More widgets

Common Controls

- SeekBar
- ToggleButton / Switch
- TimePicker
- DatePicker
- Swipe-to-Refresh and ListView
- Asset folder

SeekBar

- XML Part

```
<SeekBar
android:id="@+id/tipSeekBar"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:max="100"
android:progress="50"
android:layout_marginTop="20dp"
android:layout_gravity="fill_horizontal"/>
```

SeekBar

- Java Part (adding listener to notify changes)

- **`seekBarInstance.setOnSeekBarChangeListener(new OnSeekBarChangeListener() {…}`**

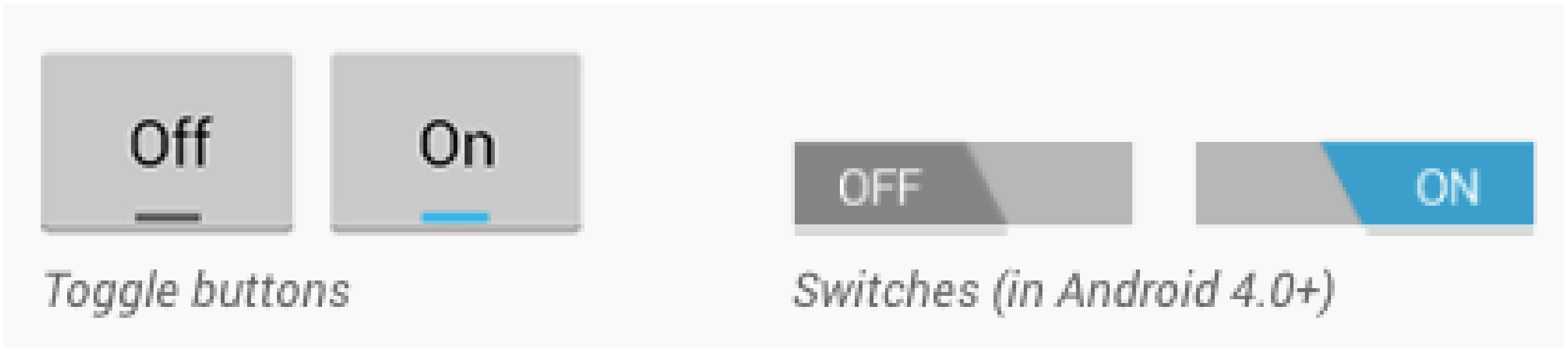
- We need to implement three abstract methods here;
- **`public void onProgressChanged(SeekBar seekBar, int progresValue, boolean fromUser) {…}`**
- This listener method will be invoked if any change is made in the SeekBar

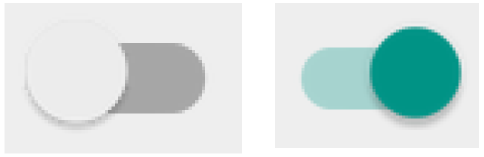
- **`public void onStartTrackingTouch(SeekBar seekBar) {...}`**
- This listener method will be invoked at the start of user's touch event.

- **`public void onStopTrackingTouch(SeekBar seekBar) {...}`**
- This listener method will be invoked at the end of user touch event.

ToggleButtons or Switch

- A toggle button allows the user to change a setting between two states.
- You can add a basic toggle button to your layout with the **ToggleButton** object.
- Android 4.0 (API level 14) introduces another kind of toggle button called a switch that provides a slider control, which you can add with a **Switch** object





ToggleButtons or Switch

- XML Part

```
<Switch
    android:id="@+id/switchButton"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textOn="Vibrate on"
    android:textOff="Vibrate off"
    android:showText="true"
    android:onClick="onSwitchClicked"/>
```

ToggleButtons or Switch

- Java Part

```java
public void onSwitchClicked(View view) {
    // Is the switch on?
    boolean on = ((Switch) view).isChecked();

    if (on) {
        // Enable vibrate
    } else {
        // Disable vibrate
    }
}
```

