CS4131
Mobile Application Development

Name: ________________________________ (   ) Date: _______________

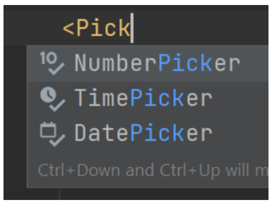
Chapter 3: Android App Building and Event Handling

3.1 Introduction to Chapter 3

Recall in the previous chapter that some of the common views were introduced and some example code was given to introduce how to code their functionalities, either, by introducing event listeners of linking methods to some of the View's attributes (like onClick, etc). This chapter will cover more advanced Widgets you may use for your apps, the associated codes to code the functionalities, as well as advanced event handling such as touch-based event-handling and the Gesture Detector class to allow you apps to have enhanced interactivity.

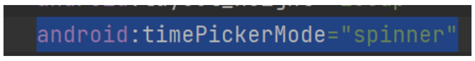
3.2 Examples Using More Advanced Widgets in Android

Example 1: DateTimePicker

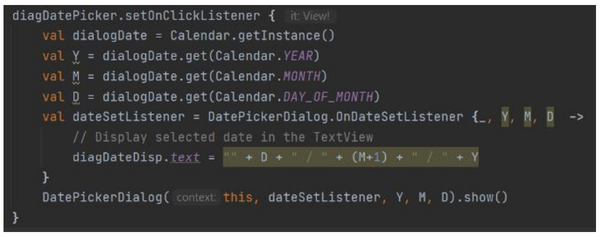


```
<Pick
10 NumberPicker
   TimePicker
   DatePicker
Ctrl+Down and Ctrl+Up will m
```

This app showcases the use of both the DatePicker and the TimePicker widgets. Note that both the DatePicker and TimePicker are NOT part of the Design Palette (similar to the NumberPicker showcased in Chapter 2 Example 2) and need to go through the XML code to code the widgets. Note that as seen in the screenshots, there are different forms for both pickers, namely the Dialog form, where the calendar and analog clocks are displayed in the form of dialog windows, and the Spinner form, where the user can choose the date and time. The Spinner form can be selected through the datePickerMode and the timePickerMode respectively. Whereas, the Dialog form is done through a listener attached to another View, for this case, the "Select Date" Button

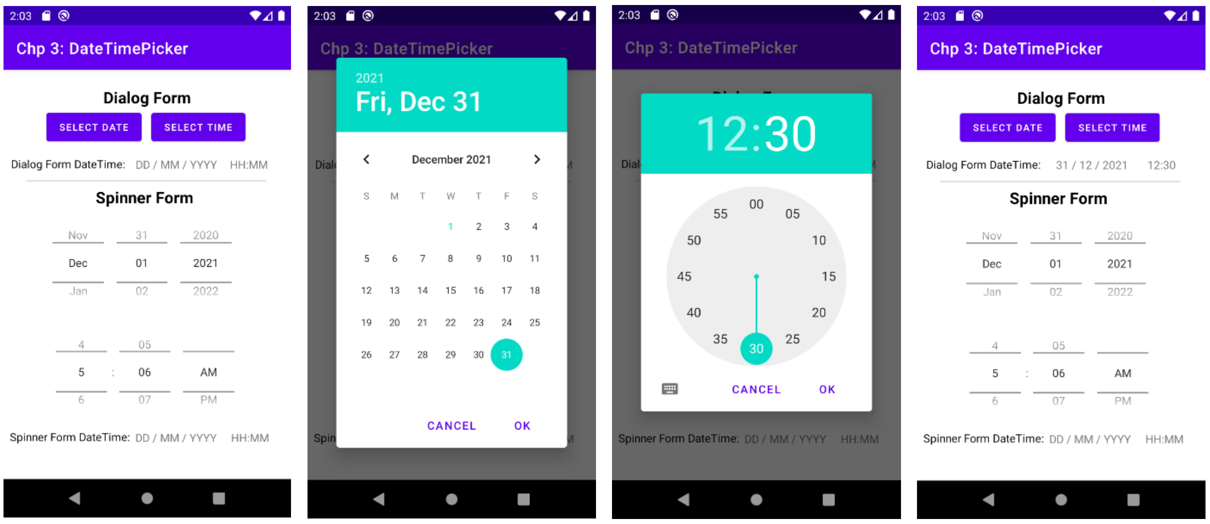


```
android:timePickerMode="spinner"
```

