ITW202: Mobile Application

Unit IV: Developing for Android

Ms. Sonam Wangmo

Gyalpozhing College of Information Technology
Royal University of Bhutan

May 3, 2021

Input Controls

Input Controls

Definition

Input controls are interactive elements in your app's UI that accept data input.

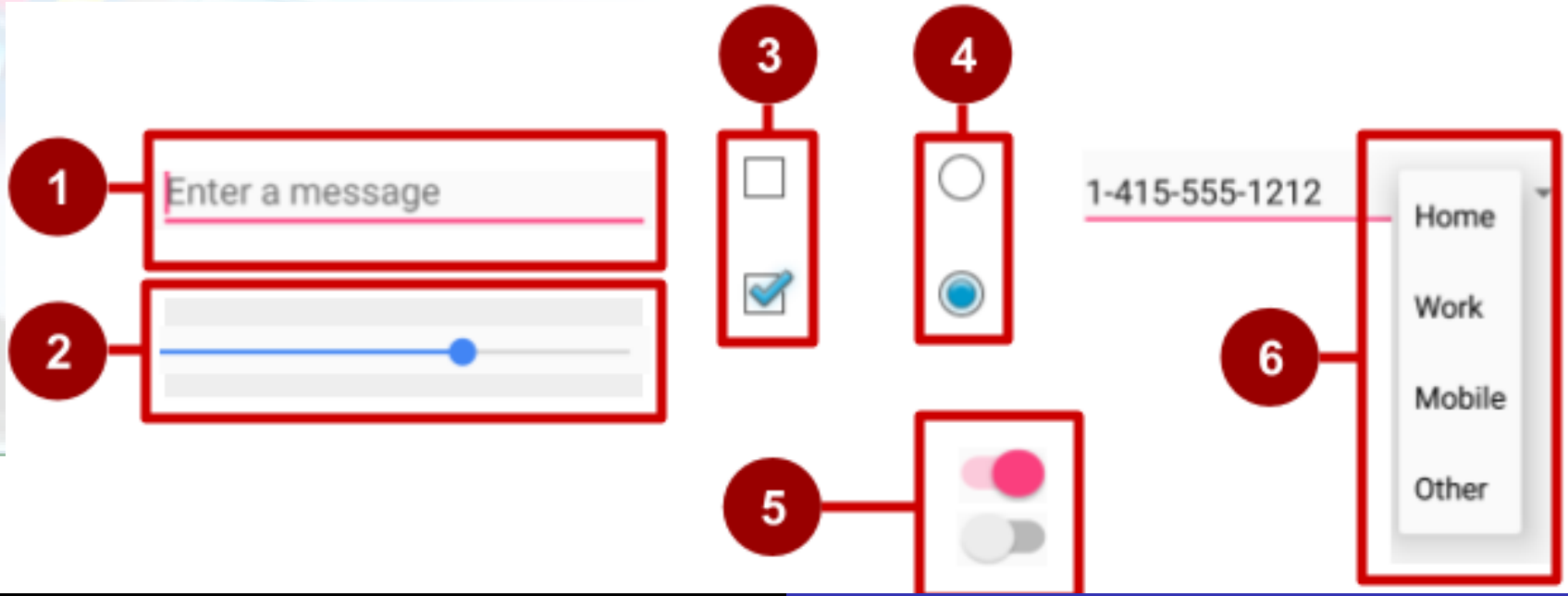
Example: Users input data to apps by entering text or numbers into fields using the on-screen keyboard. Users also select options from checkboxes, radio buttons, and drop-down menus, and they change settings and turn on or turn off certain features.

Input Controls

- Freeform text and numbers: EditText (using keyboard)
- Providing choices: CheckBox, RadioButton, Spinner
- Switching on/off: Toggle, Switch
- Choosing value in range of values: SeekBar

Examples of input controls

1. EditText
2. SeekBar
3. CheckBox
4. RadioButton
5. Switch
6. Spinner



How input controls work

1. Use EditText for entering text using keyboard
2. Use SeekBar for sliding left or right to a setting
3. Combine CheckBox elements for choosing more than one option
4. Combine RadioButton elements into RadioGroup — user makes only one choice
5. Use Switch for tapping on or off
6. Use Spinner for choosing a single item from a list

Input controls for making choices

Android offers ready-made input controls for the user to select one or more choices:

- Checkbox: Select one or more choices from a set of choices by tapping or clicking checkboxes.
- RadioGroup of radio buttons: Select one choice from a set of choices by clicking one circular "radio" button. Radio buttons are useful if you are providing only two or three choices.

