College of Science and Technology

Royal University of Bhutan

Unit II:
User-Input Control

CTE308- AS2025

Tutor: Pema Galey
#17682761

Outline

- User Interaction
- Focus
- Text input and keyboards
- Radio Buttons and Checkboxes
- Making Choices - dialogs, spinners and pickers
- Recognizing gestures

User Interaction with App

★ **User Interaction with App**

- Users are expected to interact with apps by:
  - Clicking, pressing, talking, typing, and listening.
  - Using user input controls such as:
    - buttons, menus, keyboards, text boxes, and a microphone.
  - Navigating between activities.

User Interaction Design

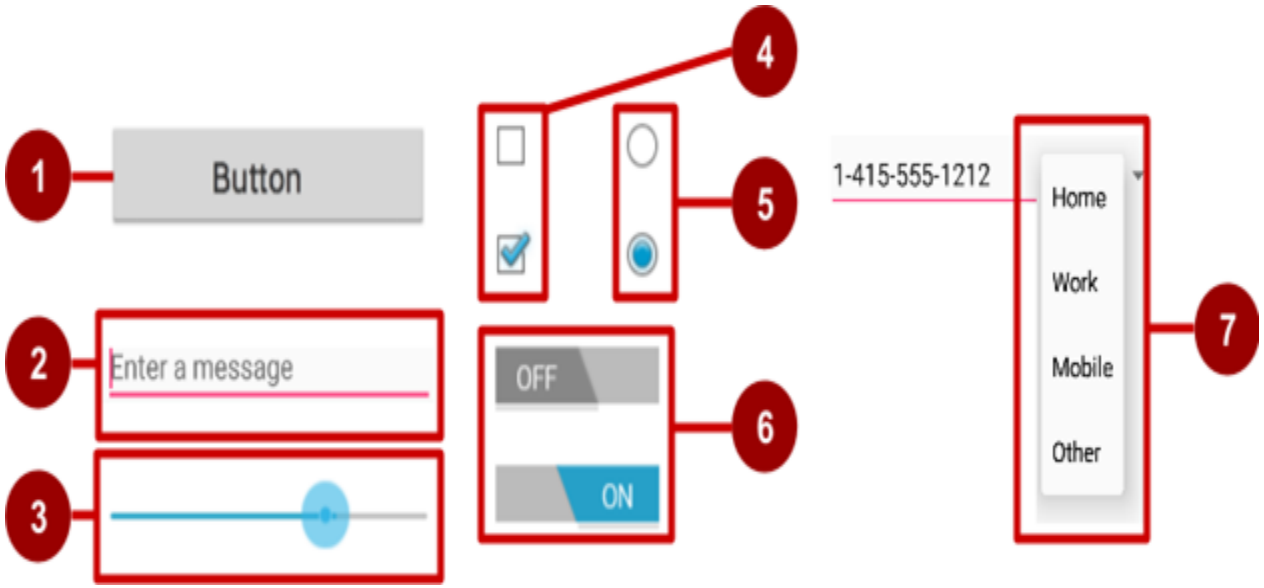
- User interaction design is very important aspect of app development
- Design must be obvious, easy, and consistent:
  - Think about how users will use your app.
  - Minimize steps.
  - Use UI elements that are easy to access, understand and use.
  - Follow Android best practices.
  - Meet user's expectations.

User Input Control

- There are different ways to get input from the user:
  1. **Free form**
     - ✓ Text and voice input
  2. **Actions**
     - ✓ Buttons
     - ✓ Contextual menus
     - ✓ Gestures
     - ✓ Dialogs
  3. **Constrained choices**
     - ✓ Pickers
     - ✓ Checkboxes
     - ✓ Radio buttons
     - ✓ Toggle buttons
     - ✓ Spinners

