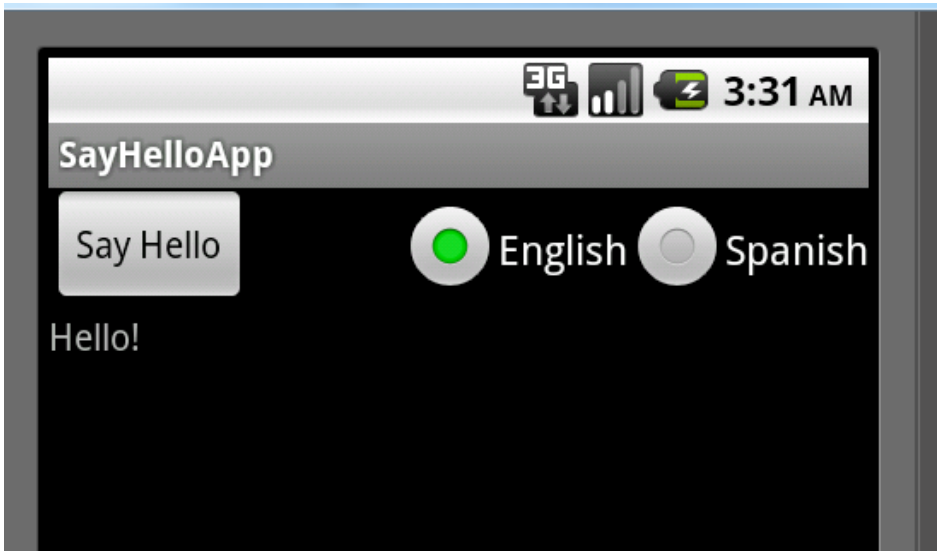
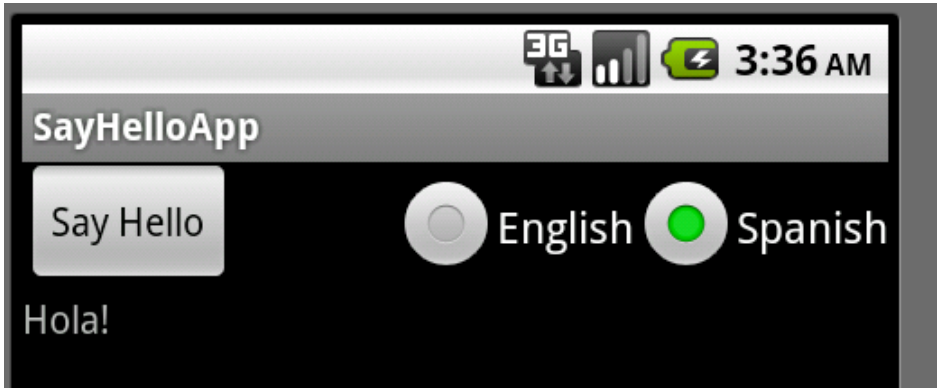
Android Programming
Lecture 5

9/16/2011

RadioButtons

- Multiple RadioButtons belong in a RadioGroup
- RadioGroup
  - Manages only-one-selected
  - Exposes functions to:
    - programmatically select a button via ID
    - determine ID of button currently selected

