I3350
Mobile Application Development

Chapter 2

WIDGETS

Recall: Android widgets

9:26:00 pm Analog/DigitalClock	Button 1 Button 2 Button 3 Button	Plain Serif Bold Italic Checkbox	12:15 PM Date/TimePicker
EditText 1 (206)555-1212 EditText	Gallery	ImageView/Button	ProgressBar
Plain Serif Bold Bold & Italic RadioButton	Spinner Spinner	Plain Serif Bold Italic TextView	MapView, WebView

Button (link)

A clickable widget with a text label

- key attributes:

`android:clickable="`***`bool`***`"`	set to false to disable the button
`android:id="@+id/`***`theID`***`"`	unique ID for use in Java code
`android:onClick="`***`function`***`"`	function to call in activity when clicked (must be public, void, and take a View arg)
`android:text="`***`text`***`"`	text to put in the button

- represented by `Button` class in Java code

```
Button b = (Button) findViewById(R.id.theID);
...
```

ImageButton

A clickable widget with an image label

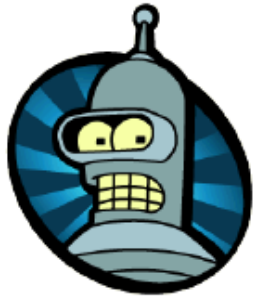
- key attributes:

`android:clickable="`***`bool`***`"`	set to false to disable the button
`android:id="@+id/`***`theID`***`"`	unique ID for use in Java code
`android:onClick="`***`function`***`"`	function to call in activity when clicked (must be public, void, and take a View arg)
`android:src="@drawable/`***`img`***`"`	image to put in the button (must correspond to an image resource)

- to set up an image resource:
  - put image file in project folder **app/src/main/res/drawable**
  - use `@drawable/foo` to refer to `foo.png`
    - use simple file names with only letters and numbers

ImageView

Displays an image without being clickable



- key attributes:

`android:id="@+id/`***`theID`***`"`	unique ID for use in Java code
`android:src="@drawable/`***`img`***`"`	image to put in the screen (must correspond to an image resource)

- to change the visible image, in Java code:
  - get the ImageView using findViewById
  - call its **setImageResource** method and pass R.drawable.***filename***

EditText (link)

An editable text input box

- key attributes:

`android:hint="`***`text`***`"`	gray text to show before user starts to type
`android:id="@+id/`***`theID`***`"`	unique ID for use in Java code
`android:inputType="`***`type`***`"`	what kind of input is being typed; `number,phone,date,time,...`
`android:lines="`***`int`***`"`	number of visible lines (rows) of input
`android:maxLines="`***`int`***`"`	max lines to allow user to type in the box
`android:text="`***`text`***`"`	initial text to put in box (default empty)
`android:textSize="`***`size`***`"`	size of font to use (e.g. "20dp")

  - others: `capitalize, digits, fontFamily, letterSpacing, lineSpacingExtra, minLines, numeric, password, phoneNumber, singleLine, textAllCaps, textColor, typeface`

CheckBox (link)

An individual toggleable on/off switch

- key attributes:

`android:checked="bool"`	set to true to make it initially checked
`android:clickable="bool"`	set to false to disable the checkbox
`android:id="@+id/theID"`	unique ID for use in Java code
`android:onClick="function"`	function to call in activity when clicked (must be public, void, and take a View arg)
`android:text="text"`	text to put next to the checkbox

- In Java code:

```
CheckBox cb = (CheckBox) findViewById(R.id.theID);
cb.toggle();
cb.setChecked(true);
cb.performClick();
```

```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">
    <CheckBox android:id="@+id/checkbox_meat"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/meat"
        android:onClick="onCheckboxClicked"/>
    <CheckBox android:id="@+id/checkbox_cheese"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/cheese"
        android:onClick="onCheckboxClicked"/>
</LinearLayout>
```

```java
public void onCheckboxClicked(View view) {
    // Is the view now checked?
    boolean checked = ((CheckBox) view).isChecked();

    // Check which checkbox was clicked
    switch(view.getId()) {
        case R.id.checkbox_meat:
            if (checked)
                // Put some meat on the sandwich
            else
                // Remove the meat
            break;
        case R.id.checkbox_cheese:
            if (checked)
                // Cheese me
            else
                // I'm lactose intolerant
            break;
        // TODO: Veggie sandwich
    }
}
```

RadioButton (link)

A toggleable on/off switch; part of a group

- key attributes:

