CMSC436: Programming Handheld Systems

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

Occupies a rectangular space on screen

Responsible for drawing itself and for handling events

Some Predefined Views

Button

ToggleButton

Checkbox

RatingBar

AutoCompleteTextView

Button

View that can be clicked on to perform an action

UIButton

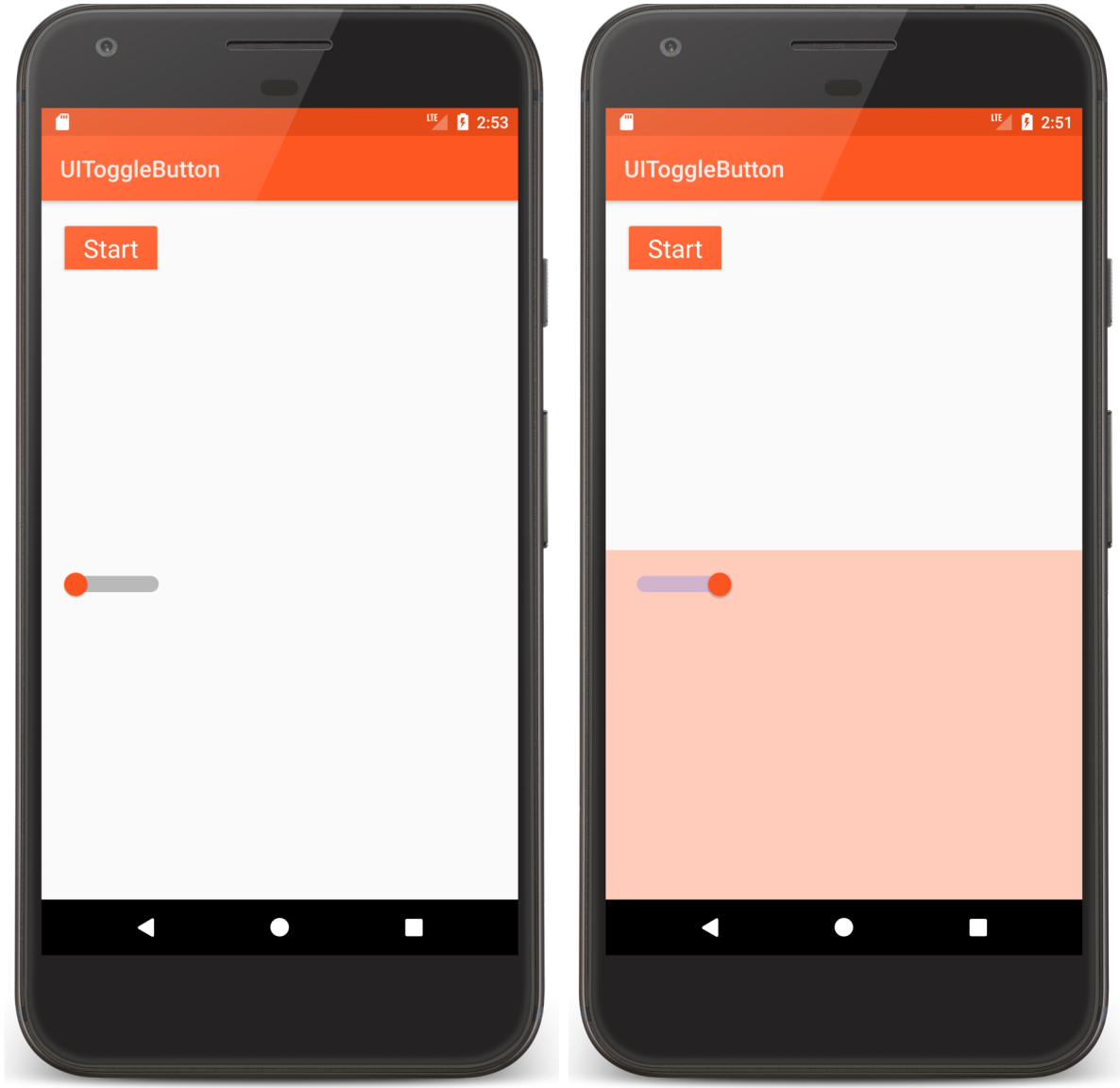
```
public class ButtonActivity extends Activity {
    private static int mCount;
    private Button mButton;

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

        // Get a reference to the Press Me Button
        mButton = findViewById(R.id.button);
```

```
    // Reset Button Text if restarting
    if (null != savedInstanceState) {
        mButton.setText(getString(R.string.got_pressed_string, mCount));
    }
    // Set an OnClickListener on this Button
    // Called each time the user clicks the Button
    mButton.setOnClickListener(new OnClickListener() {
        @Override
        public void onClick(View v) {
            // Maintain a count of user presses
            // Display count as text on the Button
            mButton.setText(getString(R.string.got_pressed_string, ++mCount));

        }
...
```

ToggleButton

- A 2-state Button
  - Checked/not checked state
  - Light indicator showing state

UIToggleButton



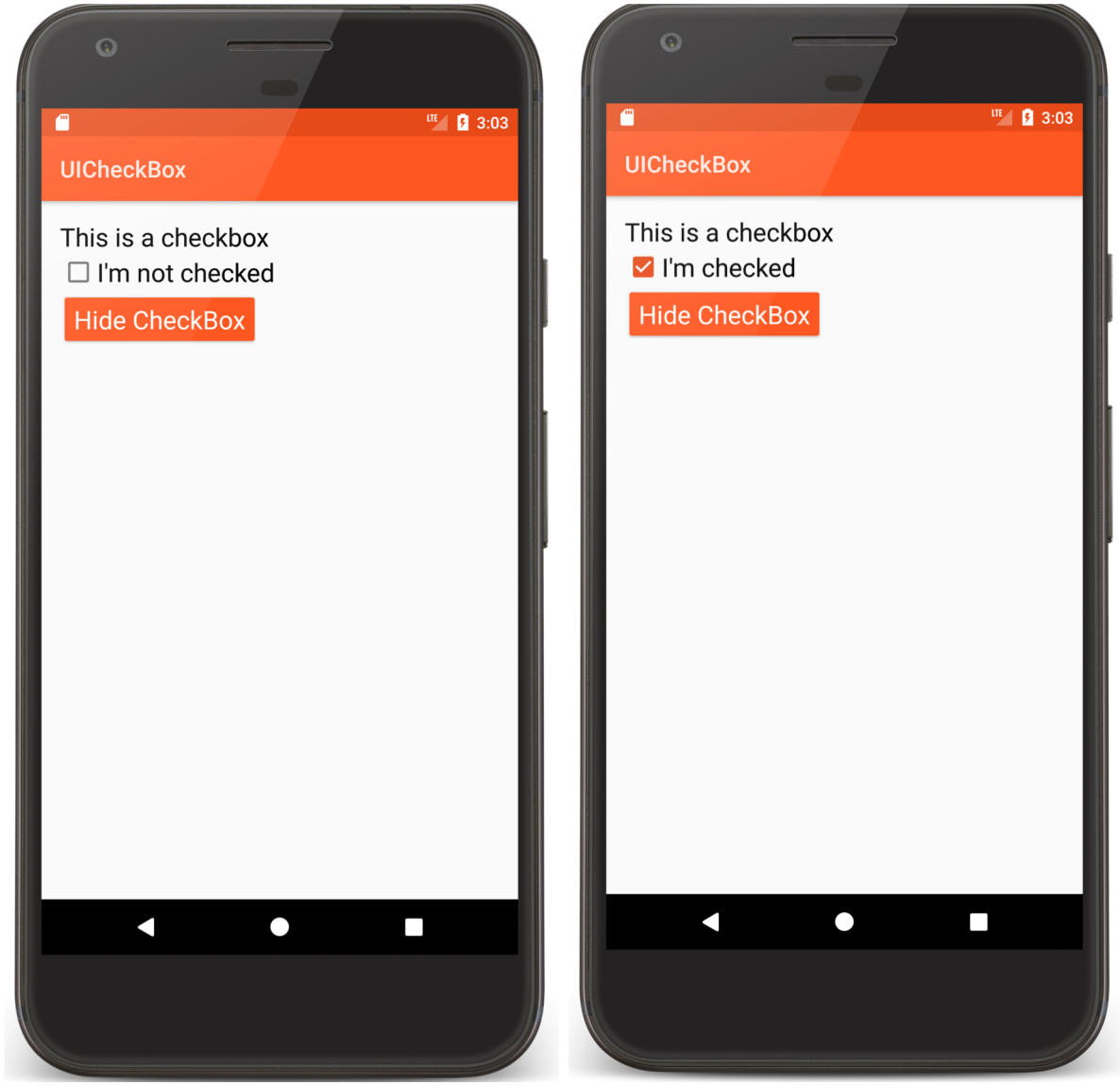
```
public class ToggleButtonActivity extends Activity {
  @Override
  public void onCreate(Bundle savedInstanceState) {
     …
    // Get a reference to the ToggleButton
     final ToggleButton toggleButton = findViewById(R.id.togglebutton);

     // Set an setOnCheckedChangeListener on the ToggleButton
     setListener(toggleButton, top);

     // Get a reference to the Switch
     final Switch switcher = findViewById(R.id.switcher);

     // Set an OnCheckedChangeListener on the Switch
     setListener(switcher, bottom);
  }
```

```
private void setListener(CompoundButton button, final View background) {
   button.setOnCheckedChangeListener(new
                              CompoundButton.OnCheckedChangeListener() {
      public void onCheckedChanged(CompoundButton compoundButton,
                                                   boolean isChecked) {
         // Toggle the Background color between a light and dark color
         if (isChecked) {
          background.setBackgroundColor(
                              getResources().getColor(R.color.primary_light,null));
         } else {
            background.setBackgroundColor(Color.TRANSPARENT);
         }
   }
});  …
```