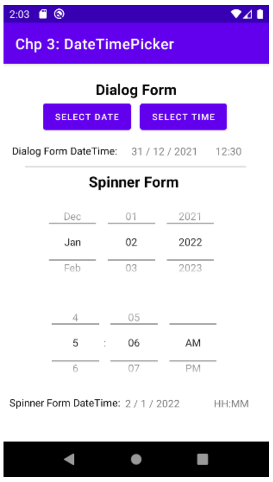


```
diagDatePicker.setOnClickListener {  it: View!
    val dialogDate = Calendar.getInstance()
    val Y = dialogDate.get(Calendar.YEAR)
    val M = dialogDate.get(Calendar.MONTH)
    val D = dialogDate.get(Calendar.DAY_OF_MONTH)
    val dateSetListener = DatePickerDialog.OnDateSetListener {_, Y, M, D  ->
        // Display selected date in the TextView
        diagDateDisp.text = "" + D + " / " + (M+1) + " / " + Y
    }
    DatePickerDialog( context: this, dateSetListener, Y, M, D).show()
}
```

Screenshots for DateTimePicker app



Chp 3: DateTimePicker
Dialog Form
SELECT DATE
SELECT TIME
Dialog Form DateTime: DD / MM / YYYY HH:MM
Spinner Form
Spinner Form DateTime: DD / MM / YYYY HH:MM
2021
Fri, Dec 31
December 2021
CANCEL
OK
12:30
CANCEL
OK
Dialog Form DateTime: 31 / 12 / 2021 12:30


2:03

Chp 3: DateTimePicker

Dialog Form

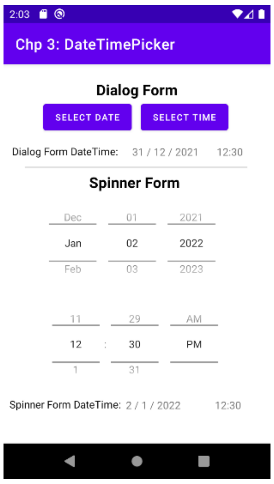
SELECT DATE SELECT TIME

Dialog Form DateTime: 31 / 12 / 2021 12:30

Spinner Form

```
Dec  01  2021
Jan  02  2022
Feb  03  2023

4    05
5  : 06  AM
6    07  PM
```

Spinner Form DateTime: 2 / 1 / 2022 HH:MM


2:03

Chp 3: DateTimePicker

Dialog Form

SELECT DATE SELECT TIME

Dialog Form DateTime: 31 / 12 / 2021 12:30

Spinner Form

```
Dec  01  2021
Jan  02  2022
Feb  03  2023

11   29  AM
12 : 30  PM
1    31
```

Spinner Form DateTime: 2 / 1 / 2022 12:30

strings.xml

```
<resources>
    <string name="app_name">Chp 3: DateTimePicker
</string>
    <string name="title1">Dialog Form</string>
    <string name="dateCalendar">Select Date</string>
    <string name="timeClock">Select Time</string>
    <string name="dialogDate"> Dialog Form DateTime:
</string>
    <string name="title2">Spinner Form</string>
    <string name="spinnerDate">Spinner Form DateTime:
</string>
</resources>
```

MainActivity.kt

```kotlin
import android.app.DatePickerDialog
import android.app.TimePickerDialog
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.widget.Button
import android.widget.DatePicker
import android.widget.TextView
import android.widget.TimePicker
import java.text.SimpleDateFormat
import java.util.*

class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        val diagDatePicker = findViewById<Button>(R.id.dialogDateBtn)
        val diagTimePicker = findViewById<Button>(R.id.dialogTimeBtn)
        val diagDateDisp = findViewById<TextView>(R.id.dialogDateDisp)
        val diagTimeDisp = findViewById<TextView>(R.id.dialogTimeDisp)
        val spinnerDPicker = findViewById<DatePicker>(R.id.spinnerDatePicker)
        val spinnerTPicker = findViewById<TimePicker>(R.id.spinnerTimePicker)
        val spinnerDDisp = findViewById<TextView>(R.id.spinnerDateDisp)
        val spinnerTDisp = findViewById<TextView>(R.id.spinnerTimeDisp)

        diagDatePicker.setOnClickListener {
            val dialogDate = Calendar.getInstance()
            val Y = dialogDate.get(Calendar.YEAR)
            val M = dialogDate.get(Calendar.MONTH)
            val D = dialogDate.get(Calendar.DAY_OF_MONTH)
            val dateSetListener = DatePickerDialog.OnDateSetListener {_, Y, M, D  ->
                // Display selected date in the TextView
                diagDateDisp.text = "" + D + " / " + (M+1) + " / " + Y
            }
            DatePickerDialog(this, dateSetListener, Y, M, D).show()
        }
        diagTimePicker.setOnClickListener{
            val dialogDate = Calendar.getInstance()
            val timeSetListener = TimePickerDialog.OnTimeSetListener{ _, hour, minute ->
                dialogDate.set(Calendar.HOUR_OF_DAY, hour)
                dialogDate.set(Calendar.MINUTE, minute)
                diagTimeDisp.text = SimpleDateFormat("HH:mm").format(dialogDate.time)
            }
            TimePickerDialog(this, timeSetListener, dialogDate.get(Calendar.HOUR_OF_DAY),
                dialogDate.get(Calendar.MINUTE), true).show()
        }
47          val today = Calendar.getInstance()
48          spinnerDPicker.init(today.get(Calendar.YEAR), today.get(Calendar.MONTH),
49              today.get(Calendar.DAY_OF_MONTH)) { _, Y, M, D ->
50              spinnerDDisp.text = "" + D + " / " + (M+1) + " / " + Y
51          }
52          spinnerTPicker.setOnTimeChangedListener{_, hour, minute ->
53              val dialogDate = Calendar.getInstance()
54              dialogDate.set(Calendar.HOUR_OF_DAY, hour)
55              dialogDate.set(Calendar.MINUTE, minute)
56              spinnerTDisp.text = SimpleDateFormat("HH:mm").format(dialogDate.time)
57          }
58      }
59  }
```

