# SSE3052: Embedded Systems Practice

Jinkyu jeong
jinkyu@skku.edu
Computer Systems Laboratory
Sungkyunkwan University
http://csl.skku.edu

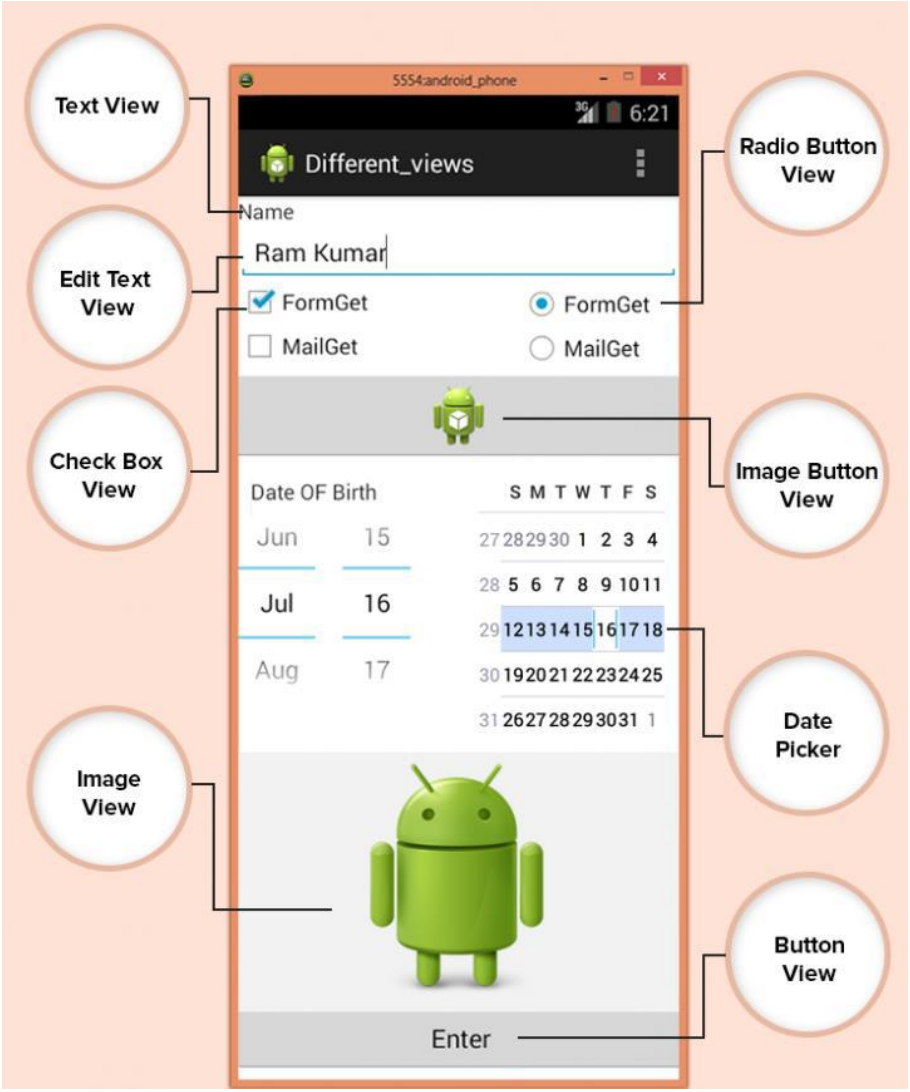
# Layouts

- View & Layout (ViewGroup)
- Attributes of Layout
- Multiple Layouts

# View

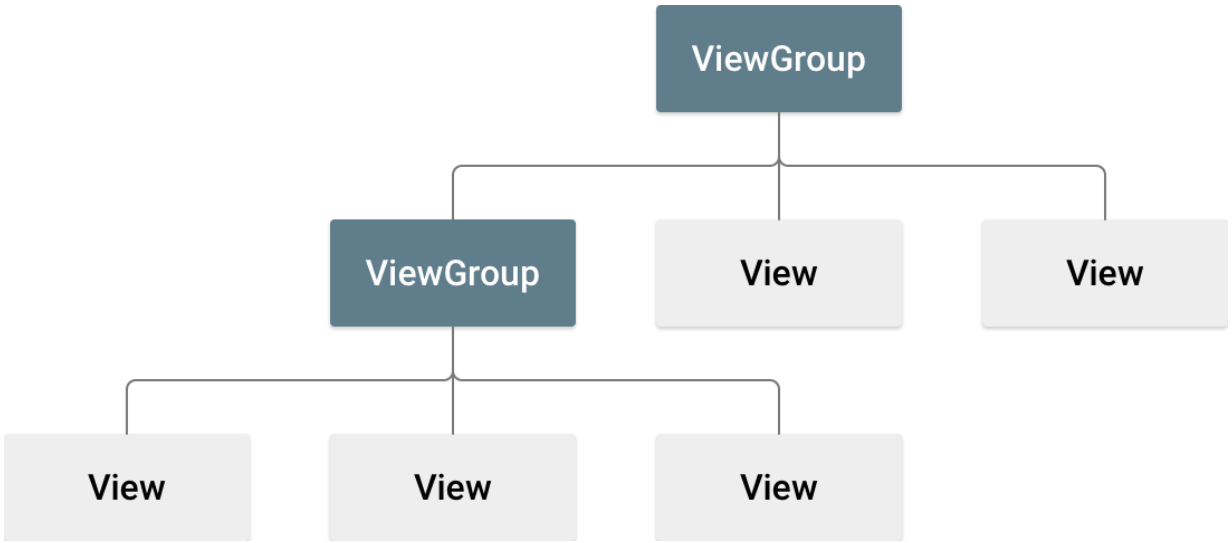
- A View is an object that draw something on the screen that the user can interact
- A Viewgroup is an object that holds other View objects in order to define the layout of the user interface.

# View Class


Text View
Edit Text View
Check Box View
Image View
Radio Button View
Image Button View
Date Picker
Button View
5554:android_phone
6:21
Different_views
Name
Ram Kumar
FormGet
MailGet
FormGet
MailGet
Date OF Birth
Jun
Jul
Aug
15
16
17
S M T W T F S
Enter

# Layouts

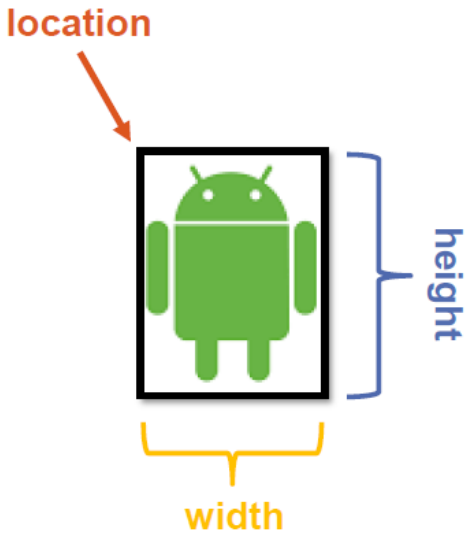
- The View objects are called "widgets"
  - Can be one of many subclasses, such as Button or TextView
- The ViewGroup objects are called "Layouts"
  - Can be one of many types that provide a different layout structure, such as LinearLayout or ConstraintLayout




View hierarchy

# Layout Position

- A View has a location, expressed as a pair of left and top coordinates, and two dimensions, expressed as a width and a height . The unit for location and dimensions is the pixel.
- It is possible to retrieve the location of a view by invoking the methods getLeft() and getTop()
  - To avoid unnessary computations, getRight() and getBottom() are offered.
  - getRight() returns getLeft() + getWidth()



# Layout Size

- The size of a View is expressed with a width and a height.
- Drawing width and drawing height
  - Define the actual size of the view on the screen
  - getWidth() and getHeight()
- Measured width and measured height
  - Define how big a View wants to be within its parent
  - getMeasuredWidth() and getMeasuredHeight()

# How to Declare a Layout?

- Declare UI elements in XML
  - Android provides a straightforward XML vocabulary that corresponds to the View classes and subclasses, such as those for widgets and Layouts. You can also use Android Studio's Layout Editor to build your XML layout using a drag and drop interface.
- Instantiate layout elements at runtime
  - Your app can create View and ViewGroup objects (and manipulate their properties) programmatically.

