

Lecture #2

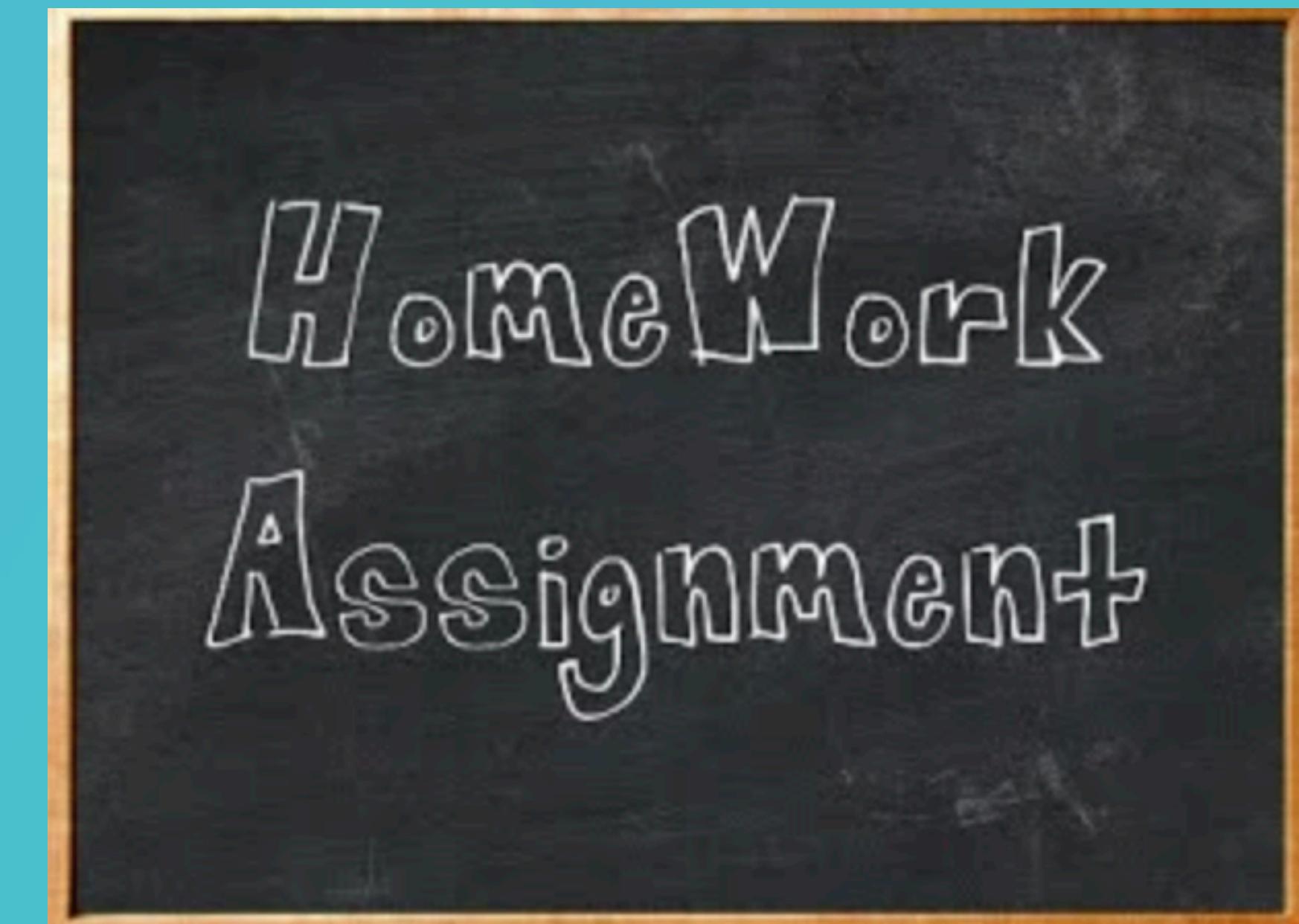
Lists and Rest

Resources

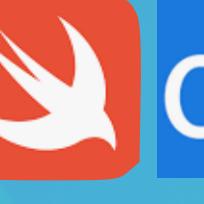
Mobile Applications
Fall 2023

Homework Assignments

- First assignment - Project details **Due: 2nd laboratory**
- One CRUD project
 - UI Only
 - In Native and Non-Native
Due: 3rd and 4th laboratory
 - DB
Due: 6th laboratory
 - NET
Due: last laboratory