Explanation:

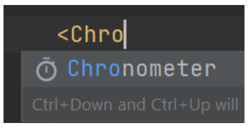
Lines 26 to 46	OnClickListeners are set on the respective buttons under the dialog mode. In the DatePicker, the Calendar object is initialised to the current date the app runs, and within it, a DateSetListener is set to reflect the date selected into the TextView. It is likewise for the TimePicker such that it is initialized to the current time the app runs and within it, a TimeSetListener is set to reflect the time selected into the TextView.
Lines 47 to 57	The DatePicker and TimePicker are initialized to contain listeners to obtain the date and time set by the user. Note the use of lambda expressions in both listeners.

activity_main.xml

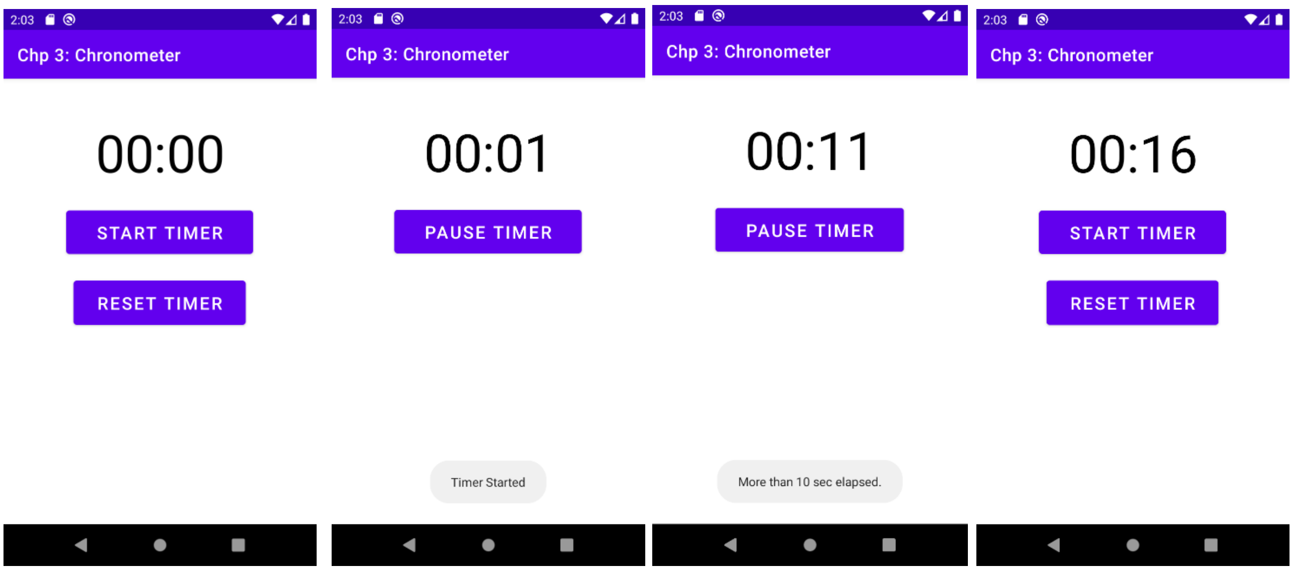
```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/textView3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="8dp"
        android:text="@string/title1"
        android:textColor="@color/black"
        android:textSize="20sp"
        android:textStyle="bold"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.026" />

    <LinearLayout
        android:id="@+id/linearLayout"
        android:layout_width="252dp"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.496"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/textView3"
        app:layout_constraintVertical_bias="0.0">

        <Button
            android:id="@+id/dialogDateBtn"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/dateCalendar"
            android:textSize="12sp" />

        <Button
            android:id="@+id/dialogTimeBtn"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginStart="12dp"
            android:text="@string/timeClock"
            android:textSize="12sp" />

    </LinearLayout>

    <androidx.constraintlayout.widget.ConstraintLayout
        android:id="@+id/constraintLayout"
        android:layout_width="339dp"
        android:layout_height="29dp"
        android:layout_marginTop="8dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.444"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/linearLayout"
        app:layout_constraintVertical_bias="0.0">

        <TextView
            android:id="@+id/textView4"
            android:layout_width="155dp"
            android:layout_height="20dp"
            android:text="@string/dialogDate"
            android:textColor="@color/black"
            android:textSize="14sp"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintEnd_toStartOf="@+id/dialogDateDisp"
            app:layout_constraintHorizontal_bias="0.206"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="parent" />

        <TextView
            android:id="@+id/dialogDateDisp"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="DD / MM / YYYY"
            android:textAlignment="center"
            android:textSize="14sp"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintHorizontal_bias="0.658"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="parent" />

        <TextView
            android:id="@+id/dialogTimeDisp"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="HH:MM"
            android:textAlignment="center"
            android:textSize="14sp"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintHorizontal_bias="0.483"
            app:layout_constraintStart_toEndOf="@+id/dialogDateDisp"
            app:layout_constraintTop_toTopOf="parent" />
    </androidx.constraintlayout.widget.ConstraintLayout>

    <TextView
        android:id="@+id/textView6"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="8dp"
        android:text="@string/title2"
        android:textColor="@color/black"
        android:textSize="20sp"
        android:textStyle="bold"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintHorizontal_bias="0.486"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/constraintLayout"
        app:layout_constraintVertical_bias="0.015" />

    <DatePicker
        android:id="@+id/spinnerDatePicker"
        android:layout_width="267dp"
        android:layout_height="137dp"
        android:calendarViewShown="false"