Checkbox

- Another kind of 2-state button
  - Checked/not checked

UICheckbox


3:03
UICheckBox
This is a checkbox
I'm not checked
Hide CheckBox
3:03
UICheckBox
This is a checkbox
I'm checked
Hide CheckBox

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
   ...
   <CheckBox
      android:id="@+id/checkbox"
      ...
      android:onClick="checkBoxClickCallback"
      android:text="@string/im_not_checked_string"
    .../>
   <Button
      android:id="@+id/button"
      ...
      android:onClick="buttonClickCallback"
      android:text="@string/hide_checkbox_string"
    .../>
</RelativeLayout>
```

```
public class CheckBoxActivity extends Activity {
...
// Set with android:onClick
 public void checkBoxClickCallback(@SuppressWarnings("unused") View view) {
      setCheckedStateAndText();
 }
 private void setCheckedStateAndText() {
    // Check whether CheckBox is currently checked. Set CheckBox text accordingly
      if (mCheckBox.isChecked()) {
         mCheckBox.setText(R.string.im_checked_string);
      } else {
         mCheckBox.setText(R.string.im_not_checked_string);
      }
   }
```

RatingBar

A view comprising a row of stars

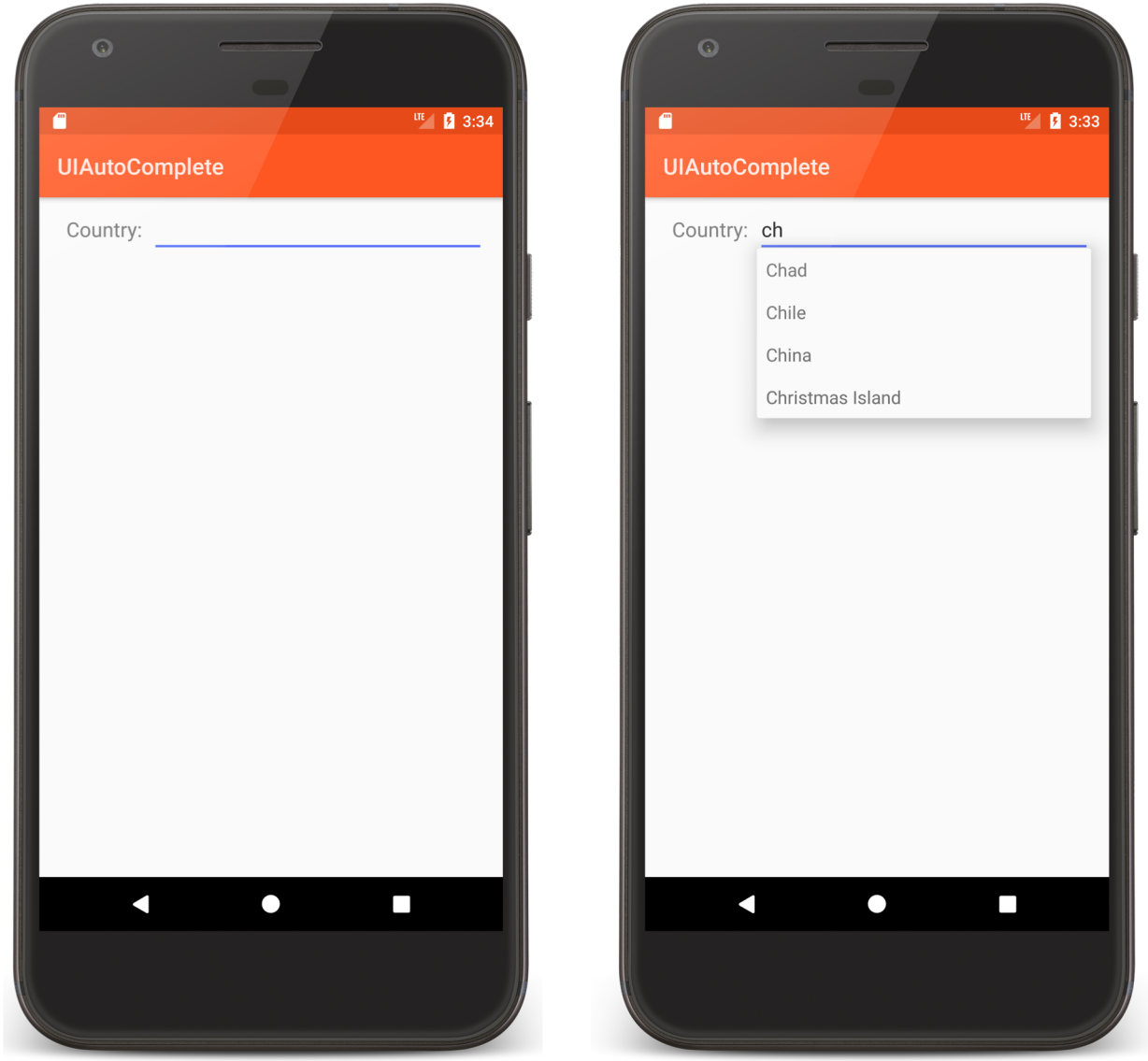
The user can click or drag the stars to highlight some number of them

UIRatingBar

AutoCompleteTextView

An editable text field that provides completion suggestions as the user types in text

UIAutoComplete TextView



Common View Operations

Set visibility: Show or hide View

Set checked state: Checked or not checked

Set listeners: Code that will be executed when specific events occur

Set properties: Opacity, background, rotation

Manage input focus: Allow View to take focus, request focus, etc.

View Event Sources

- User interaction
  - Touch
  - Keyboard/trackball/D-pad
- System control
  - Lifecycle changes

Handling View Events

Will often handle events using listeners

Many Listener interfaces defined by 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 within a UI are organized as a tree

Displaying/refreshing the UI has multiple steps

- Measure – get dimensions of each View
- Layout – Position each View
- Draw – Draw each view

Handling View Events

Typically create View subclasses

Then override View methods

Handling View Events

onMeasure()

Determine the size of this View and its children

onLayout()

Assign a size and position to all View’s children

onDraw()

Render View content

Handling View Events

onFocusChanged()

Called when View’s focus state has changed

onKeyUp(), onKeyDown()

Called when 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 and Layouts

Some Predefined ViewGroups

RadioGroup

TimePickerFragment

DatePickerFragment

WebView

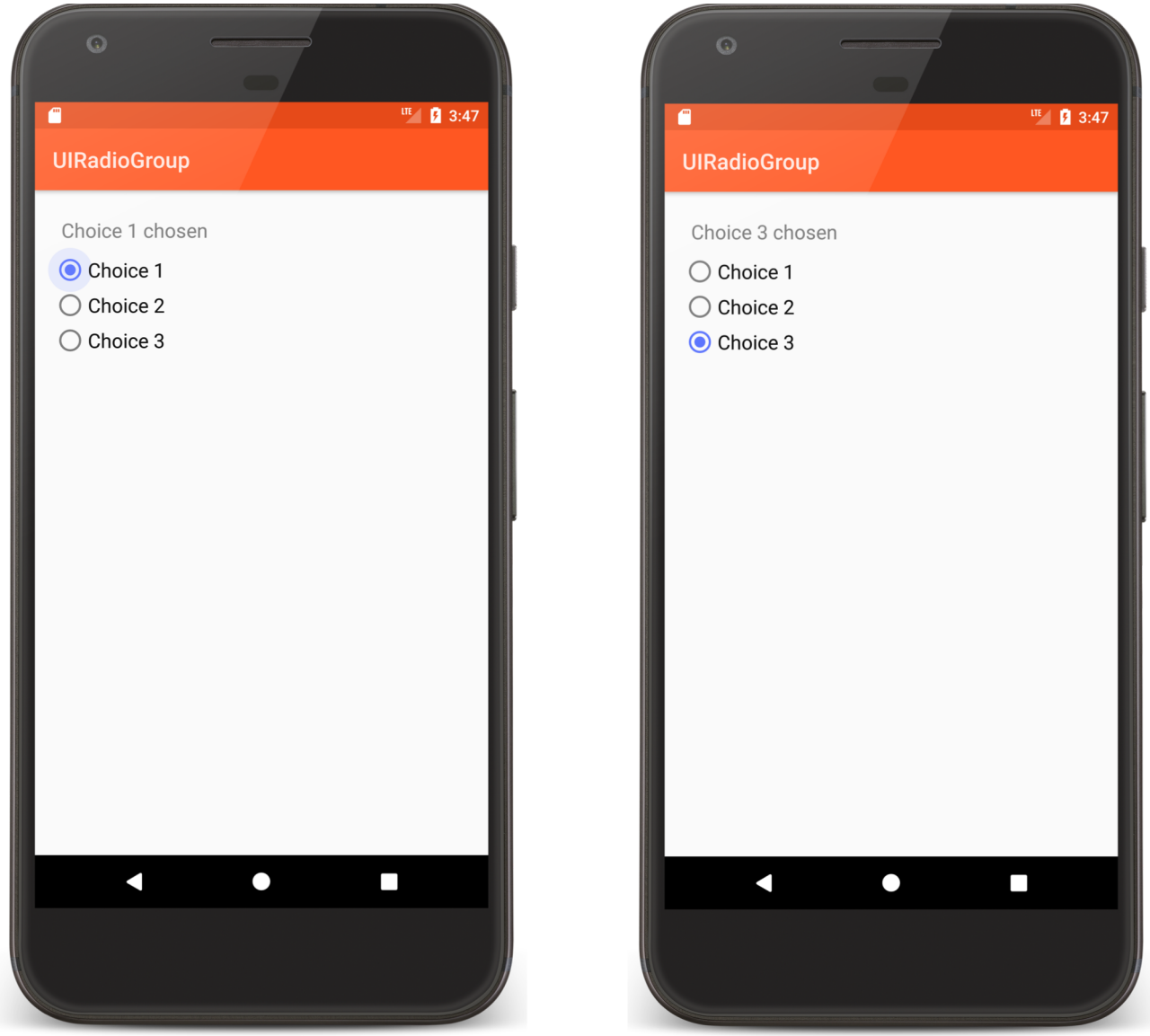
MapView

RadioGroup

A ViewGroup containing a set of Radio Buttons

Only one RadioButton can be selected at any one time

UIRadioGroup

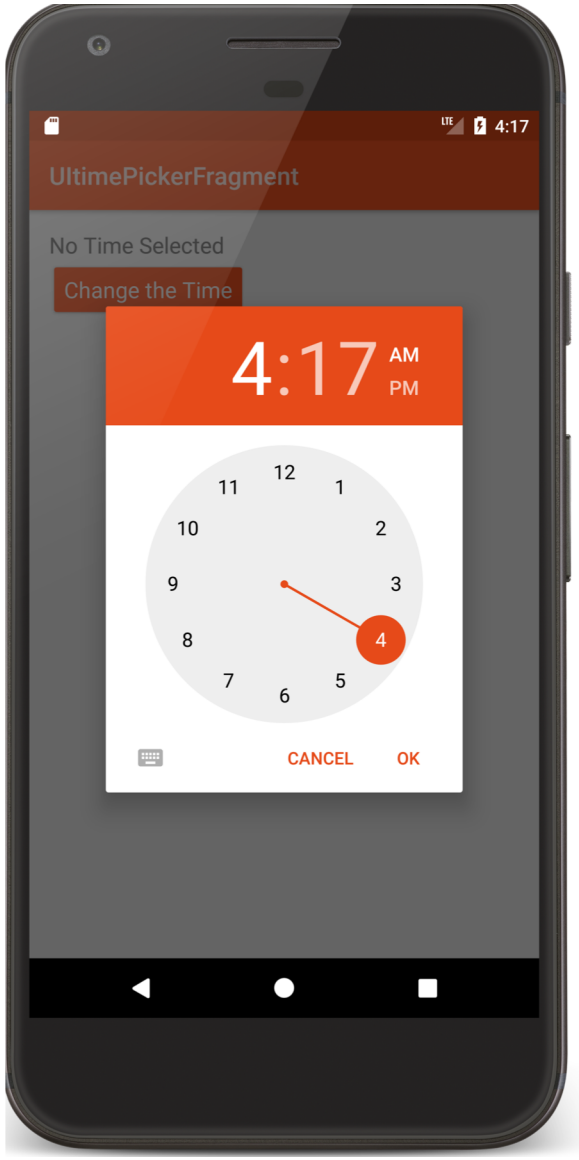


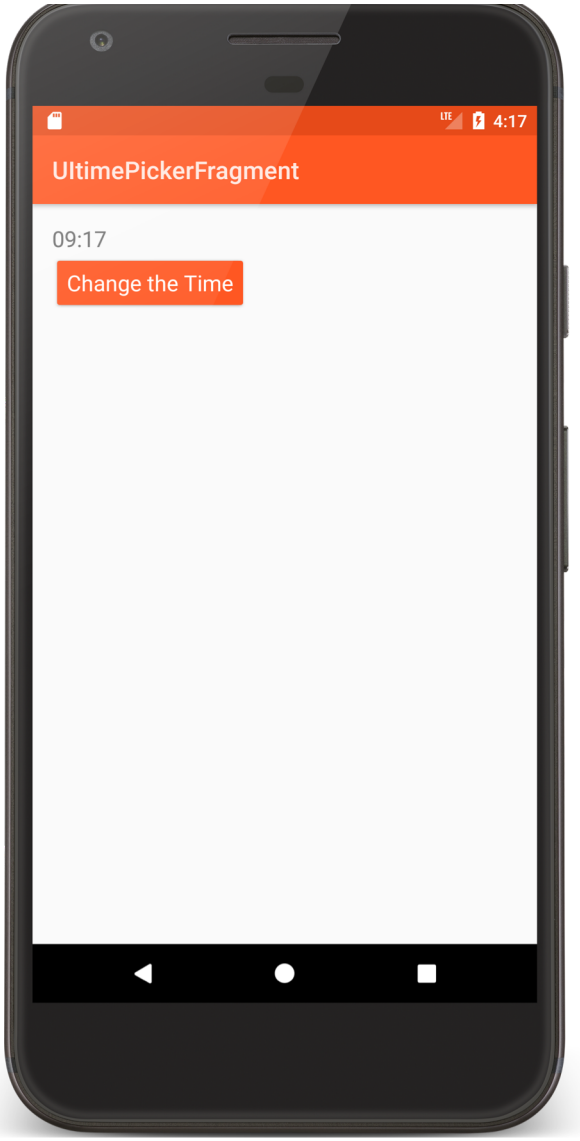
```
<RadioGroup
   android:id="@+id/radio_group"
   ... >
   <RadioButton
      android:id="@+id/choice1"
      android:onClick="radioButtonClickCallback”
    .../>
   <RadioButton
      android:id="@+id/choice2"
      android:onClick="radioButtonClickCallback"
    .../>
   <RadioButton
      android:id="@+id/choice3"
      android:onClick="radioButtonClickCallback"
   ... />
</RadioGroup>
```