SwitchCompat

- SwitchCompat is a version of the Switch widget which runs on devices back to API 7.

```
<android.support.v7.widget.SwitchCompat
    android:id="@+id/switchButton"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textOn="on"
    android:textOff="off"
    android:showText="true"
    android:onClick="onSwitchClicked"/>
```

ScrollView

- A ScrollView is a simple scrolling container you can use to scroll whatever you put inside it, which might be a list of items, or it might not

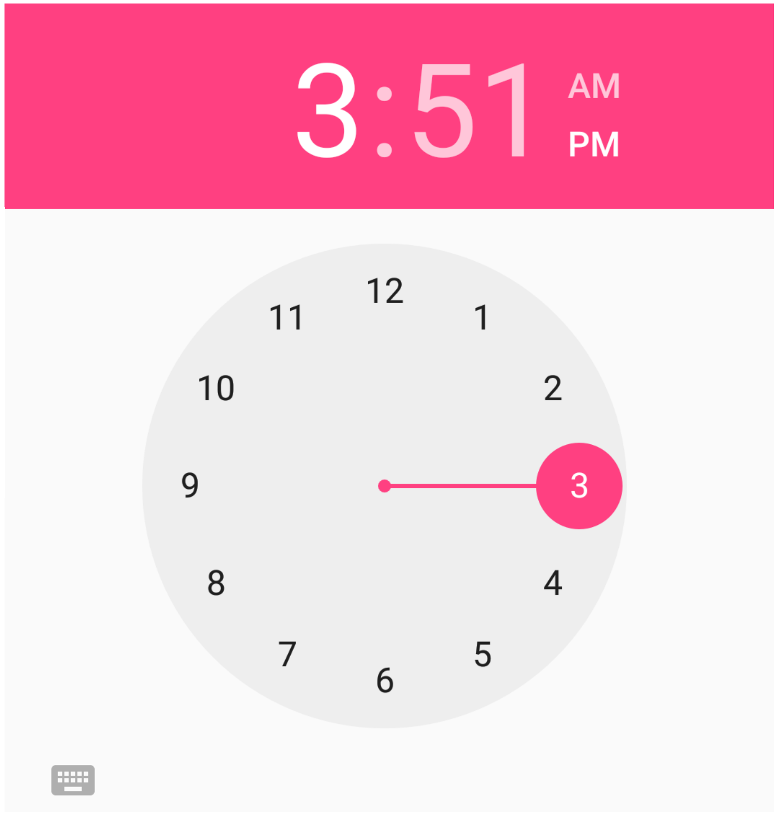
```
<ScrollView
      android:id="@+id/scrollView1"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
            <LinearLayout
         android:id="@+id/linearLayout1"
         android:orientation="vertical"
               android:layout_width="wrap_content"
         android:layout_height="wrap_content"/>

  </ScrollView>
```

TimePicker

- Android TimePicker allows you to select the time of day in either 24 hour or AM/PM mode

- Default form (clock)

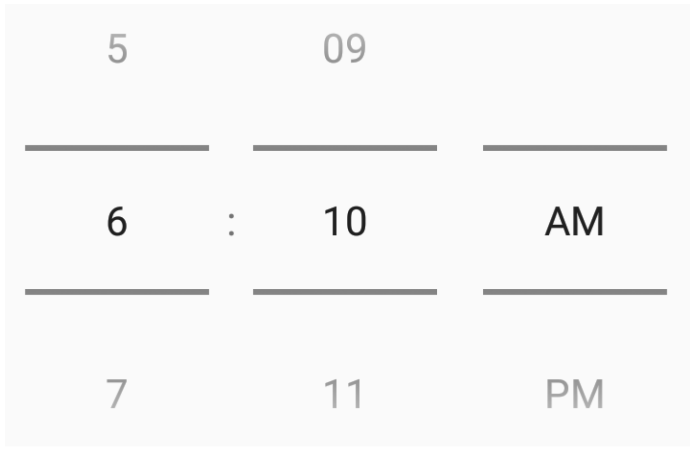
```
<TimePicker
    android:id="@+id/timePicker1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:timePickerMode="clock">
</TimePicker>
```



