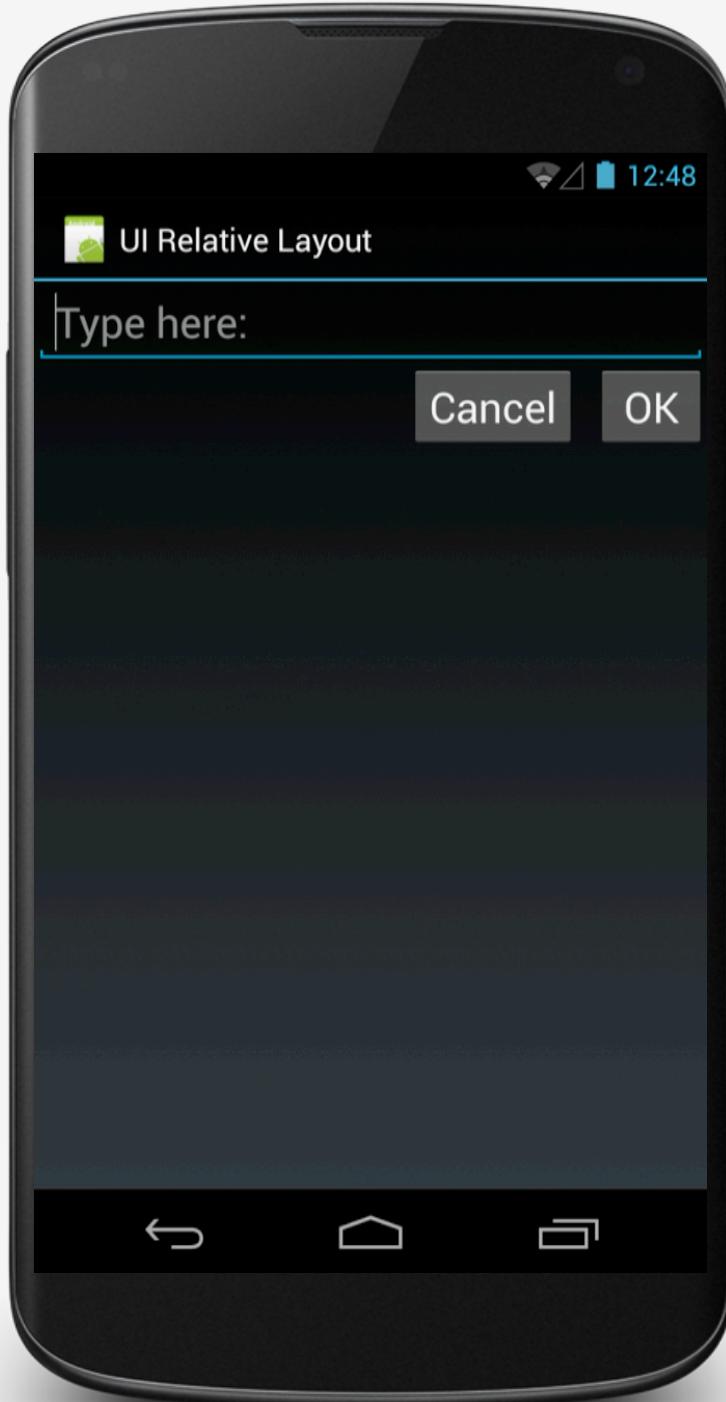
```
<LinearLayout  
    android:layout_width="match_parent"  
    android:layout_height="0dp"  
    android:layout_weight="1"  
    android:orientation="horizontal" >
```

```
<LinearLayout  
    android:layout_width="match_parent"  
    android:layout_height="0dp"  
    android:layout_weight="3"  
    android:orientation="vertical" >
```

# RELATIVE LAYOUT

CHILD VIEWS ARE POSITIONED RELATIVE TO  
EACH OTHER AND TO PARENT VIEW

# UIRELATIVE LAYOUT



# UI RELATIVE LAYOUT

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent" >  
  
    <!-- Note the use of android:hint to put explanatory text in the EditText -->  
    <EditText  
        android:id="@+id/entry"  
        android:layout_width="match_parent"
```

# TABLE LAYOUT

CHILD VIEWS ARRANGED INTO ROWS &  
COLUMNS

# UITABLELAYOUT



# UITABLELAYOUT