CRUD Application

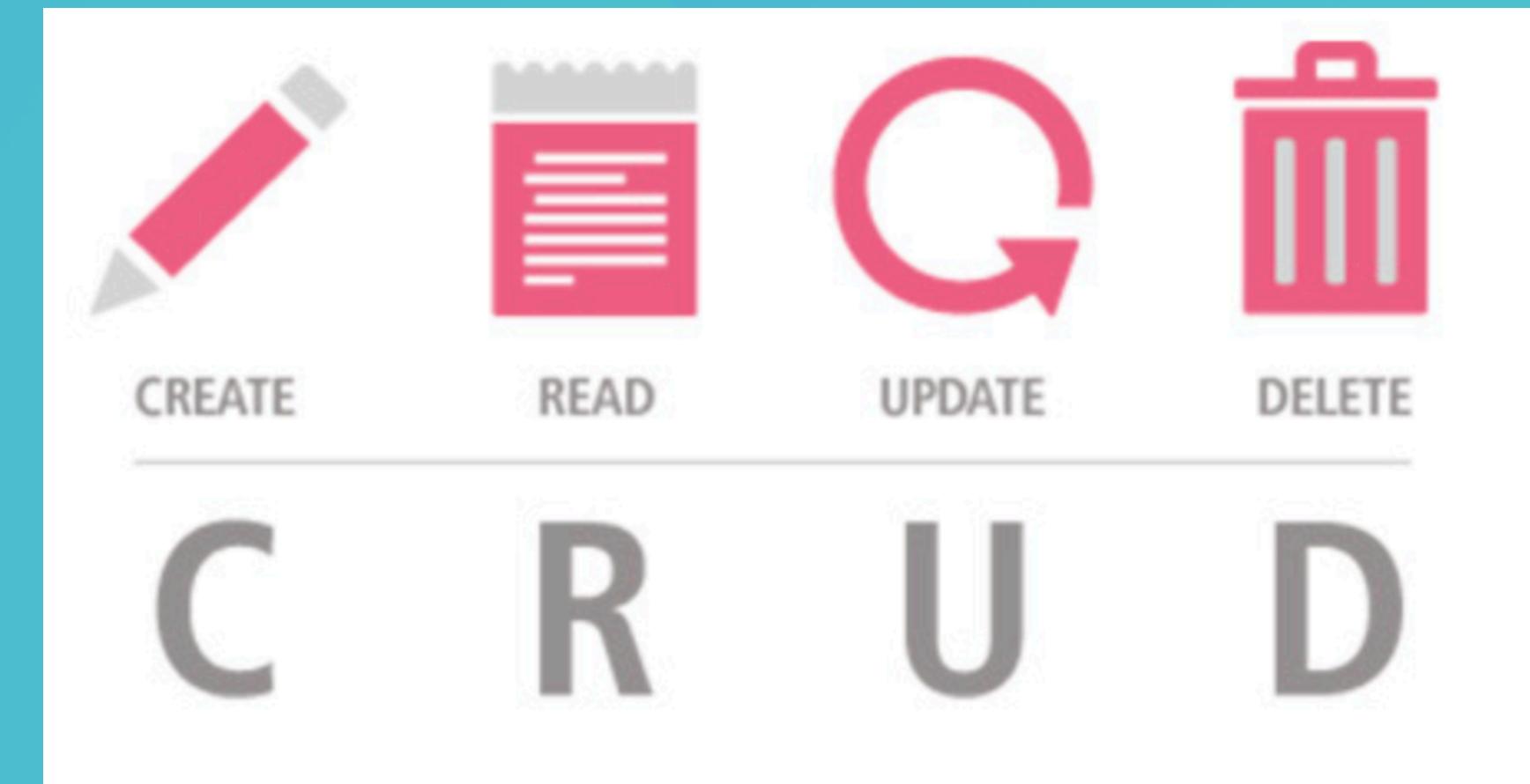
Native    OBJ-C

AND **UI Only**

Non-Native

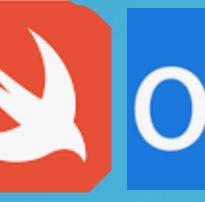


Other



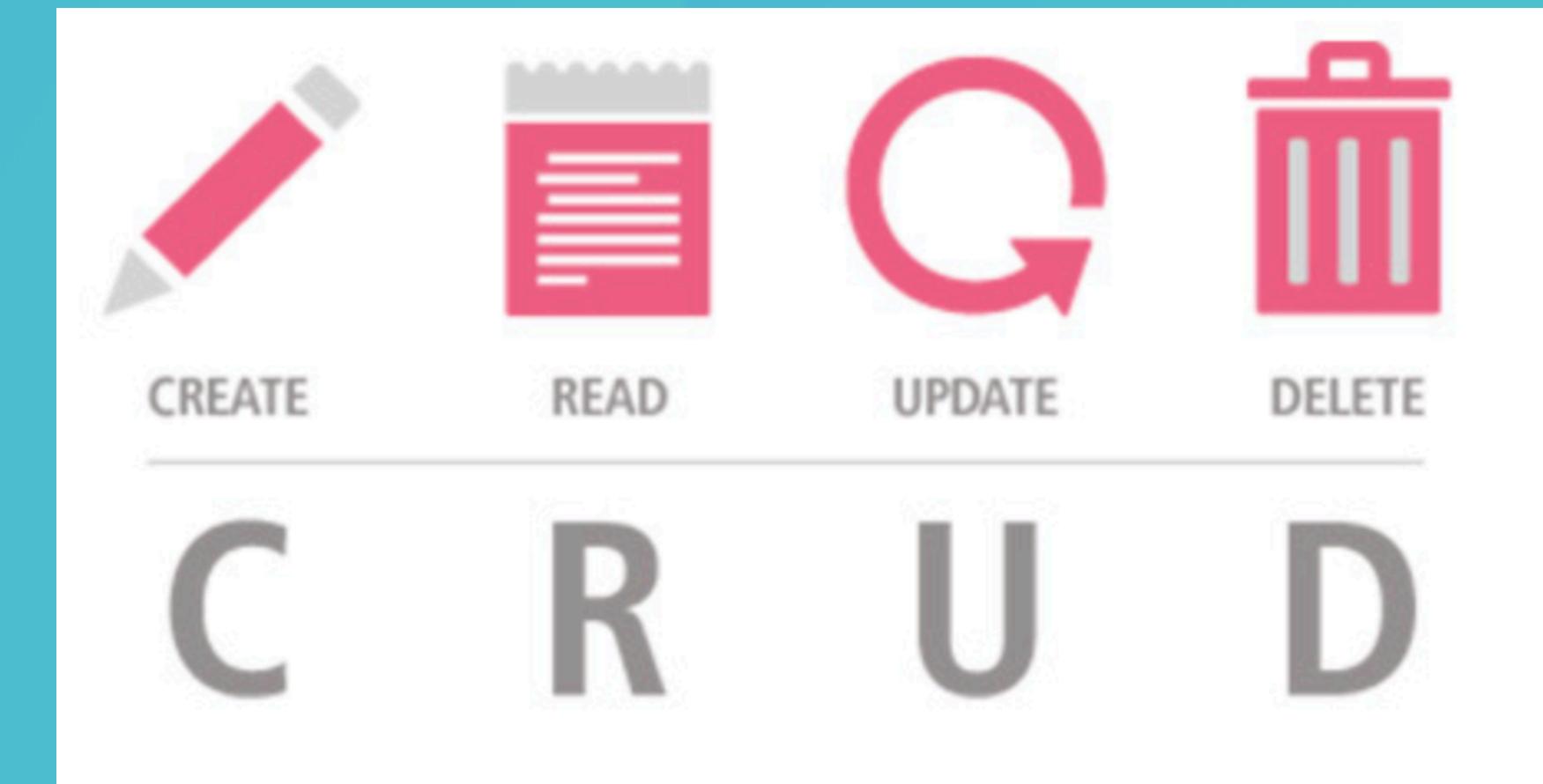
CRUD Application

DB, and .NET

Native    

Only one

Non-Native



<https://www.cs.ubbcluj.ro/~dan/ma/labPlan.html>

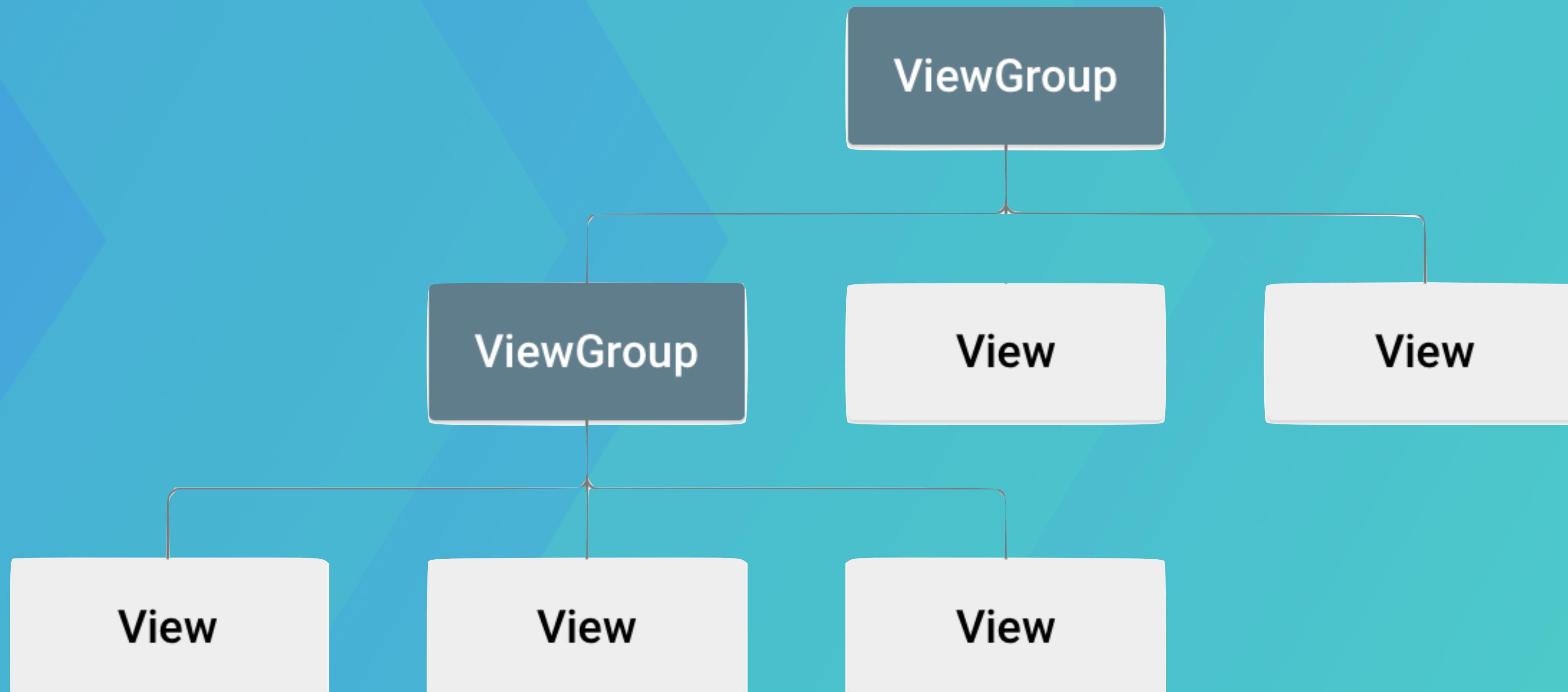
DEMO

Lifecycle



Layouts

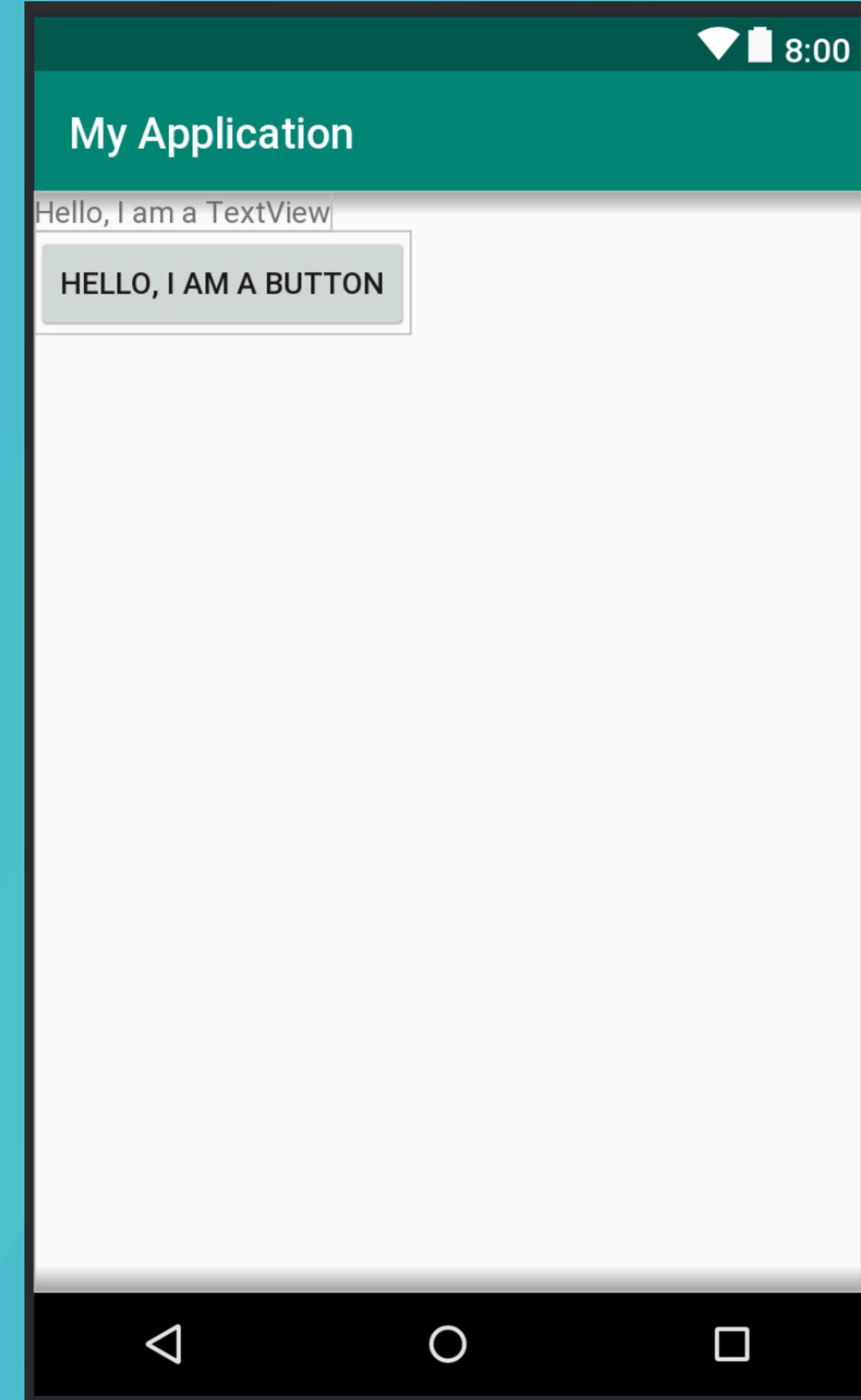
<https://developer.android.com/guide/topics/ui/declaring-layout>



XML

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://
schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">
    <TextView android:id="@+id/text"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello, I am a TextView"/>
    <Button android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello, I am a Button"/>
</LinearLayout>
```

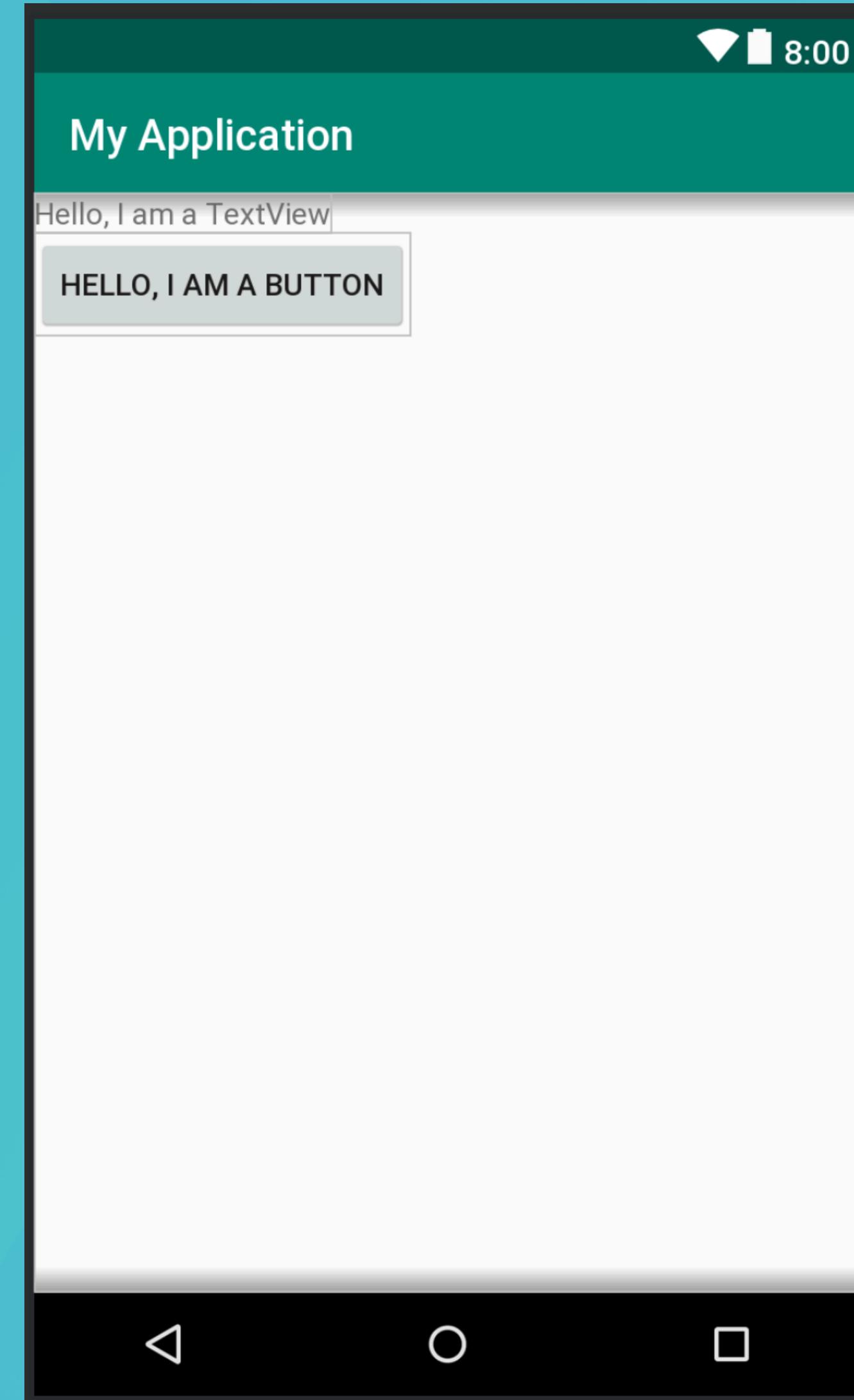
```
fun onCreate(savedInstanceState: Bundle) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.main_layout)
}
```



Accessing Assets

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://
schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">
    <TextView android:id="@+id/text"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello, I am a TextView"/>
    <Button android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello, I am a Button"/>
</LinearLayout>
```

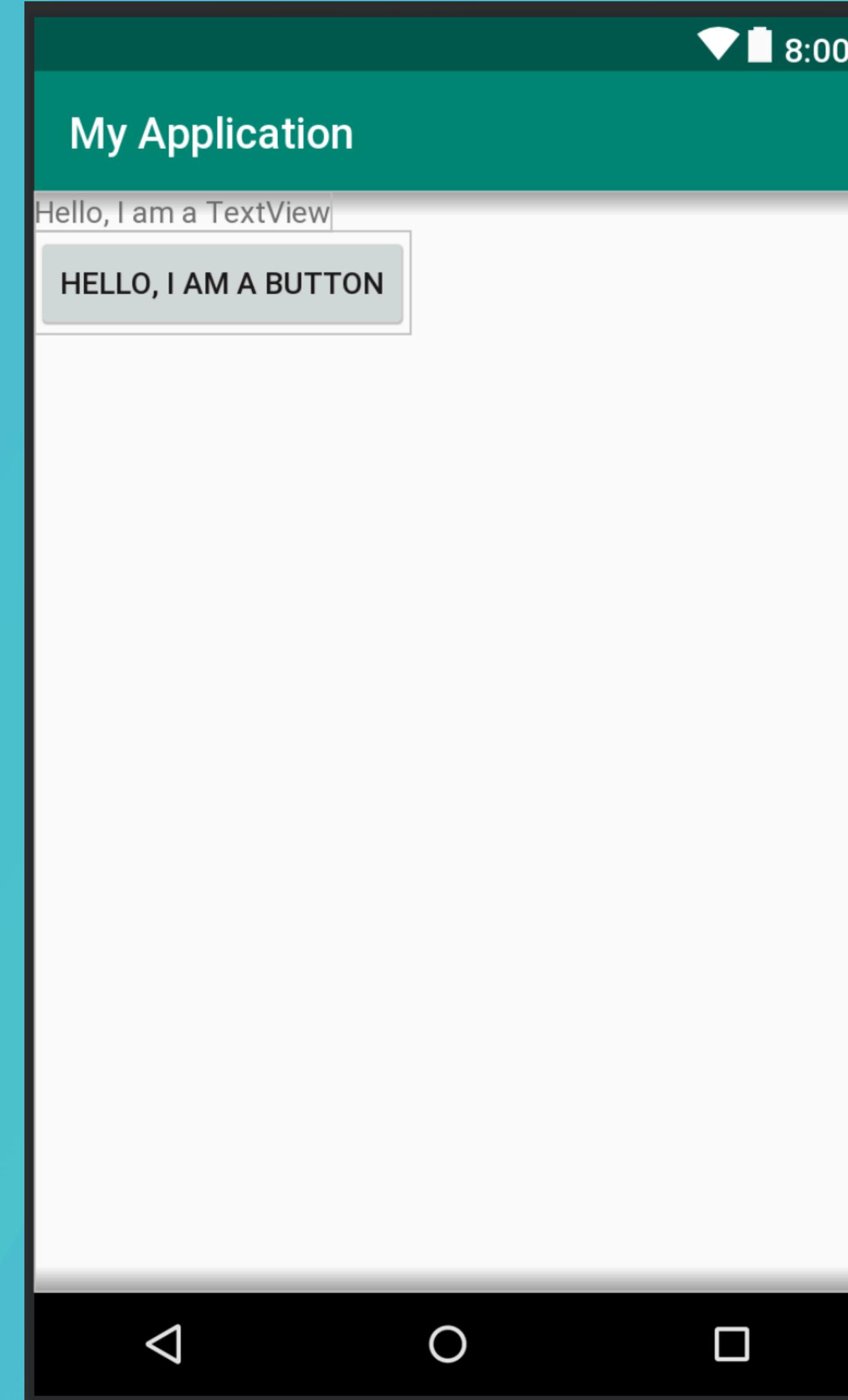
```
fun onCreate(savedInstanceState: Bundle) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.main_layout)
} val myButton: Button = findViewById(R.id.button)
}
```