        android:datePickerMode="spinner"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.473"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/textView6"
        app:layout_constraintVertical_bias="0.0" />

    <TimePicker
        android:id="@+id/spinnerTimePicker"
        android:layout_width="266dp"
        android:layout_height="136dp"
        android:timePickerMode="spinner"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.472"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/spinnerDatePicker"
        app:layout_constraintVertical_bias="0.0" />

    <androidx.constraintlayout.widget.ConstraintLayout
        android:id="@+id/constraintLayout3"
        android:layout_width="349dp"
        android:layout_height="32dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.49"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/spinnerTimePicker"
        app:layout_constraintVertical_bias="0.0">

        <TextView
            android:id="@+id/textView7"
            android:layout_width="152dp"
            android:layout_height="24dp"
            android:text="@string/spinnerDate"
            android:textAlignment="center"
            android:textColor="@color/black"
            android:textSize="14sp"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintEnd_toStartOf="@+id/spinnerDateDisp"
            app:layout_constraintHorizontal_bias="0.181"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="parent" />

        <TextView
            android:id="@+id/spinnerTimeDisp"
            android:layout_width="53dp"
            android:layout_height="21dp"
            android:layout_weight="1"
            android:text="HH:MM"
            android:textAlignment="viewStart"
            android:textSize="14sp"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintHorizontal_bias="0.945"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="parent"
            app:layout_constraintVertical_bias="0.454" />

        <TextView
            android:id="@+id/spinnerDateDisp"
            android:layout_width="103dp"
            android:layout_height="21dp"
            android:layout_weight="1"
            android:text="DD / MM / YYYY"
            android:textAlignment="viewStart"
            android:textSize="14sp"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintEnd_toStartOf="@+id/spinnerTimeDisp"
            app:layout_constraintHorizontal_bias="0.903"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="parent"
            app:layout_constraintVertical_bias="0.454" />

    </androidx.constraintlayout.widget.ConstraintLayout>

    <View
        android:id="@+id/divider2"
        android:layout_width="300dp"
        android:layout_height="2dp"
        android:layout_marginTop="4dp"
        android:background="?android:attr/listDivider"
        app:layout_constraintBottom_toTopOf="@+id/textView6"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.486"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/constraintLayout"
        app:layout_constraintVertical_bias="0.181" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

Example 2: Chronometer

The app showcases the chronometer which is effectively a stopwatch. This app allows the user to start the timer and when it runs, the user has the option to pause the timer. When more than 10 seconds have passed, a Toast will show that more than 10 seconds has elapsed, and finally when the timer is paused by the user, the user has the option to either resume (by clicking on Start Timer again) or to Reset Timer. Note that the Chronometer can only be accessed through editing the XML code itself.



Screenshots for Chronometer app



strings.xml

```
<resources>
    <string name="app_name">Chp 3: Chronometer</string>
    <string name="start">Start Timer</string>
    <string name="pause">Pause Timer</string>
    <string name="reset">Reset Timer</string>
    <string name="toastStart">Timer Started</string>
    <string name="toastStop">Timer Stopped</string>
    <string name="listenMsg">More than 10 sec elapsed.</string>
</resources>
```

MainActivity.kt