```
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:stretchColumns="1" >

    <!-- First row -->
    <TableRow>

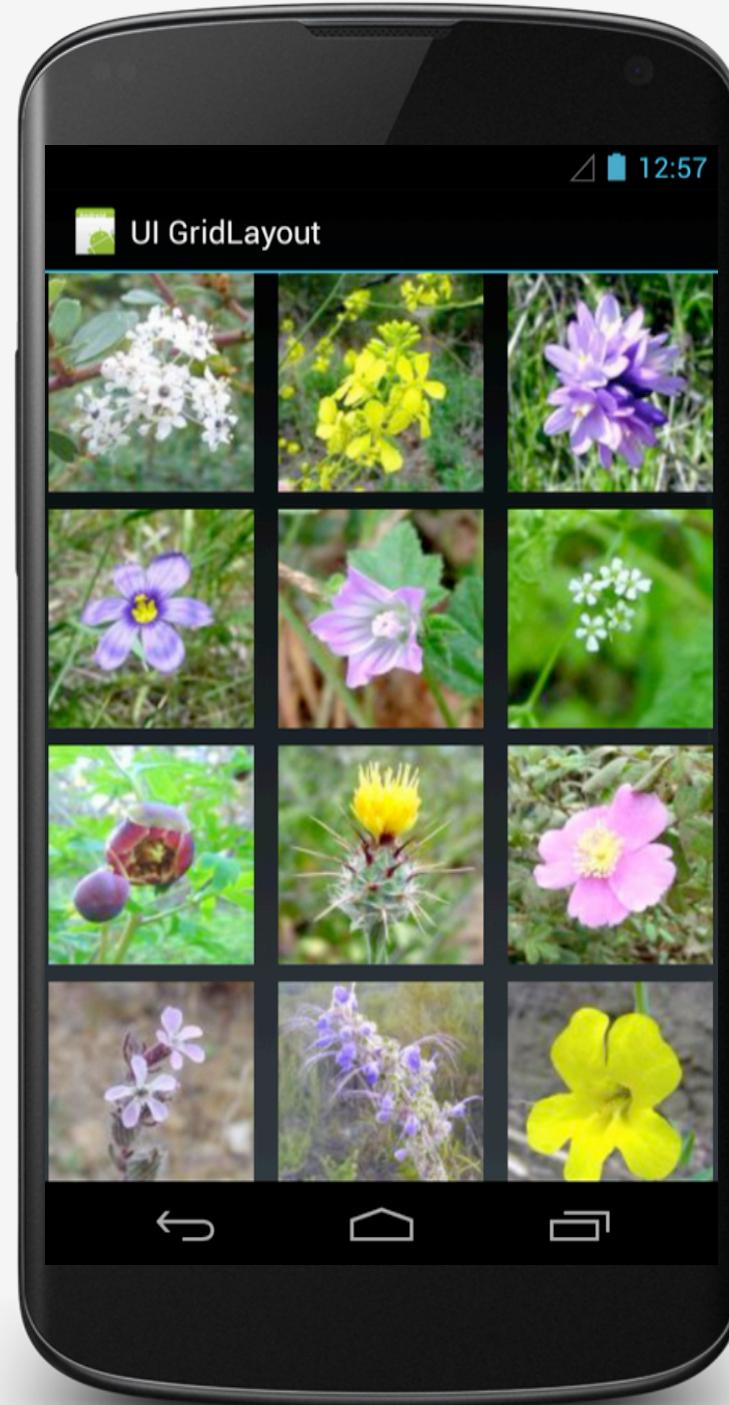
        <!-- start in column 1 -->
        <TextView
            android:layout_column="1"
            android:padding="3dip"
            android:text="@string/open_string"
            android:textSize="24sp" />

        <TextView
            android:gravity="right"
            android:padding="3dip"
            android:text="@string/ctrl_o_string"
            android:textSize="24sp" />
    </TableRow>
```

# GRIDVIEW

CHILD VIEWS ARRANGED IN A TWO-DIMENSIONAL, SCROLLABLE GRID

# UIGRIDVIEW



# UIGRIDVIEW

```
<GridView  
    android:id="@+id/gridview"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    android:columnWidth="90dp"  
    android:gravity="center"  
    android:horizontalSpacing="10dp"  
    android:numColumns="auto_fit"  
    android:stretchMode="columnWidth"  
    android:verticalSpacing="10dp" />
```

```
GridView gridview = (GridView) findViewById(R.id.gridview);  
  
// Create a new ImageAdapter and set it as the Adapter for this GridView  
gridview.setAdapter(new ImageAdapter(this, mThumbIdsFlowers));  
  
