Handling UI Events

Handling UI Events

- An event listener is an interface in the View class that contains a single callback method.
- These methods will be called by the Android framework when the View to which the listener has been registered is triggered by user interaction with the item in the UI.

Event listener callback methods

- onClick() From View.OnClickListener
  - This is called when the user either touches the item (when in touch mode), or focuses upon the item with the navigation-keys or trackball and presses the suitable "enter" key or presses down on the trackball.
- onLongClick() From View.OnLongClickListener
  - This is called when the user either touches and holds the item (when in touch mode), or focuses upon the item with the navigation-keys or trackball and presses and holds the suitable "enter" key or presses and holds down on the trackball (for one second).

Event listener callback methods

onFocusChange() From View.OnFocusChangeListener.

- This is called when the user navigates onto or away from the item, using the navigation-keys or trackball.

onKey() From View.OnKeyListener.

- This is called when the user is focused on the item and presses or releases a key on the device.

onTouch() From View.OnTouchListener.

- This is called when the user performs an action qualified as a touch event, including a press, a release, or any movement gesture on the screen (within the bounds of the item).

onCreateContextMenu() From View.OnCreateContextMenuListener.

- This is called when a Context Menu is being built (as the result of a sustained "long click").

Using Listeners

- To define one of these methods and handle your events, implement the nested interface in your Activity or define it as an anonymous class.
- Then, pass an instance of your implementation to the respective View.set...Listener() method. (E.g., call setOnClickListener() and pass it your implementation of the OnClickListener.)

Example 1

```
// Create an anonymous implementation of OnClickListener
    private OnClickListener mCorkyListener = new OnClickListener() {
        public void onClick(View v) {
          // do something when the button is clicked
        }
    };

    protected void onCreate(Bundle savedValues) {
        ...
        // Capture our button from layout
        Button button = (Button)findViewById(R.id.corky);
        // Register the onClick listener with the implementation above
        button.setOnClickListener(mCorkyListener);
        ...
    }
```

Example 2

```
public class ExampleActivity extends Activity implements
    OnClickListener {
      protected void onCreate(Bundle savedValues) {
        ...
        Button button = (Button)findViewById(R.id.corky);
        button.setOnClickListener(this);
      }

      // Implement the OnClickListener callback
      public void onClick(View v) {
       // do something when the button is clicked
      }
      ...
    }
```

What listener methods returns?

- Notice that the onClick() callback in the above example has no return value, but some other event listener methods must return a boolean. The reason depends on the event.
- Methods returning boolean value are:
  - onLongClick()
  - onKey()
  - onTouch()
- All the methods above returns a boolean to indicate whether your listener consumes this event.
- Return *true* to indicate that you have handled the event and it should stop here; return *false* if you have not handled it and/or the event should continue to any other on-click listeners.

Note: *Android will call event handlers first and then the appropriate default handlers from the class definition second.*

Touch Mode

- For a touch-capable device, once the user touches the screen, the device will enter touch mode.
- From this point onward, only Views for which isFocusableInTouchMode() is true will be focusable, such as text editing widgets.
- Other Views that are touchable, like buttons, will not take focus when touched; they will simply fire their on-click listeners when pressed.

Touch mode

- Any time a user hits a directional key or scrolls with a trackball, the device will exit touch mode, and find a view to take focus. Now, the user may resume interacting with the user interface without touching the screen.
- The touch mode state is maintained throughout the entire system (all windows and activities).
  - To query the current state, you can call isInTouchMode() to see whether the device is currently in touch mode.

Handling Focus

- The framework will handle routine focus movement in response to user input.
- Views indicate their willingness to take focus through
  - isFocusable()
- To change whether a View can take focus, call
  - setFocusable()
  - android:focusable – in XML file
- When in touch mode,
  - you may query whether a View allows focus with isFocusableInTouchMode().
  - You can change this with setFocusableInTouchMode().
  - android:focusableInTouchMode – in XML file

Handling Focus

- Focus movement is based on an algorithm which finds the nearest neighbor in a given direction.
- The default Focus algorithm can be overridden by overriding the *nextFocusDown*, *nextFocusLeft*, *nextFocusRight*, and *nextFocusUp* attributes in XML layout file.

```
<LinearLayout
      android:orientation="vertical"
      ... >
    <Button android:id="@+id/top"
          android:nextFocusUp="@+id/bottom"
          ... />
    <Button android:id="@+id/bottom"
          android:nextFocusDown="@+id/top"
          ... />
  </LinearLayout>
```