`android:checked="`***`bool`***`"`	set to true to make it initially checked
`android:clickable="`***`bool`***`"`	set to false to disable the button
`android:id="@+id/`***`theID`***`"`	unique ID for use in Java code
`android:onClick="`***`function`***`"`	function to call in activity when clicked (must be public, void, and take a View arg)
`android:text="`***`text`***`"`	text to put next to the button

- need to be nested inside a `RadioGroup` tag in XML so that only one can be selected at a time

RadioGroup example

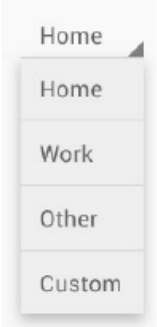
```
<LinearLayout ...
        android:orientation="vertical"
        android:gravity="center|top">
    <RadioGroup ...
            android:orientation="horizontal">
        <RadioButton ... android:id="@+id/lions"
                         android:text="Lions"
                         android:onClick="radioClick" />
        <RadioButton ... android:id="@+id/tigers"
                         android:text="Tigers"
                         android:checked="true"
                         android:onClick="radioClick" />
        <RadioButton ... android:id="@+id/bears"
                         android:text="Bears, oh my!"
                         android:onClick="radioClick" />
    </RadioGroup>
</LinearLayout>
```

Reusing onClick handler

```
// in MainActivity.java
public class MainActivity extends Activity {

    public void radioClick(View view) {
        // check which radio button was clicked
        if (view.getId() == R.id.lions) {
            // ...
        } else if (view.getId() == R.id.tigers) {
            // ...
        } else {
            // bears ...
        }
    }
```

Spinner (link)

A drop-down menu of selectable choices



- key attributes:

`android:clickable="bool"`	set to false to disable the spinner
`android:id="@+id/theID"`	unique ID for use in Java code
`android:entries="@array/array"`	set of options to appear in spinner (must match an array in `strings.xml`)
`android:prompt="@string/text"`	title text when dialog of choices pops up

- also need to handle events in Java code (see later)
  - must get the Spinner object using findViewById
  - then call its setOnItemSelectedListener method (see example)

String resources

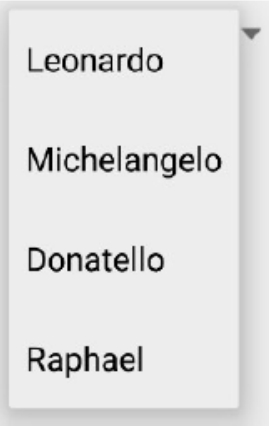
- Declare constant strings and arrays in `res/values/strings.xml`:

```
<resources>
    <string name="name">value</string>
    <string name="name">value</string>

    <string-array name="arrayname">
        <item>value</item>
        <item>value</item>
        <item>value</item>   <!-- must escape ' as \' in values -->
        ...
        <item>value</item>
    </string-array>
</resources>
```

- Refer to them in Java code:
  - as a resource: `R.string.name`, `R.array.name`
  - as a string or array: `getResources().getString(R.string.name)`, `getResources().getStringArray(R.array.name)`

Spinner example

```
<LinearLayout ...>
    <Spinner ... android:id="@+id/tmnt"
        android:entries="@array/turtles"
        android:prompt="@string/choose_turtle" />
    <TextView ... android:id="@+id/result" />
</LinearLayout>
```

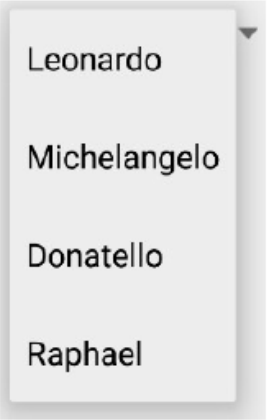


- in res/values/strings.xml:

```
<resources>
    <string name="choose_turtle">Choose a turtle:</string>
    <string-array name="turtles">
        <item>Leonardo</item>
        <item>Michelangelo</item>
        <item>Donatello</item>
        <item>Raphael</item>
    </string-array>
</resources>
```