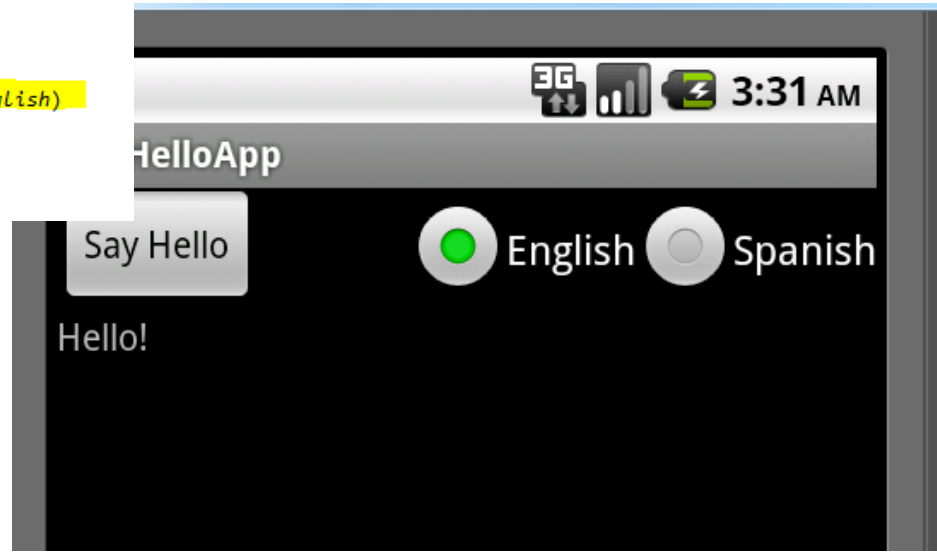




RadioButtons

```java
public class SayHelloAppActivity extends Activity implements View.OnClickListener {

    Button button;
    TextView textView;
    RadioGroup languageButtonsGroup;

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        button = (Button)findViewById(R.id.button);
        languageButtonsGroup = (RadioGroup)findViewById(R.id.radio_group);
        textView = (TextView)findViewById(R.id.textview);

        button.setOnClickListener(this);
        languageButtonsGroup.check(R.id.radio_english);

    }

    @Override
    public void onClick(View arg0) {
        // TODO Auto-generated method stub
        if (arg0.getId() == button.getId())
        {
            if (languageButtonsGroup.getCheckedRadioButtonId() == R.id.radio_english)
                textView.setText("Hello!");
            else textView.setText("Hola!");
        }
    }
```

```xml
<RadioGroup android:id="@+id/radio_group"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_alignParentRight="true"
        android:orientation="horizontal">
        <RadioButton android:id="@+id/radio_english"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="English" />
        <RadioButton android:id="@+id/radio_spanish"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Spanish" />
    </RadioGroup>
```

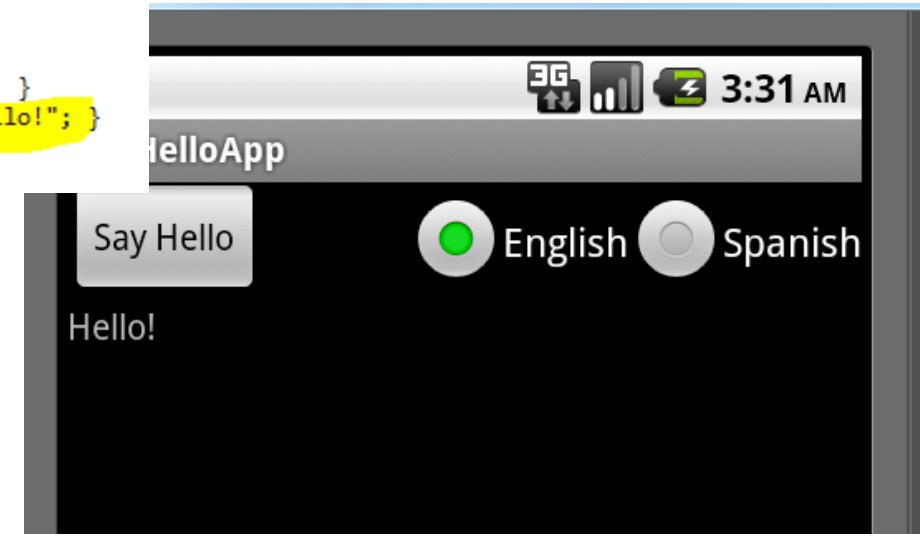


This is not explicitly using the Listener pattern for the RadioButtons (not responding to the click; just checking what is clicked)

RadioButtons

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

    button = (Button)findViewById(R.id.button);
    //languageButtonsGroup = (RadioGroup)findViewById(R.id.radio_group);
    textView = (TextView)findViewById(R.id.textview);

    englishButton = (RadioButton)findViewById(R.id.radio_english);
    spanishButton = (RadioButton)findViewById(R.id.radio_spanish);
    englishButton.setOnClickListener(this);
    spanishButton.setOnClickListener(this);

    button.setOnClickListener(this);
    //languageButtonsGroup.check(R.id.radio_english);
    text = "Hello!";
    englishButton.toggle();
}