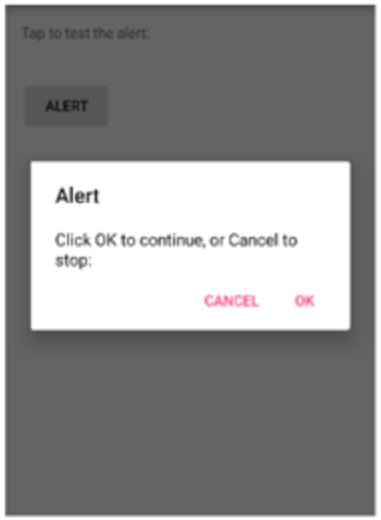
User Interaction with App

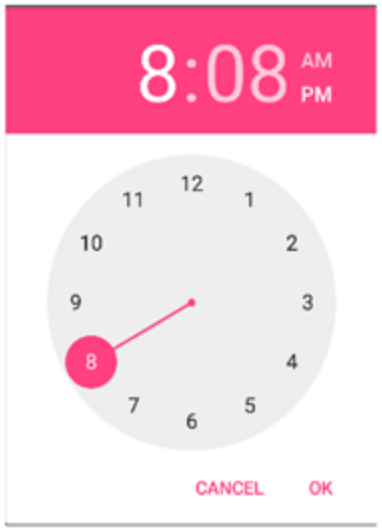
- Examples of user input controls:
  1. Button
  2. Text field
  3. Seek bar
  4. Checkboxes
  5. Radio buttons
  6. Toggle
  7. Spinner



User Interaction with App

- Examples of user input controls:
  8. Alert dialog
  9. date picker
  10. time picker





User Interaction with App

❖User Interaction with App

- ➢ 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***

Focus

- 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:
  1. User tapping a view.
  2. App guiding the user from one text input control to the next using the Return, Tab, or arrow keys.
  3. Calling ***requestFocus()*** on any view that is focusable.

Focus Vs Clickable

❖Focus Vs Clickable

- **Clickable**—View can respond to being clicked or tapped.
- **Focusable**—View can gain focus to accept input.
- Input controls such as keyboards send input to the view that has focus

Focus

❖Focus

- ➢ Which View gets focus next?
  - ✓ The first focus is given to the Topmost view under the touch.
  - ✓ After user submits input, focus moves to nearest neighbour. priority is left to right, top to bottom.
- ➢ Focus can change when user interacts with a directional control.

Focus guides users

❖Focus guides users

- Focus should visually indicate which view has focus so users knows where their input goes.
- It should visually indicate which views can have focus helps users navigate through flow.
- Predictable and logical—no surprises!

Guiding Focus

❖Guiding Focus

- Arrange input controls in a layout from left to right and top to bottom in the order you want focus assigned.
- Place input controls inside a view group in your layout
- Specify ordering in XML
  - ***android:id="@+id/top"***
  - ***android:focusable="true"***
  - ***android:nextFocusDown="@+id/bottom"***

Setting Focus

❖Setting Focus

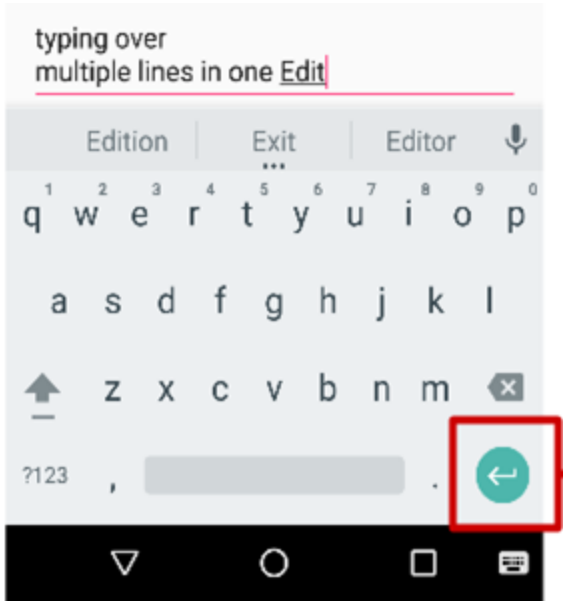
- Setting focus explicitly using methods of the View class.
  - ***setFocusable()*** sets whether a view can have focus.
  - ***requestFocus()*** gives focus to a specific view.
  - ***setOnFocusChangeListener()*** sets listener for when view gains. or loses focus
  - ***onFocusChanged()*** called when focus on a view changes.

- Finding the View with Focus
  - Activity.getCurrentFocus()
  - ViewGroup.getFocusedChild()

EditText

❖EditText

- Using ***EditText*** class, multiple lines of input can be provided.
- Input such as characters, numbers, and symbols.
- It provides Spelling correction.
- Tapping the Return (Enter) key starts a new line.
- Customizable.



