7B

Android
Hard & Soft Keyboards

Victor Matos
Cleveland State University

Notes are based on:

The Busy Coder's Guide to Android Development
by Mark L. Murphy

ISBN: 978-0-9816780-0-9
&
Android Developers
http://developer.android.com/index.html

Hard & Soft Keyboard

Android r1.5 introduced the notion of **Input Method Framework (IMF)**.

The idea is to let the IFM arbitrate the interaction between applications and the current input method chosen by the user.

The motivation behind this framework is the realization that as Android matures, more hardware /software devices, and input techniques will appear in user's applications, for instance:

- real & virtual keyboards,
- voice recognition,
- hand writing,
- etc…

Hard & Soft Keyboard

Keyboarding data into Android's applications is functionally dependent of the hardware present in the actual device.

HTC – G1
Sliding Window exposes (occasionally) a hard keyboard

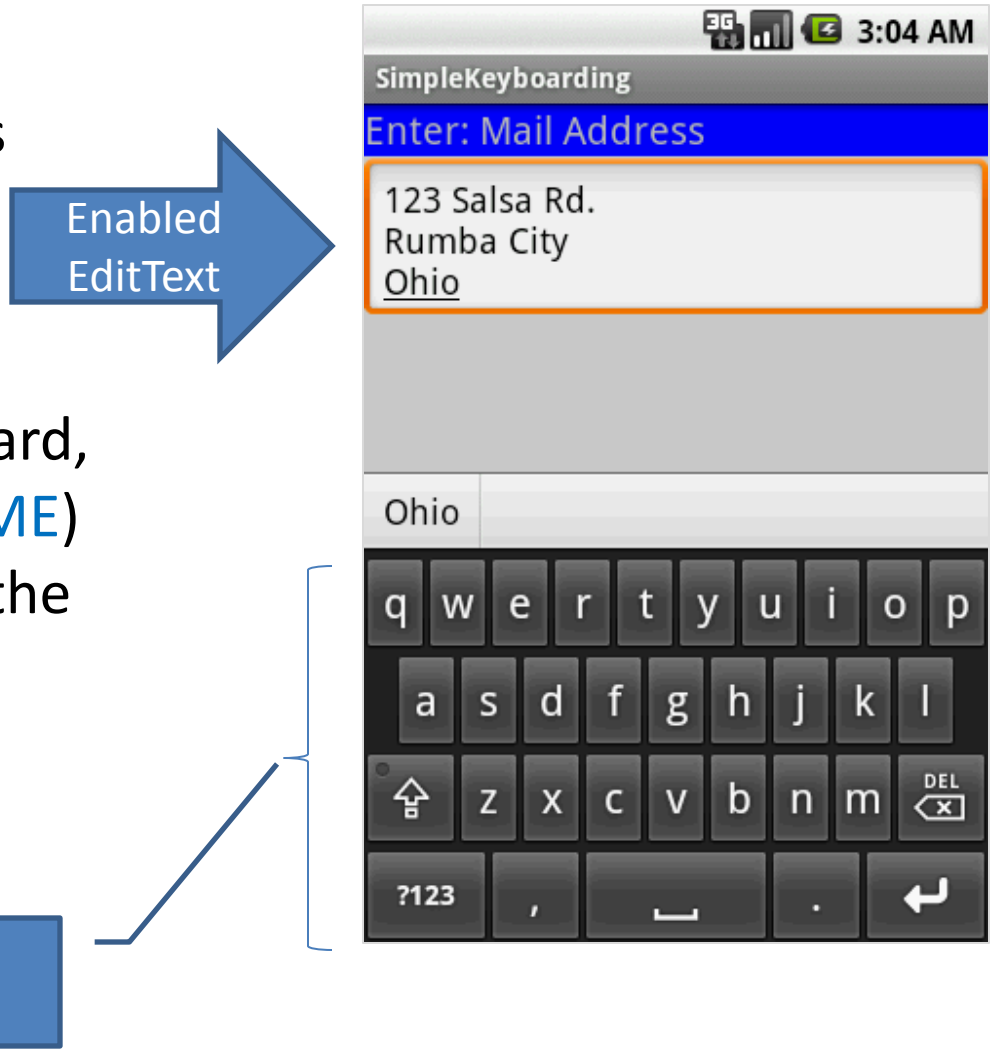
Samsung
Model shows a permanent hard keyboard

HTC - Magic
Model shown has no hard keyboard

Hard & Soft Keyboard

The IMF is aware of the available hardware and its current state.