Add Event Handler

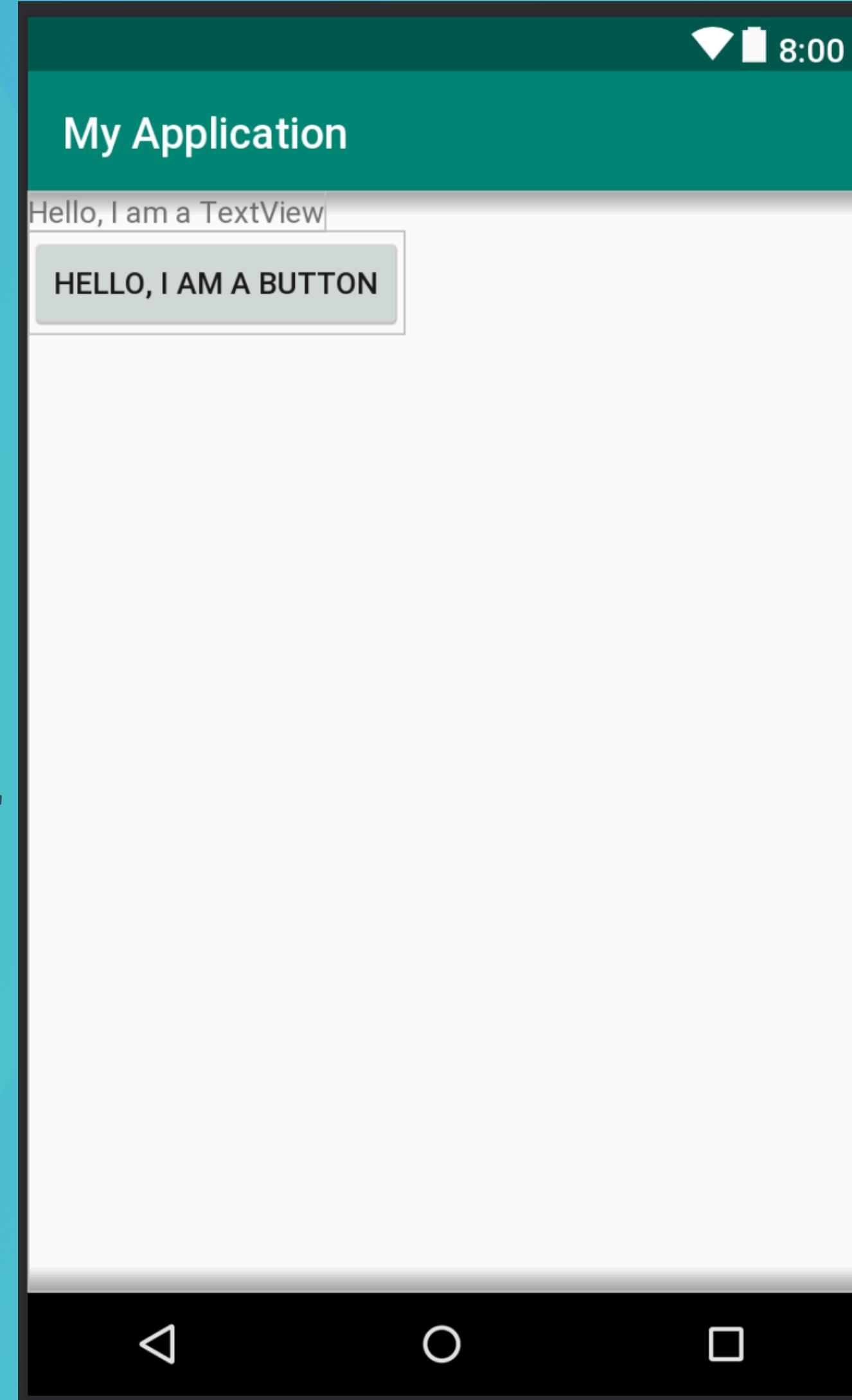
```
...  
    <Button android:id="@+id/button"  
            android:layout_width="wrap_content"  
            android:layout_height="wrap_content"  
            android:text="Hello, I am a Button"  
            android:onClick="sendMessage"  
    />  
...  
    ...
```

```
fun onCreate(savedInstanceState: Bundle) {  
    super.onCreate(savedInstanceState)  
    setContentView(R.layout.main_layout)  
    val myButton: Button = findViewById(R.id.button)  
}  
fun sendMessage(view: View) {  
    logd("Ready!")  
}
```



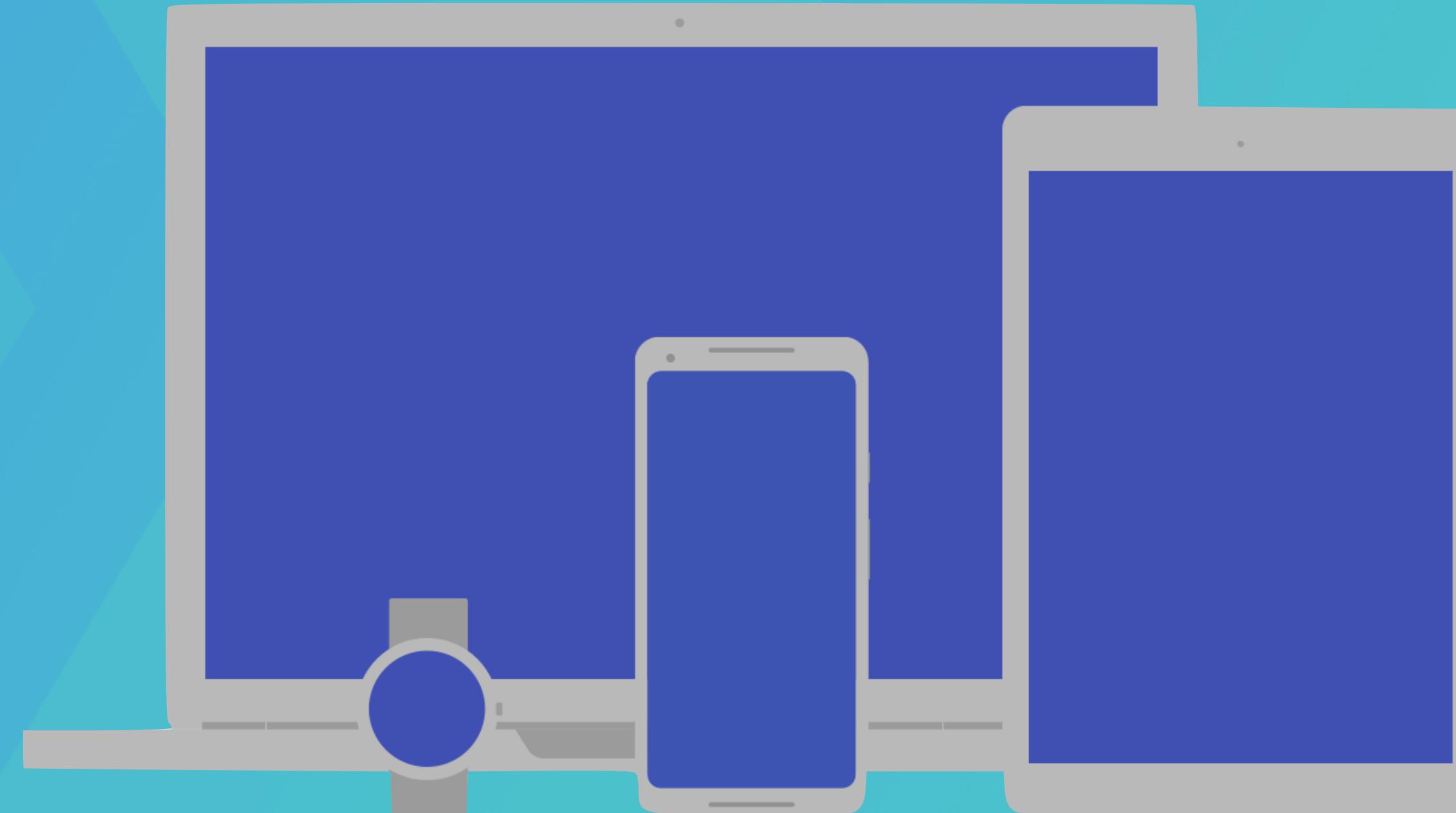
Using Android KTX

```
import kotlinx.android.synthetic.main.activity_main.*  
...  
    <Button android:id="@+id/button"  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        android:text="Hello, I am a Button"  
        android:onClick="sendMessage"  
    />  
...  
  
fun onCreate(savedInstanceState: Bundle) {  
    super.onCreate(savedInstanceState)  
    setContentView(R.layout.main_layout)  
    val myButton: Button = findViewById(R.id.button)  
    myButton.setOnClickListener{  
        text.text = "From editText: ${editText.text.toString()}"  
    }  
    fun sendMessage(view: View) {  
        editText.text = update  
        logd("Ready!")  
    }  
}
```



Supporting different screen sizes

- Flexible layouts
- Alternative layouts
- Stretchable images
- Alternative bitmaps
- Vector graphics