@Override
public void onClick(View arg0) {
    // TODO Auto-generated method stub
    if (arg0.getId() == button.getId()) { textView.setText(text); }
    else if (arg0.getId() == englishButton.getId()) { text = "Hello!"; }
    else { text = "Hola!"; }
}
```

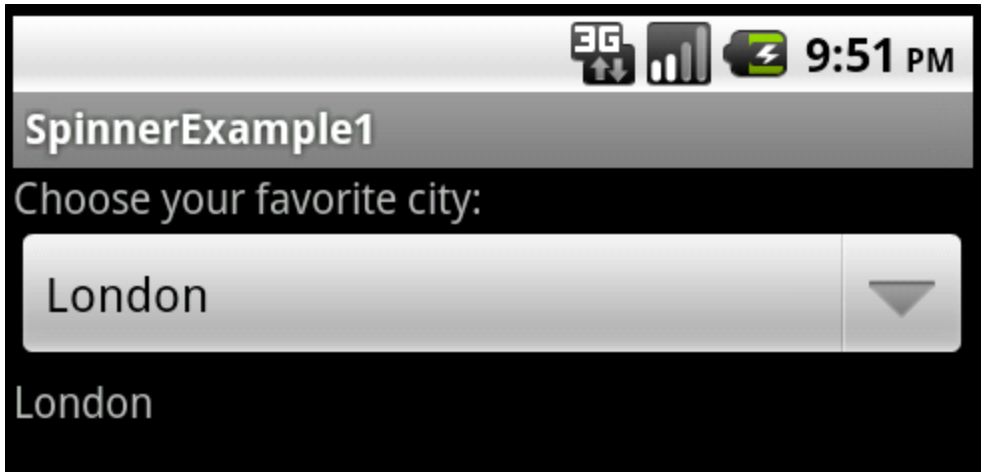
Can also listen directly to clicks on RadioButtons



Spinner

- Drop-down selection box, with a list of RadioButton options
- Provides screen real-estate savings over multiple individual RadioButtons
- Requires a different approach to setup
  - An array of choices to present
  - An ArrayAdapter to format the array data for the Spinner
  - The actual Spinner view component, associated with the ArrayAdapter

Spinner



SpinnerExample1Activity.java | main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
    <TextView android:id="@+id/city_prompt"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    >
    </TextView>
    <Spinner android:id="@+id/city_spinner"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
    />
    <TextView android:id="@+id/outputTextView"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    />
</LinearLayout>
```

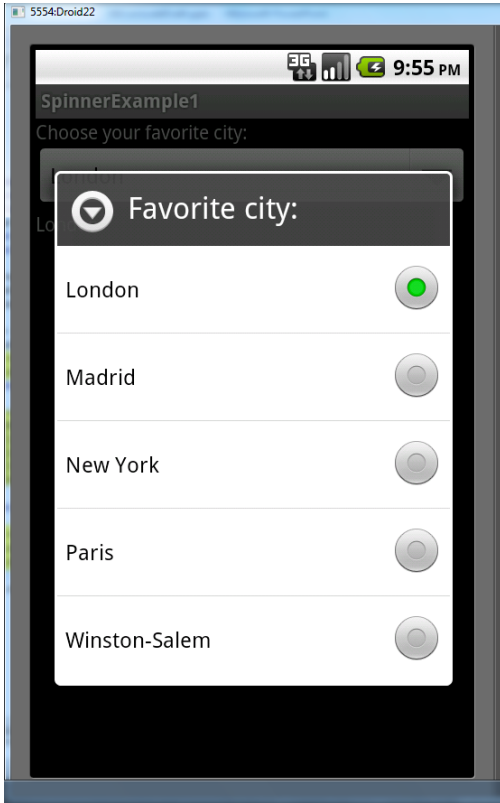
Spinner

```
TextView cityPrompt;
Spinner citySpinner;
TextView outputTextView;
ArrayAdapter<CharSequence> cityAdapter;
String[] citiesArray;
int cityLastSelected;

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

    cityPrompt = (TextView)findViewById(R.id.city_prompt);
    cityPrompt.setText("Choose your favorite city:");
    citySpinner = (Spinner)findViewById(R.id.city_spinner);
    citySpinner.setPrompt("Favorite city:"); // prompt that appears on pop-up list overlay
    outputTextView = (TextView)findViewById(R.id.outputTextView);

    citiesArray = new String[5];
    citiesArray[0] = "London";
    citiesArray[1] = "Madrid";
    citiesArray[2] = "New York";
    citiesArray[3] = "Paris";
    citiesArray[4] = "Winston-Salem";

    cityLastSelected = 0;

    cityAdapter = new ArrayAdapter<CharSequence> (this, android.R.layout.simple_spinner_item, citiesArray);
    cityAdapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
    citySpinner.setAdapter(cityAdapter);

    citySpinner.setOnItemSelectedListener(this);

}
```

Setup prompt at top of pop-up that appears

Set up array

Associate adapter with array and spinner with adapter



Spinner: Listening for Selection Events in Spinner

public static interface

AdapterView.OnItemSelectedListener

android.widget.AdapterView.OnItemSelectedListener

Class Overview

Interface definition for a callback to be invoked when an item in this view has been selected.