// Set an setOnItemClickListener on the GridView  
gridview.setOnItemClickListener(new OnItemClickListener() {  
    public void onItemClick(AdapterView<?> parent, View v,  
        int position, long id) {
```

# MENUS AND ACTIONBAR

ACTIVITIES SUPPORT MENUS

ACTIVITIES CAN

ADD ITEMS TO A MENU

HANDLE CLICKS ON THE MENU ITEMS

# MENU TYPES

## OPTIONS

MENU SHOWN WHEN USER PRESSES THE MENU  
BUTTON

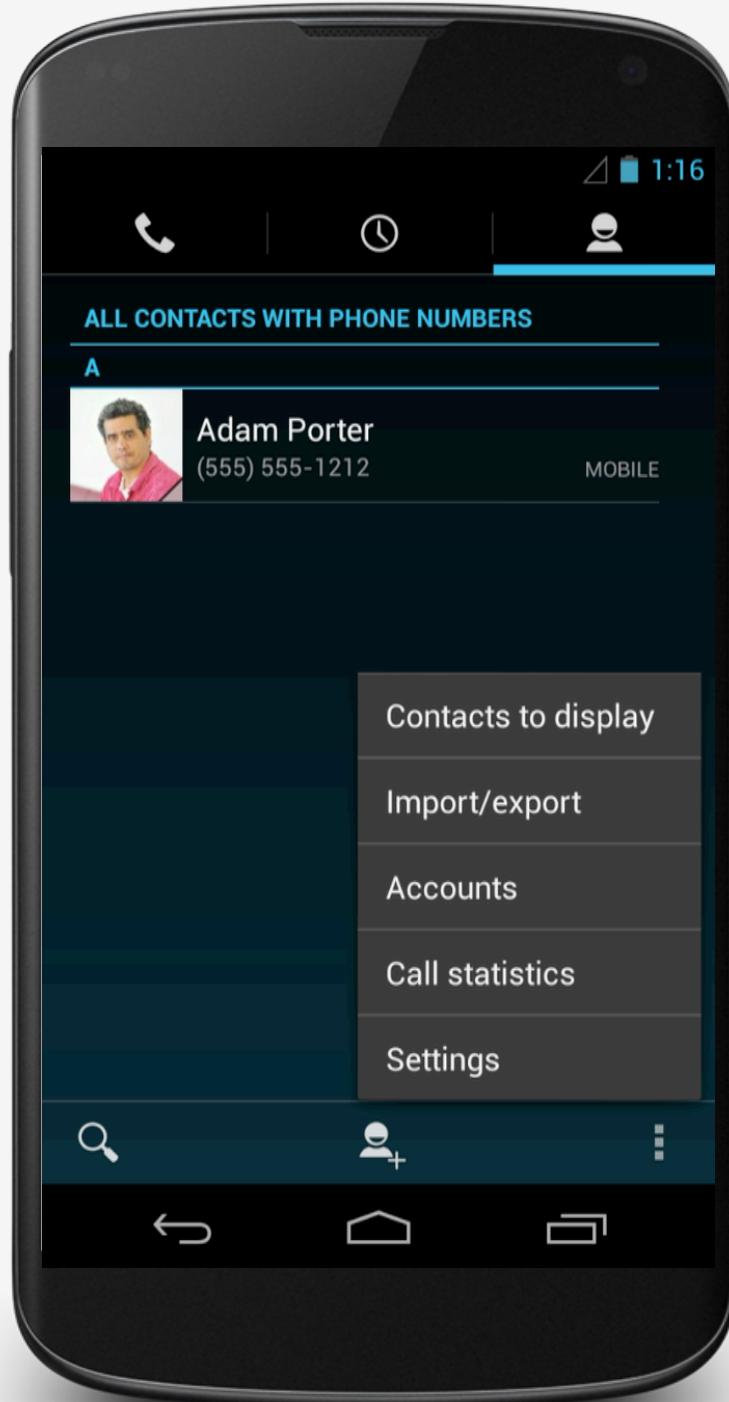
## CONTEXT

VIEW-SPECIFIC MENU SHOWN WHEN USER  
TOUCHES AND HOLDS THE VIEW

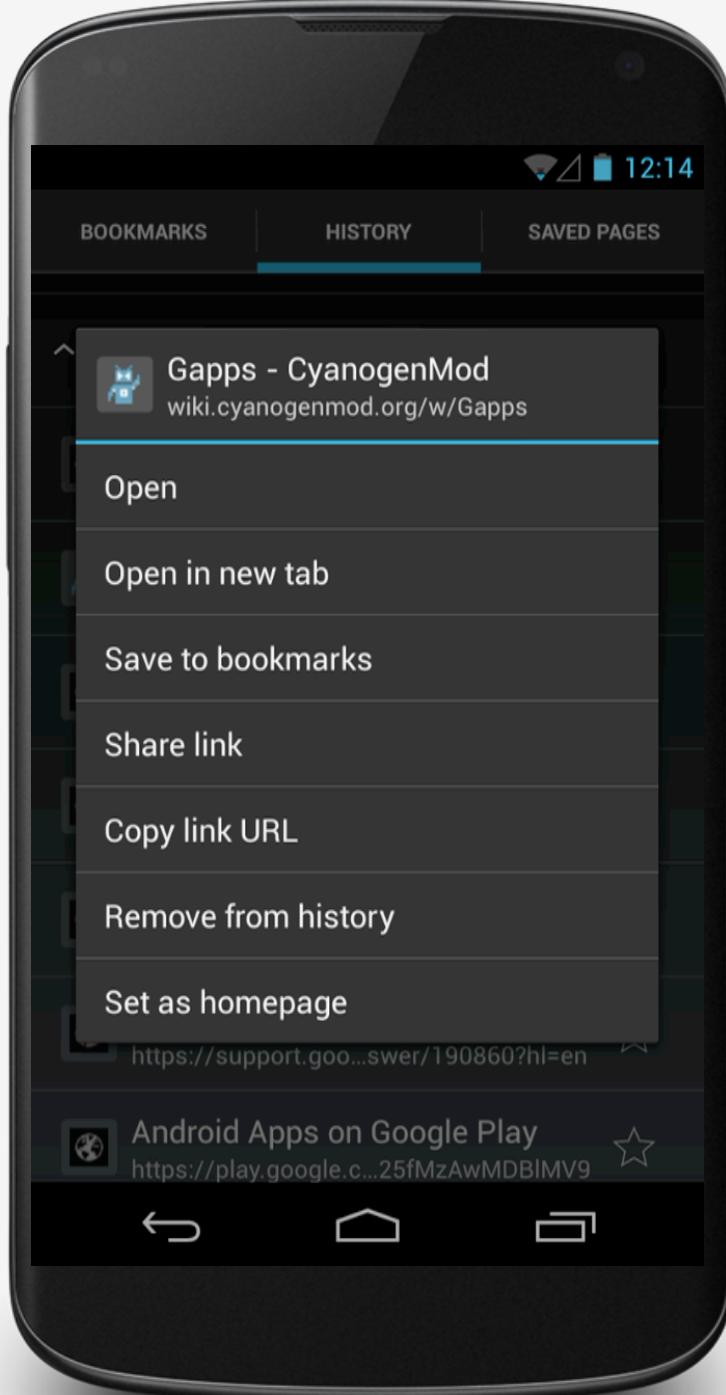
## SUBMENU

A MENU ACTIVATED WHEN USER TOUCHES A  
VISIBLE MENU ITEM

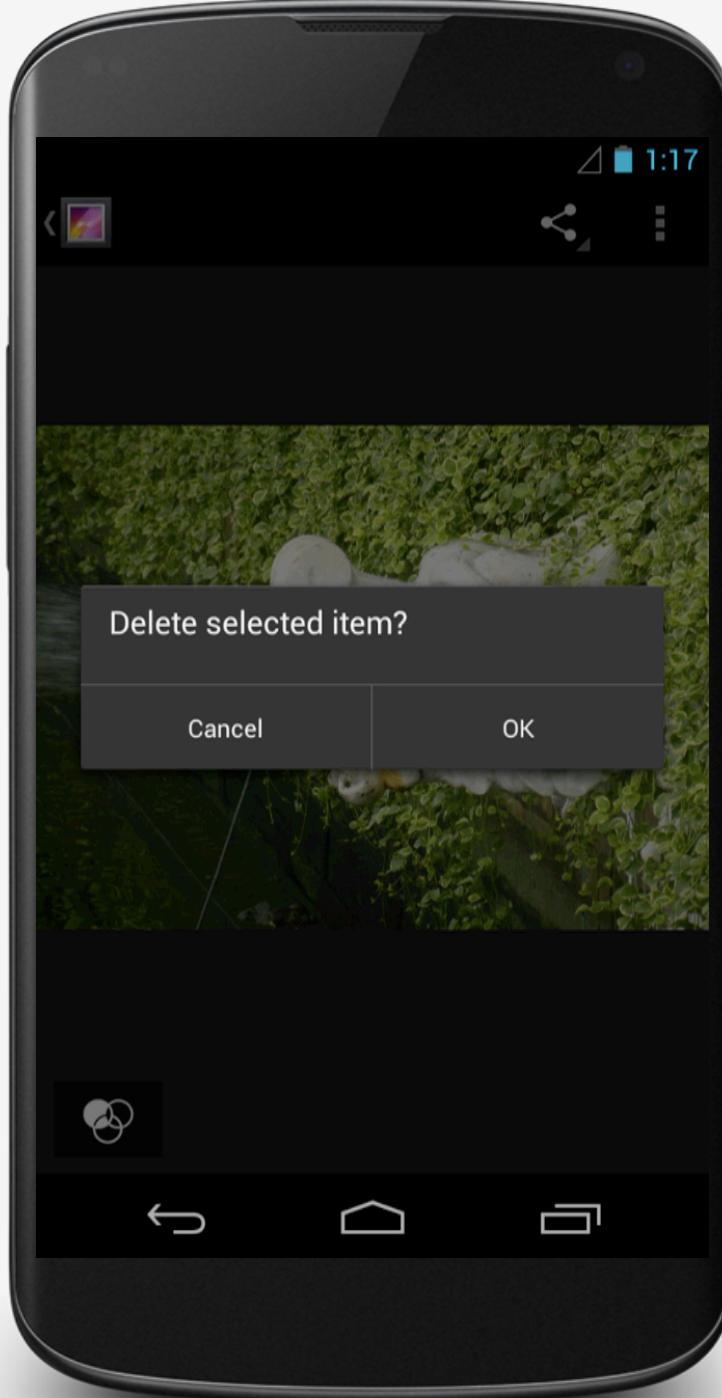