```
// Define a generic listener for all three RadioButtons in the RadioGroup
public void radioButtonClickCallback(View v) {
    RadioButton rb = (RadioButton) v;

    // RadioButtons report each click, even if the toggle state doesn't change
    if (rb.isChecked()) {
        mTextView.setText(getString(R.string.chosen_string, rb.getText()));
    }
}
```

TimePickerFragment

A ViewGroup that allows the user to select a time





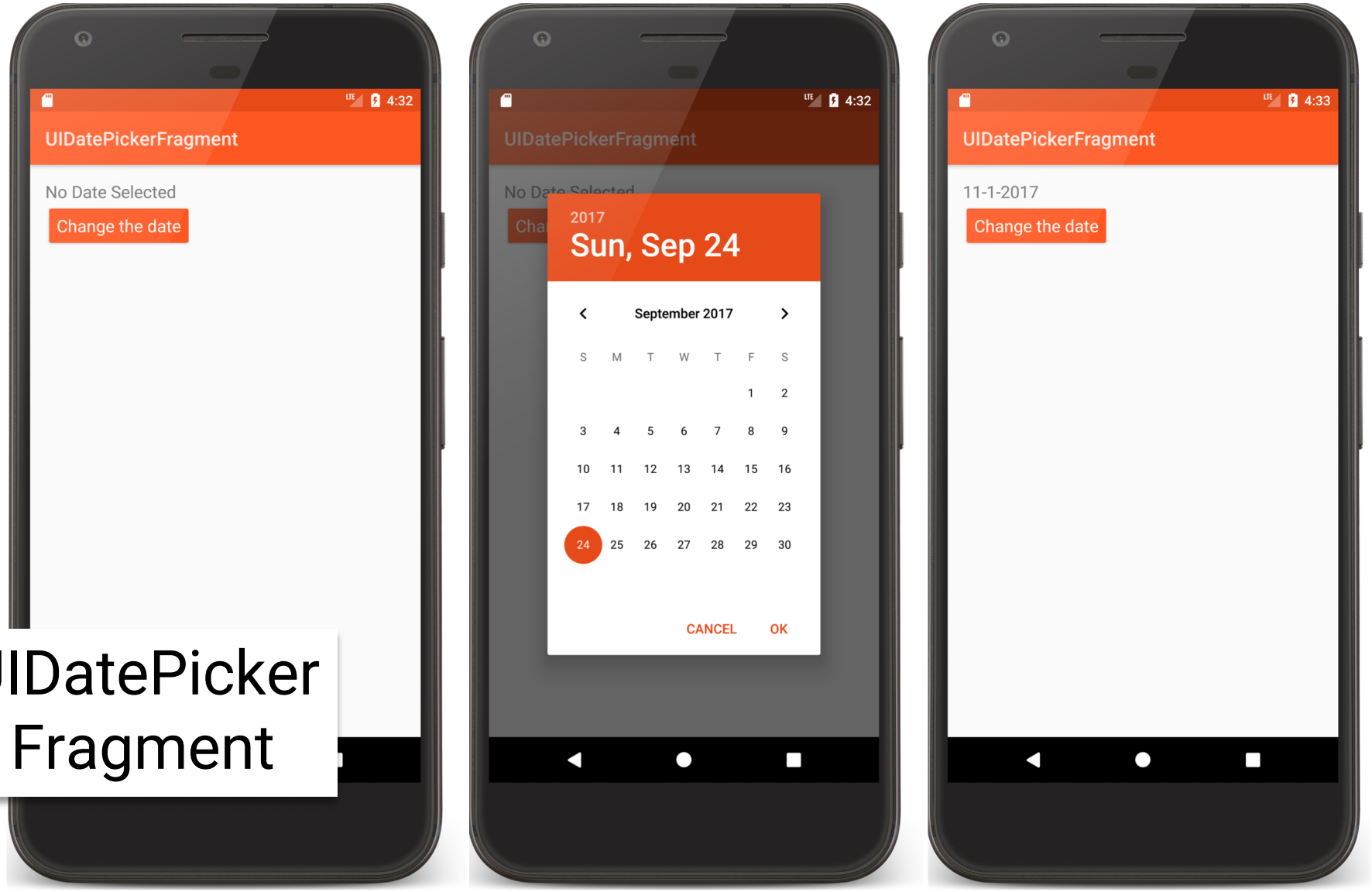
UITimePicker Fragment