If there is no a readily available hardware keyboard, an *input method editor* (IME) will be made available to the user when they tap on an enabled *EditText*.



Hard & Soft Keyboard

Telling Android what data to expect

TextViews can indicate by *XML attribute* or *Java method* the expected type of a text field:

XML → `android:inputType="..."`

Java → `editTextBox.setRawInputType(int)`

This way Android knows the type of data to be placed in a text field.

Knowing the type is useful in deciding what appropriated input method could be applied to help the user enter text.

Hard & Soft Keyboard

Android:inputType Values

Constant	Value	Description
`none`	0x00000000	There is no content type. The text is not editable.
`text`	0x00000001	Just plain old text.
`textCapCharacters`	0x00001001	Can be combined with *text* and its variations to request capitalization of all characters.
`textCapWords`	0x00002001	Can be combined with *text* and its variations to request capitalization of the first character of every word.
`textCapSentences`	0x00004001	Can be combined with *text* and its variations to request capitalization of the first character of every sentence.
`textAutoCorrect`	0x00008001	Can be combined with *text* and its variations to request auto-correction of text being input.

Hard & Soft Keyboard

Android:inputType Values

Constant	Value	Description
`textAutoComplete`	0x00010001	Can be combined with *text* and its variations to specify that this field will be doing its own auto-completion and talking with the input method appropriately.
`textMultiLine`	0x00020001	Can be combined with *text* and its variations to allow multiple lines of text in the field. If this flag is not set, the text field will be constrained to a single line.
`textImeMultiLine`	0x00040001	Can be combined with *text* and its variations to indicate that though the regular text view should not be multiple lines, the IME should provide multiple lines if it can.

Hard & Soft Keyboard

Android:inputType Values

Constant	Value	Description
`textUri`	0x00000011	Text that will be used as a URI.
`textEmailAddress`	0x00000021	Text that will be used as an e-mail address.
`textEmailSubject`	0x00000031	Text that is being supplied as the subject of an e-mail.
`textShortMessage`	0x00000041	Text that is the content of a short message.
`textLongMessage`	0x00000051	Text that is the content of a long message.
`textPersonName`	0x00000061	Text that is the name of a person.
`textPostalAddress`	0x00000071	Text that is being supplied as a postal mailing address.
`textPassword`	0x00000081	Text that is a password.
`textVisiblePassword`	0x00000091	Text that is a password that should be visible.
`textWebEditText`	0x000000a1	Text that is being supplied as text in a web form.

Hard & Soft Keyboard

Android:inputType Values

Constant	Value	Description
`textFilter`	0x000000b1	Text that is filtering some other data.
`textPhonetic`	0x000000c1	Text that is for phonetic pronunciation, such as a phonetic name field in a contact entry.
`number`	0x00000002	A numeric only field.
`numberSigned`	0x00001002	Can be combined with *number* and its other options to allow a signed number.
`numberDecimal`	0x00002002	Can be combined with *number* and its other options to allow a decimal (fractional) number.
`phone`	0x00000003	For entering a phone number.
`datetime`	0x00000004	For entering a date and time.
`date`	0x00000014	For entering a date.
`time`	0x00000024	For entering a time.

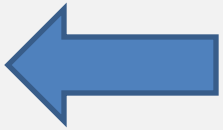
Hard & Soft Keyboard

Example1: Using android:text="*inputType: text|textCapWords*"

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
android:id="@+id/widget31"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:background="#ffcccccc"
android:orientation="vertical"
xmlns:android="http://schemas.android.com/apk/res/android" >

<TextView
android:id="@+id/caption"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:background="#ff0000ff"
android:text="inputType: text|textCapWords"
android:textStyle="bold"
android:textSize="22sp" />