Summary

Public Methods	
abstract void	onItemSelected (AdapterView<?> parent, View view, int position, long id) Callback method to be invoked when an item in this view has been selected.
abstract void	onNothingSelected (AdapterView<?> parent) Callback method to be invoked when the selection disappears from this view.

Public Methods

public abstract void **onItemSelected** (AdapterView<?> parent, View view, int position, long id)

Callback method to be invoked when an item in this view has been selected. Impelmenters can call getItemAtPosition(position) if they need to access the data associated with the selected item.

Parameters

parent	The AdapterView where the selection happened
view	The view within the AdapterView that was clicked
position	The position of the view in the adapter
id	The row id of the item that is selected

public abstract void **onNothingSelected** (AdapterView<?> parent)

Callback method to be invoked when the selection disappears from this view. The selection can disappear for instance when touch is activated or when the adapter becomes empty.

Parameters

parent	The AdapterView that now contains no selected item.

Spinner: Listening for Selection Events in Spinner

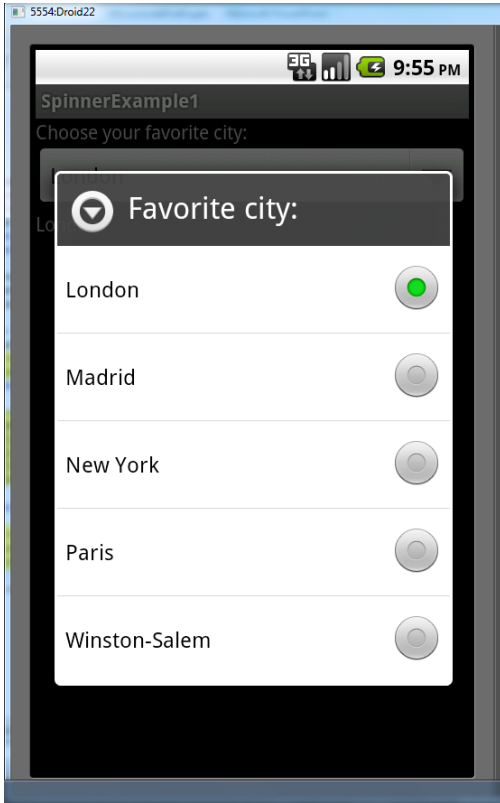
```
public class SpinnerExample1Activity extends Activity implements AdapterView.OnItemSelectedListener {

    citiesArray[3] = "Paris";
    citiesArray[4] = "Winston-Salem";

    cityLastSelected = 0;

    cityAdapter = new ArrayAdapter<CharSequence> (this, android.R.layout.simple_spinner_item, citiesArray);
    cityAdapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
    citySpinner.setAdapter(cityAdapter);

    citySpinner.setOnItemSelectedListener(this);

}

public void onItemSelected(AdapterView<?> parent, View view, int position, long id) {
    if (position != cityLastSelected)
    {
        outputTextView.setText((String)parent.getItemAtPosition(position));
        cityLastSelected = position;
    }
}

public void onNothingSelected(AdapterView<?> parent) {
}
```

Wait a Minute!

```
TextView cityPrompt;
Spinner citySpinner;
TextView outputTextView;
ArrayAdapter<CharSequence> cityAdapter;
String[] citiesArray;
int cityLastSelected;

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

    cityPrompt = (TextView)findViewById(R.id.city_prompt);
    cityPrompt.setText("Choose your favorite city:");
    citySpinner = (Spinner)findViewById(R.id.city_spinner);
    citySpinner.setPrompt("Favorite city:"); // prompt that appears on pop-up list overlay
    outputTextView = (TextView)findViewById(R.id.outputTextView);

    citiesArray = new String[5];
    citiesArray[0] = "London";
    citiesArray[1] = "Madrid";
    citiesArray[2] = "New York";
    citiesArray[3] = "Paris";
    citiesArray[4] = "Winston-Salem";

    cityLastSelected = 0;

    cityAdapter = new ArrayAdapter<CharSequence> (this, android.R.layout.simple_spinner_item, citiesArray);
    cityAdapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
    citySpinner.setAdapter(cityAdapter);

    citySpinner.setOnItemSelectedListener(this);

}
```

Set up array

Isn't this dealing with
setting up the interface, in the code?
Can I make this an XML 'resource' too?