# OPTIONS MENUS



# CONTEXT MENUS



# SubMenus



# CREATING MENUS

DEFINE MENU RESOURCE IN XML FILE

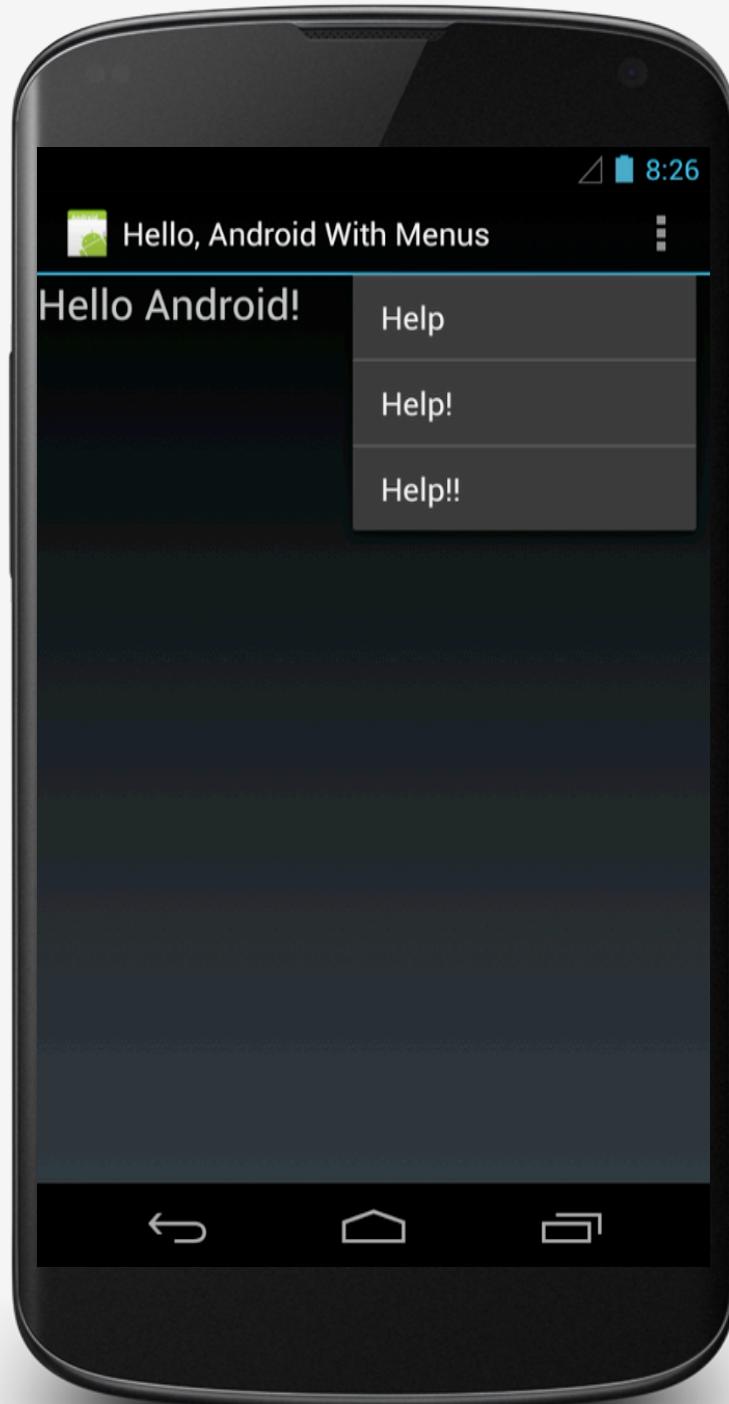
STORE IN RES/MENU/FILENAME.XML

# CREATING MENUS

INFLATE MENU RESOURCE USING MENU  
INFLATER IN ONCREATE...MENU() METHODS

HANDLING ITEM SELECTION IN APPROPRIATE  
ON...ITEMSELECTED() METHODS

# HELLOANDROID WITHMENUS



# HELLOANDROIDWITHMENUS