```
1   import androidx.appcompat.app.AppCompatActivity
2   import android.os.Bundle
3   import android.os.SystemClock
4   import android.view.View
5   import android.widget.Button
6   import android.widget.Chronometer
7   import android.widget.Toast
8
9   class MainActivity : AppCompatActivity() {
10      private var isWorking = false
11      private var pauseOffSet : Long = 0
12
13      override fun onCreate(savedInstanceState: Bundle?) {
14          super.onCreate(savedInstanceState)
15          setContentView(R.layout.activity_main)
16
17          val meter : Chronometer = findViewById(R.id.timeDisp)
18          meter.base = SystemClock.elapsedRealtime()
19          meter.setOnChronometerTickListener { cmeter ->
20              if ((SystemClock.elapsedRealtime() - cmeter.base) - 10000 > 0)
21                  Toast.makeText(this@MainActivity, R.string.listenMsg,
22                                                       Toast.LENGTH_LONG).show()
23          }
24          val startStopBtn : Button = findViewById(R.id.startStopBtn)
25          val resetBtn : Button = findViewById(R.id.resetBtn)
26          startStopBtn.setOnClickListener {
27              if (!isWorking) {
28                  meter.base = SystemClock.elapsedRealtime() - pauseOffSet
29                  meter.start()
30                  isWorking = true
31                  resetBtn.visibility = View.INVISIBLE
32              } else {
33                  meter.stop()
34                  pauseOffSet = SystemClock.elapsedRealtime() - meter.base
35                  isWorking = false
36                  resetBtn.visibility = View.VISIBLE
37              }
38              if (isWorking) startStopBtn.setText(R.string.pause)
39              else startStopBtn.setText(R.string.start)
40
41              Toast.makeText(this@MainActivity, getString(
42                      if (isWorking) R.string.toastStart else R.string.toastStop),
43                          Toast.LENGTH_SHORT).show()
44          }
45          resetBtn.setOnClickListener{
46              meter.base = SystemClock.elapsedRealtime()
47              pauseOffSet = 0
48          }
49      }
50  }
```

Explanation:

Lines 18 to 23	The Chronometer takes reference from the current time in the system clock as its base and an OnChronometerTickListener is set such that the Toast is displayed notifying the user that more than 10 seconds (10 000 milliseconds) have elapsed.
Lines 26 to 44	The startStopBtn has an OnClickListener which will run the Chronometer. When the timer starts, the “Reset” button will disappear. When the timer is paused and resumes, the Chronometer sets the new timer base to be the time which the pause happened to provide a more accurate measure of the time elapsed
Lines 45 to 48	The resetBtn has an OnClickListener which will set the base to the current system clock time.

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <Chronometer
        android:id="@+id/timeDisp"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textColor="@color/black"
        android:textSize="60sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.103" />

    <Button
        android:id="@+id/startStopBtn"
        android:layout_width="215dp"
        android:layout_height="62dp"
        android:text="@string/start"
        android:textSize="20sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.498"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/timeDisp"
        app:layout_constraintVertical_bias="0.065" />

    <Button
        android:id="@+id/resetBtn"
        android:layout_width="198dp"
        android:layout_height="63dp"
        android:text="@string/reset"
        android:textSize="20sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.497"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/startStopBtn"
        app:layout_constraintVertical_bias="0.077" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

3.3 Android Multi-Touch Event-Handling and Gesture Detection

Recall in Chapter 2 that Android Views contain at least one of the following listeners:

Listener	Description
onClick	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	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).
onTouch	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	Called when a Context Menu is being built (as the result of a sustained "long click"). See the discussion on context menus in the Menus developer guide.

onFocusChange	Detects when focus moves from the current view as the result of interaction with a track-ball or navigation key.
onKey	Called when the user is focused on the item and presses or releases a hardware key on the device.