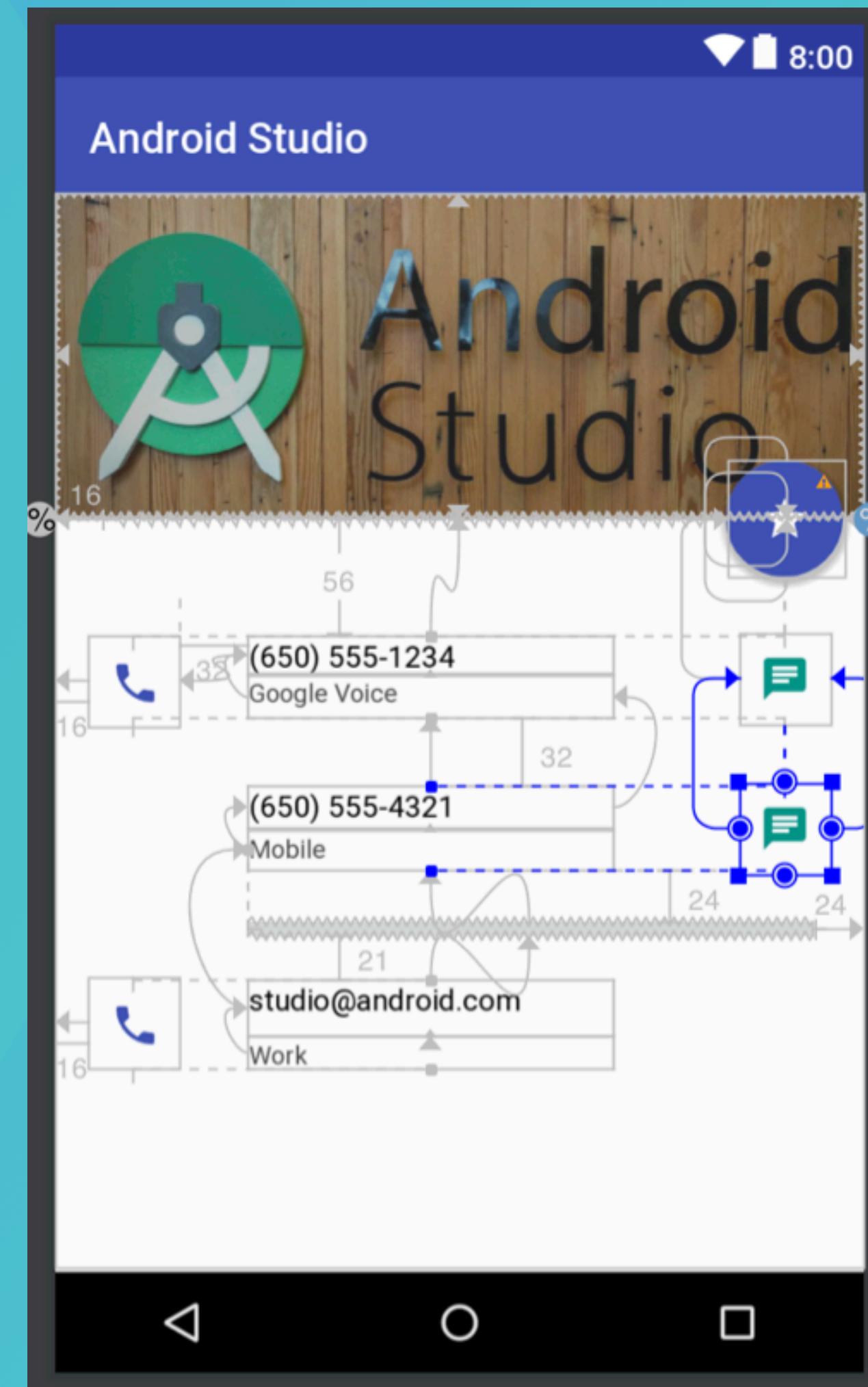
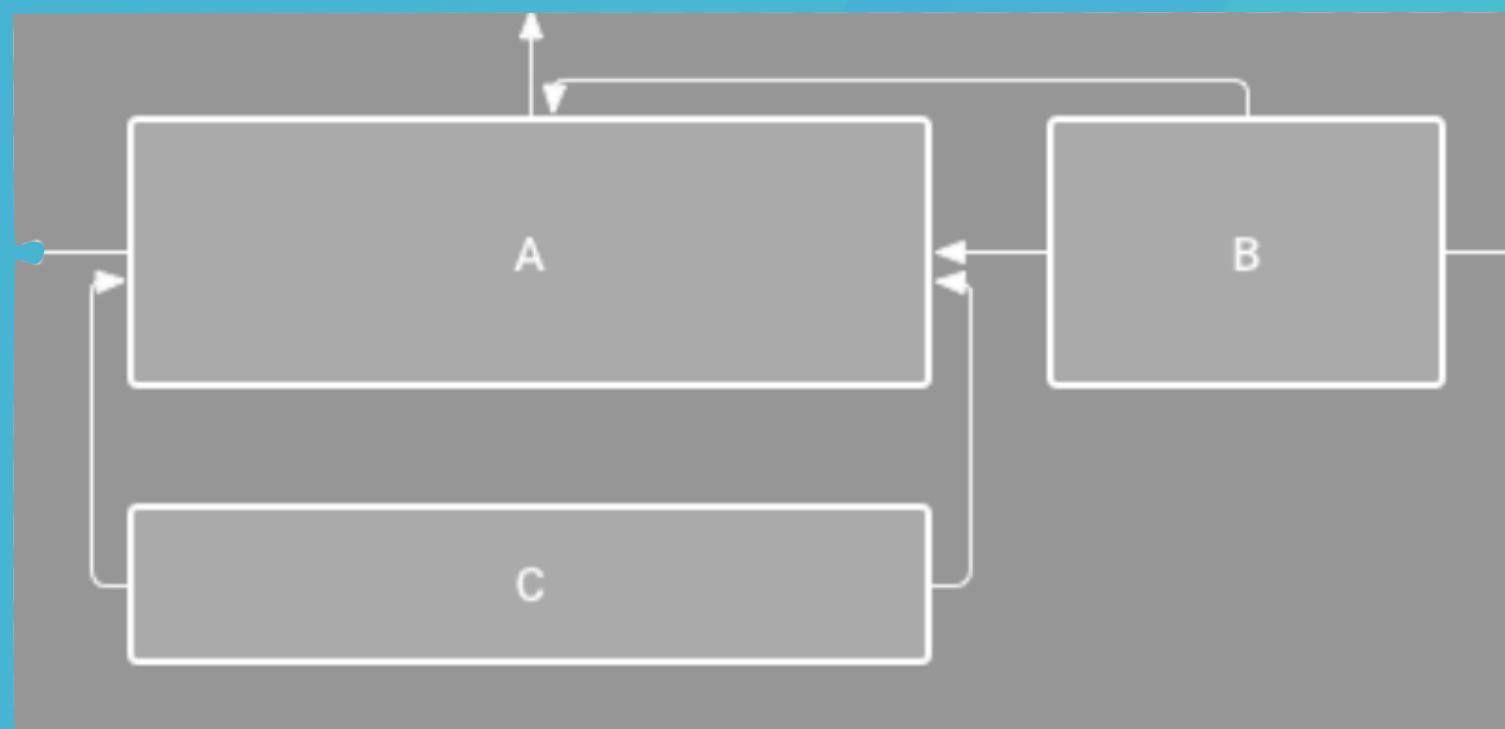
https://developer.android.com/guide/practices/screens_support

Flexible Layouts

ConstraintLayout

In module-level `gradle.build`:

```
repositories {  
    google()  
}  
  
dependencies {  
    implementation  
        'com.android.support.constraint:constraint-layout:2.1.4'  
}
```

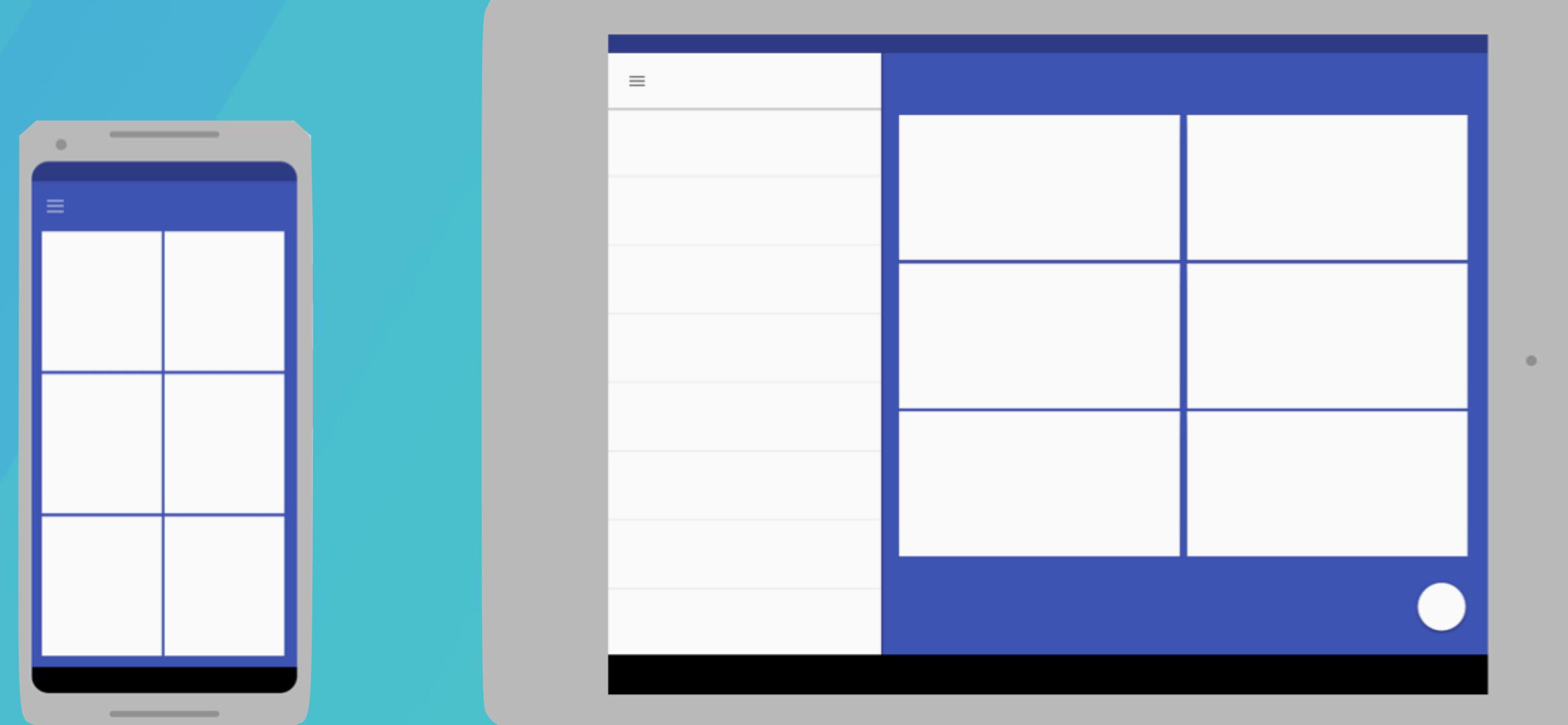


DEMO

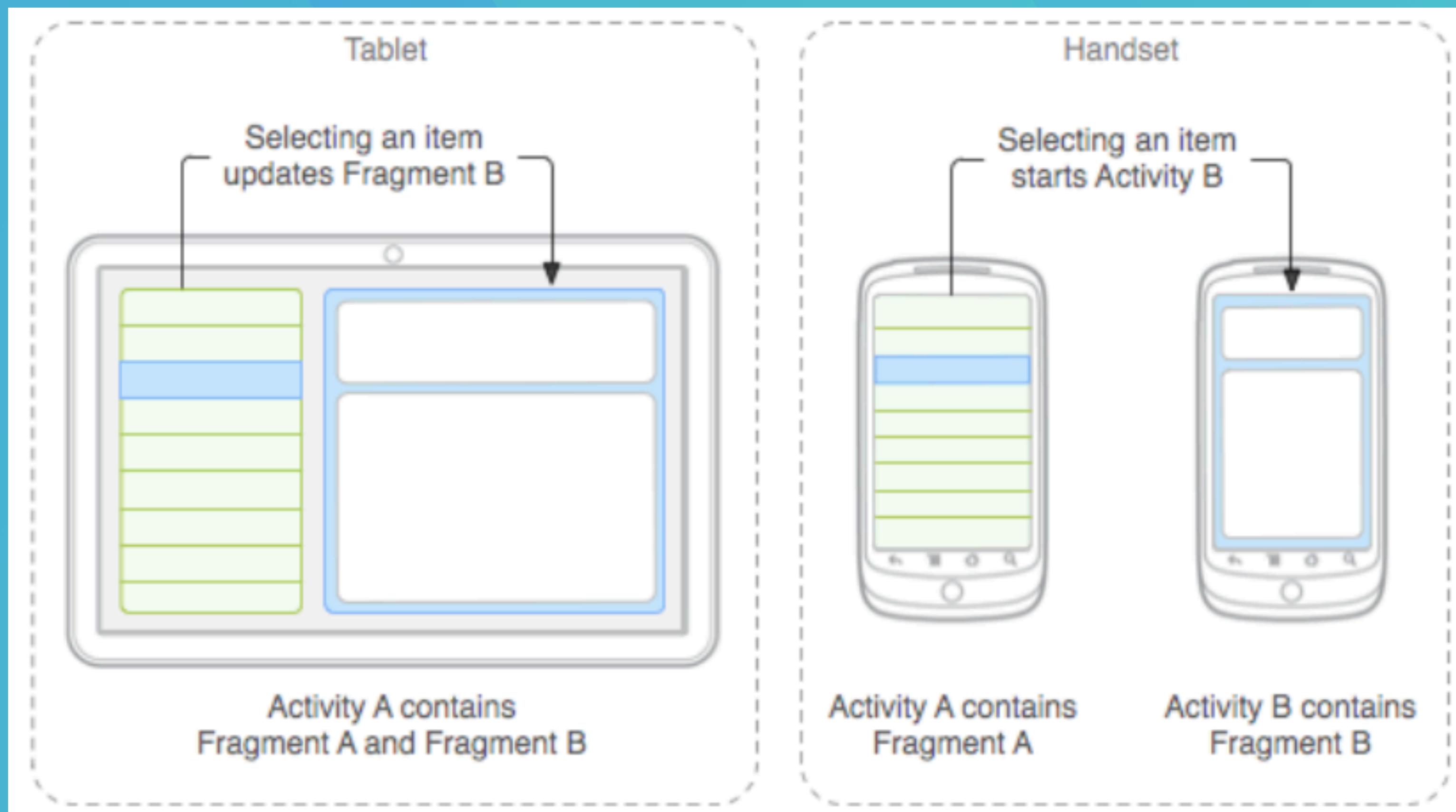
Alternative layouts

```
res/layout/main_activity.xml  
res/layout-land/main_activity.xml  
res/layout-sw600dp/main_activity.xml  
res/layout-sw600dp-land/main_activity.xml
```

```
# Default layout  
# When in landscape mode  
# For 7" tablets  
# For 7" tablets in landscape
```