<EditText
android:id="@+id/editTextBox"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:padding="10px"
android:textSize="18sp"
android:inputType="text|textCapWords"   />
</LinearLayout>
```

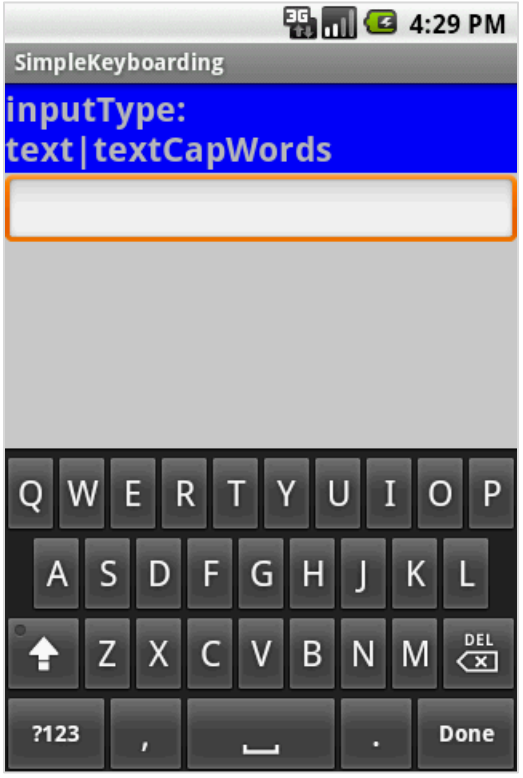


Multiple types of input methods could be combined. Use “pipe” symbol **|** to separate the options.

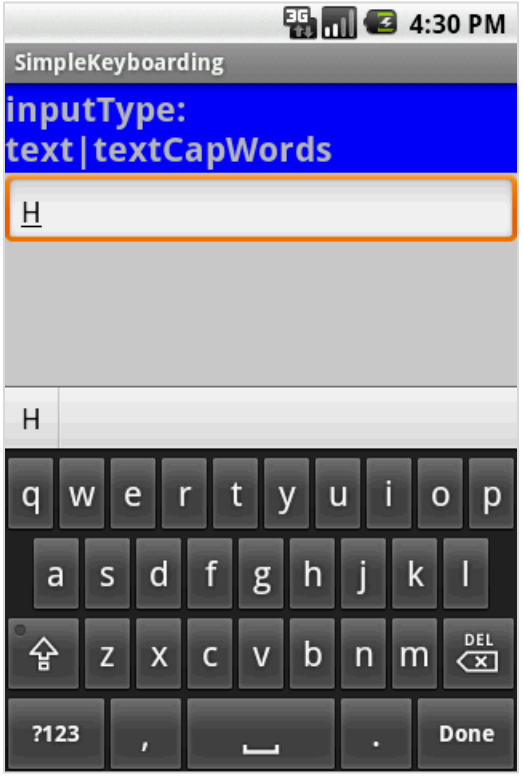
In the example a soft text keyboard is used, in addition it should *proper capitalize* each word

Hard & Soft Keyboard

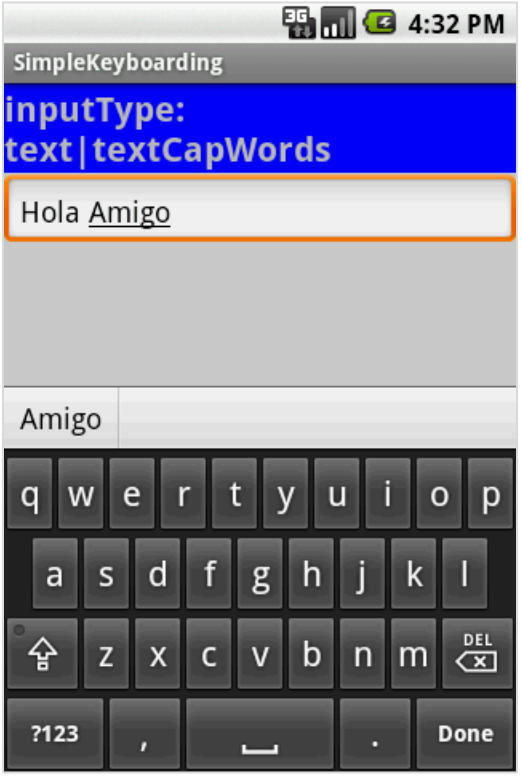
Example1: Using android:text="*inputType: text|textCapWords*"



After tapping the EditBox a soft keyboard appears showing CAPITAL letters



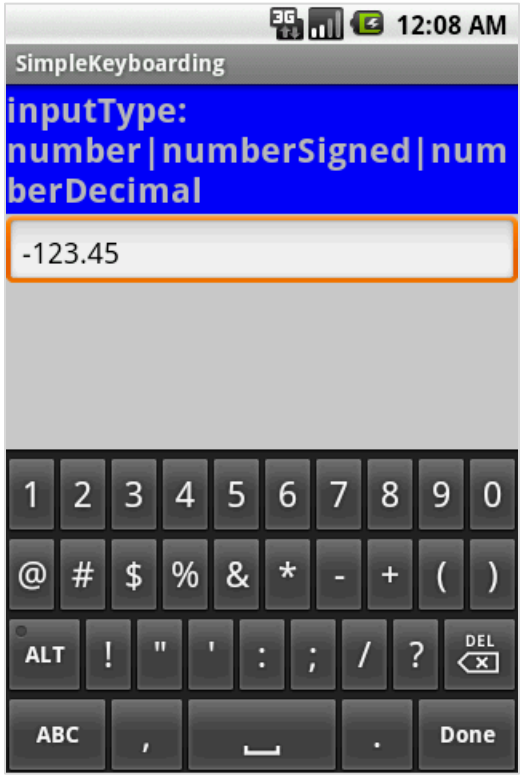