"Action" key

Getting Text

❖Getting Text

- Get the ***EditText*** object for the EditText view
  - ***EditText simpleEditText =***
    (EditText) findViewById(R.id.edit_simple);

- Retrieve the CharSequence and convert it to a string
  - ***String strValue =***
    simpleEditText.getText().toString();

Getting Text

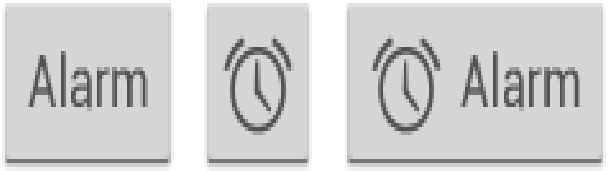
❖Getting Text

- ➢ Common Input Types:
  - ✓ textShortMessage—Limit input to 1 line
  - ✓ textCapSentences—Set keyboard to caps at beginning of sentences
  - ✓ textAutoCorrect—Enable autocorrecting
  - ✓ textPassword—Conceal typed characters
  - ✓ textEmailAddress—Show an @ sign on the keyboard
  - ✓ phone—numeric keyboard for phone numbers.
    - ***android:inputType="phone"***
    - ***android:inputType="textAutoCorrect|textCapSentences"***

Button

❖Button

- ➢ View that responds to clicking or pressing.
- ➢ Usually text or visuals indicate what will happen when it is pressed.
- ➢ **Views:** Button > ToggleButton, ImageView > FloatingActionButton (FAB)
- ➢ **State**: normal, focused, disabled, pressed, on/off
- ➢ **Visuals:** raised, flat, clipart, images, text



Responding to button tap

- *In your code*: Use OnClickListener event listener.
- *In XML*: use android:onClick attribute in the XML layout:

```
<Button
    android:id="@+id/button_send"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/button_send"
    android:onClick="sendMessage" />
```

Button

Button

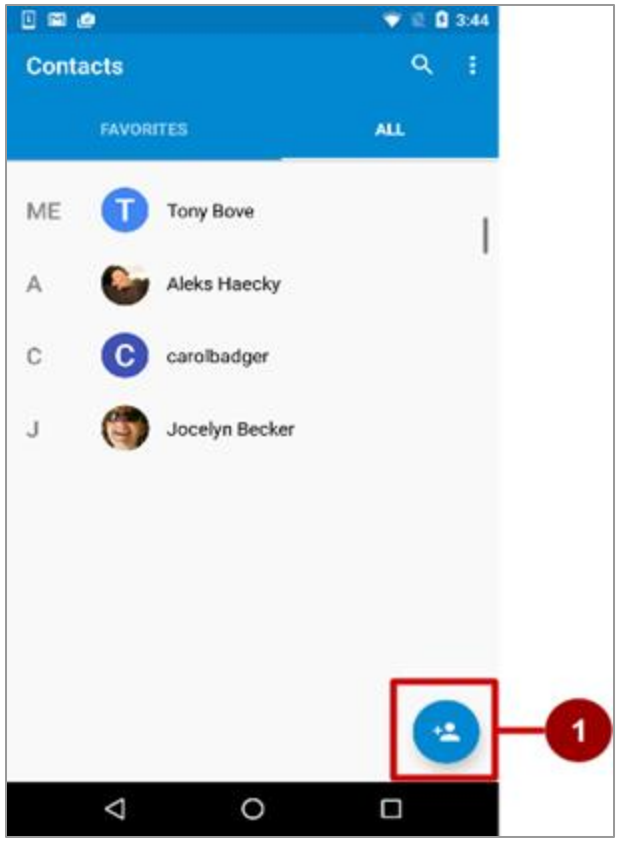
- Setting listener with onClick callBack.

```
Button button = (Button) findViewById(R.id.button);
button.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
        // Do something in response to button click
    }
});
```

Floating Action Button

❖Floating Action Button

- ➢ FAB is raised and circular, floats above layout.
- ➢ Primary or "promoted" action for a screen.
- ➢ You can not have more then one FAB per screen
- ➢ One per screen.
- ➢ For example:
  - ✓ Add Contact button in Contacts app



Using FAB

- Add design support library to build.gradle

```
compile 'com.android.support:design:a.b.c'
```