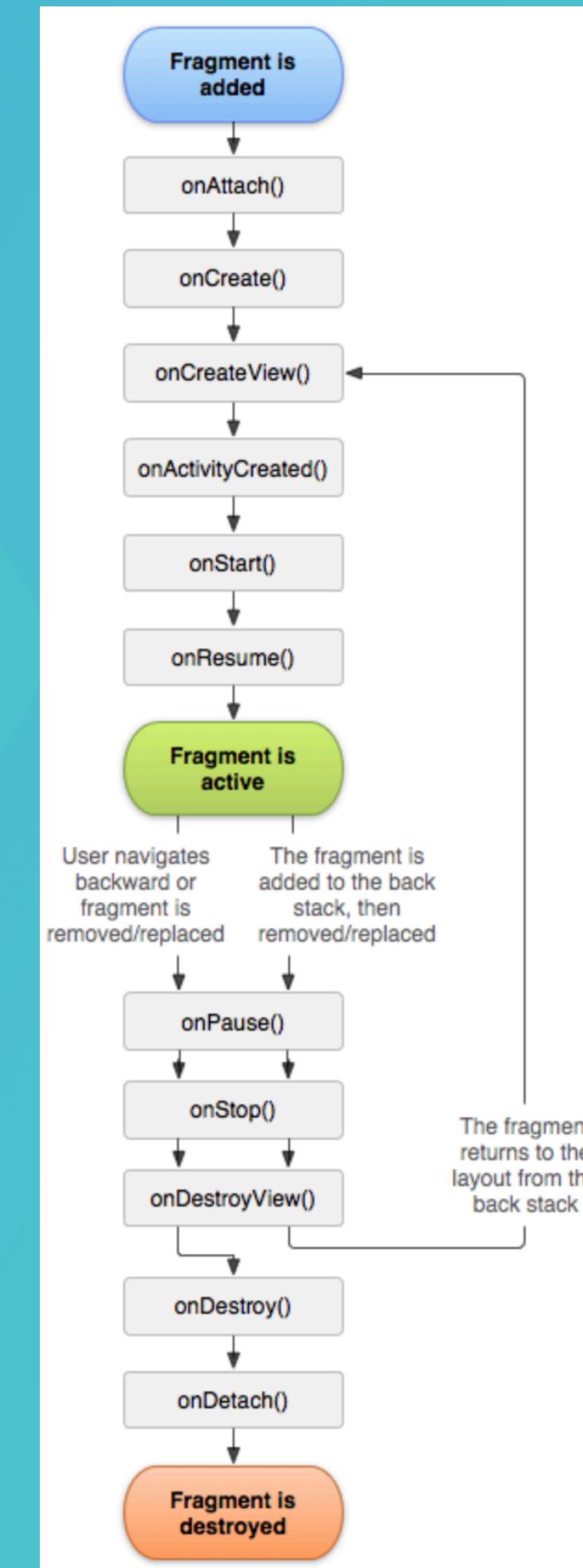


Building a Dynamic UI with Fragments



Creating a Fragment

- New callbacks
 - onAttach
 - onCreateView
 - onActivityCreated
 - onDestroyView
 - onDetach



Creating a Fragment

```
class ArticleListFragment : Fragment() {  
  
    override fun onCreateView(  
        inflater: LayoutInflater,  
        container: ViewGroup?  
        ) <-- Declare the fragment inside the activity's layout file.  
<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
    android:orientation="horizontal"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent">  
    <fragment android:name="com.example.news.ArticleListFragment"  
        android:id="@+id/list"  
        android:layout_weight="1"  
        android:layout_width="0dp"  
        android:layout_height="match_parent" />  
    <fragment android:name="com.example.news.ArticleReaderFragment"  
        android:id="@+id/viewer"  
        android:layout_weight="2"  
        android:layout_width="0dp"  
        android:layout_height="match_parent" />  
    </LinearLayout>
```

DEMO

Creating a Fragment

```
class ArticleListFragment : Fragment() {  
  
    override fun onCreateView(  
        inflater: LayoutInflater,  
        container: ViewGroup?,  
        savedInstanceState: Bundle?  
    ): View {  
        // Inflate the layout for this fragment  
        return inflater.inflate(R.layout.example_fragment, container, false)  
    }  
}  
Or, programmatically add the fragment to an existing ViewGroup  
  
val fragmentManager = supportFragmentManager  
val fragmentTransaction = fragmentManager.beginTransaction()  
  
val fragment = ArticleListFragment()  
fragmentTransaction.add(R.id.fragment_container, fragment)  
fragmentTransaction.commit()
```

Density-independent pixels



Density-independent pixels

```
<Button android:layout_width="wrap_content"  
       android:layout_height="wrap_content"  
       android:text="@string/clickme"  
       android:layout_marginTop="20dp" />
```

```
<TextView android:layout_width="match_parent"  
         android:layout_height="wrap_content"  
         android:textSize="20sp" />
```

dp units to pixel units

medium-density screen
"baseline" density



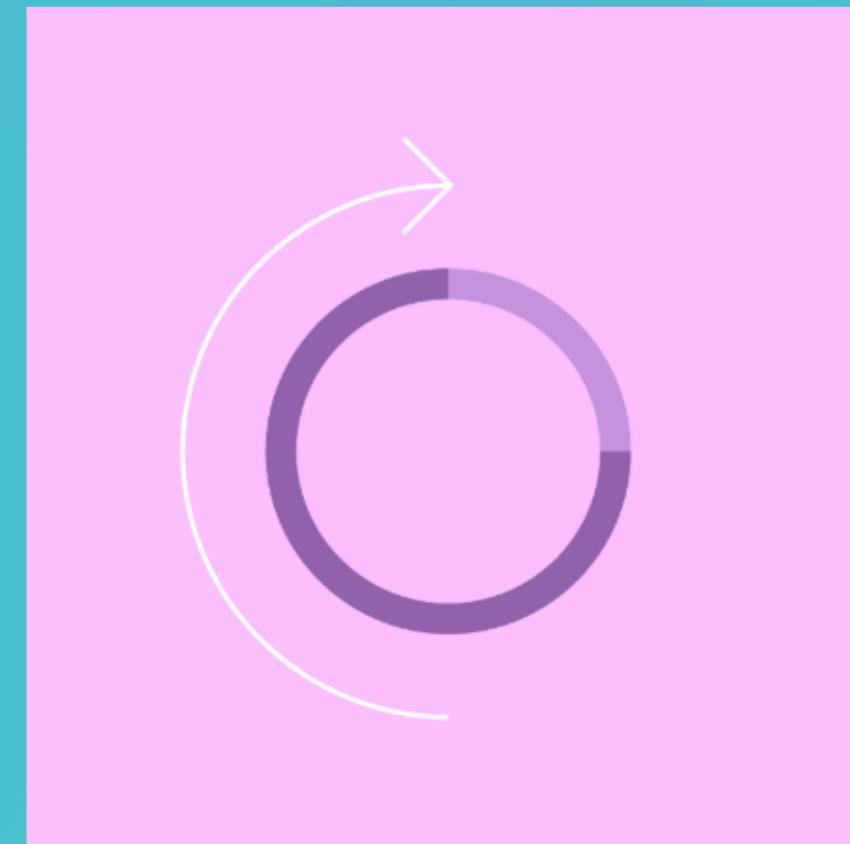
$$px = dp * (dpi / 160)$$



Actual device pixels

Progress Indicators

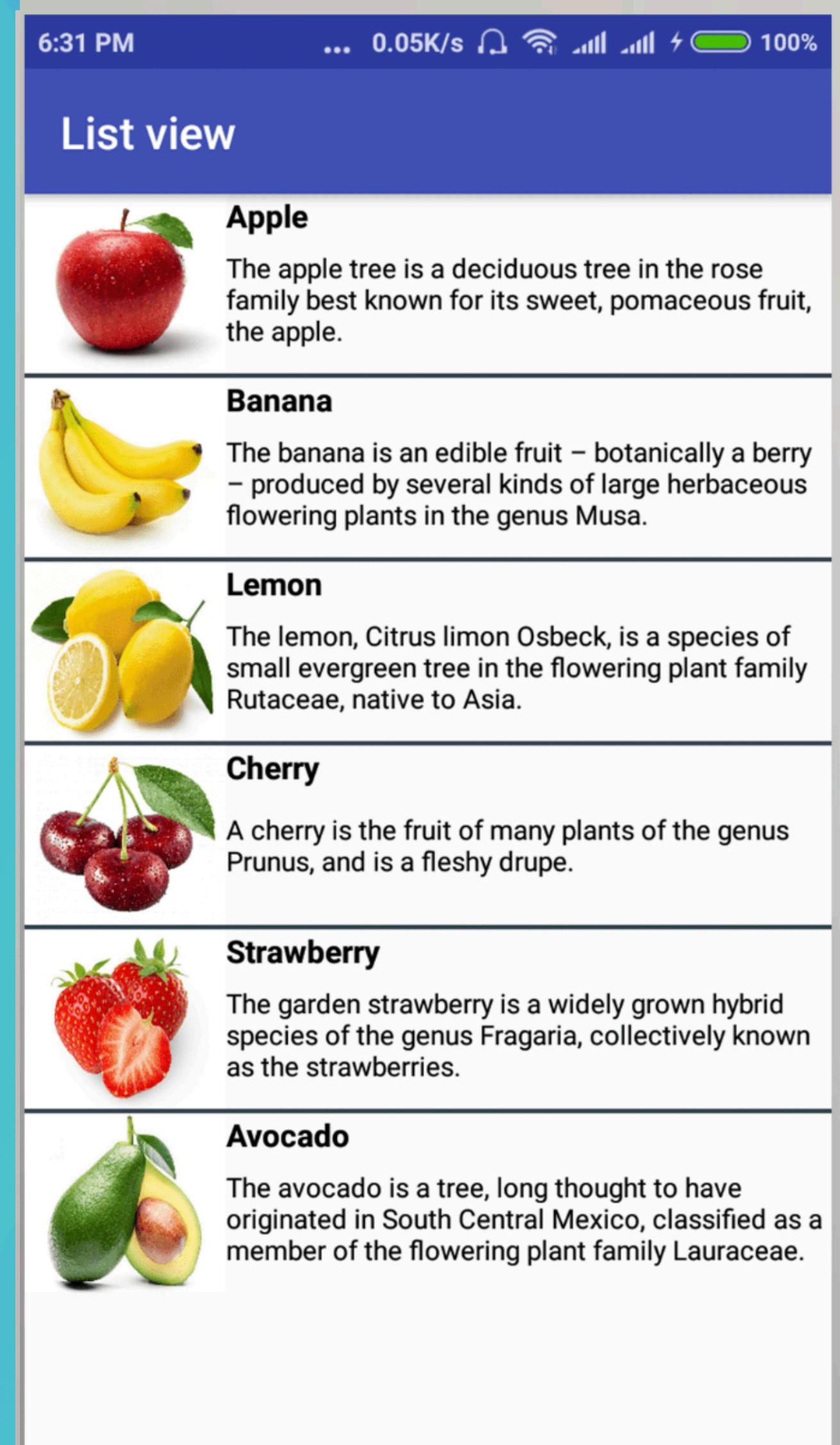
```
• ProgressBar  
<ProgressBar  
    android:id="@+id/indicator"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:visibility="gone"  
/>  
• RatingBar  
• SeekBar  
  
//before starting the action  
indicator.setVisibility = View.VISIBLE  
  
//when the action is done  
indicator.setVisibility = View.GONE
```



<https://developer.android.com/reference/android/widget/ProgressBar>

Lists

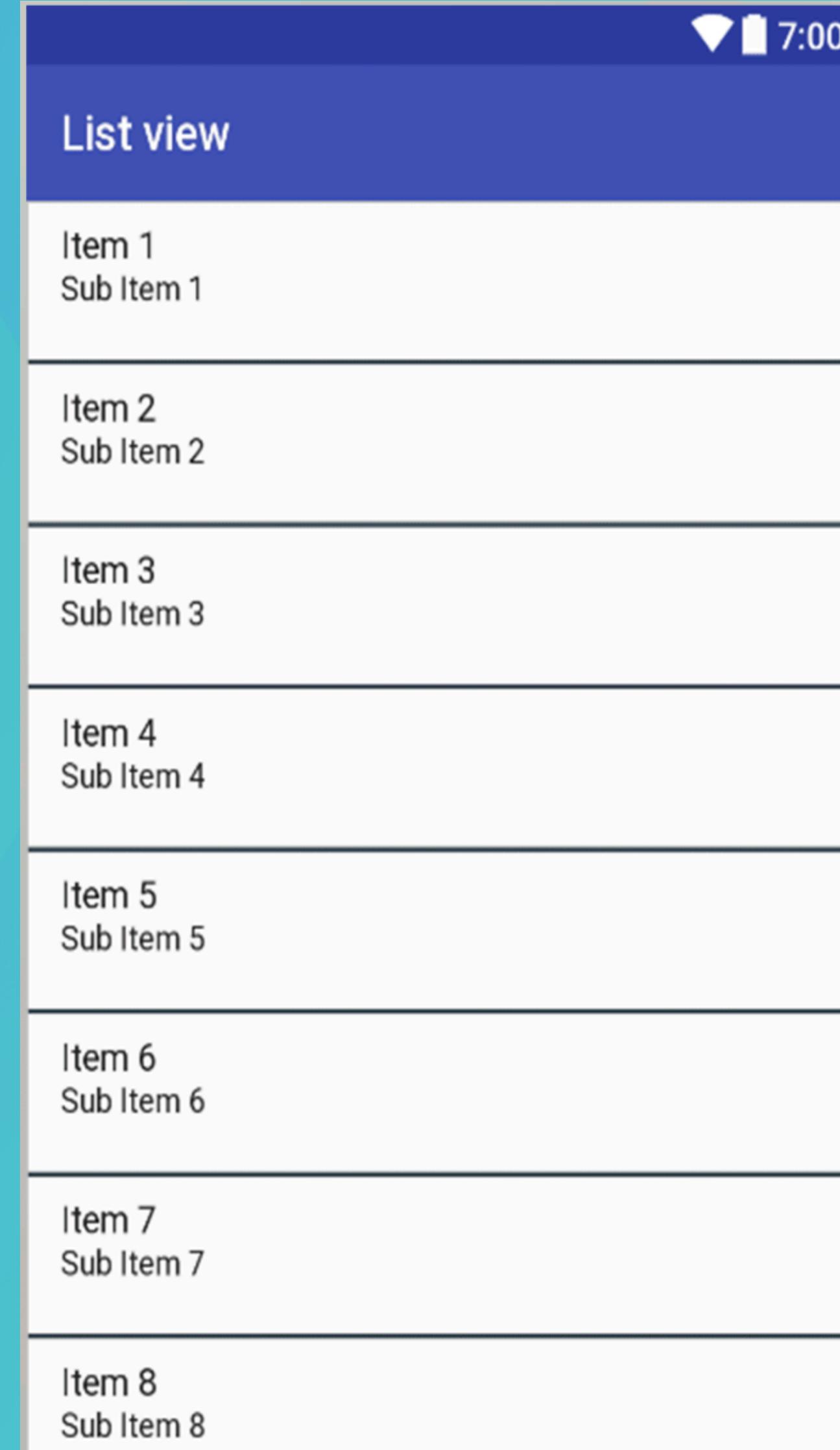
- ListView
- RecyclerView



ListView

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="8dp"
    android:paddingRight="8dp">
    <ListView android:id="@+id/list"
        android:layout_width="match_parent"
        android:layout_height="match_parent"/>
</LinearLayout>