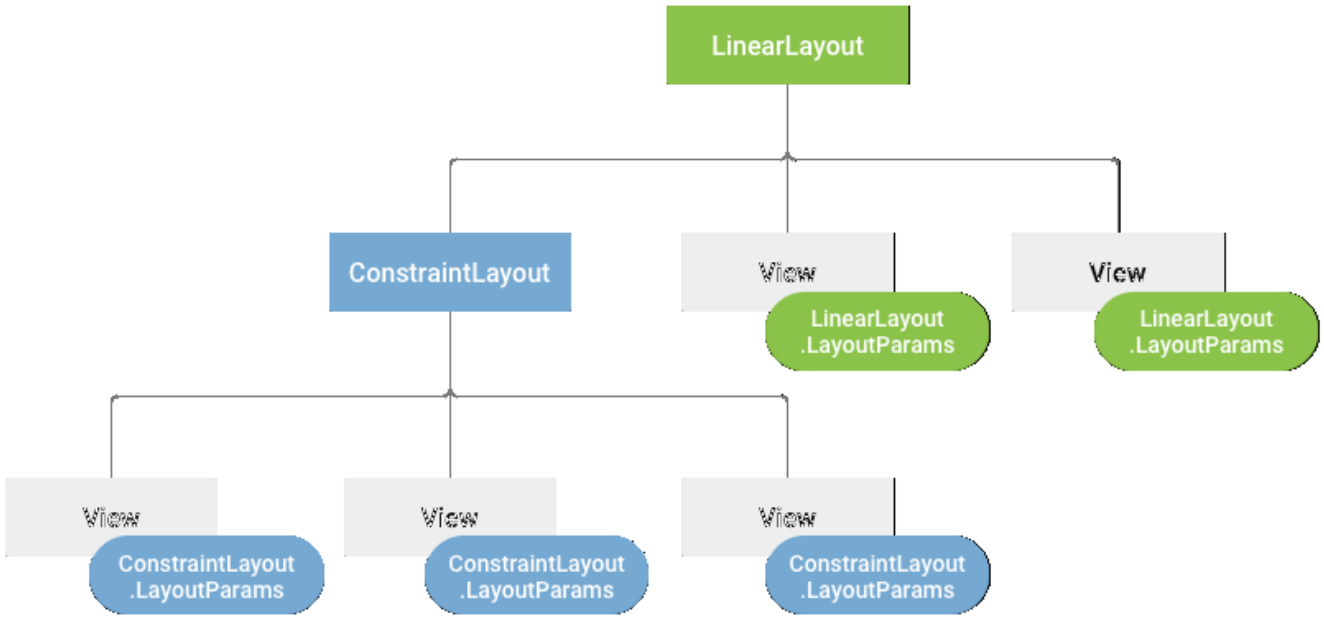
# XML (Extensible Markup Language)

- Using Androids' XML vocabulary, you can quickly design UI layouts and the screen elements they contain.

```
<?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, Iam a TextView“ />
    <Button android:id="@+id/button“
        android:layout_width="wrap_content“
        android:layout_height="wrap_content“
        android:text="Hello, IamaButton“ />
</LinearLayout>
```

# Attributes of Layout (1)

- XML layout attributes named layout_something define layout parameters for the View that are appropriate for the ViewGroup in which it resides.