```
public class TimePickerFragmentActivity extends Activity implements
      OnTimeSetListener {
...
 // OnClickListener for the "Change the Time" Button
 public void buttonPressedCallback(@SuppressWarnings("unused") View view) {
      // Create and display DatePickerFragment
      new TimePickerFragment().show(getFragmentManager(), "TimePicker");
   }

   // Callback called by TimePickerFragment when user sets the time
 public void onTimeSet(TimePicker view, int hourOfDay, int minute) {
      mTimeDisplay.setText(/* construct time string from params*/);
   }
```

```
public static class TimePickerFragment extends DialogFragment implements
                                                        OnTimeSetListener {
   public Dialog onCreateDialog(Bundle savedInstanceState) {
   // Create a new instance of TimePickerDialog and return it
    …
    return new TimePickerDialog(getActivity(), this, hourOfDay, minute,  false);
  }

 // Callback called by TimePickerDialog when user sets the time
  public void onTimeSet(TimePicker view, int hourOfDay, int minute) {
    ((OnTimeSetListener) getActivity()).onTimeSet(view, hourOfDay, minute);
      }
   }
}
```

DatePickerFragment

A ViewGroup that allows the user to select a date



UIDatePicker Fragment

WebView

A ViewGroup that displays a web page

UIWebView

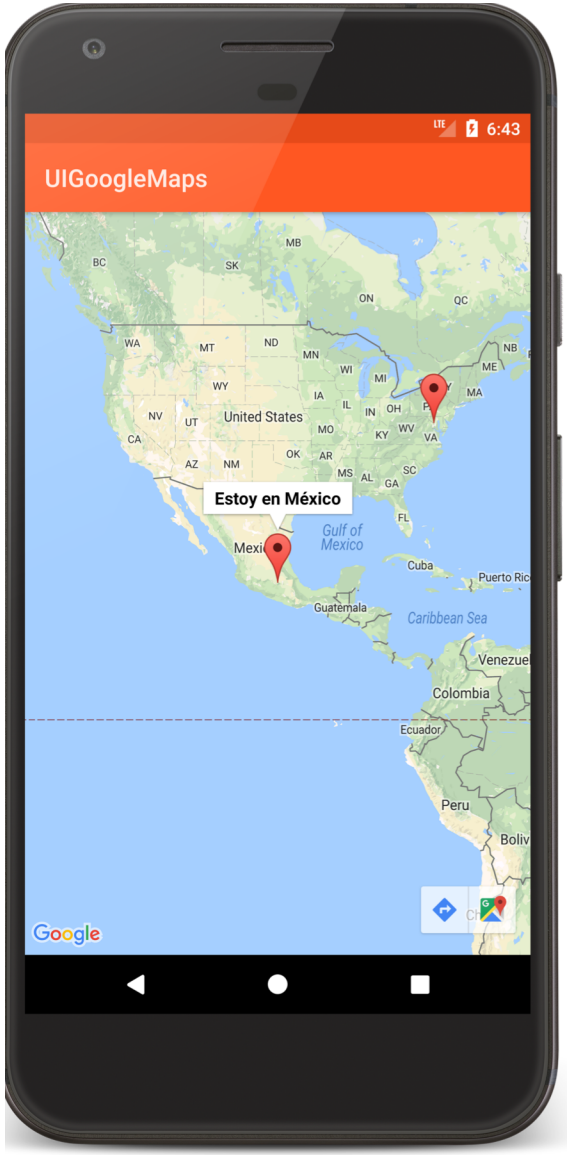
```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:background="@color/primary_light">

   <WebView
      android:id="@+id/webview"
      android:layout_width="match_parent"
      android:layout_height="match_parent" />

</RelativeLayout>
```

```
public class WebViewActivity extends Activity {
   ...
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.main);
      mWebView = 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());
      // Add Zoom controls to WebView
      mWebView.getSettings().setBuiltInZoomControls(true);
      mWebView.loadUrl("https://www.cs.umd.edu/~aporter/Tmp/bee.html");
   }
}
```

MapView

A ViewGroup that displays a Map

UIGoogleMaps



Adapters & AdapterViews

AdapterViews are Views whose children and data are managed by an Adapter

Interaction pattern

- Adapter manages the data and provides data Views to AdapterView
- AdapterView displays the data Views

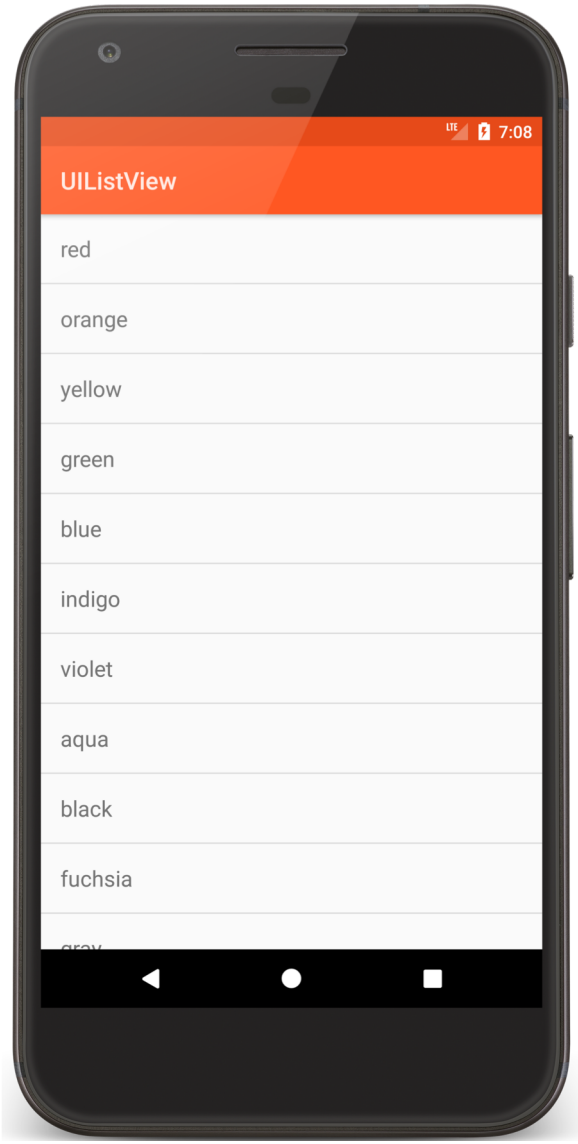
ListView

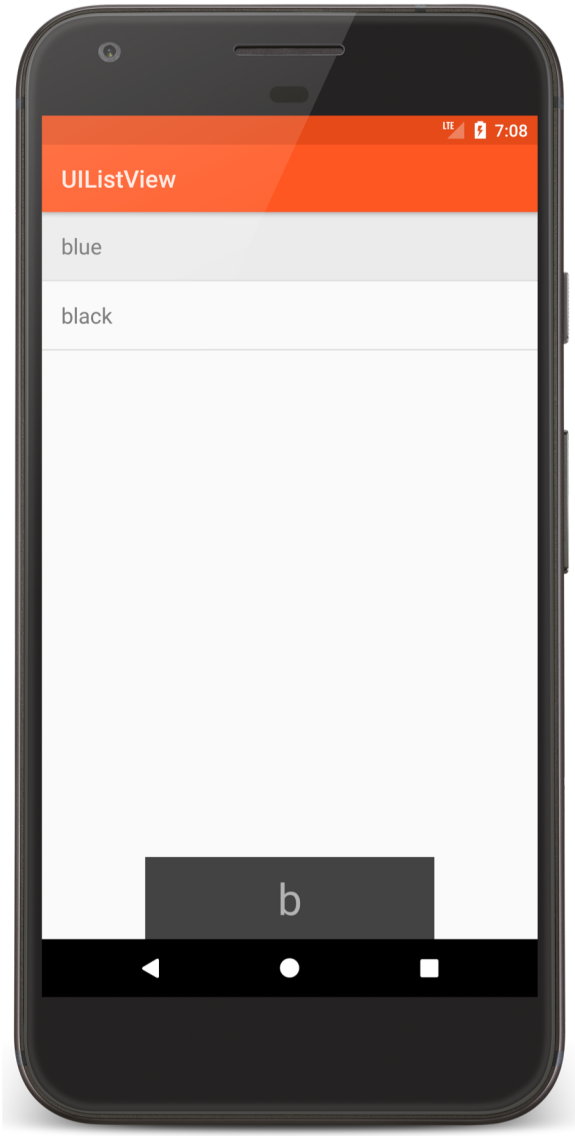
An AdapterView that displays a scrollable list of selectable items

Data items managed by a ListAdapter

ListView can filter the list of items based on text input

UIListView


7:08
UIListView
red
orange
yellow
green
blue
indigo
violet
aqua
black
fuchsia

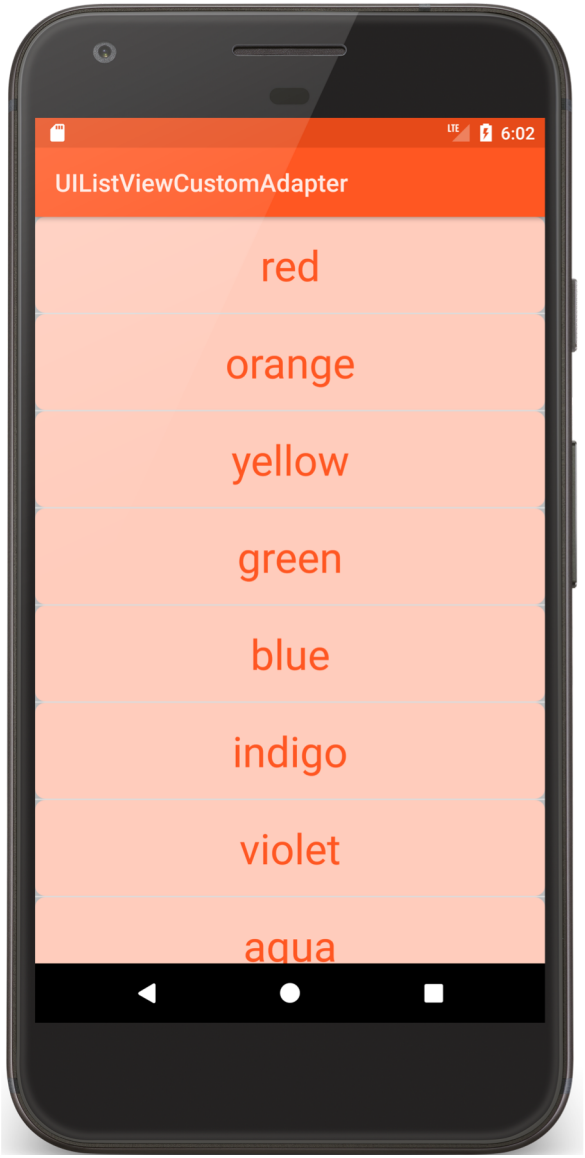

7:08
UIListView
blue
black
b