Spinner event example

```
// in MainActivity.java
public class MainActivity extends Activity {
  ...
  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    Spinner spin = (Spinner) findViewById(R.id.tmnt);
    spin.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
      public void onItemSelected(AdapterView<?> spin, View v, int i, long id) {
        TextView result = (TextView) findViewById(R.id.turtle_result);
        result.setText("You chose " + spin.getSelectedItem());
      }

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



TMNT app exercise

- Write an app to select TMNT characters from a spinner.
  - When a character is selected, an image about that character and other information is presented to the user.
  - Assume that relevant image files are already available for each character.

ScrollView

A container with scrollbars around another widget or container

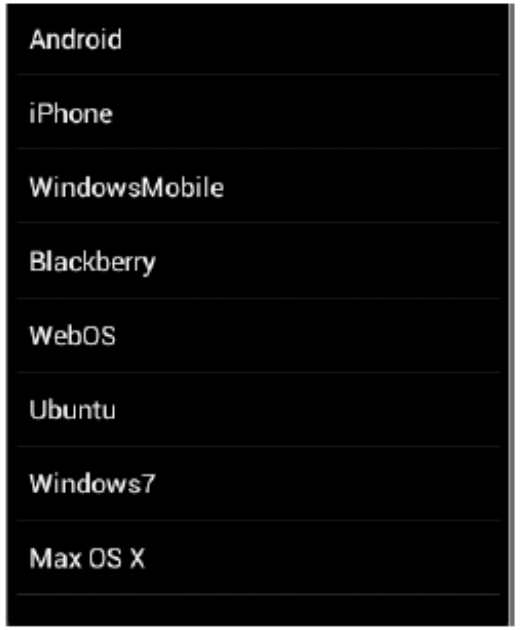
Michelangelo, Mike or Mikey (as he is usually called), is a fictional character and one of the four protagonists of the Teenage Mutant Ninja Turtles comics and all related media. His mask is typically portrayed as orange outside of the Mirage/Image Comics and his weapons are dual nunchucks, though he has also been portrayed using other weapons, such as a grappling hook, manriki-gusari,

```
<LinearLayout ...>
    ...
    <ScrollView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content">
        <TextView ... android:id="@+id/turtle_info" />
    </ScrollView>
</LinearLayout>
```

List (link)

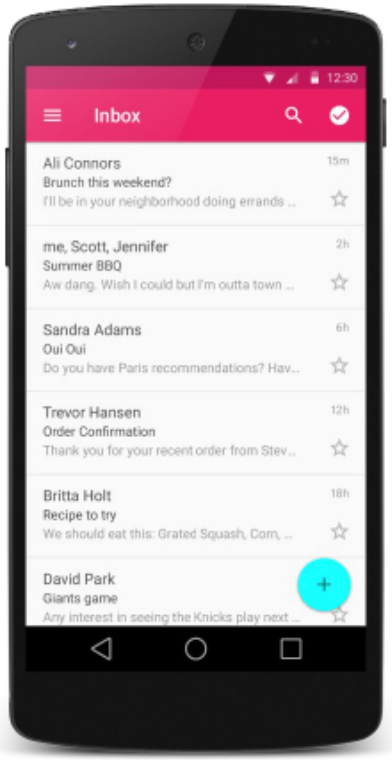
A visible menu of selectable choices

- lists are more complicated, so we'll cover them later ...



ListView (link)

An ordered collection of selectable choices

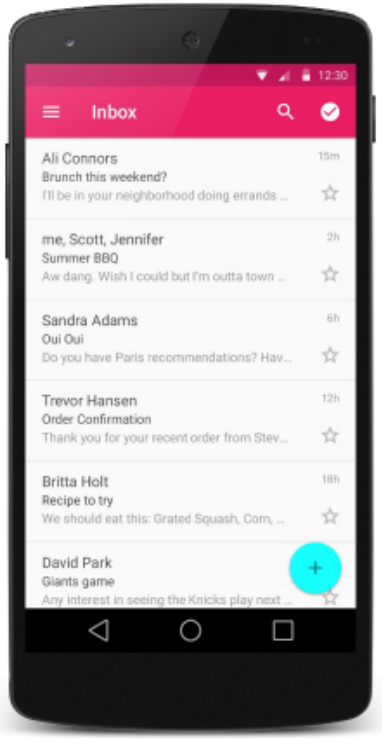


- key attributes in XML:

`android:clickable="`***`bool`***`"`	set to false to disable the list
`android:id="@+id/`***`theID`***`"`	unique ID for use in Java code
`android:entries="@array/`***`array`***`"`	set of options to appear in the list (must match an array in `strings.xml`)

ListView (link)

An ordered collection of selectable choices



- key attributes in XML:

`android:clickable="`***`bool`***`"`	set to false to disable the list
`android:id="@+id/`***`theID`***`"`	unique ID for use in Java code
`android:entries="@array/`***`array`***`"`	set of options to appear in the list (must match an array in `strings.xml`)

Static lists

- **static list**: Content is fixed and known before the app runs.
  - Declare the list elements in the **strings.xml** resource file.

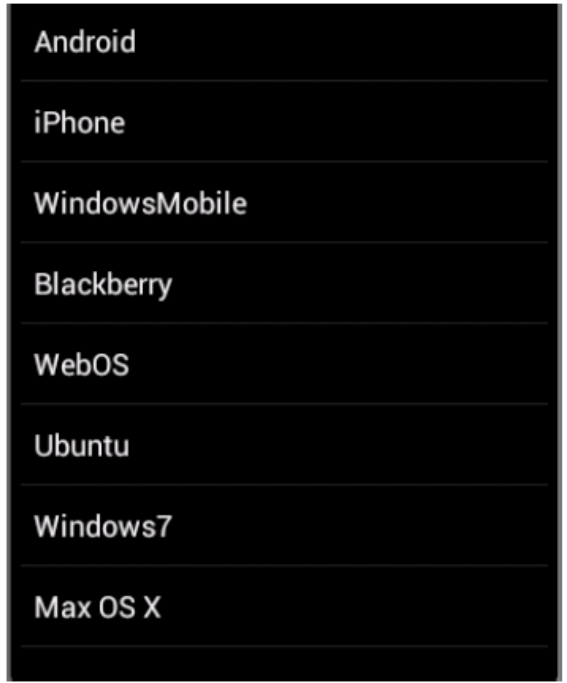
```
<!-- res/values/strings.xml -->
<resources>
    <string-array name="oses">
        <item>Android</item>
        <item>iPhone</item>
        ...
        <item>Max OS X</item>
    </string-array>
</resources>