```
// Create Options Menu
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    MenuInflater inflater = getMenuInflater();
    inflater.inflate(R.menu.top_menu, menu);
    return true;
}

// Process clicks on Options Menu items
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case R.id.help:
            Toast.makeText(getApplicationContext(), "you've been helped",
                    Toast.LENGTH_SHORT).show();
            return true;
        case R.id.more_help:
            Toast.makeText(getApplicationContext(), "you've been helped more",
                    Toast.LENGTH_SHORT).show();
            return true;
        case R.id.even_more_help:
            return true;
        default:
            return false;
    }
}
```

# HELLOANDROIDWITHMENUS

```
// Create Context Menu
@Override
public void onCreateContextMenu(ContextMenu menu, View v,
        ContextMenuInfo menuInfo) {
    super.onCreateContextMenu(menu, v, menuInfo);
    MenuInflater inflater = getMenuInflater();
    inflater.inflate(R.menu.context_menu, menu);
}

// Process clicks on Context Menu Items
@Override
public boolean onContextItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case R.id.help_guide:
            Toast.makeText(getApplicationContext(), "ContextMenu Shown",
                    Toast.LENGTH_SHORT).show();
            return true;
        default:
            return false;
    }
}
```

# MENUS

MANY OTHER FEATURES SUPPORTED

GROUPING MENU ITEMS

BINDING SHORTCUT KEYS TO MENU ITEMS

BINDING INTENTS TO MENU ITEMS

# ACTIONBAR

SIMILAR TO APPLICATION BAR IN MANY  
DESKTOP APPLICATIONS

ENABLES QUICK ACCESS TO COMMON  
OPERATIONS

# FRAGMENTDYNAMICLAYOUT WITHACTIONBAR

SHOWS PLAY TITLES AND ONE QUOTE FROM  
THE SELECTED PLAY

PROVIDES ACTIONS FOR THE ACTIONBAR  
THREE MAIN OBJECTS

QUOTEVIEWERACTIVITY

TITLEFRAGMENT

QUOTEFRAGMENT

# ACTIONBAR.TAB

SCREEN IS DIVIDED INTO TAB & CONTENT AREAS

ALLOWS MULTIPLE FRAGMENTS TO SHARE SINGLE CONTENT AREA

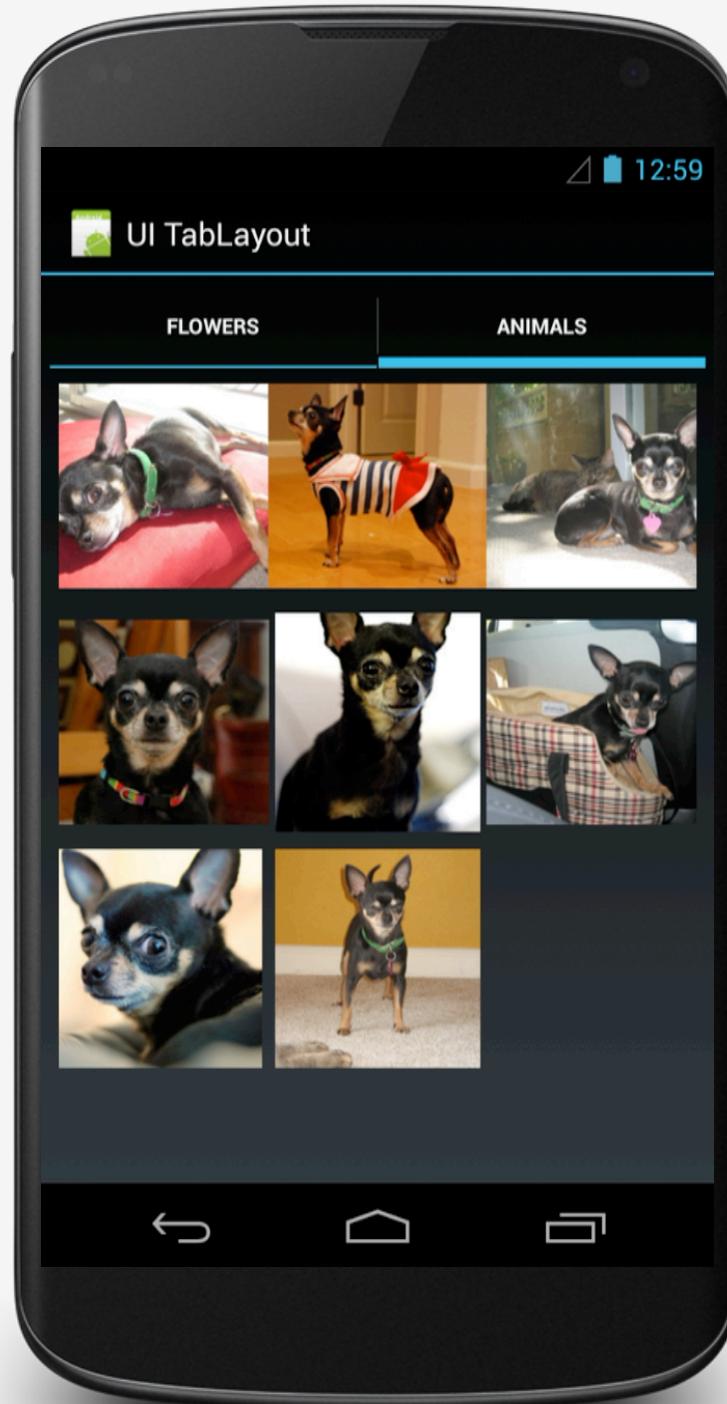
# ACTIONBAR.TAB

EACH TAB IS ASSOCIATED WITH ONE  
FRAGMENT

EXACTLY ONE TAB IS SELECTED AT  
ANY GIVEN TIME

FRAGMENT CORRESPONDING TO THE  
SELECTED TAB IS VISIBLE IN THE  
CONTENT AREA

# UITABLAYOUT



# UITABLAYOUT