# Attributes of Layout (2)

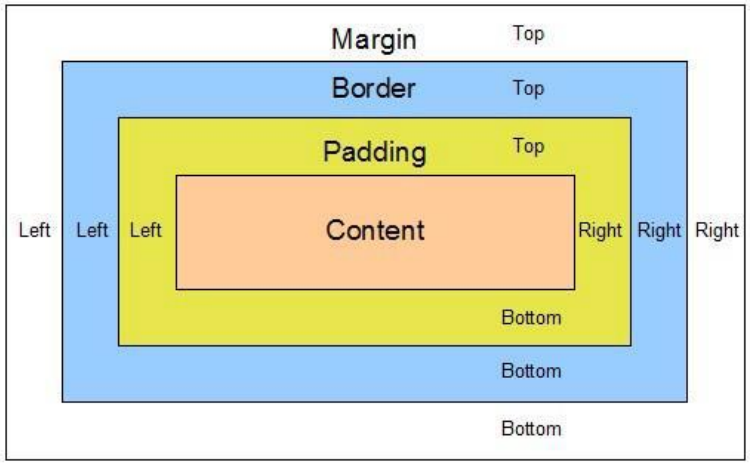
- ID
  - Any view object may have an integer ID
  - android:id="@+id/my_button"
    - @ symbol at the beginning of the string indicates that the XML parser should parse and expand the rest of ID string and identify it as an ID resource
    - + symbol means that this is a new resource name that must be created and added to our resource(in R.java file)
  - Each View object can be access with ID
    - Button myButton = (Button)findViewByID(R.id.my_button);

# Attributes of Layout (3)

- Layout parameters
  - Help defining other attributes with relative values
    - wrap_content tells your view to size itself to the dimensions required by its content
    - match_parent tells your view to become as big as its parent view group will allow

# Attributes of Layout (4)

- Margin
  - The space between border and its parent
- Padding
  - The space between border and content
  - setPadding(int, int, int, int): left, top, right, bottom
  - getPaddingLeft(), getPaddingTop(), getPaddingRight(), getPaddingBottom()



# Common Layouts

- Each subclass of the ViewGroup class provides a unique way to display the views you nest within it.
- Below are some of the more common layout types that are built into the Android platform.

<Linear layout>

<Relative layout>


<Web view>

# Example of LinearLayout

```
<?xmlversion="1.0“ encoding="utf-8"?>
<LinearLayoutxmlns:android=http://schemas.android.com/apk/res/android”
        android:layout_width="match_parent“
        android:layout_height="match_parent“
        android:paddingLeft="16dp“
        android:paddingRight="16dp“
        android:orientation="vertical“ >
        <EditText
                android:layout_width="match_parent“
                android:layout_height="wrap_content“
                android:hint="@string/to“ />
        <EditText
                android:layout_width="match_parent“
                android:layout_height="wrap_content“
                android:hint="@string/subject“ />
        <EditText
                android:layout_width="match_parent“
                android:layout_height="0dp“
                android:layout_weight=“1“
                android:gravity="top“
                android:hint="@string/message“ />
        <Button
                android:layout_width="100dp“
                android:layout_height="wrap_content“
                android:layout_gravity="right“
                android:text="@string/send” />
</LinearLayout>
```

# Example of RelativeLayout

```
<?xmlversion="1.0“ encoding="utf-8"?>
<RelativeLayoutxmlns:android=http://schemas.android.com/apk/res/android”
        android:layout_width="match_parent“
        android:layout_height="match_parent“
        android:paddingLeft="16dp“
        android:paddingRight="16dp“ >
        <EditText
                android:id="@+id/name“
                android:layout_width="match_parent“
                android:layout_height="wrap_content“
                android:hint="@string/reminder“ />
        <Spinner
                android:id="@+id/dates“
                android:layout_width="0dp“
                android:layout_height="wrap_content“
                android:layout_below="@id/name“
                android:layout_alignParentLeft="true“
                android:layout_toLeftOf="@+id/times“ />
        <Spinner
                android:id="@id/times“
                android:layout_width="96dp“
                android:layout_height="wrap_content“
                android:layout_below="@id/name“
                android:layout_alignParentRight="true“ />
        <Button
                android:layout_width="96dp“
                android:layout_height="wrap_content“
                android:layout_below="@id/times“
                android:layout_alignParentRight="true“
                android:text="@string/done“ />
</RelativeLayout>
```

# Layout Types

- Linear Layout
  - LinearLayout is a view group that aligns all children in a single direction, vertically or horizontally
- Relative Layout
  - RelativeLayout is a view group that displays child views in relative positions
- Table Layout
  - TableLayout is a view that groups views into rows and columns
- Absolute Layout
  - AbsoluteLayout enables you to specify the exact location of its children
- Frame Layout
  - The FrameLayout is a placeholder on screen that you can use to display a single view
- List View
  - ListView is a view group that displays a list of scrollable items
- Grid View
  - GridView is a ViewGroup that displays items in a two dimensional, scrollable grid.