- Layout

```
<android.support.design.widget.FloatingActionButton
    android:id="@+id/fab"
    android:layout_gravity="bottom|end"
    android:layout_margin="@dimen/fab_margin"
    android:src="@drawable/ic_fab_chat_button_white"
    .../>
```

Button Image Assets

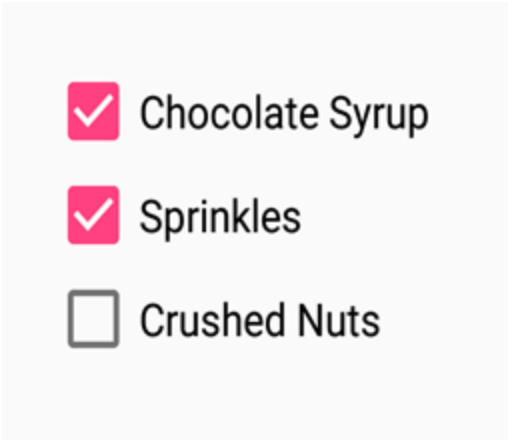
❖Button Image Assets

1. Right-click app/res/drawable
2. Choose New > Image Asset
3. Choose Action Bar and Tab Items from drop down menu
4. Click the Clipart: image
   (the Android logo)

Checkboxes

❖checkboxes

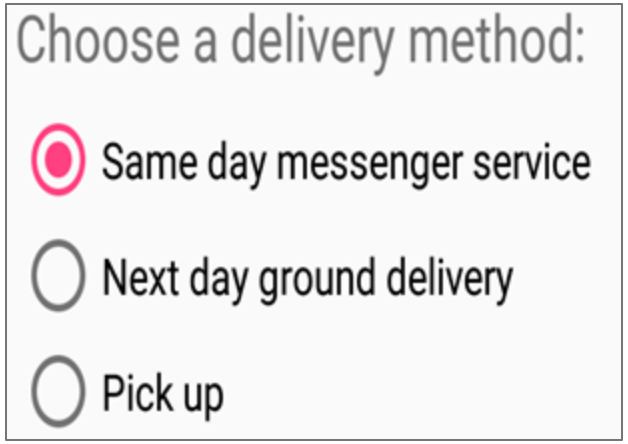
- ➢ User can select any number of choices.
- ➢ Checking one box does not uncheck another.
- ➢ Users expect checkboxes in a vertical list.
- ➢ Commonly used with a submit button.
- ➢ Every checkbox is a view and can have an onClick handler.



Radio Buttons

❖Radio Buttons

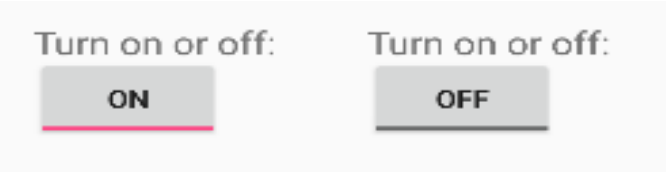
- User can select one of a number of choices.
- Put radio buttons in a RadioGroup.
- Checking one unchecks another.
- Put radio buttons in a vertical list or horizontally if labels are short.
- Every radio button can have an onClick handler.
- Commonly used with a submit button for the RadioGroup.

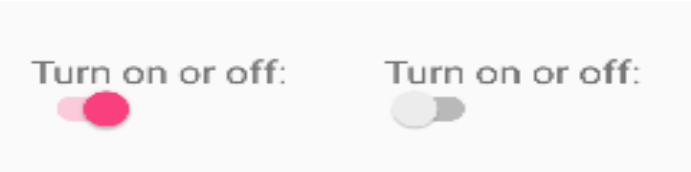


Toggle Buttons and Switches

❖Toggle Buttons and Switches

- ➢ User can switch between 2 exclusive states (on/off)
- ➢ Use android:onClick+callback—or handle clicks in code