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical">
    <TextView android:id="@+id/text"
        android:textSize="16sp"
        android:textStyle="bold"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"/>
    <TextView android:id="@+id/subText"
        ...
</LinearLayout>
```



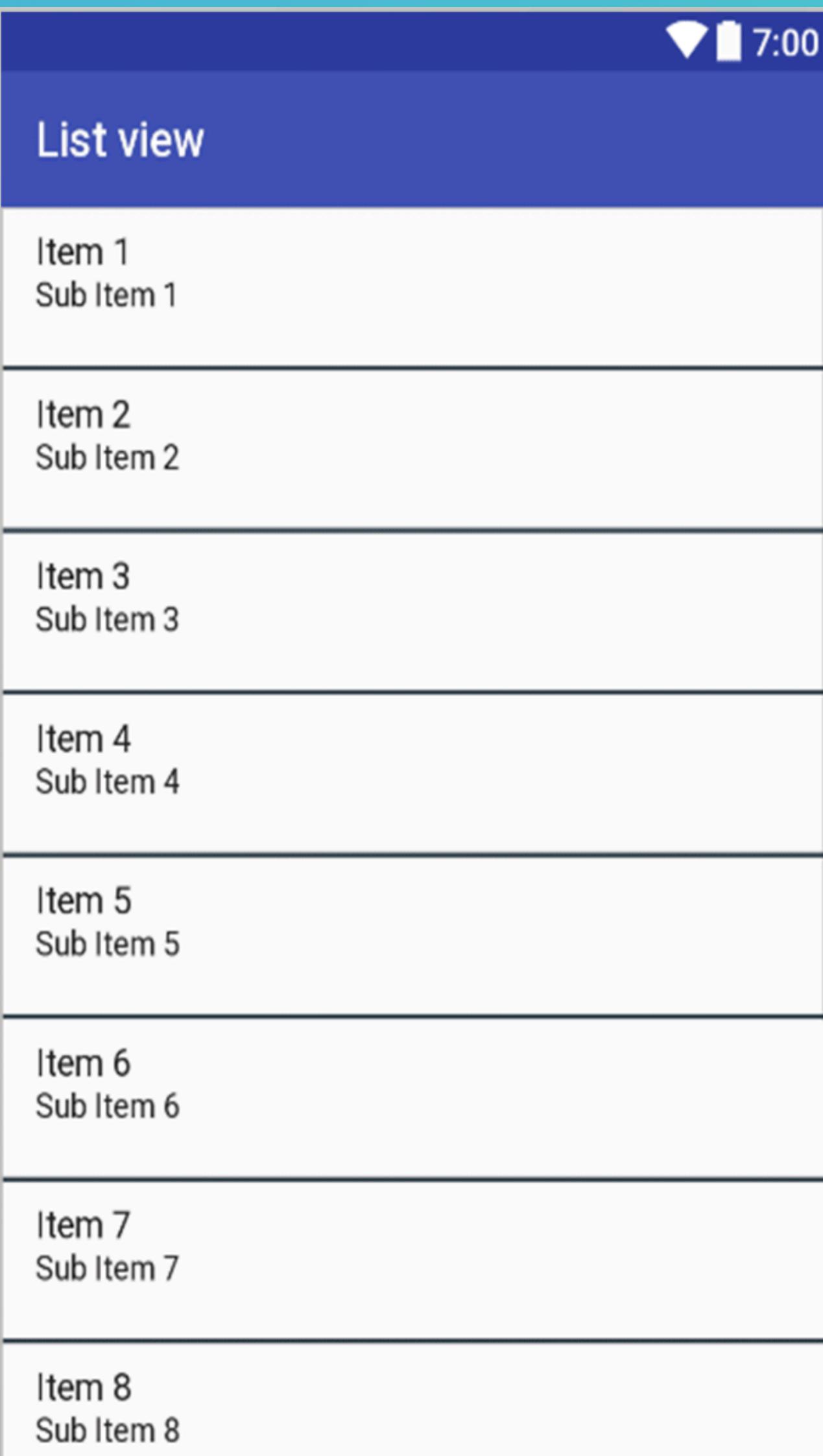
DEMO

ListView

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="..."
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="8dp"
    android:paddingRight="8dp">
    <ListView android:id="@+id/myList"
        android:layout_width="match_parent"
        android:layout_height="match_parent"/>
</LinearLayout>

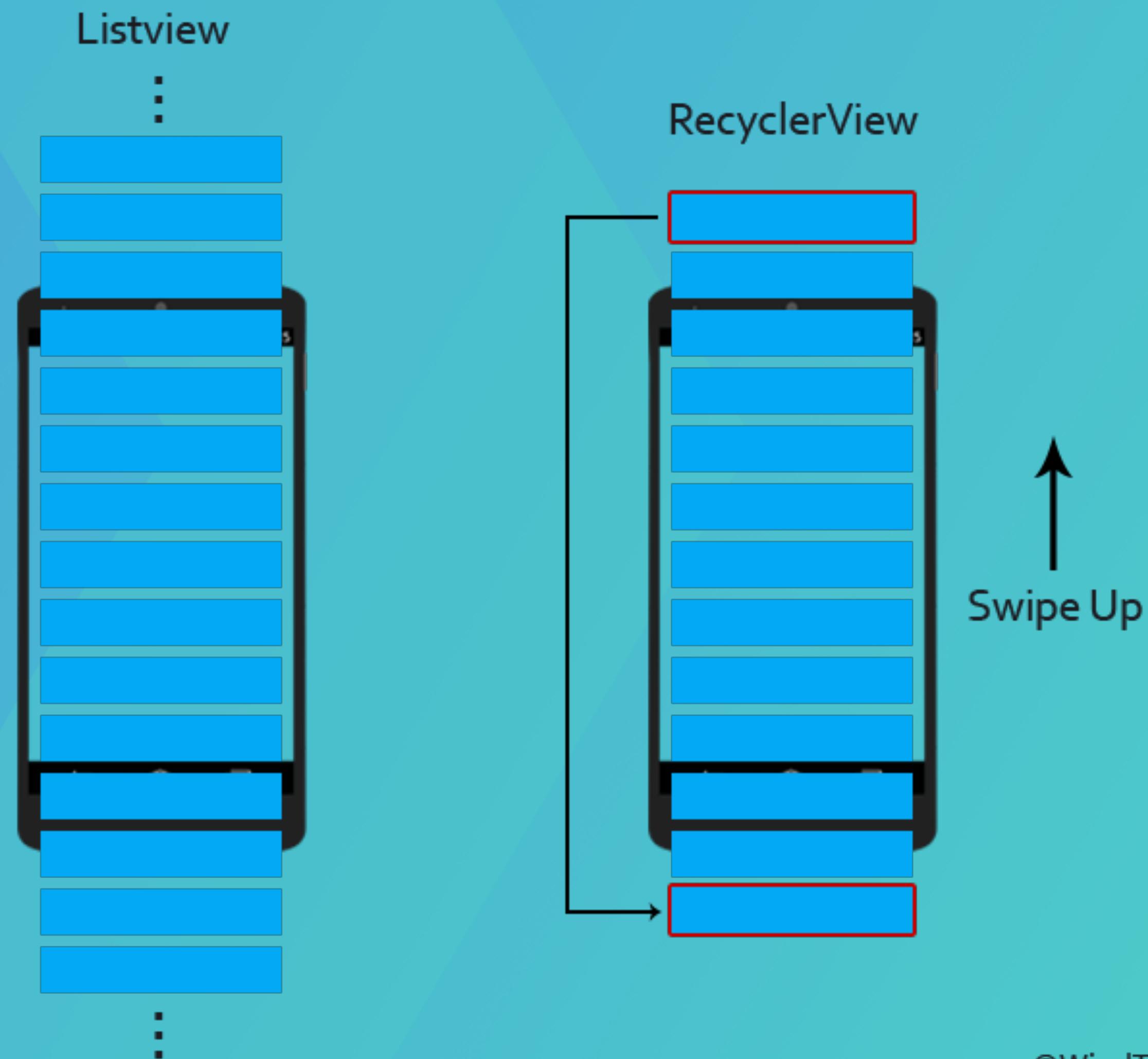
val arrayAdapter = ArrayAdapter<String>(this,
    android.R.layout.simple_list_item_1, arrayList)

myList.adapter = arrayAdapter
```



RecyclerView

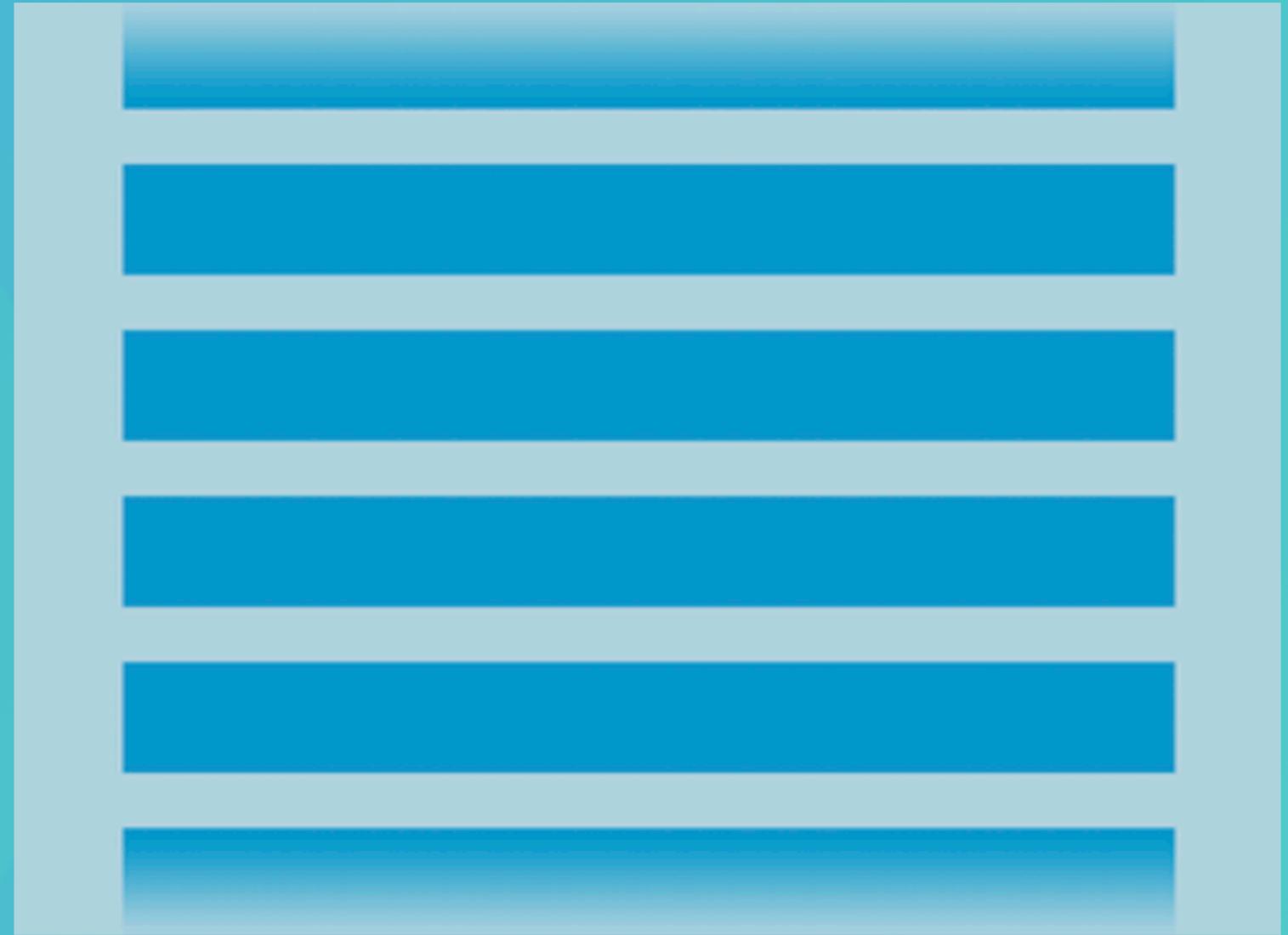
- ListView
- RecyclerView



RecyclerView

```
<?xml version="1.0" encoding="utf-8"?>
<!-- A RecyclerView with some commonly used attributes -->
<android.support.v7.widget.RecyclerView
    android:id="@+id/my_recycler_view"
    android:scrollbars="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"/>

class MyActivity : Activity() {
    private lateinit var recyclerView: RecyclerView
    private lateinit var viewAdapter: RecyclerView.Adapter<*>
    private lateinit var viewManager: RecyclerView.LayoutManager
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.my_activity)
        viewManager = LinearLayoutManager(this)
        viewAdapter = MyAdapter(myDataset)
        recyclerView = findViewById<RecyclerView>(R.id.my_recycler_view).apply {
            setHasFixedSize(true)
            layoutManager = viewManager
            adapter = viewAdapter
        }
    }
    // ...
}
```