```
@Override  
public void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
  
    setContentView(R.layout.main);  
  
    // Put ActionBar in Tab Mode  
    final ActionBar tabBar = getActionBar();  
    tabBar.setNavigationMode(ActionBar.NAVIGATION_MODE_TABS);  
  
    // Create a GridFragment for the Flower thumbnails  
    GridFragment flowerFrag = new GridFragment();  
  
    // Store the list of thumbnails as an argument to the GridFragment  
    Bundle args = new Bundle();  
    args.putIntegerArrayList(THUMBNAIL_IDS, mThumbIdsFlowers);  
    flowerFrag.setArguments(args);  
  
    // Configure a tab for the Flower thumbnail GridFragment  
    tabBar.addTab(tabBar.newTab().setText(FLOWERS_TABSTRING)  
        .setTabListener(new TabListener(flowerFrag)));  
  
    // Create a GridFragment for the Animal thumbnails  
    GridFragment animalFrag = new GridFragment();  
  
    // Store the list of thumbnails as an argument to the GridFragment  
    args = new Bundle();  
    args.putIntegerArrayList(THUMBNAIL_IDS, mThumbIdsAnimals);  
    animalFrag.setArguments(args);  
  
    // Configure a tab for the Animal thumbnail GridFragment  
    tabBar.addTab(tabBar.newTab().setText(ANIMALS_TABSTRING)  
        .setTabListener(new TabListener(animalFrag)));  
  
}
```

# UITABLAYOUT

```
// This class handles user interaction with the tabs
public static class TabListener implements ActionBar.TabListener {
    private static final String TAG = "TabListener";
    private final Fragment mFragment;

    public TabListener(Fragment fragment) {
        mFragment = fragment;
    }

    @Override
    public void onTabReselected(Tab tab, FragmentTransaction ft) {}

    // When a tab is selected, change the currently visible Fragment
    @Override
    public void onTabSelected(Tab tab, FragmentTransaction ft) {
        Log.i(TAG, "onTabSelected called");

        if (null != mFragment) {
            ft.replace(R.id.fragment_container, mFragment);
        }
    }