Input controls for making choices

- ToggleButton and Switch: Turn an option on or off.
- Spinner: Select one choice from a set of choices in a drop-down menu. A Spinner is useful for three or more choices, and takes up little room in your layout.

View is base class for input controls

- The View class is the basic building block for all UI components, including input controls
- View is the base class for classes that provide interactive UI components
- View provides basic interaction through android:onClick

View focus

Focus

- Focus indicates which View is selected.
- The View that receives user input has "Focus"
- Only one View can have focus
- Focus makes it unambiguous which View gets the input
- Focus is assigned by
  - User tapping a View
  - App guiding the user from one text input control to the next using the Return, Tab, or arrow keys
  - programmatically : Calling requestFocus() on any View that is focusable

Clickable versus focusable

- A focusable view is allowed to gain focus from a touchscreen, external keyboard, or other input device.
- A clickable view is any view that reacts to being tapped or clicked.

Which View gets focus next?

Focus movement is based on a natural algorithm that finds the nearest neighbor in a given direction:

- When the user taps the screen, the topmost View under the tap is in focus, providing touch access for the child View elements of the topmost View.

Which View gets focus next?

- If you set an EditText view to a single line (such as the textPersonName value for the android:inputType attribute), the user can tap the right-arrow key on the on-screen keyboard to close the keyboard and shift focus to the next input control View based on what the Android system finds.

Which View gets focus next?

The system usually finds the nearest input control in the same direction the user was navigating (up, down, left, or right).
If there are multiple input controls that are nearby and in the same direction, the system scans from left to right, top to bottom.

Which View gets focus next?

- Focus can also shift to a different View if the user interacts with a directional control, such as a D-pad or trackball.

Guiding focus

If the algorithm does not give you what you want, you can override it by adding the nextFocusDown, nextFocusLeft, nextFocusRight, and nextFocusUp XML attributes to your layout file:

Guiding focus

```
<LinearLayout
    <!-- Other attributes... -->
    android:orientation="vertical" >

    <Button android:id="@+id/top"
          <!-- Other Button attributes... -->
          android:nextFocusUp="@+id/bottom" />

    <Button android:id="@+id/bottom"
          <!-- Other Button attributes... -->
          android:nextFocusDown="@+id/top"/>

</LinearLayout>
```

Set focus explicitly

Use methods of the View class to set focus

- setFocusable() sets whether a view can have focus
- requestFocus() gives focus to a specific view
- setOnFocusChangeListener() sets listener for when view gains or loses focus
- onFocusChanged() called when focus on a view changes

Find the view with focus

- Activity.getCurrentFocus()
- ViewGroup.getFocusedChild()

