Lesson 5 – User Interfaces Part 3

In this part, we're going to take a look at the EditText Views, and how they function.

Open the existing Lesson5_UI project, and create a new Layout file (see Part 2, if you're not sure how) called **edittext_example.xml**. It should use a vertically oriented LinearLayout, and have a layout_margin of 20dp. The margin will help keep other objects placed inside the LinearLayout from touching the edge of the screen. Set it's Layout_Gravity to center_horizontal as well.

Add the following EditText Views to the LinearLayout, one after the other:

1 – Numbers only prompt:

- ID: @+id/txtNumOnly
- Hint: Numbers Only
- Input Type: number
- Layout Margin Bottom: 5dp
- Text: - Leave the text property blank

2 – Multi-line prompt:

- ID: @+id/txtMulti
- Hint: Multi-line
- Input Type: textMultiLine
- Layout Margin Bottom: 5dp
- Gravity: top|left
- Lines: 3
- Text: - Leave the text property blank

3 – Phone Number prompt:

- ID: @+id/txtPhone
- Hint: Phone Number
- Input Type: Phone
- Layout Margin Bottom: 5dp
- Text: - Leave the text property blank

Now, add a Button below the phone prompt. The button should have the following properties:

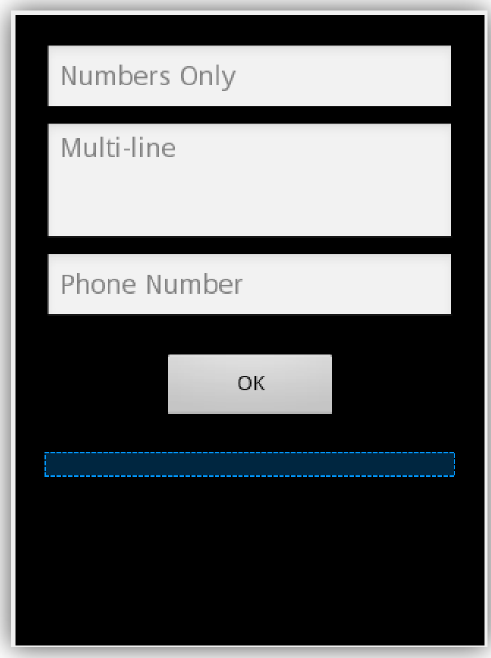
- ID: @+id/btnTextOK
- Text: OK
- Layout Gravity: center_horizontal
- Layout Margin Top: 15dp

- Layout Width: 120dp

Add a TextView control to the screen below the button with the following properties:

- ID: @+id/txtTextOut
- Layout width: fill_parent
- Text: - Leave the text property blank

Your screen should look like this:



Save and close the xml file.

Open the EditText_Example.java file and add the following code:

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

                final Button btn1 = (Button) this.findViewById(R.id.btnTextOK);

                btn1.setOnClickListener(new View.OnClickListener() {

                        @Override
                        public void onClick(View v) {

                                final EditText txtNum = (EditText) findViewById(R.id.txtNumOnly);
                                final EditText txtMulti = (EditText) findViewById(R.id.txtMulti);
                                final EditText txtPhone = (EditText) findViewById(R.id.txtPhone);
                                final TextView txtOut = (TextView) findViewById(R.id.txtTextOut);

                                String sText = "";

                                sText = "txtNum = " + txtNum.getText() + "\n";
                                sText += "txtMulti = " + txtMulti.getText() + "\n";
                                sText += "txtPhone = " + txtPhone.getText() + "\n";

                                txtOut.setText(sText);

                        }
                });

        }
```

We’ve added the following functionality:

When the OK button is clicked we collect the data from each of our prompts and display it back on the screen through the txtOut textview. Note the keyboard layout when editing the numbers only prompt vs the multi-line text prompt vs the phone number prompt. Be sure to review the other input types that are available.

In the XML layout file, we used the HINT property to specify to the user what they need to enter into the prompt. The hint disappeared when the user started entering text. This is useful when there is a lot of information on the screen.