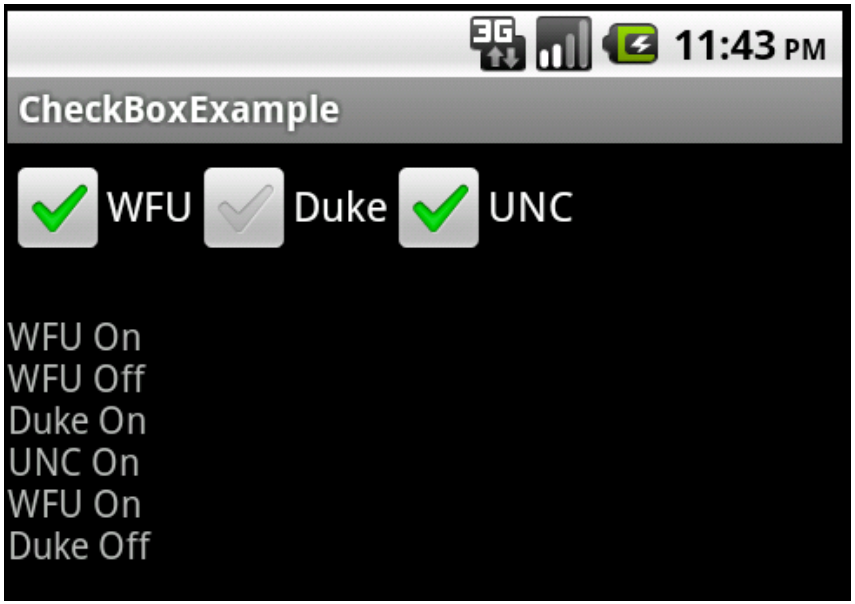


CheckBox

- 2-state toggles (on/off)
- Not organized in a group
  - Some, all, or none can be checked in a logical group
- Generates Click events
  - A Click event only says a click happened: Need to be able to check state of CheckBox
    - Call *boolean isChecked()* method on CheckBox

CheckBox

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

    wfuCheckBox = (CheckBox)findViewById(R.id.wfu_checkbox);
    dukeCheckBox = (CheckBox)findViewById(R.id.duke_checkbox);
    uncCheckBox = (CheckBox)findViewById(R.id.unc_checkbox);

    wfuCheckBox.setOnClickListener(this);
    dukeCheckBox.setOnClickListener(this);
    uncCheckBox.setOnClickListener(this);

