







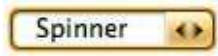


6. GUI: Widgets

Recall: Android widgets

 <p>Analog/DigitalClock</p>	 <p>Button</p>	 <p>Checkbox</p>	 <p>Date/TimePicker</p>
 <p>EditText</p>	 <p>Gallery</p>	 <p>ImageView/Button</p>	 <p>ProgressBar</p>
 <p>RadioButton</p>	 <p>Spinner</p>	 <p>TextView</p>	 <p>MapView, WebView</p>

Button

A clickable widget with a text label



- key attributes:

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

- represented by Button class in Java code

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

...

TextView

Displays non-editable text

- key attributes:

<code>android:id="@+id/<i>theID</i>"</code>	unique ID for use in Java code
<code>android:text="<i>string</i>"</code>	text to display

// to change the visible text in Java code

```
TextView myTextView = (TextView) findViewById(R.id.theID);  
myTextView.setText("text");
```

ScrollView

A container with scrollbars around another widget or container

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

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.

EditText

An editable text input box



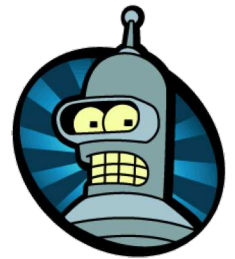
- key attributes:

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

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

ImageView

Displays an image without being clickable



- key attributes:

<code>android:id="@+id/<i>theID</i>"</code>	unique ID for use in Java code
<code>android:src="@drawable/<i>img</i>"</code>	image to put in the view (must correspond to an image resource)
<code>android:tag="<i>string</i>"</code>	a text tag to associate with the image
<code>android:scaleType="<i>type</i>"</code>	causes the image to grow/shrink; can be "center", "centerCrop", "fitCenter", "matri x", ...

// to change the visible image in Java code

```
ImageView myImageView = (ImageView) findViewById(R.id.theID);  
myImageView.setImageResource(R.drawable.filename);
```

Resources

- In the project directory structure:
 - *res/type/name.extension*
 - example: *res/drawable/pikachu.png*
 - (on disk: **AndroidStudioProjects/ProjectName/app/src/main/res/type/name**)
- Referring to a resource, in the XML:
 - *@type/name*
 - example: *@drawable/horses*
- Referring to a resource ID, in the Java code:
 - *R.type.name*
 - example: *R.drawable.horses*



ImageButton

A clickable widget with an image label



- key attributes:

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

CheckBox

An individual toggleable on/off switch



- key attributes:

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

- In Java code:

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

RadioButton

A toggleable on/off switch; part of a group



- key attributes:

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

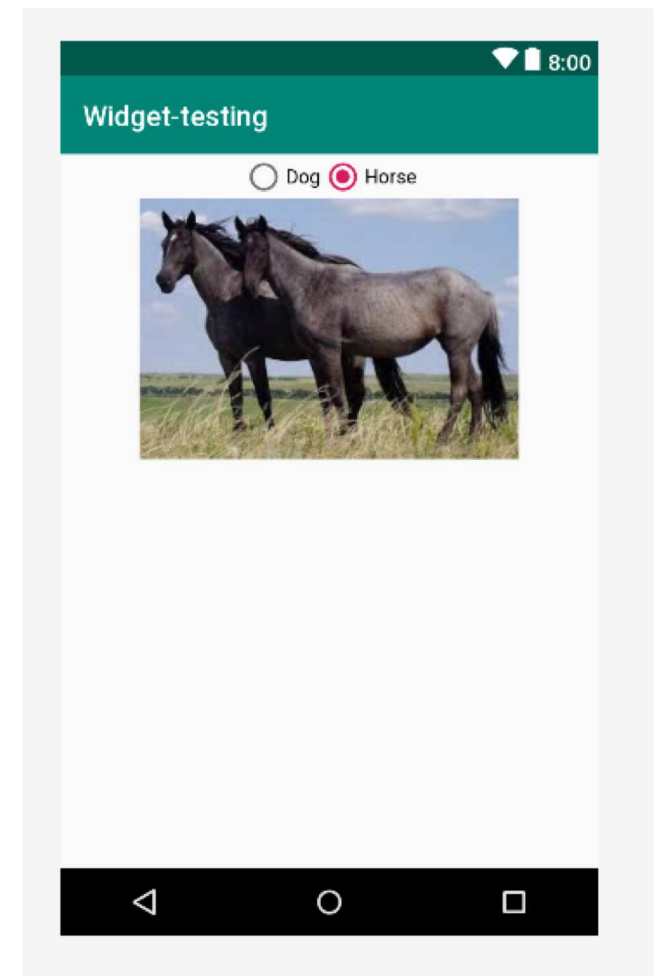
```
// in MainActivity.java
```

```
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 ...  
        }  
    }  
}
```



Exercise

- Using radio buttons to select between images
- Add images to res
- Create a radio group
- Use a single onclick for both buttons
- Switch between images



Spinner

A drop-down menu of selectable choices

- key attributes:

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

Home

Home

Work

Other

Custom

String resources

- Declare constant strings and arrays in res/values/[strings.xml](#):

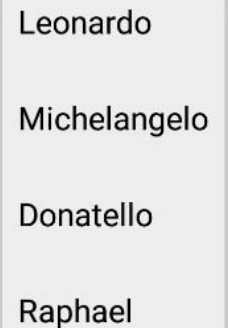
```
<resources>
    <string name="name">value</string>
    <string name="name">value</string>

    <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: `getString(R.string.name)`,
`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>
```



Leonardo
Michelangelo
Donatello
Raphael

- 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>
```

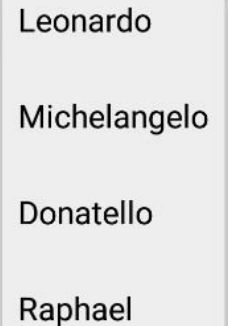
Spinner event example

// 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.setOnItemClickListener(new AdapterView.OnItemClickListener() {
            public void onItemClick(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
        });
    }
}
```



- Leonardo
- Michelangelo
- Donatello
- Raphael

TMNT app exercise

- Write an app to select Disney 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.