So far, all the app examples we have seen only include onClick events. For the onLongClick event, which can also be applied to Buttons, more is required (return of a Boolean value) to indicate to Android runtime if the callback has consumed the event, such that upon true, the event is discarded by the framework; whereas upon false, the Android framework will consider the event still active and consequently pass it along to the next matching event listener that is registered on the same View.

Other than detecting clicks, an important feature on mobile devices is to detect touch. If an onTouch listener is setup, it can detect multiple touches, for example, use 2 fingers simultaneously to perform **gestures** (motions) such as zooming in or out. Each touch performed on a view are referred to as **pointers**, and are further classified as primary or non-primary pointers. The event whereby the user's gesture is detected is classified as a MotionEvent, and MotionEvent objects have their own classification of different gestures, some of which are as described below:

MotionEvent	**Description**
ACTION_DOWN	A pressed gesture has started, the motion contains the initial starting location.
ACTION_UP	A pressed gesture has finished, the motion contains the final release location as well as any intermediate points since the last down or move event.
ACTION_SCROLL	The motion event contains relative vertical and/or horizontal scroll offsets.
ACTION_POINTER_DOWN	A **non-primary** pointer has gone down.
ACTION_POINTER_UP	A **non-primary** pointer has gone up.
ACTION_MOVE	A change has happened during a press gesture (between ACTION_DOWN and ACTION_UP).
ACTION_CANCEL	The current gesture has been aborted.
ACTION_HOVER_ENTER	The pointer is not down but has entered the boundaries of a window or view.
ACTION_HOVER_EXIT	The pointer is not down but has exited the boundaries of a window or view.
ACTION_HOVER_MOVE	A change happened but the pointer is not down

A touch on a view, particularly one involving motion across the screen, will generate a stream of events before the point of contact with the screen is lifted. As such, it is likely that an application will need to track individual touches over multiple touch events. While the ID of a specific touch gesture will not change from one event to the next, it is important to keep in mind that the index value will change as other touch events come and go. When working with a touch gesture over multiple events, therefore, it is essential that the ID value be used as the touch reference in order to make sure the same touch is being tracked.

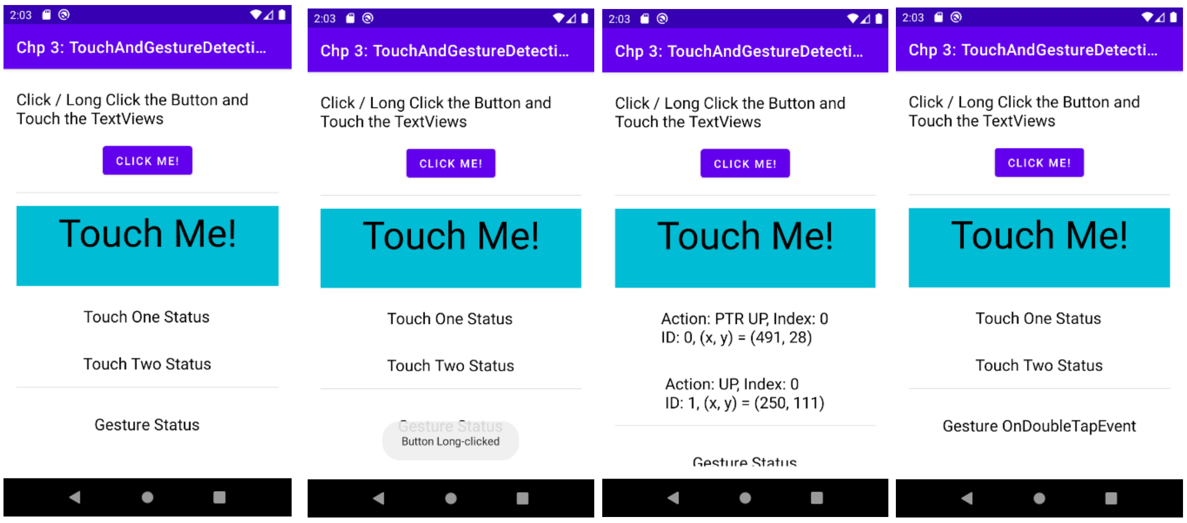
The Android SDK also provides mechanisms for the detection of both ***common gestures*** such as a tap, double tap, long press or a swiping motion in either a horizontal or a vertical direction (fling), through using the onGestureListener in the GesutureDetectorCompat class. The GestureDetectorCompat class is part of the **Support Library** which allow for provision of newer features on earlier functions of Android or gracefully fall back to equivalent functionality. This is especially important in the case of app development where your users utlise a wide range of API versions of Android. It is strongly recommended to utilize the support library in app development.

The descriptions will be clearer when we move to the example app in the next section.

3.4 Example of Multi-Touch Event-Handling and Gesture Detection

The app is subdivided into three parts. The top-most part is where a Toast will be generated indicating if the button has been clicked or long-clicked. The middle part is where the touch status on the cyan “Touch Me!” View will be reflected in the two TextViews below, meaning to say that it will detect and reflect the presence of more than one pointer. For the bottom part, a GestureDectector is implemented such that any gestures on the layout itself will be reflected on the TextView showing “Gesture Status”.

Screenshots for TouchAndGestureDetection app


2:03
Chp 3: TouchAndGestureDetecti...
Click / Long Click the Button and Touch the TextViews
CLICK ME!
Touch Me!
Touch One Status
Touch Two Status
Gesture Status
2:03
Chp 3: TouchAndGestureDetecti...
Click / Long Click the Button and Touch the TextViews
CLICK ME!
Touch Me!
Touch One Status
Touch Two Status
Gesture Status
Button Long-clicked
2:03
Chp 3: TouchAndGestureDetecti...
Click / Long Click the Button and Touch the TextViews
CLICK ME!
Touch Me!
Action: PTR UP, Index: 0
ID: 0, (x, y) = (491, 28)
Action: UP, Index: 0
ID: 1, (x, y) = (250, 111)
Gesture Status
2:03
Chp 3: TouchAndGestureDetecti...
Click / Long Click the Button and Touch the TextViews
CLICK ME!
Touch Me!
Touch One Status
Touch Two Status
Gesture OnDoubleTapEvent

strings.xml