    textView = (TextView)findViewById(R.id.textview);
}

public void onClick(View args0) {
    if (args0.getId() == R.id.wfu_checkbox) {
        if (wfuCheckBox.isChecked())
            textView.setText(textView.getText() + "\nWFU On");
        else
            textView.setText(textView.getText() + "\nWFU Off");
    }
    else if (args0.getId() == R.id.duke_checkbox) {
        if (dukeCheckBox.isChecked())
            textView.setText(textView.getText() + "\nDuke On");
        else
            textView.setText(textView.getText() + "\nDuke Off");
    }
    else if (args0.getId() == R.id.unc_checkbox) {
        if (uncCheckBox.isChecked())
            textView.setText(textView.getText() + "\nUNC On");
        else
            textView.setText(textView.getText() + "\nUNC Off");
    }
}
```

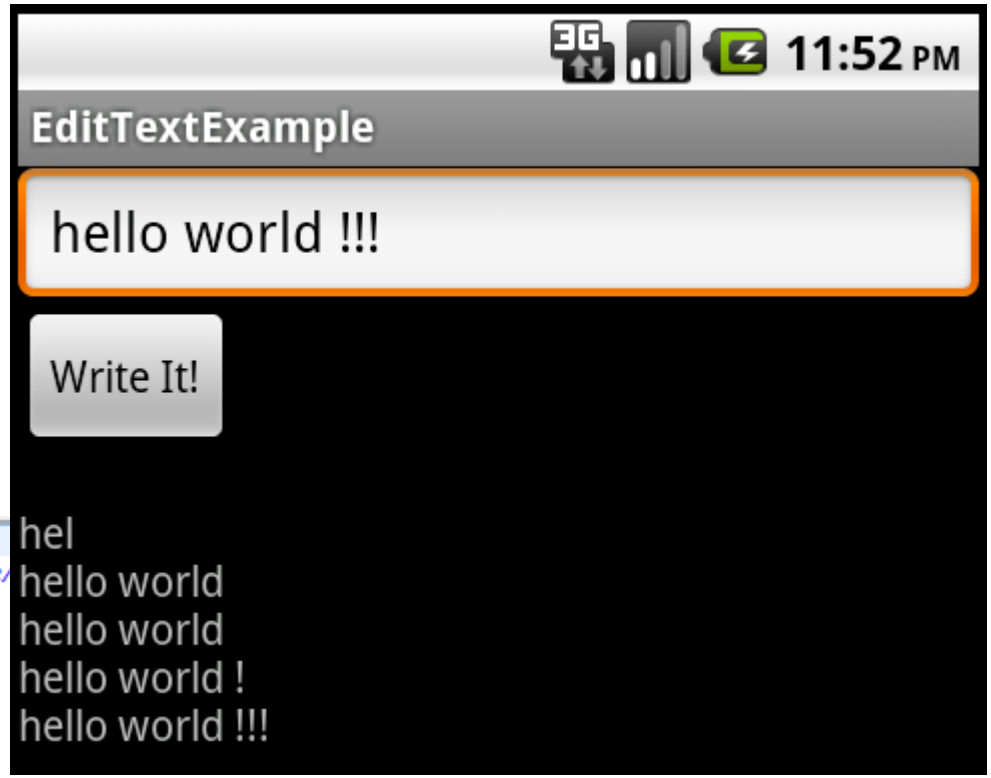


EditText

- Allows entry of arbitrary text from physical or virtual keyboard
- Typically end of entry is signified by
  - Associated button being clicked OR (a Click event)
  - Return being pressed in the editable field (a KeyPress event)
- Can constrain to only certain types of input

EditText

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
    <EditText android:id="@+id/edit_text"
        android:layout_height="wrap_content"
        android:layout_width="fill_parent"
        ></EditText>
    <Button android:id="@+id/button"
        android:layout_height="wrap_content"
        android:layout_width="wrap_content"
        android:text="Write It!"
        ></Button>
    <TextView android:id="@+id/text_view"
        android:text=""
        android:layout_width="fill_parent"
        android:layout_height="fill_parent">
        </TextView>
</LinearLayout>
```



EditText

```
public class EditTextExampleActivity extends Activity implements View.OnClickListener, View.OnKeyListener {

    EditText editText;
    Button writeButton;
    TextView textView;
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        editText = (EditText)findViewById(R.id.edit_text);
        writeButton = (Button)findViewById(R.id.button);
        textView = (TextView)findViewById(R.id.text_view);

        writeButton.setOnClickListener(this);
        editText.setOnKeyListener(this);
    }

    public void onClick(View v)    {
        if (v.getId() == R.id.button) {
            textView.setText(textView.getText() + "\n" + editText.getText());
        }
    }

    public boolean onKey(View v, int keyCode, KeyEvent event)
    {
        if (v.getId() == R.id.edit_text) {
            if ((event.getAction() == KeyEvent.ACTION_DOWN) && (keyCode == KeyEvent.KEYCODE_ENTER)) {
                textView.setText(textView.getText() + "\n" + editText.getText());
                return true;
            }
        }
        return false;
    }
}
```