DEMO

RecyclerView.Adapter

```
class MyAdapter(private val myDataset: Array<String>) :  
    RecyclerView.Adapter<MyAdapter.MyViewHolder>() {  
  
    class MyViewHolder(val textView: TextView) : RecyclerView.ViewHolder(textView)  
  
    override fun onCreateViewHolder(parent: ViewGroup,  
                                     viewType: Int): MyAdapter.MyViewHolder {  
        val textView = LayoutInflater.from(parent.context)  
            .inflate(R.layout.my_text_view, parent, false) as TextView  
        ...  
        return MyViewHolder(textView)  
    }  
  
    // Replace the contents of a view (invoked by the layout manager)  
    override fun onBindViewHolder(holder: MyViewHolder, position: Int) {  
        // - get element from your dataset at this position  
        // - replace the contents of the view with that element  
        holder.textView.text = myDataset[position]  
    }  
  
    override fun getItemCount() = myDataset.size  
}
```



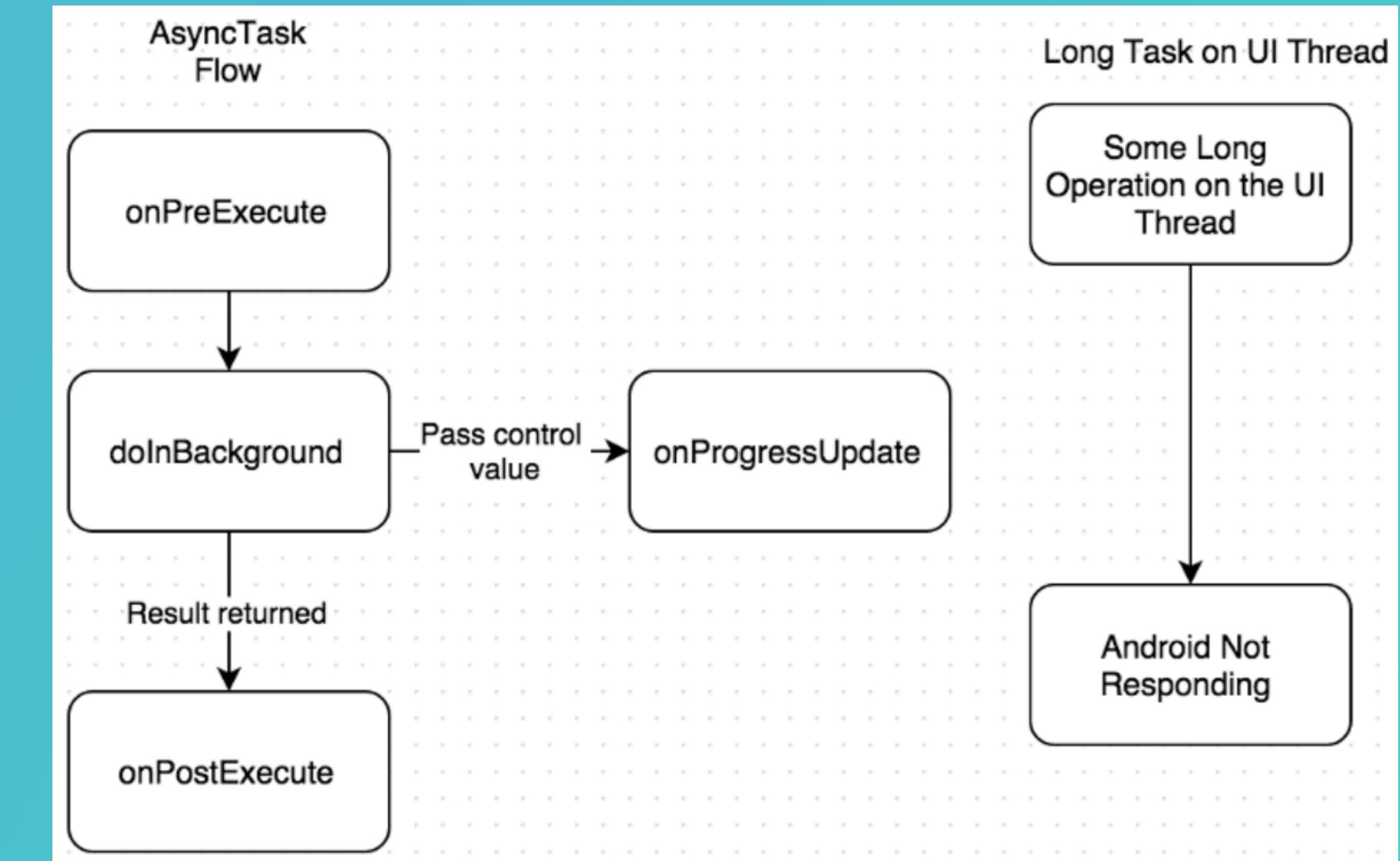
<https://developer.android.com/guide/topics/ui/layout/recyclerview>

AsyncTask

Deprecated

```
class SomeTask():  
    AsyncTask<Void, Int, String>() {  
  
        override fun doInBackground(  
            vararg params: Void?): String? {  
            // ...  
        }  
  
        override fun onPreExecute() {  
            super.onPreExecute()  
            // ...  
        }  
  
        override fun onPostExecute(  
            result: String?) {  
            super.onPostExecute(result)  
            // ...  
        }  
  
        override fun onProgressUpdate(  
            vararg values: Int){  
            super.onProgressUpdate(result)  
            // ...  
        }  
    }  
}
```

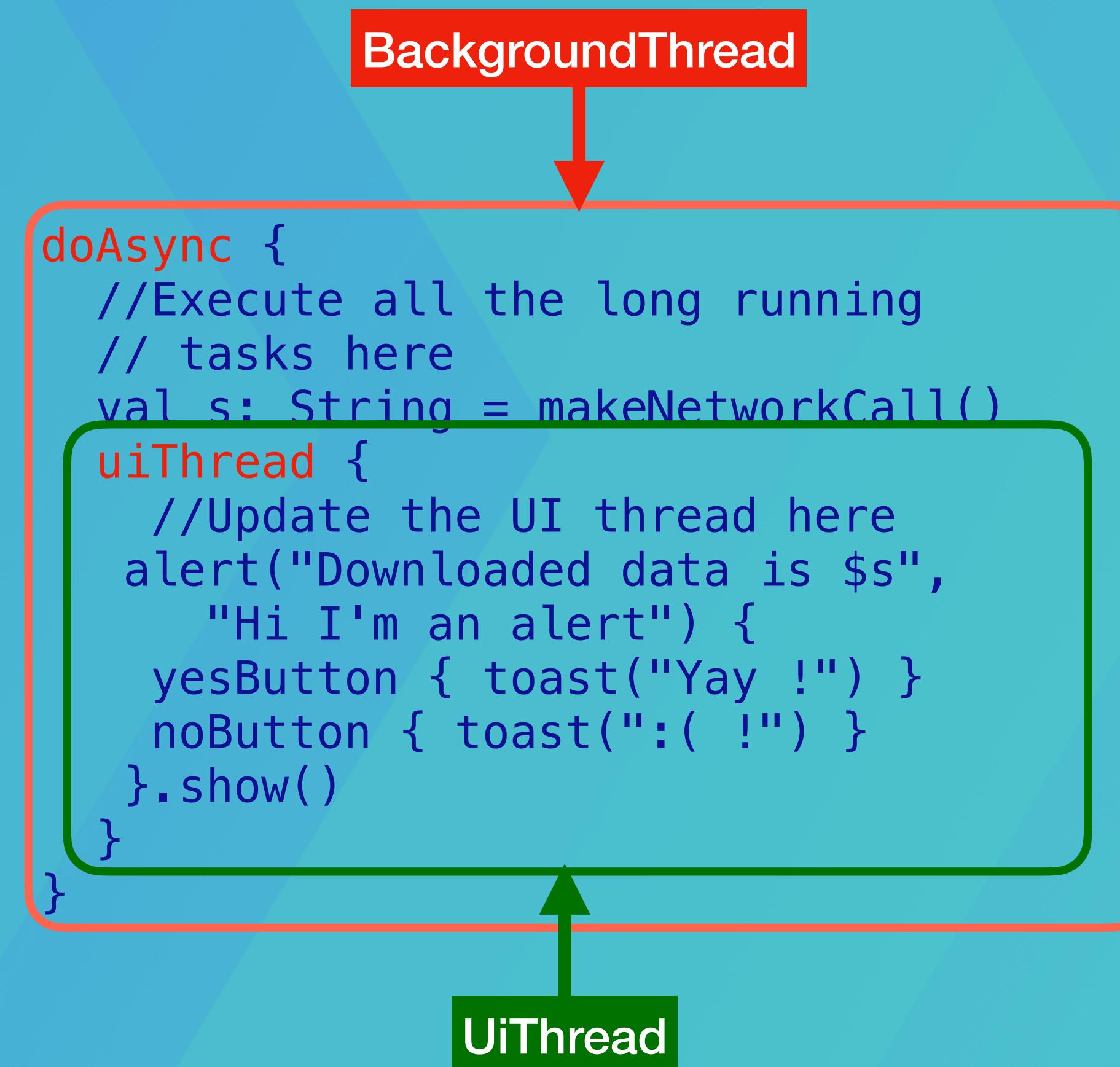
BackgroundThread



UiThread

Anko AsyncTask Alternative

DEMO



Jetpack Compose



<https://developer.android.com/jetpack/compose>

Compose Layout Basics



Compose transforms state into UI elements, via:

- Composition of elements
- Layout of elements
- Drawing of elements



Goals

- High performance
- Ability to easily write custom layouts



Basics

```
@Composable  
fun ArtistCard() {  
    Text("Alfred Sisley")  
    Text("3 minutes ago")  
}
```

Alfred Sisley



Basics

```
@Composable  
fun ArtistCard() {  
    Column {  
        Text("Alfred Sisley")  
        Text("3 minutes ago")  
    }  
}
```

Alfred Sisley
3 minutes ago





Basics

@Composable

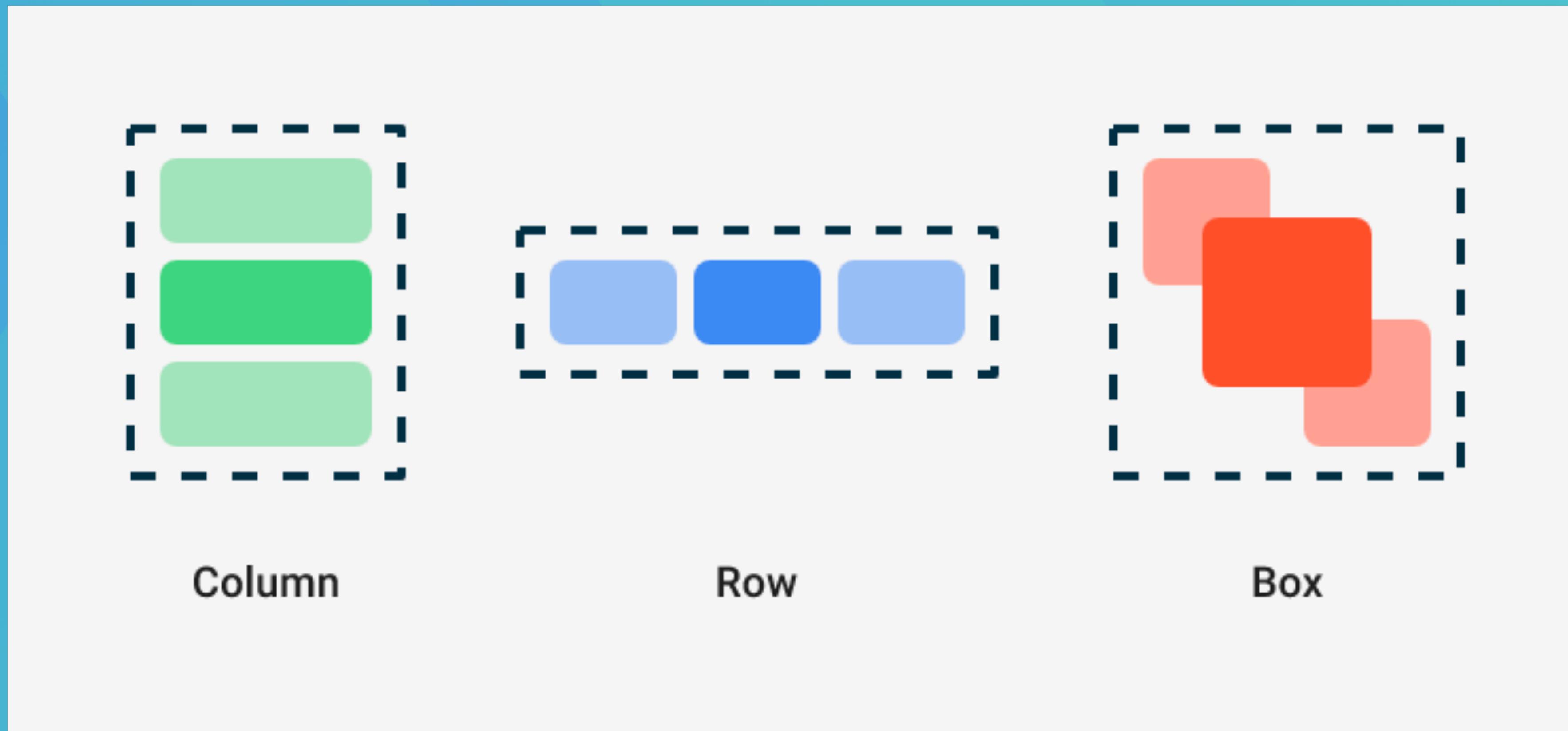
```
fun ArtistCard(artist: Artist) {  
    Row(verticalAlignment = Alignment.CenterVertically) {  
        Image(/*...*/)  
        Column {  
            Text(artist.name)  
            Text(artist.lastSeenOnline)  
        }  
    }  
}
```