```
public class ListViewActivity extends ListActivity {
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    // Create a new Adapter containing a list of colors
    // Set the adapter on this ListActivity's built-in ListView
    setListAdapter(new ArrayAdapter<>(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);
    …
```

```
<TextView xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:padding="@dimen/activity_margin"
   android:textAppearance=
                        "@android:style/TextAppearance.Material.Medium”/>
```

```
    ...
    // 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 indicting the selected item
        Toast.makeText(getApplicationContext(),
            ((TextView) view).getText(), Toast.LENGTH_SHORT).show();
      }
    });
  }
}
```

UIListViewWith CustomAdapter



```
<?xml version="1.0" encoding="utf-8"?>
<TextView xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/text"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="@drawable/list_background"
    android:gravity="center"
    android:padding="16dp"
    android:textAppearance="@android:style/TextAppearance.Material.Display1"
    android:textColor="@color/primary">

</TextView>
```

```
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="rectangle">
    <solid android:color="@color/primary_light" />
    <corners android:radius="8dp" />
</shape>
```

```
public class ListViewActivity extends ListActivity {
   @Override
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);

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

      …
```

```
class ListViewAdapter extends ArrayAdapter<String> {
   private final LayoutInflater mLayoutInflater;
  …
   public View getView(int position, View convertView, ViewGroup parent) {
      View dataView = convertView;
      // Check for recycled View
      if (null == dataView) {
         // Not recycled. Create the View
         dataView = mLayoutInflater.inflate(R.layout.list_item, parent, false);
         // Cache View information in ViewHolder Object
         ViewHolder viewHolder = new ViewHolder();
          viewHolder.text = dataView.findViewById(R.id.text);
          dataView.setTag(viewHolder);
      }
…
```

```
        …
        // Set the View's data. Retrieve the viewHolder Object
        ViewHolder storedViewHolder = (ViewHolder) dataView.getTag();
        //Set the data in the data View
        storedViewHolder.text.setText(getItem(position));
        return dataView;
    }

    // The ViewHolder class. See:http://developer.android.com/training/
    //                       improving/layouts/Smooth-scrolling.html#ViewHolder
    static class ViewHolder {
        public TextView text;
    }
}
```

UIRecyclerView

```
public class RecyclerViewActivity extends Activity {
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      …
      //Set the layout manager
      mRecyclerView.setLayoutManager(new LinearLayoutManager(this));
      // Set up the adapter
      ArrayList<String> names = new ArrayList<>();
      Collections.addAll(names, getResources().getStringArray(R.array.colors));
      MyRecyclerViewAdapter mAdapter =
                              new MyRecyclerViewAdapter(names,R.layout.list_item);
      mRecyclerView.setAdapter(mAdapter);
   }
}
```

```
class MyRecyclerViewAdapter extends
   RecyclerView.Adapter<MyRecyclerViewAdapter.ViewHolder> {
    …
   // Create ViewHolder which holds a View to be displayed
   public ViewHolder onCreateViewHolder(ViewGroup viewGroup, int i) {
      View v = LayoutInflater.from(viewGroup.getContext())
                                          .inflate(mRowLayout, viewGroup, false);
      return new MyRecyclerViewAdapter.ViewHolder(v);
   }
   // Binding: Preparing a child view to display data corr. to a position within the adapter.
   public void onBindViewHolder(ViewHolder viewHolder, int i) {
      viewHolder.mName.setText(mNames.get(i));
   }
   public int getItemCount() {
      return (null == mNames) ? 0 : mNames.size();
   }
```

```
    public static class ViewHolder extends RecyclerView.ViewHolder
                                                    implements View.OnClickListener {
        …
        public ViewHolder(View itemView) {
            super(itemView);
            mName = itemView.findViewById(R.id.text);
            itemView.setOnClickListener(this);
        }

        public void onClick(View view) {
            // Display a Toast message indicting the selected item
            Toast.makeText(view.getContext(),
                    mName.getText(), Toast.LENGTH_SHORT).show();
        }
    }
}
```

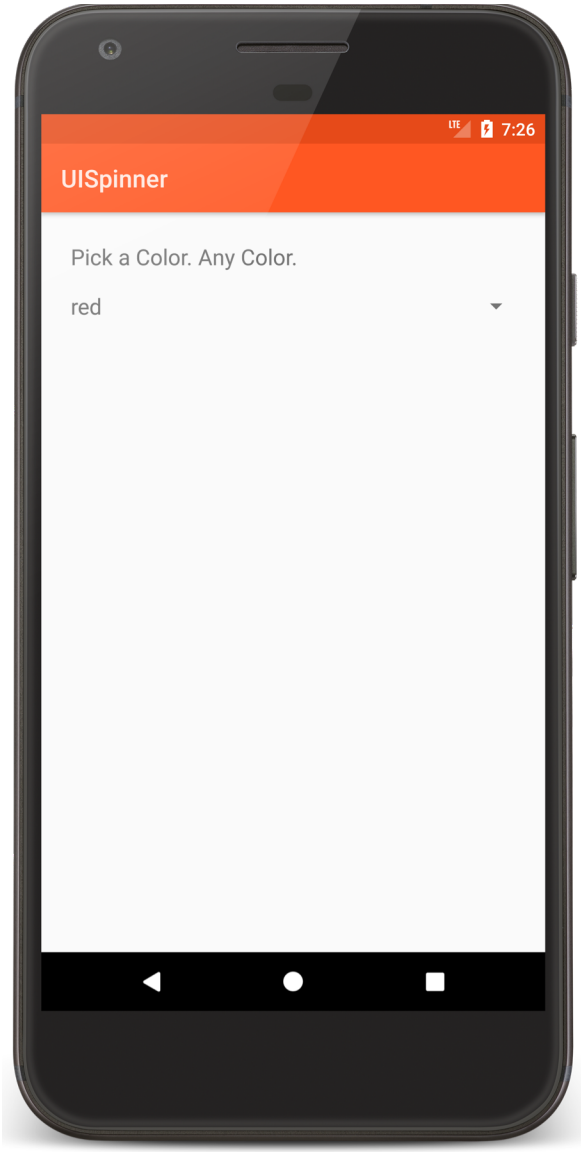
Spinner

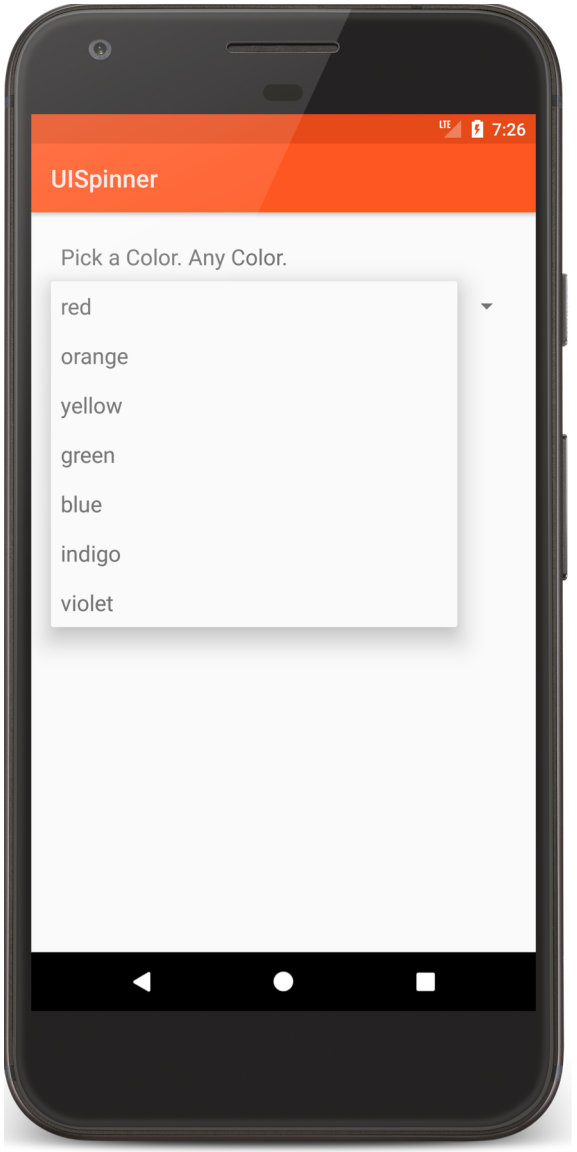
An AdapterView that provides a scrollable list of items

User can select one item from the list

Items managed by a SpinnerAdapter

UISpinner


7:26
UISpinner
Pick a Color. Any Color.
red


7:26
UISpinner
Pick a Color. Any Color.
red
orange
yellow
green
blue
indigo
violet

ViewPager

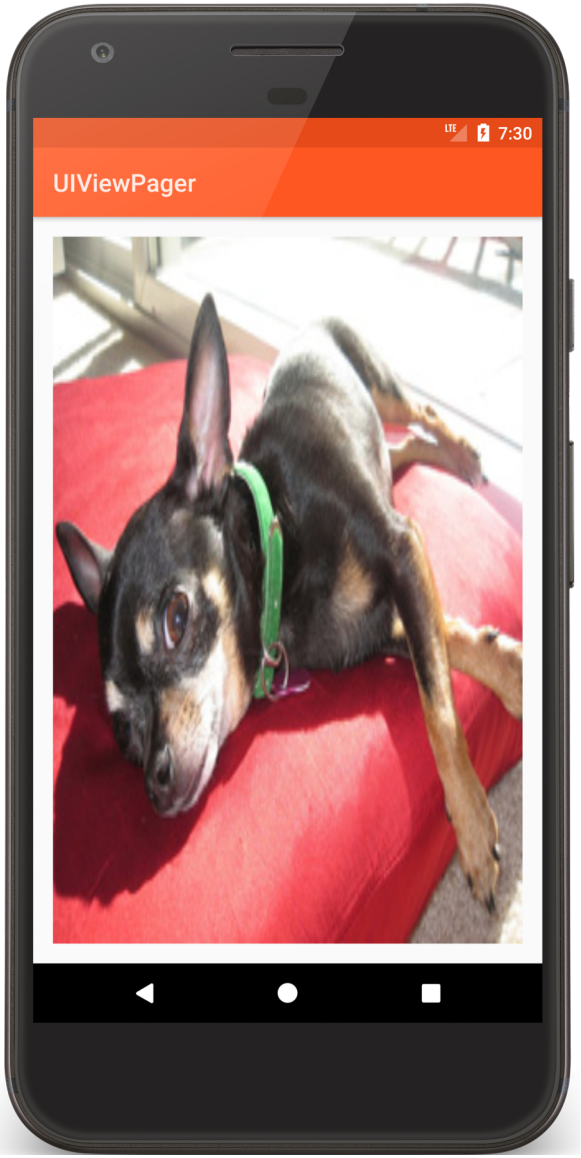
A ViewGroup showing a horizontally scrolling list

Items managed by a PagerAdapter

Two builtin PagerAdapters using Fragments

- FragmentPagerAdapter
- FragmentStatePagerAdapter

UIViewPager