```
<resources>
    <string name="app_name">Chp 3: TouchAndGestureDetection</string>
    <string name="instru">Click / Long Click the Button and Touch the TextViews</string>
    <string name="clickBtn">Click Me!</string>
    <string name="touchMe">Touch Me!</string>
    <string name="textView1">Touch One Status</string>
    <string name="textView2">Touch Two Status</string>
    <string name="gestureText">Gesture Status</string>
    <string name="gestureOnDown">Gesture OnDown</string>
    <string name="gestureOnLongPress">Gesture OnLongPress</string>
    <string name="gestureOnShowPress">Gesture OnShowPress</string>
    <string name="gestureOnSingleTapConfirmed">Gesture OnSingleTapConfirmed</string>
    <string name="gestureOnSingleTapUp">Gesture OnSingleTapUp</string>
    <string name="gestureOnDoubleTap">Gesture OnDoubleTap</string>
    <string name="gestureOnDoubleTapEvent">Gesture OnDoubleTapEvent</string>
</resources>
```

MainActivity.kt

```kotlin
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.view.GestureDetector
import android.view.MotionEvent
import android.widget.Button
import android.widget.TextView
import android.widget.Toast
import androidx.constraintlayout.widget.ConstraintLayout
import androidx.core.view.GestureDetectorCompat

class MainActivity : AppCompatActivity(), GestureDetector.OnGestureListener,
    GestureDetector.OnDoubleTapListener{
    private lateinit var textView1: TextView
    private lateinit var textView2: TextView
    private lateinit var clickBtn: Button
    private lateinit var mainLayout: ConstraintLayout
    private lateinit var handleTouch: TextView
    private lateinit var gestureText: TextView
    private lateinit var gestureDetector: GestureDetectorCompat

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        clickBtn = findViewById(R.id.clickBtn)
        textView1 = findViewById(R.id.textView1)
        textView2 = findViewById(R.id.textView2)
        handleTouch = findViewById(R.id.handleTouch)
        mainLayout = findViewById(R.id.mainLayout)
        gestureText = findViewById(R.id.gestureTextView)

        clickBtn.setOnClickListener{ view ->
            Toast.makeText(this, "Button clicked", Toast.LENGTH_SHORT).show()
        }
        clickBtn.setOnLongClickListener {  view ->
            Toast.makeText(this, "Button Long-clicked", Toast.LENGTH_SHORT).show()
            true
        }
        handleTouch.setOnTouchListener { _, motionEvent ->
            handleTouch(motionEvent)
            true
        }
        gestureDetector = GestureDetectorCompat(this, this)
        //Set gesture detector as the double tap listener
        gestureDetector.setOnDoubleTapListener(this)
    }