# Layout Attributes

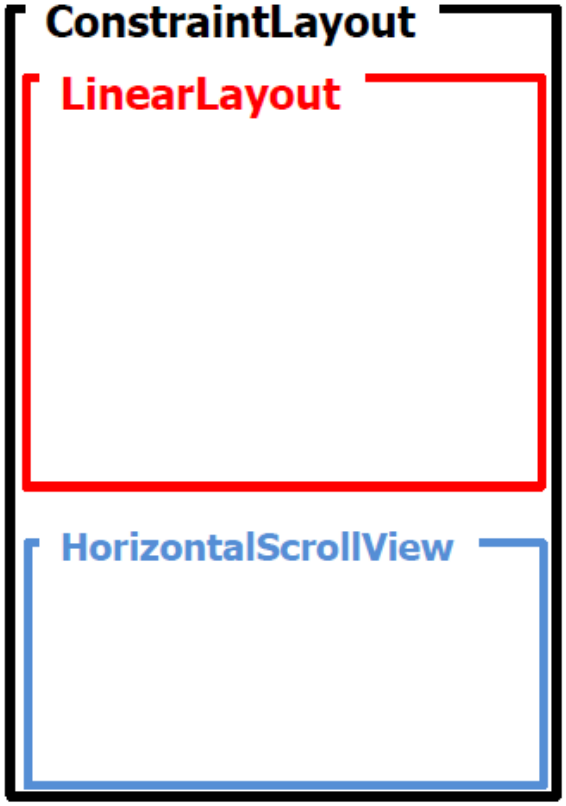
- android:id
  - This is the ID which uniquely identifies the view
- android:layout_width
  - This is the width of the layout
- android:layout_height
  - This is the height of the layout
- android:layout_marginTop
  - This is the extra space on the top side of the layout
- android:layout_marginBottom
  - This is the extra space on the bottom side of the layout
- android:layout_marginLeft
  - This is the extra space on the left side of the layout
- android:layout_marginRight
  - This is the extra space on the right side of the layout
- android:layout_gravity
  - This specifies how child Views are positioned
- android:layout_weight
  - This specifies how much of the extra space in the layout should be allocated to the View
- android:layout_x
  - This specifies the x coordinate of the layout
- android:layout_y
  - This specifies the y coordinate of the layout
- android:layout_width
  - This is the width of the layout
- android:layout_width
  - This is the width of the layout
- android:paddingLeft
  - This is the left padding filled for the layout
- android:paddingRight
  - This is the right padding filled for the layout
- android:paddingTop
  - This is the top padding filled for the layout
- android:paddingBottom
  - This is the bottom padding filled for the layout

# Multiple Layouts (1)

- Drawing more than one layout
- A layout is allowed to contain other layouts.
  - All layouts are also have constrains and attributes for arrangement.
  - It actually similar with inserting views into a layout.

# Multiple Layouts (2)

- Let's insert HorizontalScrollView and LinearLayout into ConstraintLayout.