Freeform text and numbers

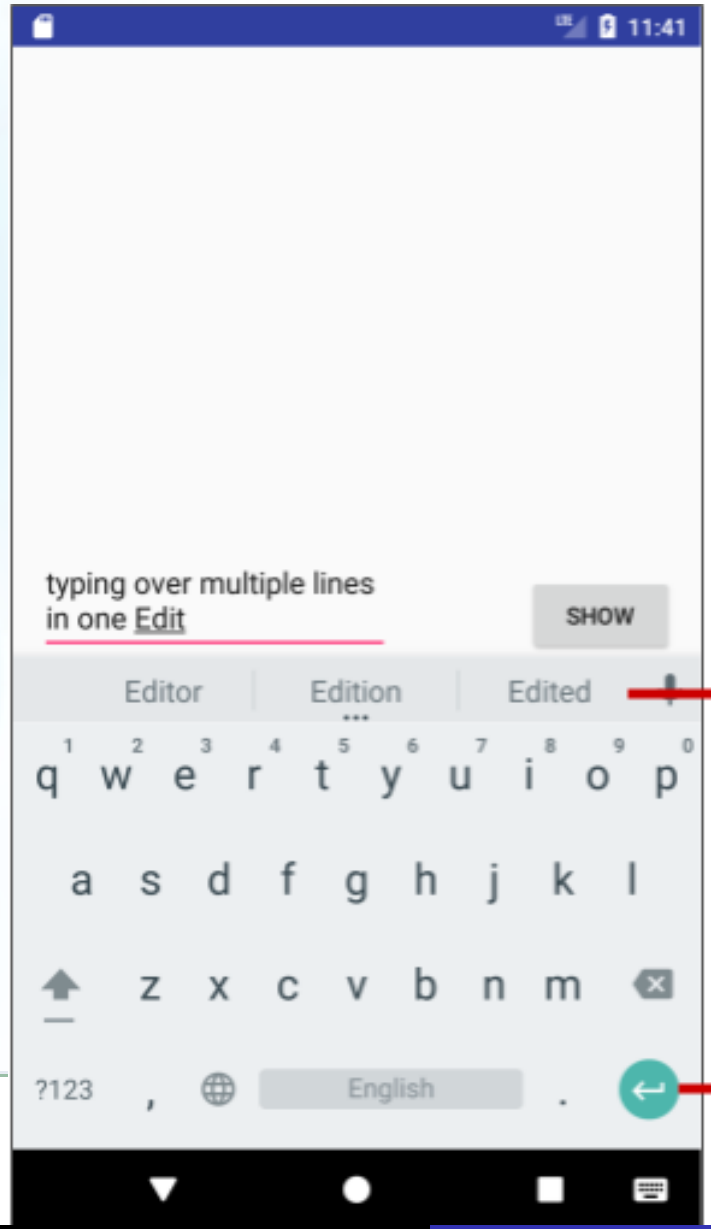
Text input

EditText class gets user input that consists of textual characters, including numbers and symbols. EditText extends the TextView class, to make the TextView editable.

Multiple lines of text

- By default, the EditText view allows multiple lines of input as shown in the figure below, and suggests full words the user can tap.
- Tapping the Return (also known as Enter) key on the keyboard starts a new line in the same EditText.
- Alphanumeric keyboards

Multiple lines of text



Multiple lines of text

In the figure above:

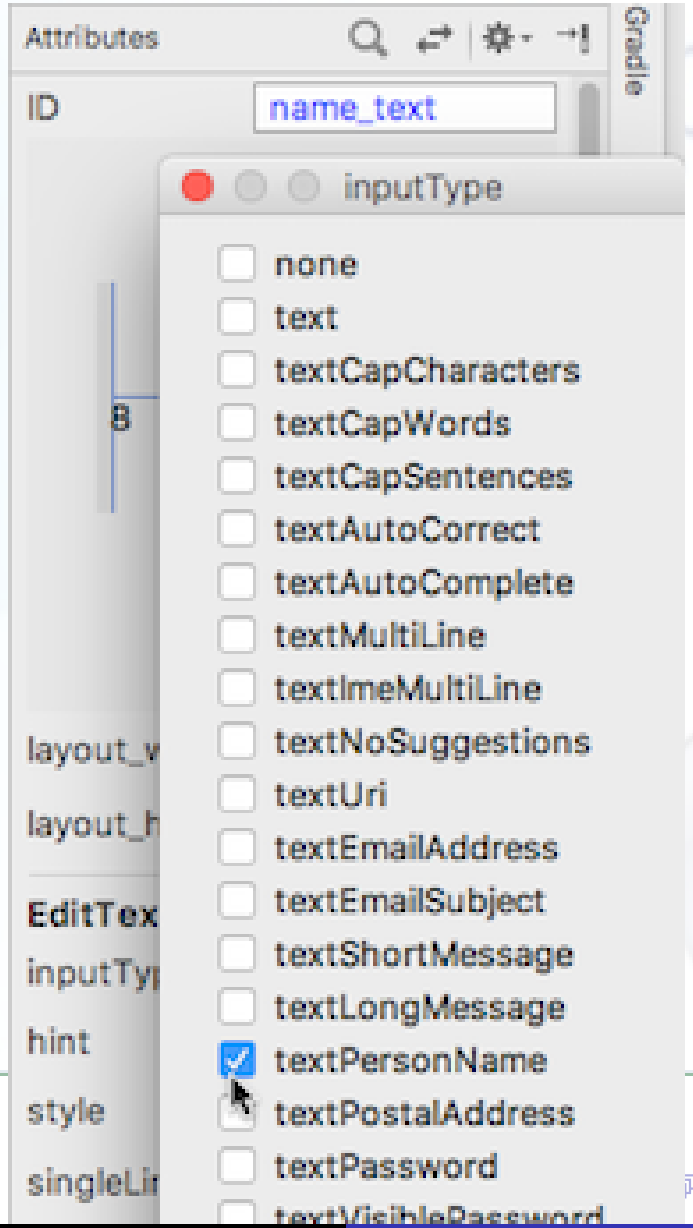
1. Suggestions to tap
2. Return (Enter) key