    private fun handleTouch(motionEvent: MotionEvent){
        val pointerCount = motionEvent.pointerCount
        for (i in 0 until pointerCount){
            val x = motionEvent.getX(i).toInt()
            val y = motionEvent.getY(i).toInt()
            val id = motionEvent.getPointerId(i)
            val action = motionEvent.actionMasked
            val actionIndex = motionEvent.actionIndex
            val actionString : String
            when(action){
                MotionEvent.ACTION_DOWN ->{ actionString = "DOWN" }
                MotionEvent.ACTION_UP -> {actionString = "UP"}
                MotionEvent.ACTION_POINTER_DOWN -> {actionString = "PTR DOWN"}
                MotionEvent.ACTION_POINTER_UP -> {actionString = "PTR UP"}
                MotionEvent.ACTION_MOVE -> {actionString = "MOVE"}
                else -> {actionString = "Nothing"}
63              }
64              val touchStatus = "Action: $actionString, Index: $actionIndex\n" +
65                      "ID: $id, (x, y) = ($x, $y)"
66              if (id == 0)
67                  textView1.text = touchStatus
68              else
69                  textView2.text = touchStatus
70          }
71      }
72
73      override fun onTouchEvent(event: MotionEvent): Boolean {
74          return if (gestureDetector.onTouchEvent(event)) {true} else {super.onTouchEvent(event)}
75      }
76      override fun onDown(event: MotionEvent): Boolean {
77          gestureText.text = getString(R.string.gestureOnDown)
78          return true
79      }
80      override fun onFling(event1: MotionEvent, event2: MotionEvent,
81          velocityX: Float, velocityY: Float): Boolean {
82          gestureText.text = "Gesture OnFling;\n Velocity (x, y) = ($velocityX, $velocityY)"
83          return true
84      }
85      override fun onLongPress(event: MotionEvent) {
86          gestureText.text = getString(R.string.gestureOnLongPress)
87      }
88      override fun onScroll(event1: MotionEvent, event2: MotionEvent,
89          distanceX: Float, distanceY: Float): Boolean {
90          gestureText.text = "Gesture OnScroll;\n Displacement (x, y) = ($distanceX, $distanceY)"
91          return true
92      }
93      override fun onShowPress(event: MotionEvent) {
94          gestureText.text = getString(R.string.gestureOnShowPress)
95      }
96      override fun onSingleTapUp(event: MotionEvent): Boolean {
97          gestureText.text = getString(R.string.gestureOnSingleTapUp)
98          return true
99      }
100     override fun onDoubleTap(event: MotionEvent): Boolean {
101         gestureText.text = getString(R.string.gestureOnDoubleTap)
102         return true
103     }
104     override fun onDoubleTapEvent(event: MotionEvent): Boolean {
105         gestureText.text = getString(R.string.gestureOnDoubleTapEvent)
106         return true
108     }
109     override fun onSingleTapConfirmed(event: MotionEvent): Boolean {
110         gestureText.text = getString(R.string.gestureOnSingleTapConfirmed)
111         return true
112     }
113 }
```

Explanation:

Lines 31 to 37	Both an onClick and onLongClick listeners are set to the Button clickBtn. Note that for onLongClick listeners, a Boolean return is expected and Kotlin allows for returns just by stating “true” within the listener written as a lambda expression, unlike Java.
Lines 38 to 41 Lines 47 to 71	The TextView handleTouch has a onTouchListener implemented and calls the method handleTouch(). Within handleTouch, note the different attributes that are available for you to use. The pointerCount attribute contains the number of pointers within the MotionEvent object passed in, and for each

	pointer, you can retrieve the x and y coordinates, and even the type of action (actionMasked) done, where the switch statement (when…) will show the touch status in textView1 and textView2. Note that in the case of multiple touches, each pointer is tagged with an ID which can be retrieved by the getPointerID() method, hence, displaying the ID number as seen in the output. A touch on a View will generate a stream of events before the point of contact on the screen is lifted, and hence, as the app needs to track individual touches over multiple touch events, **the index value of the touch events may change but the ID will not**, hence, it is recommended to use the ID as the touch reference to ensure consistency in the touch event processed.
Lines 11 to 12 Lines 42 to 45 Lines 73 to 112	The Activity implements both the OnGestureListener class and the OnDoubleTapListener classes from the GestureDetector class. A OnDoubleTapListener is implemented as we wish to detect not just a single tap on the layout, but also a double tap. Note the overridden methods from line 73 onwards are methods part of the OnGestureListener class and some have special requirements to return a Boolean value while some do not.

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/mainLayout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="16dp"
    tools:context=".MainActivity">

    <View
        android:id="@+id/divider2"
        android:layout_width="0dp"
        android:layout_height="1dp"
        android:layout_marginTop="24dp"
        android:background="?android:attr/listDivider"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/textView2" />

    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="24dp"
        android:text="@string/instru"
        android:textColor="@color/black"
        android:textSize="20sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintHorizontal_bias="0.533"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.0" />
    <Button
        android:id="@+id/clickBtn"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="16dp"
        android:text="@string/clickBtn"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/textView"
        app:layout_constraintVertical_bias="0.0" />

    <View
        android:id="@+id/divider"
        android:layout_width="0dp"
        android:layout_height="1dp"
        android:layout_marginTop="32dp"
        android:background="?android:attr/listDivider"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/clickBtn" />

    <TextView
        android:id="@+id/handleTouch"
        android:layout_width="0dp"
        android:layout_height="100dp"
        android:layout_marginTop="16dp"
        android:background="#00BCD4"
        android:text="@string/touchMe"
        android:textAlignment="center"
        android:textColor="@android:color/black"
        android:textSize="48sp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/divider" />

    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="24dp"
        android:text="@string/textView1"
        android:textColor="@color/black"
        android:textSize="20sp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.494"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/handleTouch" />

    <TextView
        android:id="@+id/textView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="32dp"
        android:text="@string/textView2"
        android:textColor="@color/black"
        android:textSize="20sp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.498"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/textView1" />

    <TextView
        android:id="@+id/gestureTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="32dp"
        android:text="@string/gestureText"
        android:textColor="@color/black"
        android:textSize="20sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.498"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/divider2"
        app:layout_constraintVertical_bias="0.0" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

[Reference]

[1] Android MotionEvent class:
https://developer.android.com/reference/android/view/MotionEvent
[2] Detecting Common Gestures: https://developer.android.com/training/gestures/detector
[3] Android Support Library: https://developer.android.com/topic/libraries/support-library
[3] Introduction to JShell https://docs.oracle.com/javase/9/jshell/introduction-jshell.htm