<!-- res/layout/activity_main.xml -->
<ListView ... android:id="@+id/mylist"
    android:entries="@array/oses" />
```

Dynamic lists

- **dynamic list**: Content is read or generated as the program runs.
  - Comes from a data file, or from the internet, etc.
  - Must be set in the Java code.

  - Suppose we have the following file and want to make a list from it:

```
// res/raw/oses.txt
Android
iPhone
...
Max OS X
```



List adapters

- **adapter**: Helps turn list data into list view items.
  - common adapters: `ArrayAdapter`, `CursorAdapter`

- Syntax for creating an adapter:

```
ArrayAdapter<String> name =
  new ArrayAdapter<String>(activity, layout, array);
```

  - the ***activity*** is usually `this`
  - the default ***layout*** for lists is `android.R.layout.simple_list_item_1`
  - get the ***array*** by reading your file or data source of choice
    (it can be an array like `String[]`, or a list like `ArrayList<String>`)

  - Once you have an adapter, you can attach it to your list by calling the `setAdapter` method of the `ListView` object in the Java code.

List adapter example

```
ArrayList<String> myArray = ...;  // load data from file

ArrayAdapter<String> adapter =
  new ArrayAdapter<String>(
      this,
      android.R.layout.simple_list_item_1,
      myArray);

ListView list = (ListView) findViewById(R.id.mylist);
list.setAdapter(myAdapter);
```

Handling list events

- Unfortunately lists don't use a simple `onClick` event.
  - Several fancier GUI widgets use other kinds of events.
  - The event listeners must be attached in the Java code, not in the XML.
  - Understanding how to attach these event listeners requires the use of Java **anonymous inner classes**.

- **anonymous inner class**: A shorthand syntax for declaring a small class without giving it an explicit name.
  - The class can be made to extend a given super class or implement a given interface.
  - Typically the class is declared and a single object of it is constructed and used all at once.

Attaching event listener in Java

```
<!-- activity_main.xml -->
<Button ... android:onClick="mybuttonOnClick" />
<Button ... android:id="@+id/mybutton" />

// MainActivity.java
public void mybuttonOnClick() { ... }
Button button = (Button) findViewById(R.id.mybutton);
button.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
        // code to run when the button gets clicked
    }
});

// this was the required style for event listeners
// in older versions of Android :-/
```

List events

- List views respond to the following events:
  - **setOnItemClickListener**(AdapterView.OnItemClickListener)

    Listener for when an item in the list has been clicked.

  - **setOnItemLongClickListener**(AdapterView.OnItemLongClickListener)

    Listener for when an item in the list has been clicked and held.

  - **setOnItemSelectedListener**(AdapterView.OnItemSelectedListener)

    Listener for when an item in the list has been selected.

- Others:
  - onDrag, onFocusChanged, onHover, onKey, onScroll, onTouch, ...

List event listener example

```
ListView list = (ListView) findViewById(R.id.id);
list.setOnItemClickListener(
    new AdapterView.OnItemClickListener() {
        @Override
        public void onItemClick(AdapterView<?> list,
                                View row,
                                int index,
                                long rowID) {
            // code to run when user clicks that item
            ...
        }
    }
);
```