```
public class GalleryWithViewPagerActivity extends Activity {

  protected void onCreate(Bundle savedInstanceState) {
  …

    // Create a new ImageAdapter (subclass of FragmentStatePagerAdapter)
    ImageAdapter imageAdapter = new ImageAdapter(getFragmentManager());

    // Set the Adapter on the ViewPager
    viewPager.setAdapter(imageAdapter);

  }
}
```

```
// Manages Fragments holding ImageViews
class ImageAdapter extends FragmentStatePagerAdapter {
…
public Fragment getItem(int i) {
        Fragment fragment = new ImageHolderFragment();
        Bundle args = new Bundle();
        args.putInt(ImageHolderFragment.RES_ID, mImageIds[i]);
        args.putString(ImageHolderFragment.POS, String.valueOf(i));
        fragment.setArguments(args);
        return fragment;
    }

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

```
// Each instance holds one image to be displayed in the ViewPager
public class ImageHolderFragment extends Fragment {
…
  public View onCreateView(LayoutInflater inflater, ViewGroup container,
                                                    Bundle savedInstanceState) {
    final Bundle args = getArguments();
    mPos = args.getString(POS);
    int mID = args.getInt(RES_ID);
    ImageView imageView = (ImageView) inflater.inflate(R.layout.page,
                                                    container, false);
    // Set the Image for the ImageView
    imageView.setImageResource(mID);
    …
   return imageView;
  }
}
```

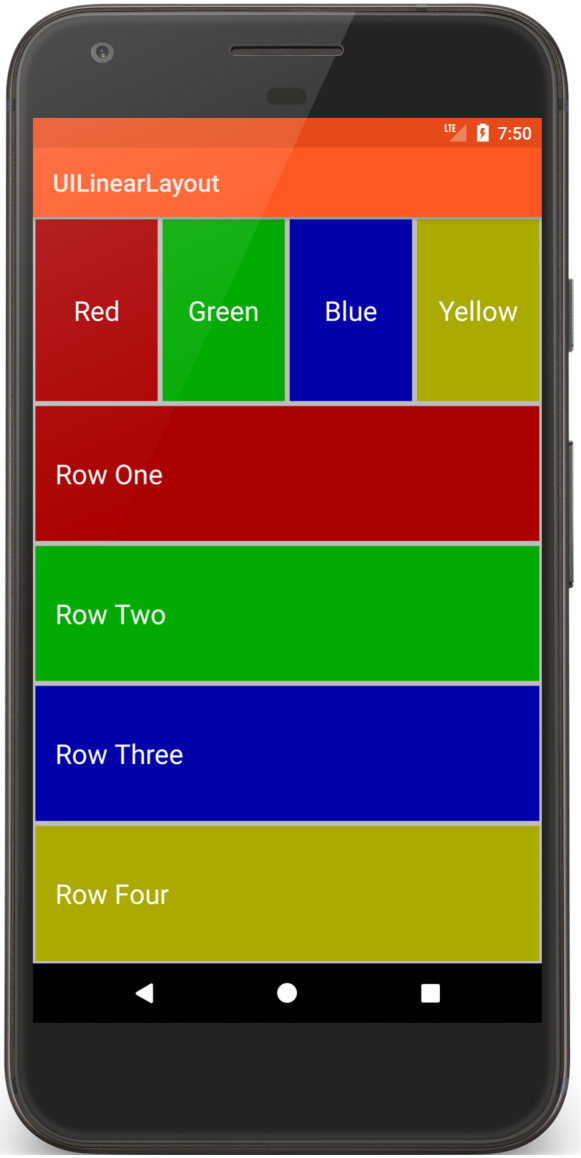
Layouts

A generic Viewgroup that defines a structure/rules for positioning the Views it contains

LinearLayout

Child Views arranged in a single horizontal or vertical row

UILinearLayout

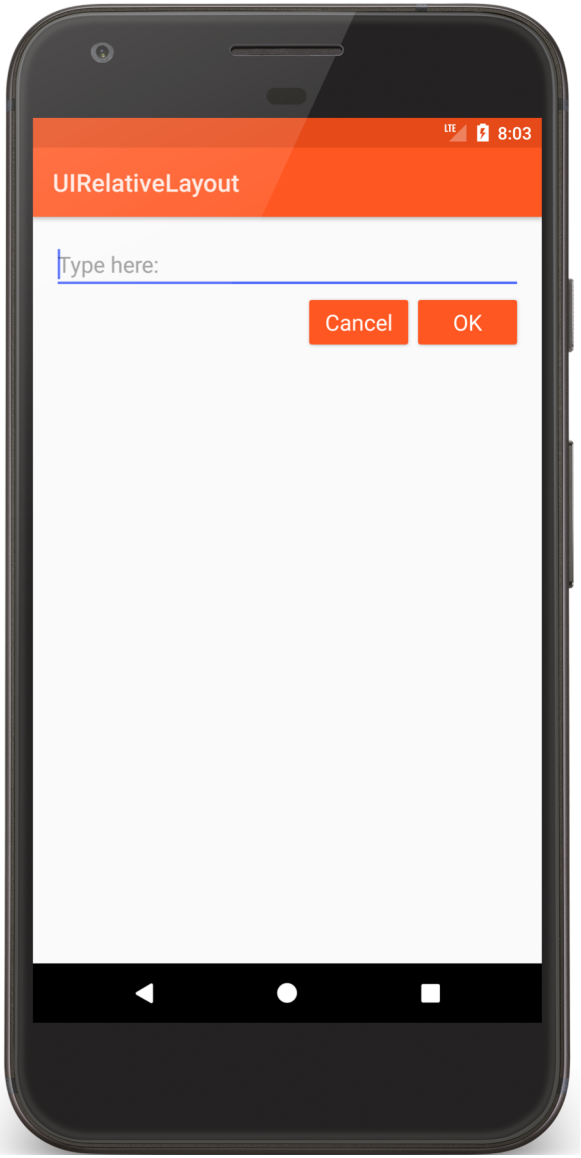


```
<!-- Outermost  LinearLayout with vertical orientation -->
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:background="@color/divider"
   android:orientation="vertical">
   <!--
    Inner LinearLayout with horizontal orientation
     and layout weight of 1 out of 4
   -->
   <LinearLayout
      android:layout_width="match_parent"
      android:layout_height="0dp"
      android:layout_weight="1"
      android:orientation="horizontal">
...
```