Customize with inputType

1. Set in Attributes pane of layout editor
2. XML code for EditText:

```
<EditText
    android:id="@+id/name_field"
    android:inputType =
                "textPersonName"
    ...
```

Customize with inputType



EditText for message

- android:inputType ="textShortMessage"
- Single line of text

Getting text

- Get the EditText object for the EditText view

  ```
  EditText simpleEditText =
  findViewById(R.id.edit_simple);
  ```

- Retrieve the CharSequence and convert it to a string

  ```
  String strValue =
  simpleEditText.getText().toString();
  ```

Common input types

- textCapCharacters: Set to all capital letters
- textCapSentences: Start each sentence with a capital letter
- textPassword: Conceal an entered password
- number: Restrict text entry to numbers
- textEmailAddress: Show keyboard with @ conveniently located
- phone: Show a numeric phone keypad
- datetime: Show a numeric keypad with a slash and colon for entering the date and time

Providing choices

UI elements for providing choices

- CheckBox and RadioButton
- ToggleButton and Switch
- Spinner

CheckBox

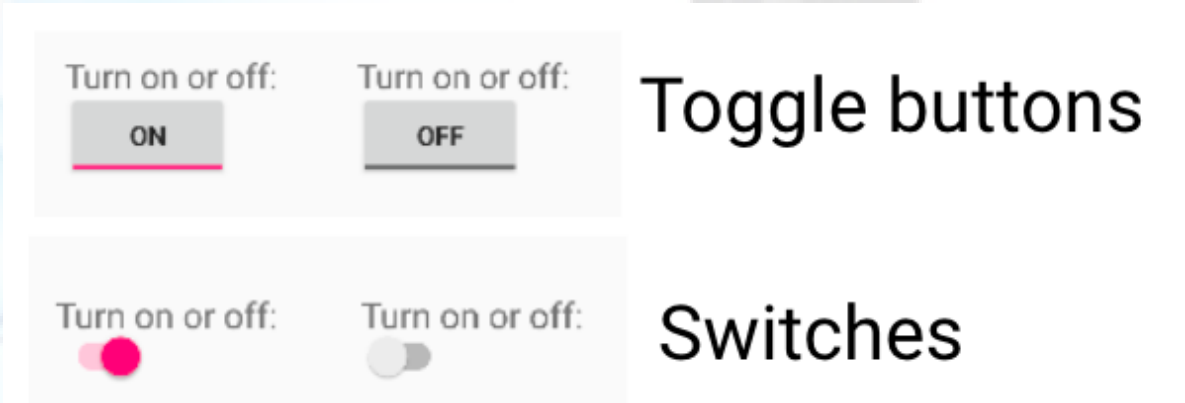
- User can select any number of choices
- Checking one box does not uncheck another
- Users expect checkboxes in a vertical list
- Commonly used with a Submit button
- Every CheckBox is a View and can have an onClick handler

RadioButton

- Put RadioButton elements in a RadioGroup in a vertical list (horizontally if labels are short)
- User can select only one of the choices
- Checking one unchecks all others in group
- Each RadioButton can have onClick handler
- Commonly used with a Submit button for the RadioGroup

Toggle buttons and switches

- User can switch between on and off
- Use android:onClick for click handler




Mobile Application
Development
THANK YOU