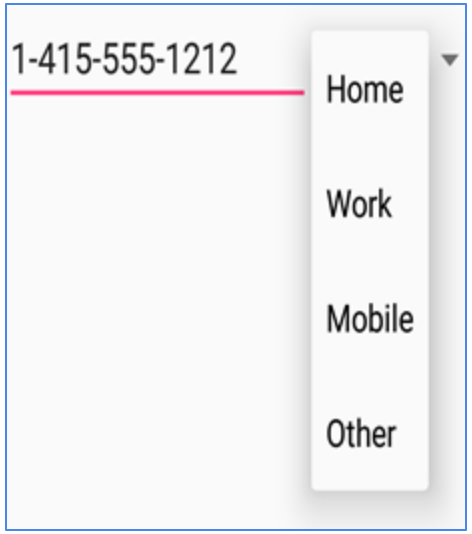




Spinners

❖Spinners

- ➢ Quick way to select value from a set.
- ➢ Drop-down list shows all values, user can select only one
- ➢ Spinners scroll automatically if necessary.
- ➢ Use the Spinner class.



Implementing Spinners

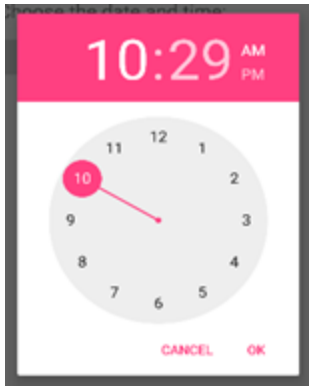
❖Implementing Spinners

- Spinner can be implemented using following steps:
  1. Create Spinner UI element in the XML layout
  2. Define spinner choices in an array
  3. Create Spinner and set ***onItemSelectedListener***
  4. Create an adapter with default spinner layouts
  5. Attach the adapter to the spinner
  6. Implement ***onItemSelectedListener*** method

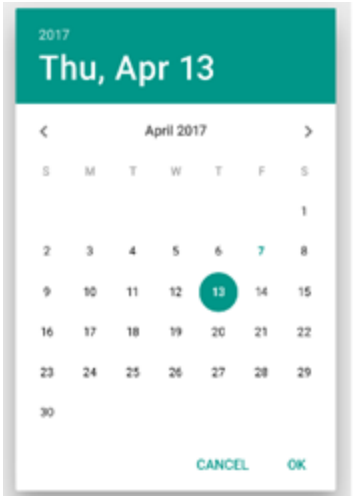
Dialogs

❖Dialogs

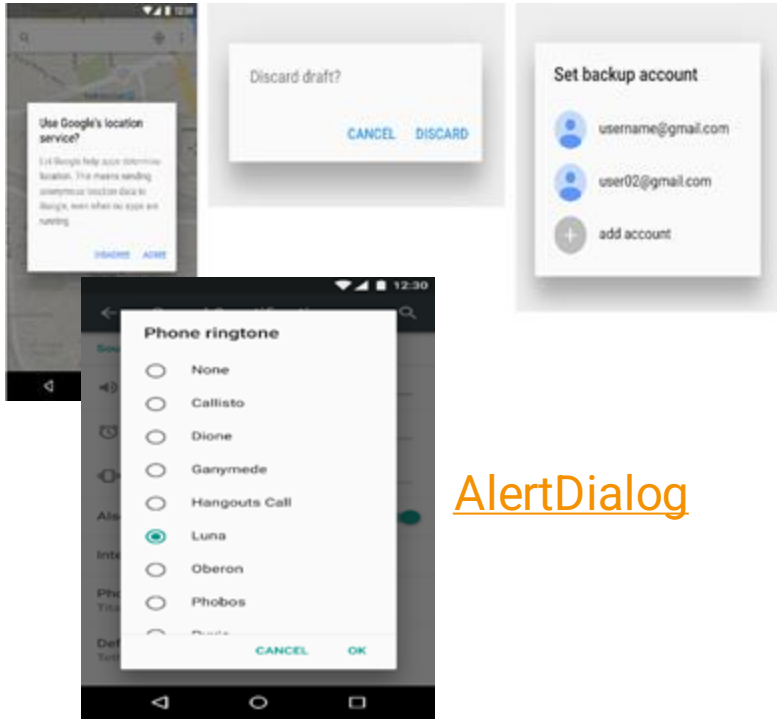
- ➢ Dialog appears on top, interrupting the flow of activity.
- ➢ Require user action to dismiss.