Alfred Sisley

3 minutes ago

Basics



LazyLists

DEMO

```
import androidx.compose.foundation.lazy.items
```

```
@Composable
fun MessageList(messages: List<Message>) {
    LazyColumn {
        items(messages) { message ->
            MessageRow(message)
        }
    }
}
```



Constraint Layout

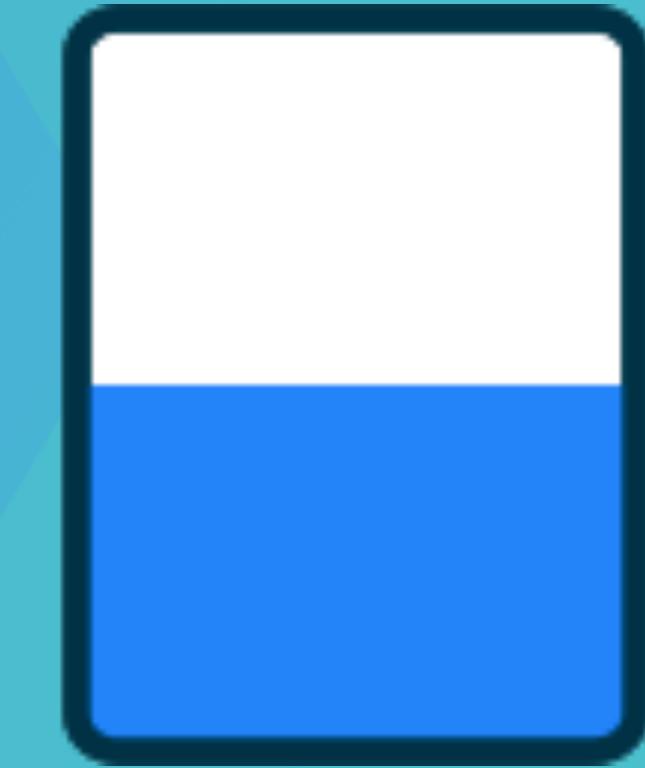
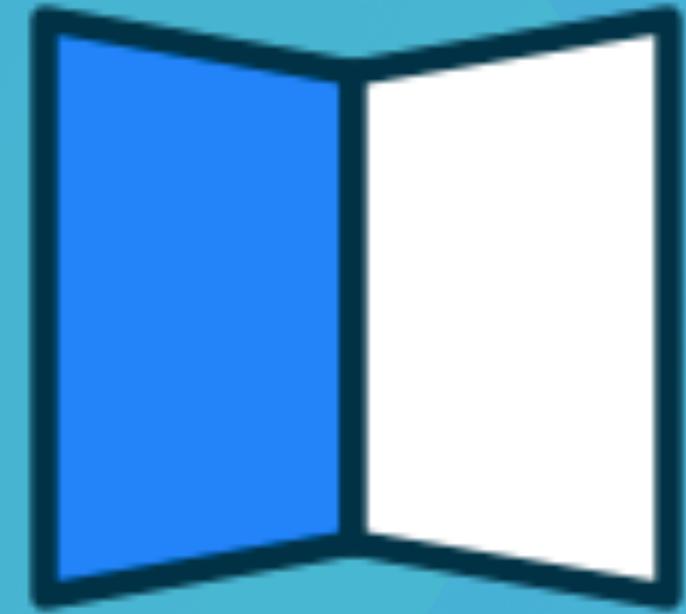
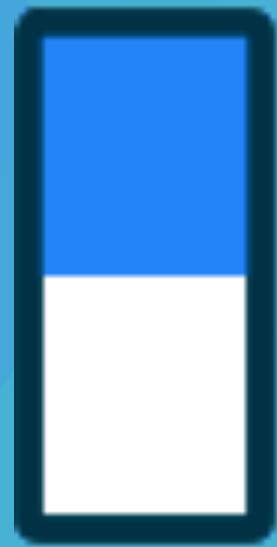
DEMO

```
@Composable
fun ConstraintLayoutContent() {
    ConstraintLayout {
        // Create references for the composables to constrain
        val (button, text) = createRefs()
        Button(
            onClick = { /* Do something */ },
            // Assign reference "button" to the Button composable
            // and constrain it to the top of the ConstraintLayout
            modifier = Modifier.constrainAs(button) {
                top.linkTo(parent.top, margin = 16.dp)
            }
        ) {
            Text("Button")
        }
        // Assign reference "text" to the Text composable
        // and constrain it to the bottom of the Button composable
        Text("Text", Modifier.constrainAs(text) {
            bottom.linkTo(button.bottom, margin = 16.dp)
        })
    }
}
```

<https://developer.android.com/jetpack/compose/layouts/constraintlayout>



Adaptive Layouts



Adaptive Layouts

```
enum class WindowSizeClass { Compact, Medium, Expanded }
```

```
@Composable
fun MyApp(windowSizeClass: WindowSizeClass) {
    // Perform logic on the size class to decide whether to show
    // the top app bar.
    val showTopAppBar = windowSizeClass != WindowSizeClass.Compact

    // MyScreen knows nothing about window sizes, and performs logic
    // based on a Boolean flag.
    MyScreen(
        showTopAppBar = showTopAppBar,
        /* ... */
    )
}
```





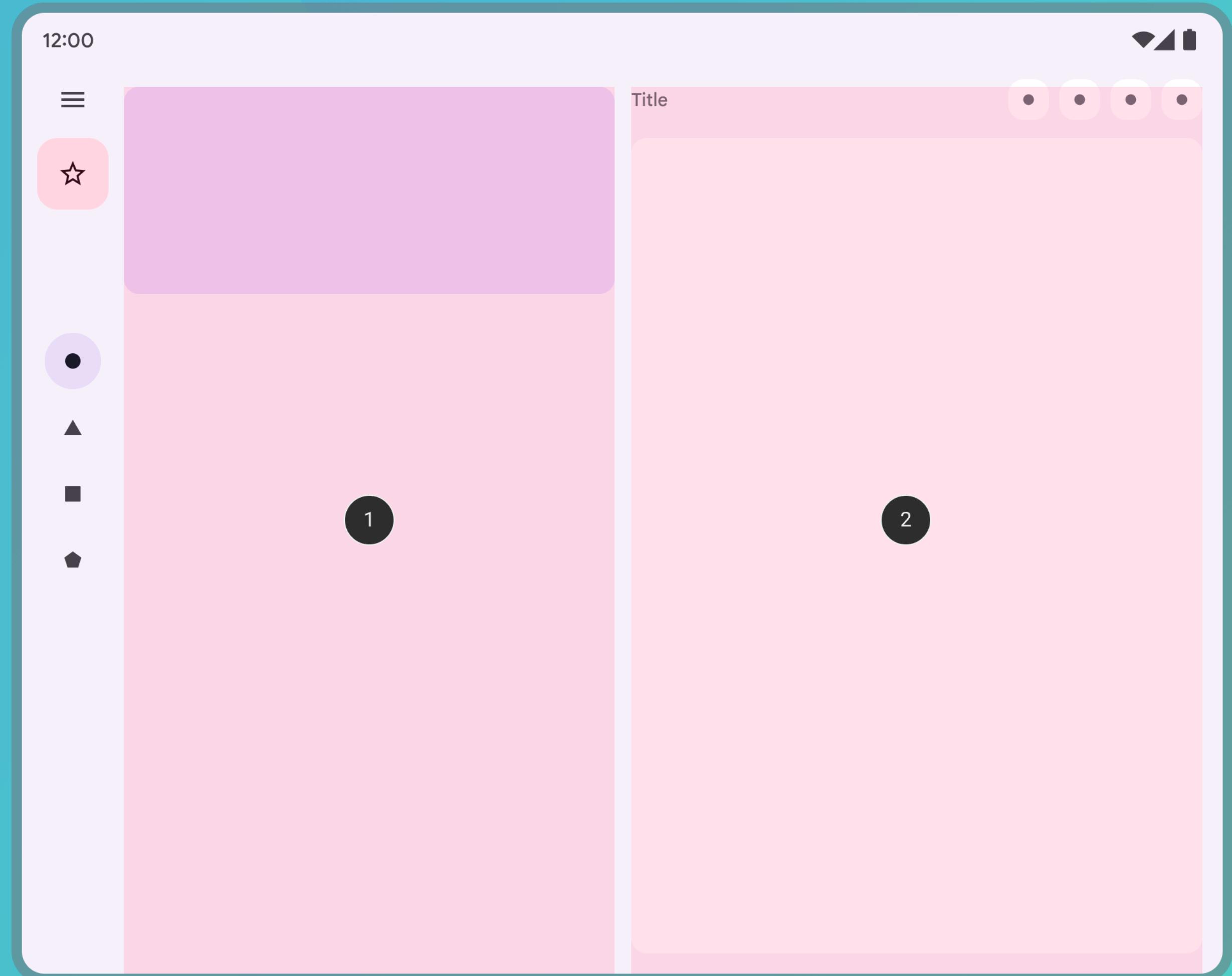
Flexible Layouts

```
@Composable  
fun AdaptivePane(  
    showOnePane: Boolean,  
    /* ... */  
) {  
    if (showOnePane) {  
        OnePane(/* ... */)  
    } else {  
        TwoPane(/* ... */)  
    }  
}
```

Flexible Layouts

@Composable

```
fun AdaptivePane(  
    showOnePane: Boolean,  
    /* ... */  
) {  
    if (showOnePane) {  
        OnePane(/* ... */)  
    } else {  
        TwoPane(/* ... */)  
    }  
}
```



Flexible Layouts

@Composable

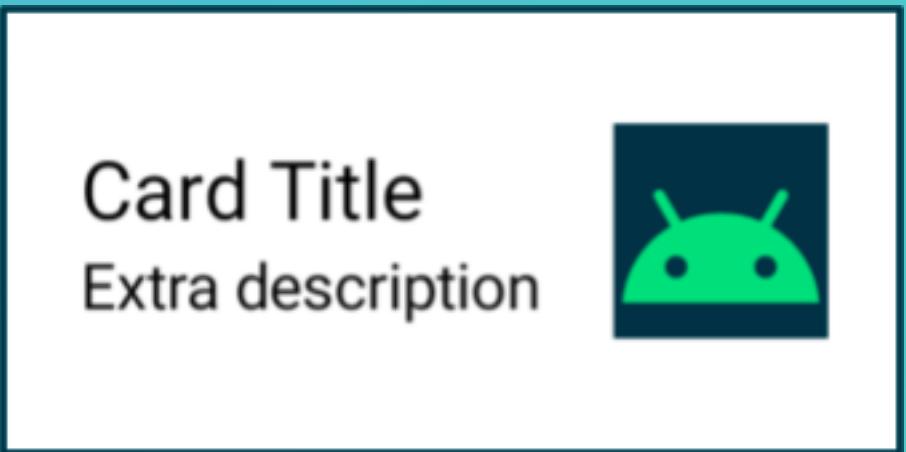
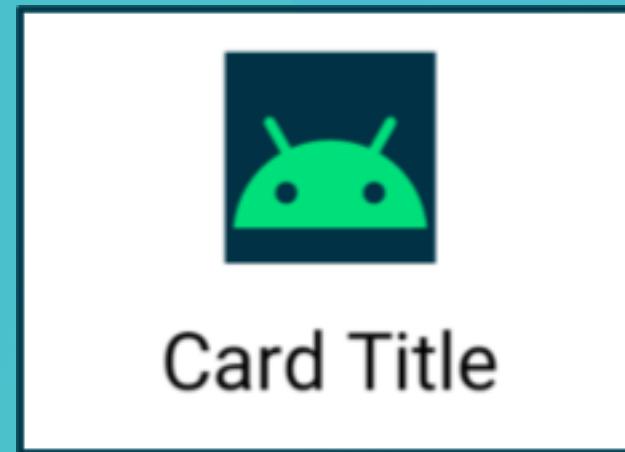
```
fun Card(/* ... */) {  
    BoxWithConstraints {  
        if (maxWidth < 400.dp) {  
            Column {  
                Image(/* ... */)  
                Title(/* ... */)  
            }  
        } else {  
            Row {  
                Column {  
                    Title(/* ... */)  
                    Description(/* ... */)  
                }  
                Image(/* ... */)  
            }  
        }  
    }  
}
```



Flexible Layouts

DEMO

```
@Composable  
fun Card(/* ... */) {  
    BoxWithConstraints {  
        if (maxWidth < 400.dp) {  
            Column {  
                Image(/* ... */)  
                Title(/* ... */)  
            }  
        } else {  
            Row {  
                Column {  
                    Title(/* ... */)  
                    Description(/* ... */)  
                }  
                Image(/* ... */)  
            }  
        }  
    }  
}
```



Lecture outcomes

- Support different screen, using layouts and fragments
- ListView, RecyclerView, Progress Indicators
- Retrieve data on background threads