TimePicker

- Spinner mode

```
<TimePicker
    android:id="@+id/timePicker1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:timePickerMode="spinner">
</TimePicker>
```



```
TimePicker timePicker = (TimePicker)findViewById(R.id.timerPicker1);
// to get the time selected by the user on the screen
int min = timePicker.getCurrentMinute(); // This method was deprecated in API
level 23. Use getMinute()

int hour = timePicker.getCurrentHour(); // This method was deprecated in API
level 23. Use getHour()
```

TimePicker

- setOnTimeChangedListener(TimePicker.OnTimeChangedListener onTimeChangedListener)
  - This method set the callback that indicates the time has been adjusted by the user

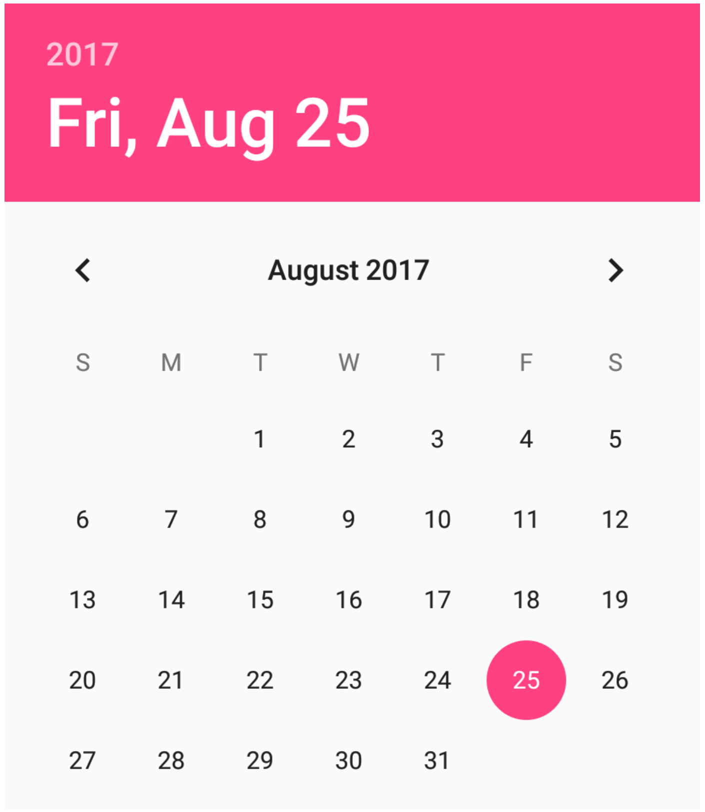
```
timePicker.setOnTimeChangedListener(new TimePicker.OnTimeChangedListener() {
    @Override
    public void onTimeChanged(TimePicker timePicker, int hourOfDay, int minute)
        {

        }
    });
```

DatePicker

- Provides a widget for selecting a date.

- Default form (calendar)

```xml
<DatePicker
    android:id="@+id/datePicker1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:datePickerMode="calendar">
</DatePicker>
```