    // When a tab is unselected, remove the currently visible Fragment
    @Override
    public void onTabUnselected(Tab tab, FragmentTransaction ft) {
        Log.i(TAG, "onTabUnselected called");

        if (null != mFragment)
            ft.remove(mFragment);
    }
}
```

# DIALOGS

INDEPENDENT SUBWINDOWS USED BY  
ACTIVITIES TO COMMUNICATE WITH USER

# DIALOG SUBCLASSES

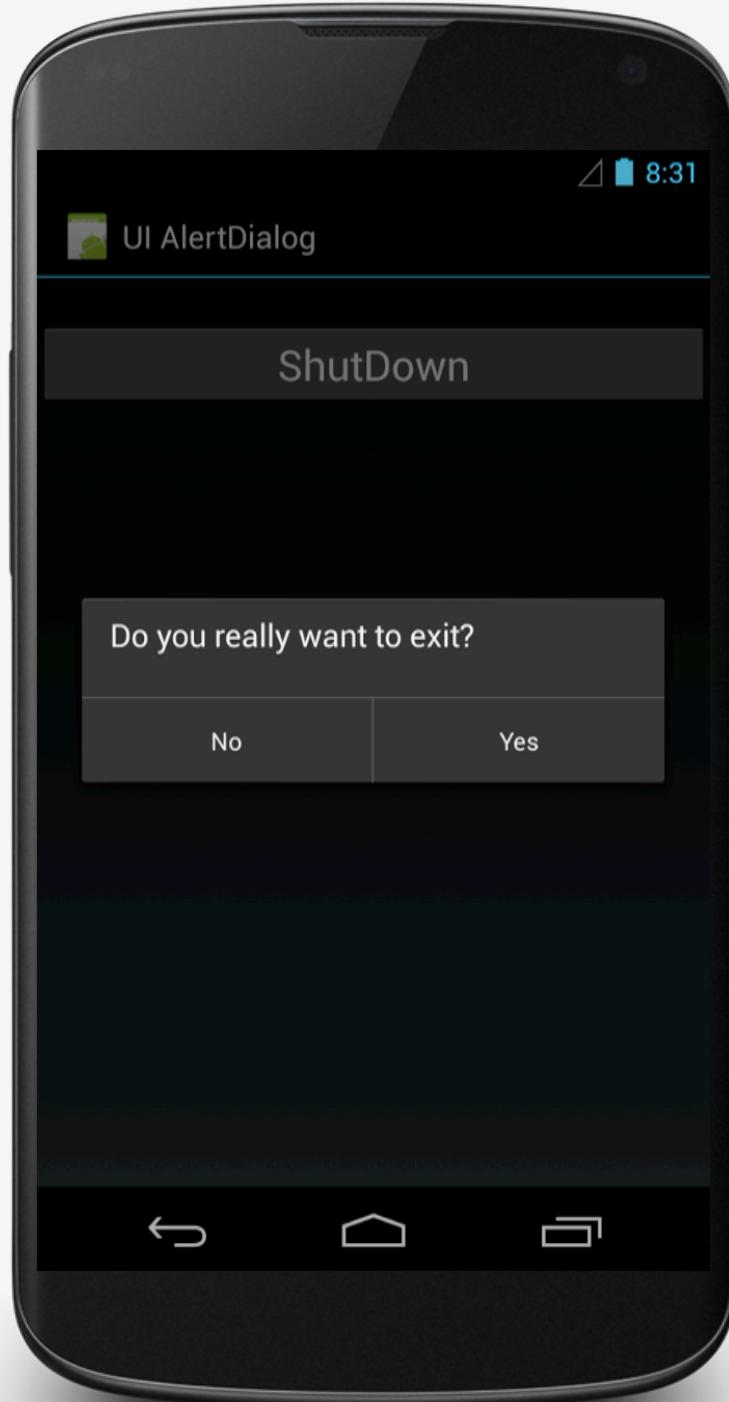
ALERTDIALOG

PROGRESSDIALOG

DATEPICKERDIALOG

TIMEPICKERDIALOG

# UIALERTDIALOG PROGRESSDIALOG



# UI ALERT DIALOG

```
// Class that creates the AlertDialog
public static class AlertDialogFragment extends DialogFragment {

    public static AlertDialogFragment newInstance() {
        return new AlertDialogFragment();
    }

    // Build AlertDialog using AlertDialog.Builder
    @Override
    public Dialog onCreateDialog(Bundle savedInstanceState) {
        return new AlertDialog.Builder(getActivity())
            .setMessage("Do you really want to exit?")

        // User cannot dismiss dialog by hitting back button
        .setCancelable(false)

        // Set up No Button
        .setNegativeButton("No",
            new DialogInterface.OnClickListener() {
                public void onClick(DialogInterface dialog,
                    int id) {
                    ((AlertDialogActivity) getActivity())
                        .continueShutdown(false);
                }
            })
        // Set up Yes Button
        .setPositiveButton("Yes",
            new DialogInterface.OnClickListener() {
                public void onClick(
                    final DialogInterface dialog, int id) {
                    ((AlertDialogActivity) getActivity())
                        .continueShutdown(true);
                }
            }).create();
    }
}
```

NEXT TIME

USER NOTIFICATIONS