Edit Text: Constraining Type

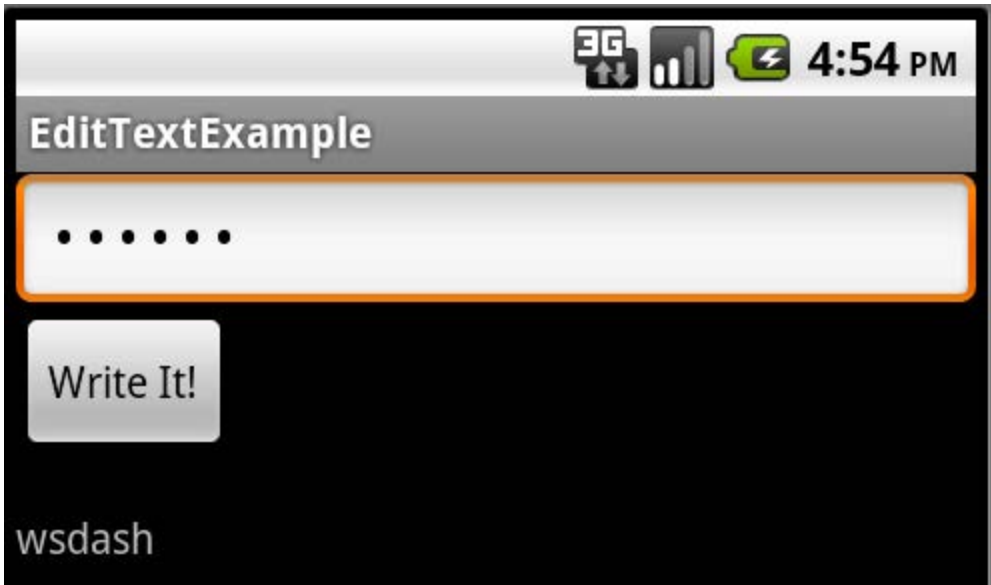
int	TYPE_CLASS_DATETIME	Class for dates and times.
int	TYPE_CLASS_NUMBER	Class for numeric text.
int	TYPE_CLASS_PHONE	Class for a phone number.
int	TYPE_CLASS_TEXT	Class for normal text.
int	TYPE_DATETIME_VARIATION_DATE	Default variation of TYPE_CLASS_DATETIME: allows entering only a date.
int	TYPE_DATETIME_VARIATION_NORMAL	Default variation of TYPE_CLASS_DATETIME: allows entering both a date and time.
int	TYPE_DATETIME_VARIATION_TIME	Default variation of TYPE_CLASS_DATETIME: allows entering only a time.
int	TYPE_MASK_CLASS	Mask of bits that determine the overall class of text being given.
int	TYPE_MASK_FLAGS	Mask of bits that provide addition bit flags of options.
int	TYPE_MASK_VARIATION	Mask of bits that determine the variation of the base content class.
int	TYPE_NULL	Special content type for when no explicit type has been specified.
int	TYPE_NUMBER_FLAG_DECIMAL	Flag of TYPE_CLASS_NUMBER: the number is decimal, allowing a decimal point to provide fractional values.
int	TYPE_NUMBER_FLAG_SIGNED	Flag of TYPE_CLASS_NUMBER: the number is signed, allowing a positive or negative sign at the start.
int	TYPE_NUMBER_VARIATION_NORMAL	Default variation of TYPE_CLASS_NUMBER: plain normal numeric text.
int	TYPE_NUMBER_VARIATION_PASSWORD	Variation of TYPE_CLASS_NUMBER: entering a numeric password.
int	TYPE_TEXT_FLAG_AUTO_COMPLETE	Flag for TYPE_CLASS_TEXT: the text editor is performing auto-completion of the text being entered based on its own semantics, which it will present to the user as they type.
int	TYPE_TEXT_FLAG_AUTO_CORRECT	Flag for TYPE_CLASS_TEXT: the user is entering free-form text that should have auto-correction applied to it.
int	TYPE_TEXT_FLAG_CAP_CHARACTERS	Flag for TYPE_CLASS_TEXT: capitalize all characters.
int	TYPE_TEXT_FLAG_CAP_SENTENCES	Flag for TYPE_CLASS_TEXT: capitalize first character of each sentence.
int	TYPE_TEXT_FLAG_CAP_WORDS	Flag for TYPE_CLASS_TEXT: capitalize first character of all words.
int	TYPE_TEXT_FLAG_IME_MULTI_LINE	Flag for TYPE_CLASS_TEXT: the regular text view associated with this should not be multi-line, but when a fullscreen input method is providing text it should use multiple lines if it can.
int	TYPE_TEXT_FLAG_MULTI_LINE	Flag for TYPE_CLASS_TEXT: multiple lines of text can be entered into the field.