After first letter is typed the Keyboard switches automatically to LOWER case to complete the word.



After entering *space* the keyboard repeats cycle beginning with UPPER case, then LOWER case letters.

Hard & Soft Keyboard

Example2: Using
android:inputType="*number|numberSigned|numberDecimal*"



1. The keyboard displays numbers.
2. In general other *non-numeric* keys are visible but disable.
3. Only valid numeric expressions can be entered.
4. Type **number|numberSigned** accepts integers.
5. Type **numberDecimal** accepts real numbers.

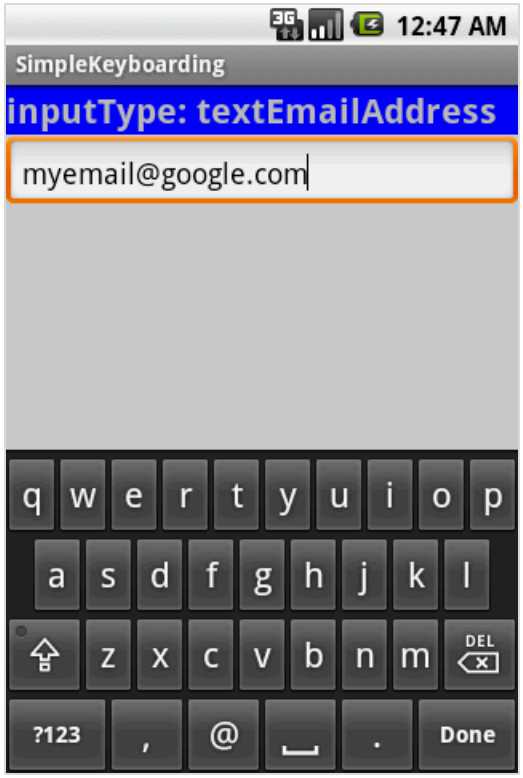
Assume the EditText field is named: **editTextBox**,
In Java code we could at run-time set the input method by issuing the command:

```
editTextBox.setRawInputType(
    android.text.InputType.TYPE_CLASS_PHONE );
```

Hard & Soft Keyboard

Example2: Using android:inputType=*"textPassword"*

Example3: Using android:inputType=*"textEmailAddress"*

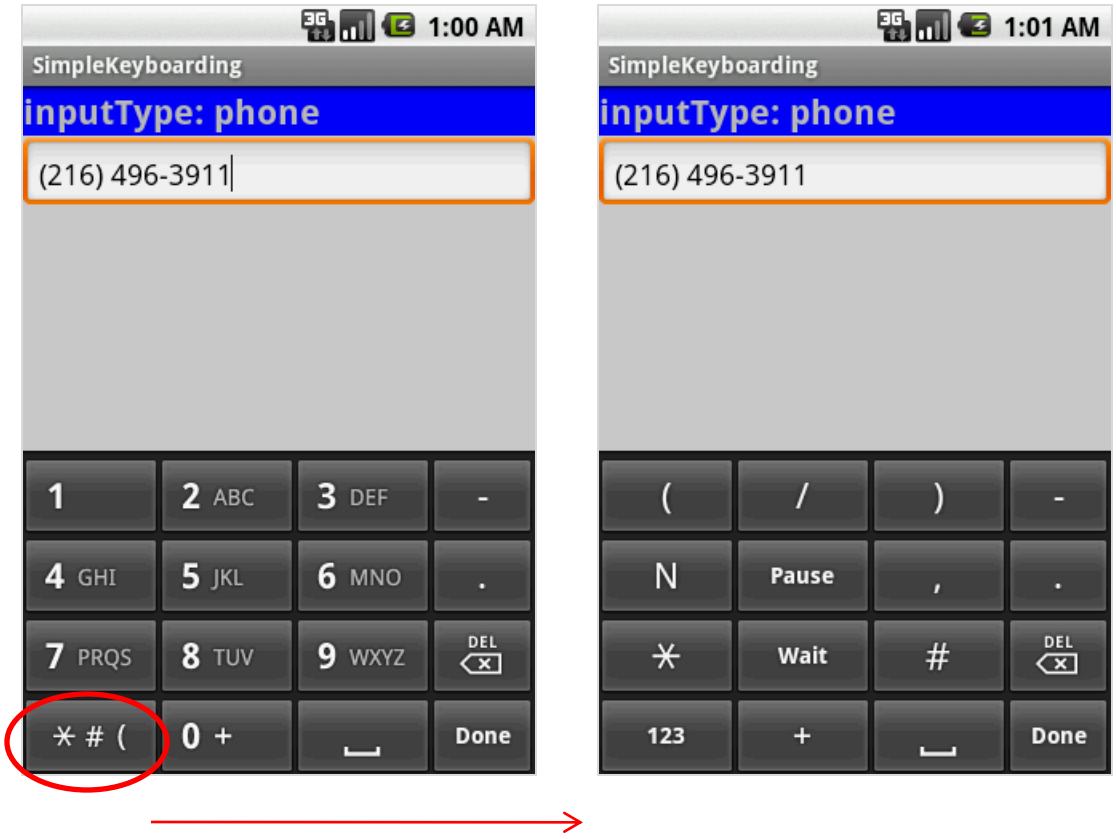


Soft keyboard favors characters commonly used in email addresses such as letters, **@**

- The keyboard displays all possible keys.
- Current character is briefly displayed for verification purposes.
- The current character is hidden and a *heavy-dot* is displayed.

Hard & Soft Keyboard

Example4: Using android:inputType= *"phone"*

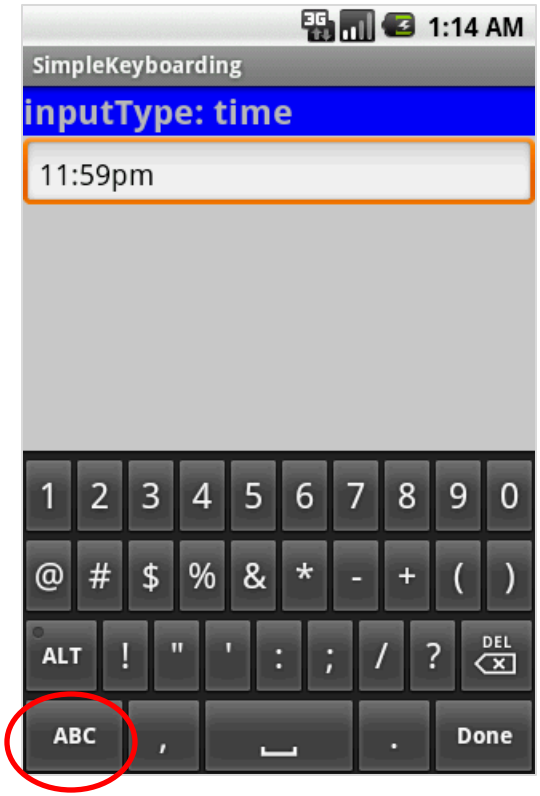


Soft keyboard displays the layout of a typical *phone keypad* plus additional non digit symbols such as:

( ) . / Pause Wait # - +

Hard & Soft Keyboard

Example5: Using android:inputType="*time*"



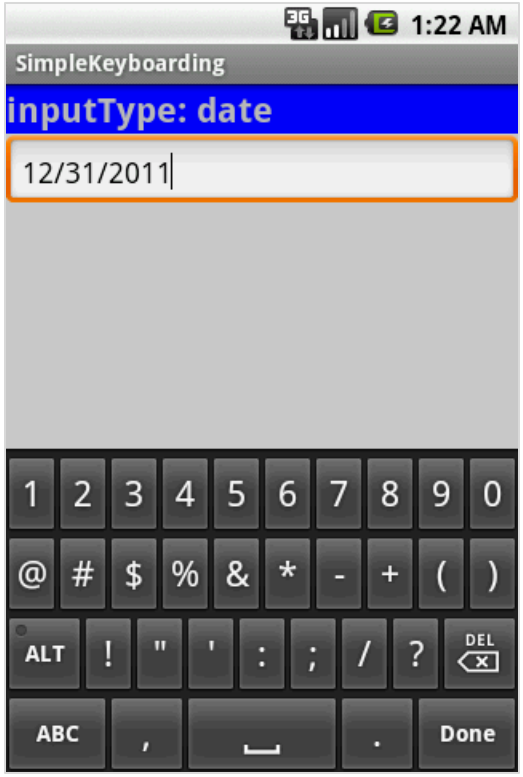
Soft keyboard displays a numerical layout.

Only digits and colon-char **:** can be used.

When clicking on alphabetic choice ***ABC*** only character to make **am** and **pm** are allowed.

Hard & Soft Keyboard

Example6: Using android:inputType="*date*"



Soft keyboard displays a numerical layout.

Only digits and date valid characters are allowed.

Examples of valid dates are:
12/31/2011
12-31-2011
12.31.2011

Hard & Soft Keyboard

Disable Soft Keyboarding on an EditText View

Assume *txtBox1* is an EditText box. To **disable** the action of the soft keyboard on an EditText you should set its input type to null, as indicated below:

```
txtBox.setInputType( InputType.TYPE_NULL );
```

You may also try (deaf touch listener)

```
txtBox.setOnTouchListener(new OnTouchListener() {
        @Override
        public boolean onTouch(View arg0, MotionEvent arg1) {
        // return true to consume the touch event without
        // allowing virtual keyboard to be called
        return true;
        }
});
```

Hard & Soft Keyboard

Close SoftKeyboard Window / Hide SoftKeyboard

Once it has opened, you may close the virtual keyboard by tapping the hardware ***BackArrow*** key or issuing the following commands:

```
InputMethodManager imm =
        (InputMethodManager) getSystemService(Context.INPUT_METHOD_SERVICE);

imm.hideSoftInputFromWindow (theEditTextField.getWindowToken(), 0);
```

Hard & Soft Keyboard

Do not Allow Soft-Keyboard to be Shown

When you click on an EditText the soft-keyboard is normally displayed. To avoid this from happening do the following:

```
//do not allow soft-keyboard to show
    txtMsg.setOnTouchListener(new OnTouchListener() {
    @Override
    public boolean onTouch(View v, MotionEvent event) {
        //do something-nothing here
        return true; // true: we have consumed the event
                     // false: it will be passed along
    }
});
```

Hard & Soft Keyboard

TextWatcher Control

Assume *txtBox1* is an **Editable** box. A listener of the type **onKeyListener** could be used to follow the actions made by the hardware keyboard; however *it will not properly work with the **Virtual Keyboard***.

A solution to this problem is to attach to the Editable control a **TextWatcher** and let its methods be called when the Editable text is changed.

The main methods of a **TextWatcher** are:

```
public void afterTextChanged (Editable theWatchedText)
public void beforeTextChanged ( … )
public void onTextChanged ( … )
```

Hard & Soft Keyboard

Example 7: TextWatcher Demo

EditText uses .addTextChangedListener

IMF suggestions

Hard & Soft Keyboard

Example 7: TextWatcher Demo

```
<?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"
   android:background="#ffaabbcc"
   >
<EditText
android:id="@+id/txtInput"
   android:layout_width="fill_parent"
   android:layout_height="wrap_content"
   android:layout_margin="10px"
   android:padding="4px"
   android:textStyle="bold"
   />

<TextView
android:id="@+id/txtMsg"
   android:layout_width="fill_parent"
   android:layout_height="wrap_content"
   android:layout_margin="10px"
   android:padding="4px"
   android:background="#ff0000ff"
   android:textStyle="bold"
   />
</LinearLayout>
```

Hard & Soft Keyboard

Example 7: TextWatcher Demo

```
// demonstrate the use of a simple TEXTWATCHER control
package cis493.keyboarding;
…
public class TextWatcherDemo extends Activity  {
   EditText txtInput;
   TextView txtMsg;
   int keyCount = 0;
   @Override
   public void onCreate (Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        txtMsg = (TextView)findViewById(R.id.txtMsg);
        txtInput = (EditText)findViewById(R.id.txtInput);

        txtInput.addTextChangedListener(new TextWatcher() {
            public void afterTextChanged (Editable theWatchedText) {
                String msg = "count: " + txtInput.getText().toString().length() + " " + theWatchedText.toString();
                txtMsg.setText( msg );
            }
            public void beforeTextChanged (CharSequence arg0, int arg1, int arg2, int arg3) {
                //Toast.makeText(getApplicationContext(), "BTC " + arg0, 1).show();
            }
            public void onTextChanged (CharSequence arg0, int arg1, int arg2, int arg3) {
                //Toast.makeText(getApplicationContext(), "OTC " + arg0, 1).show();
            }
        }); //addTextChangedListener

    } //onCreate
}
```

Hard & Soft Keyboard

Questions?