TimePickerDialog



DatePickerDialog

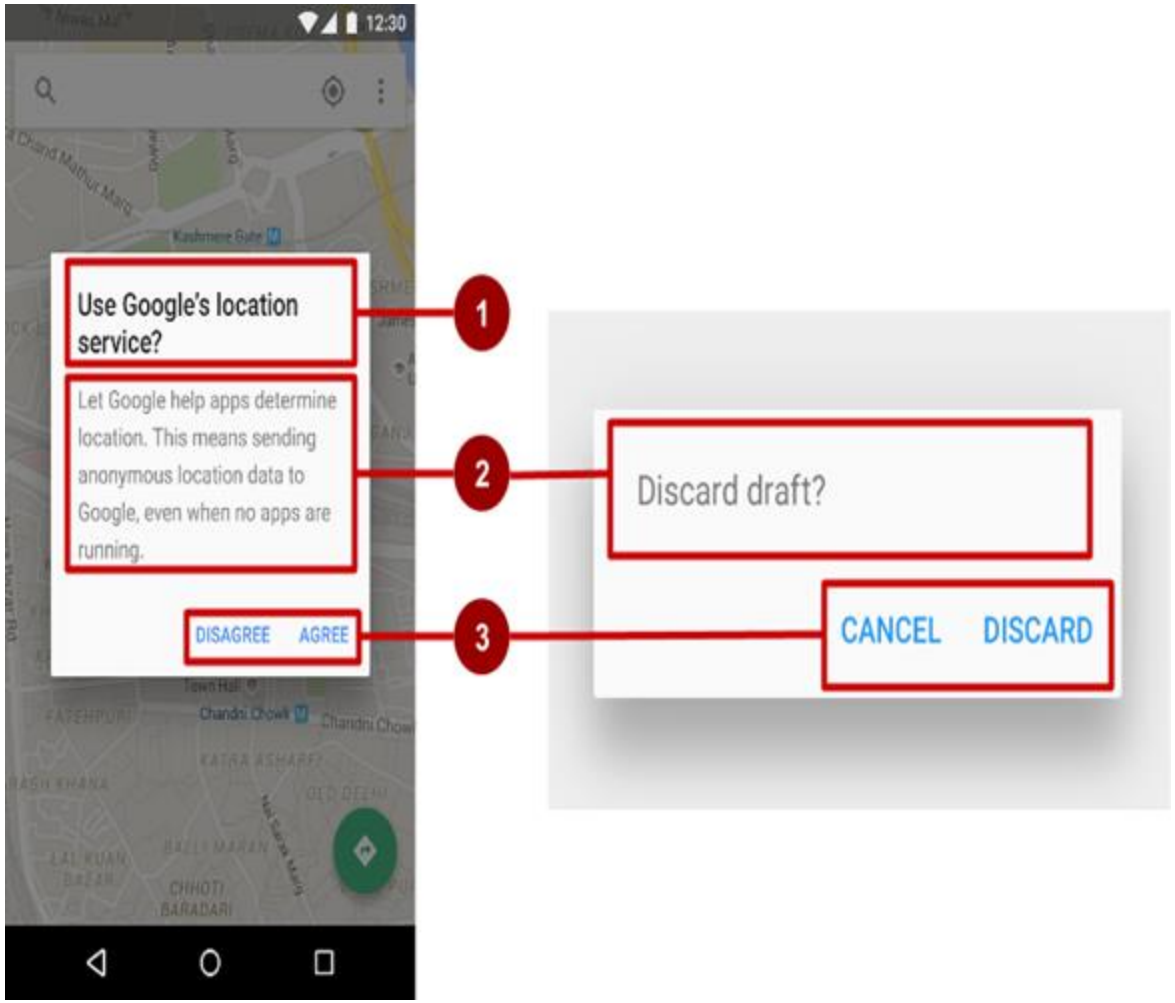


AlertDialog

AlertDialogs

❖ AlertDialogs

- ➢ AlertDialog can show:
  1. Title (optional)
  2. Content area
  3. Action buttons



Touch Gestures and Detect Gestures

Touch Gestures

- Touch gestures include:
  - long touch
  - double-tap
  - fling
  - drag
  - scroll
  - pinch

Detect Gestures

- Classes and methods are available to help you handle gestures.
  - ***GestureDetectorCompat*** class for common gestures
  - ***MotionEvent*** class for motion events
- For more read Android Gesture Documentation

Thank you!