Even more: developer.android.com/reference/android/text/InputType.html

EditText: Constraining Type

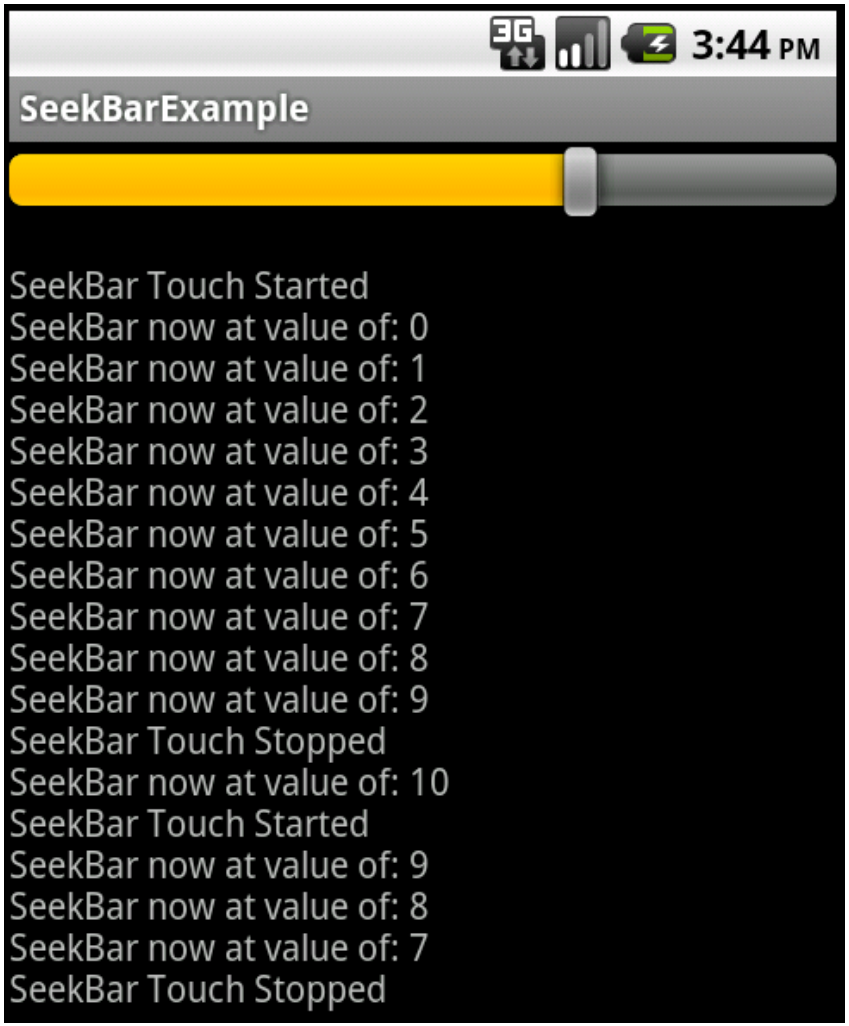
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
    <EditText android:id="@+id/edit_text"
        android:layout_height="wrap_content"
        android:layout_width="fill_parent"
        android:inputType="textPassword"
        ></EditText>
    <Button android:id="@+id/button"
        android:layout_height="wrap_content"
        android:layout_width="wrap_content"
        android:text="Write It!"
        ></Button>
    <TextView android:id="@+id/text_view"
        android:text="" |
        android:layout_width="fill_parent"
        android:layout_height="fill_parent">
        </TextView>
</LinearLayout>
```

Password type -> hides each letter after it is typed



SeekBar

- SeekBar (slider) allow selection of *integer values* using a natural interface
- Constraints:
  - Min: 0
  - Max: settable
  - Changes by: 1
  - Starting point for knob can be set
- Senses initiation of touch ,ending of touch, and movement



SeekBar

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
<SeekBar    android:id="@+id/seekbar"
   android:layout_width="fill_parent"
   android:layout_height="wrap_content"
   android:max="10"
   android:progress="1"
   />
<TextView  android:id="@+id/textview"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:text=""
    />
</LinearLayout>
```

SeekBar

```
public class SeekBarExampleActivity extends Activity implements SeekBar.OnSeekBarChangeListener {

    SeekBar seekBar;
    TextView textView;

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        seekBar = (SeekBar)findViewById(R.id.seekbar);
        textView = (TextView)findViewById(R.id.textview);
        seekBar.setOnSeekBarChangeListener(this);
    }

    public void onProgressChanged(SeekBar seekBar, int progress, boolean fromUser) {
            textView.setText(textView.getText() + "\n" + "SeekBar now at value of: " + progress);
        }

    public void onStartTrackingTouch(SeekBar seekBar) {
            textView.setText(textView.getText() + "\nSeekBar Touch Started");
        }

    public void onStopTrackingTouch(SeekBar seekBar) {
            textView.setText(textView.getText() + "\nSeekBar Touch Stopped");
        }
}
```