RelativeLayout

Child Views are positioned relative to each other and to parent View

UIRelativeLayout



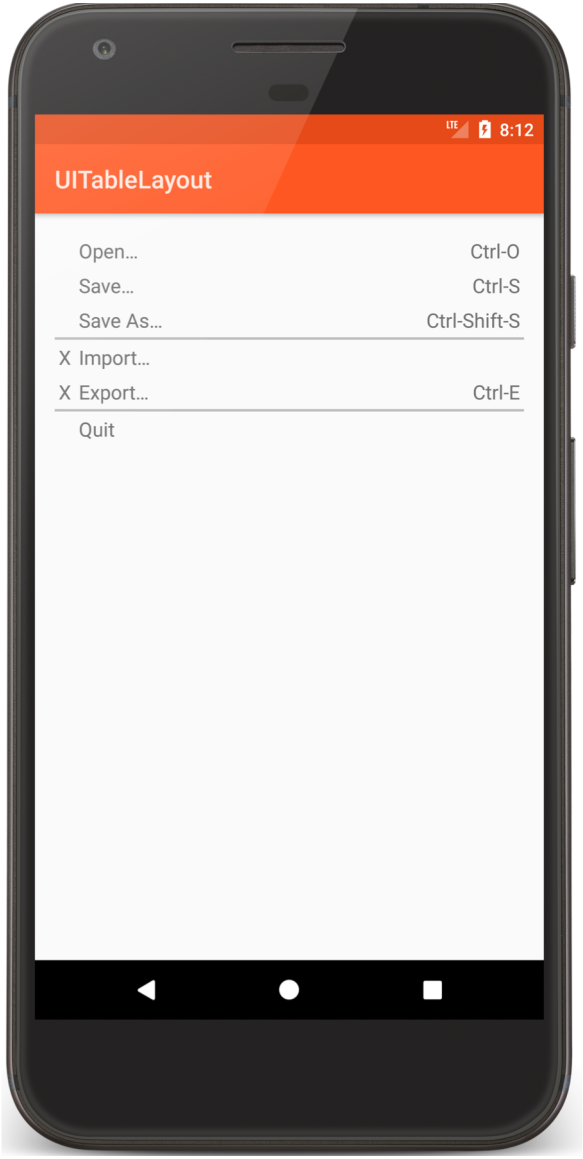
```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:padding="@dimen/activity_margin">

 <EditText
      android:id="@+id/entry"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:hint="@string/type_here_string”
  …/>
```

```
  ...
  <Button
     android:id="@+id/ok_button"
     android:layout_width="wrap_content"
     android:layout_height="wrap_content"
     android:layout_alignParentEnd="true"
     android:layout_below="@id/entry"
    .../>
   <Button
     android:id="@+id/cancel_button"
     android:layout_width="wrap_content"
     android:layout_height="wrap_content"
     android:layout_alignTop="@id/ok_button"
     android:layout_toStartOf="@id/ok_button”
   .../>
</RelativeLayout>
```