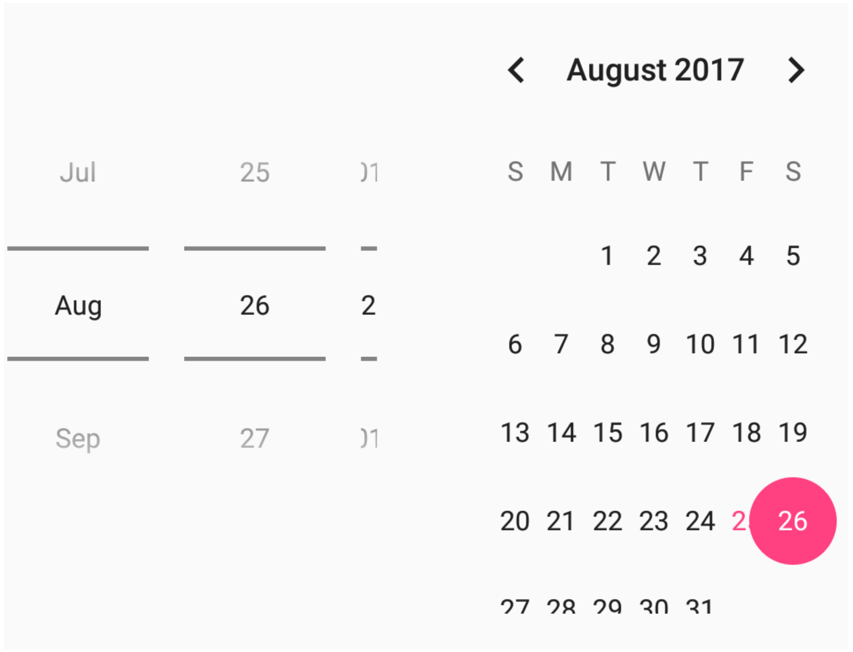


S	M	T	W	T	F	S
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31		

DatePicker

- spinner form

```
<DatePicker
    android:id="@+id/datePicker1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:datePickerMode="spinner">
</DatePicker>
```



To update the selected date:

```
DatePicker datePicker= (DatePicker)findViewById(R.id.datePicker1);
datePicker.updateDate(year, month, dayOfMonth);
```

Get the selected date:

```
int day = datePicker.getDayOfMonth();
int month = datePicker.getMonth() + 1;
int year = datePicker.getYear();
```

DatePicker

- callback used to indicate the user changed the date
  - Use the init method

```
DatePicker datePicker= (DatePicker)findViewById(R.id.datePicker1);

Calendar gregorianCalendar = new GregorianCalendar();


int day = gregorianCalendar.get(Calendar.DAY_OF_MONTH);
int month =  gregorianCalendar.get(Calendar.MONTH);
int year = gregorianCalendar.get(Calendar.YEAR);

datePicker.init(year, month, year, new DatePicker.OnDateChangedListener() {
    @Override
    public void onDateChanged(DatePicker datePicker, int year, int monthOfYear,
int dayOfMonth) {


    }
});
```

The DatePicker public method setOnDateChangedListener is added in API level 26
➔ Min API level should be 26 in order to use it.

Swipe-to-Refresh and ListView

To add the swipe to refresh widget to an existing app, add **SwipeRefreshLayout** as the parent of a **single ListView** or **GridView**

```
<androidx.swiperefreshlayout.widget.SwipeRefreshLayout

    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/swiperefresh"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <ListView
        android:id="@android:id/list"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />

</androidx.swiperefreshlayout.widget.SwipeRefreshLayout>
```

Swipe-to-Refresh - Responding to a Refresh Request

Implement the SwipeRefreshLayout.OnRefreshListener interface and its onRefresh() method.

```
mySwipeRefreshLayout.setOnRefreshListener(
    new SwipeRefreshLayout.OnRefreshListener() {
        @Override
        public void onRefresh() {

            /* This method performs the actual data-refresh operation.
               The method calls setRefreshing(false) when it's finished */

            MyUpdateOperation();
        }
    }
);
```

Swipe-to-Refresh – Update the ListView

```
private void MyUpdateOperation()
{
   for(int j = 0; j < 10; j++, i++)
     objects.add("text " +i);       // objects is an ArrayList

  ArrayAdapter<String> arrayAdapter= new ArrayAdapter<String>(this,
android.R.layout.simple_list_item_1, objects);

                listView.setAdapter(arrayAdapter);

                Toast.makeText(getApplicationContext(), "refreshed",
                Toast.LENGTH_LONG).show();


                mySwipeRefreshLayout.setRefreshing(false);
}
```

ImageView and Asset folder

- You can load an unknown number of images from assets folder (app\src\main\assets\...)

```
AssetManager assets = this.getAssets();
try {
    String [] images = assets.list("png");

    for(int i = 0; i < images.length; i++)
    {
        ImageView imageView = new ImageView(this);
        String imageName = "png/" + images[i];
        Drawable image =
       Drawable.createFromStream(assets.open(imageName), imageName);
        imageView.setImageDrawable(image);

        //addView to add the image
    }

} catch (IOException e) {
    e.printStackTrace();
}
```