TableLayout

Child views arranged into rows & columns

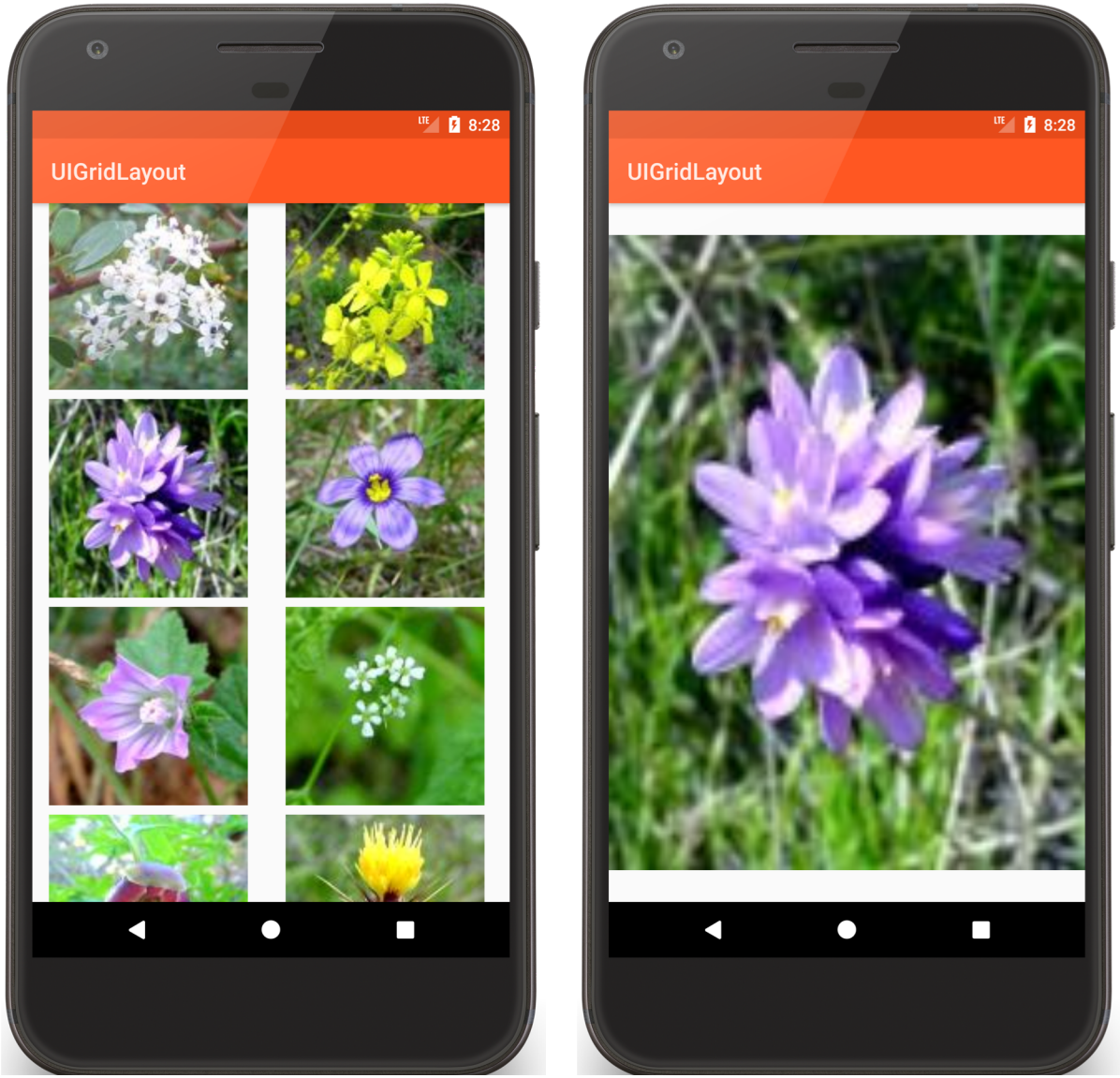
UITableLayout



GridView

Child views arranged in a two-dimensional, scrollable grid

UIGridView



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

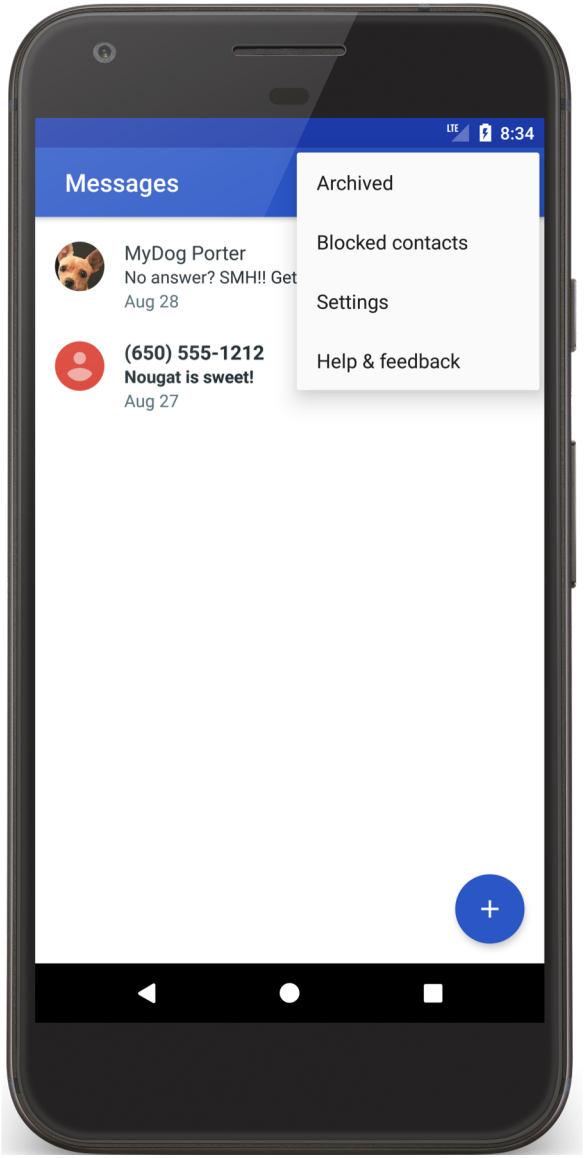
Context

View-specific menu shown when user touches and holds the View

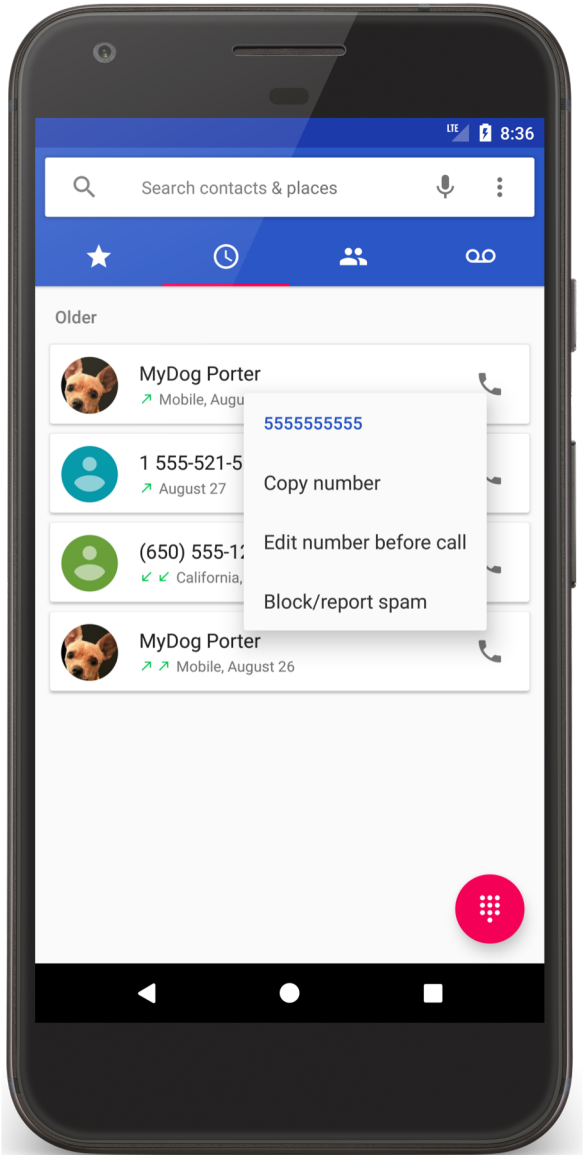
Submenu

A menu activated when user touches a visible menu item

Options Menus



Context Menus



Creating Menus

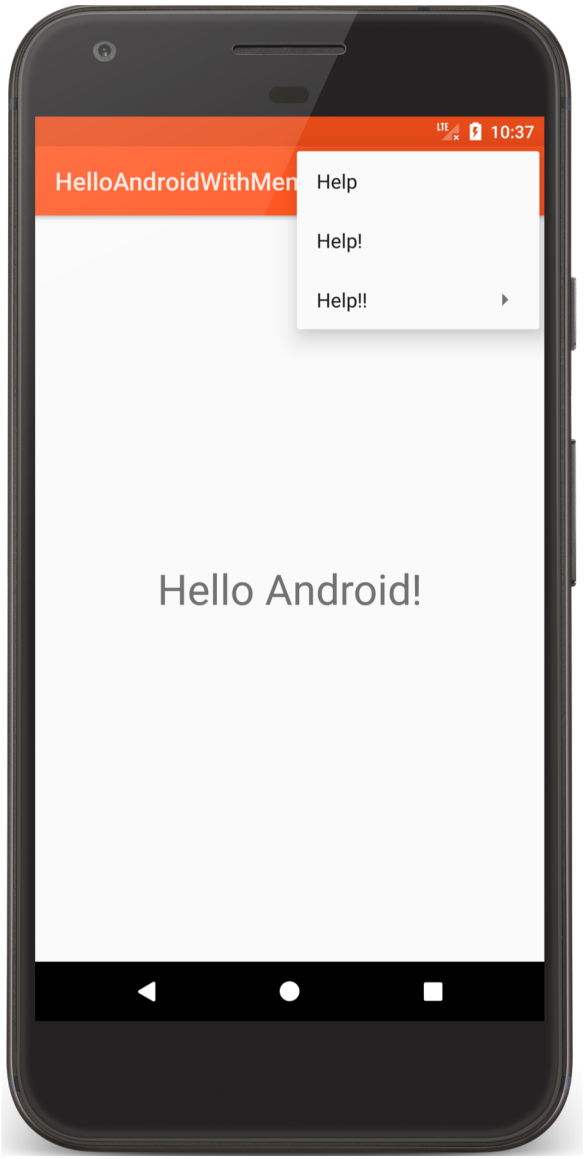
Define menu resource in XML file

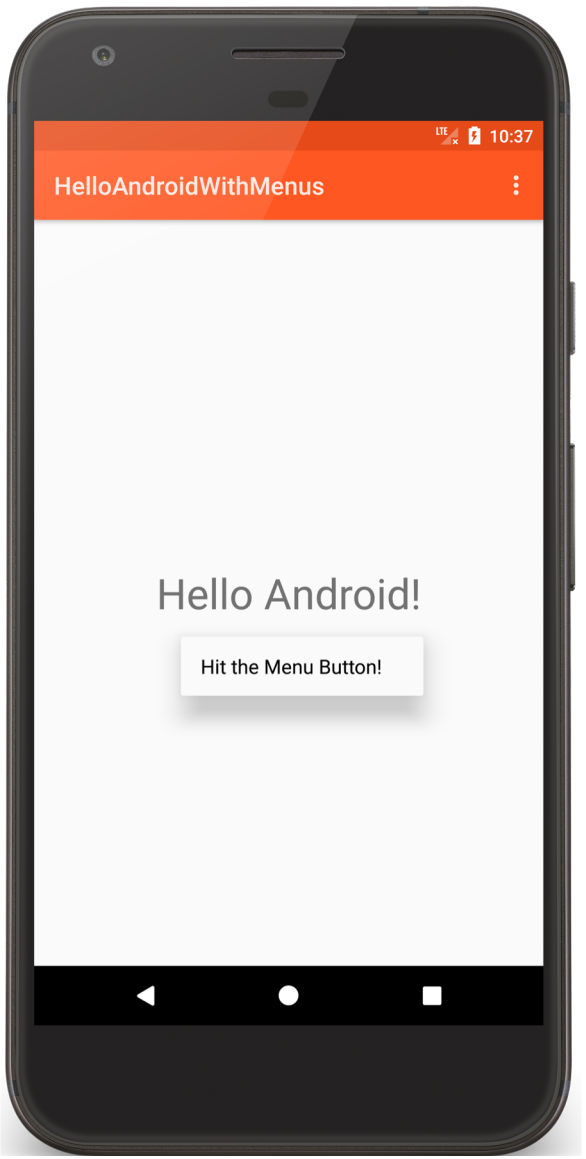
- Store in res/menu/filename.xml

Creating Menus

Inflate menu resource using Menu Inflater in onCreate{Options,Context}Menu() methods

Handling item selection in appropriate on{Options,Context}ItemsSelected() methods





HelloAndroid WithMenus

```
public class HelloAndroidWithMenuActivity extends Activity {
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    setContentView(R.layout.main);
     TextView tv = findViewById(R.id.text_view);

     // Long presses on TextView invoke Context Menu
     registerForContextMenu(tv);

  }
```

```
// Create Options Menu
public boolean onCreateOptionsMenu(Menu menu) {
  MenuInflater inflater = getMenuInflater();
  inflater.inflate(R.menu.top_menu, menu);
  return true;
}
// Process clicks on Options Menu items
public boolean onOptionsItemSelected(MenuItem item) {
  switch (item.getItemId()) {
  case R.id.help:
       // Display message and return true
  case R.id.more_help:
       // Display message and return true
  case R.id.even_more_help:
    return true;
  …
```

```
// Create Context Menu
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
 public boolean onContextItemSelected(MenuItem item) {
    switch (item.getItemId()) {
    case R.id.help_guide:
          // Display message and 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

FragmentDynamicLayoutWithActionBar

- Shows play titles and one quote from selected play
- Provides actions for the ActionBar
- Three main objects
  - QuoteViewerActivity
  - TitleFragment
  - QuoteFragment

FragmentDynamic LayoutWithActionBar

Dialogs

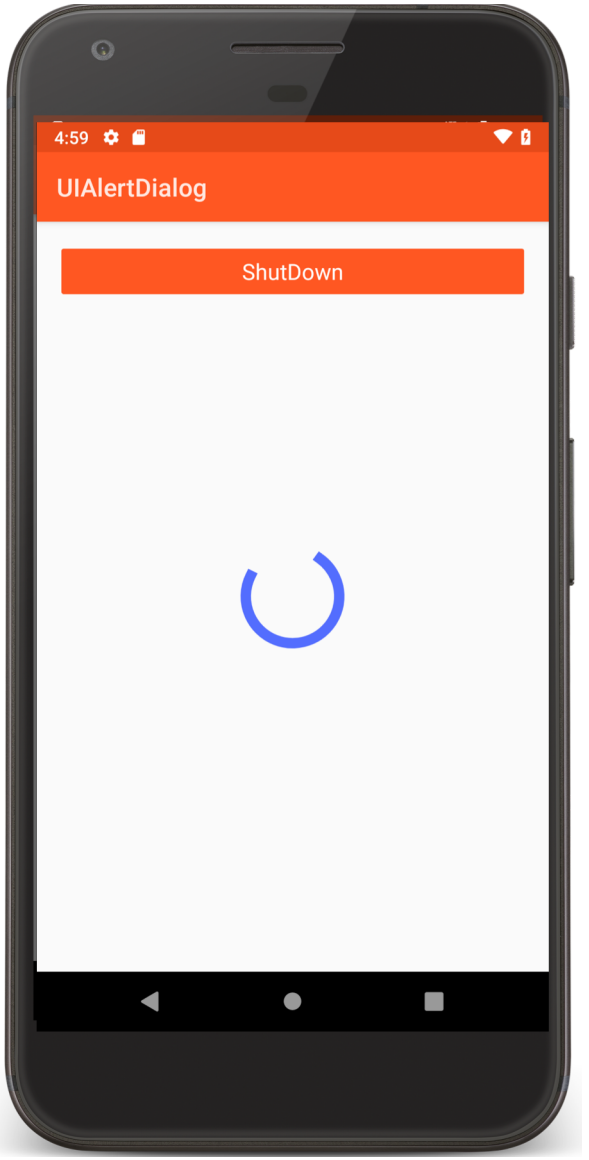
Independent subwindows used by Activities to communicate with user

Dialog Subclasses

AlertDialog

DatePickerDialog

TimePickerDialog



UIAlertDialog

Next

Data Management

Example Applications

UIButton

UIToggleButton

UICheckbox

UIRatingBar

UIAutoCompleteTextView

UIRadioGroup

UITimePickerFragment

UIDatePickerFragment

UIWebView

UIGoogleMaps

UIListView

UIListViewWithCustom Adapter

UIRecyclerView

UISpinner

UIViewPager

UILinearLayout

UIRelativeLayout

UITableLayout

UIGridView

HelloAndroidWithMenus

UIAlertDialog