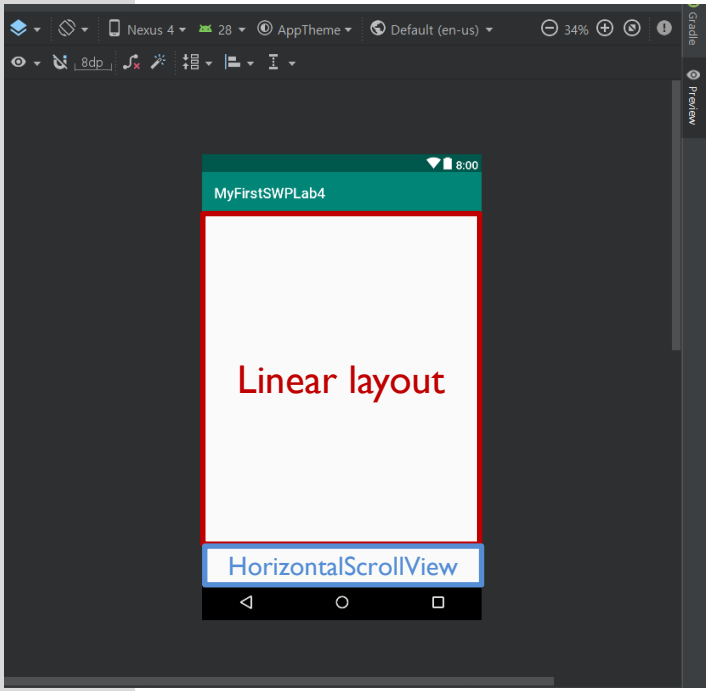

```
<ConstraintLayout>
<LinearLayout>
…
</LinearLayout>
<HorizontalScrollView>
…
</HorizontalScrollView>
</ConstraintLayout>
```

# HorizontalScrollView

- Horizontally aligned ScrollView
- Scrollable but not clickable or focusable
- It's not layout!
  - Only one view/layout can be located in this component.
  - To put multiple views on it, we should insert layout first

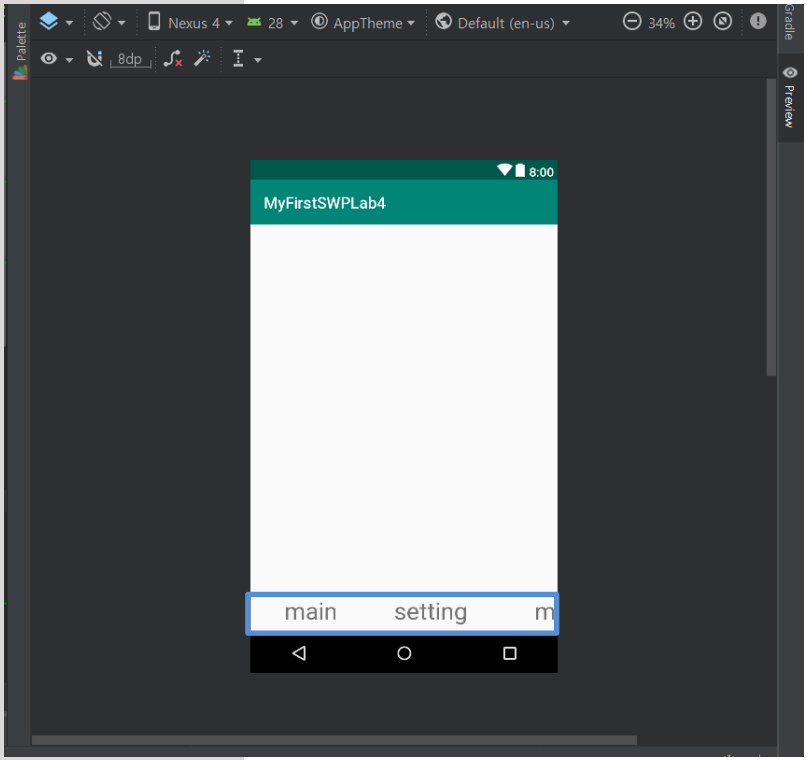
# Multiple Layouts – Example (1)

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.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">
        <LinearLayout
                android:id="@+id/linearLayout“
                android:layout_width="match_parent“
                android:layout_height="match_parent“
                android:layout_marginBottom="50dp"
                android:orientation="vertical“
                app:layout_constraintBottom_toBottomOf="parent">
        </LinearLayout>
        <HorizontalScrollView
                android:id="@+id/horizontalscroll“
                android:layout_width="match_parent“
                android:layout_height="50dp“
                app:layout_constraintTop_toBottomOf="@+id/linearLayout">
        </HorizontalScrollView>
</android.support.constraint.ConstraintLayout>
```



# Multiple Layouts – Example (2)

```
<HorizontalScrollView
        android:id="@+id/horizontalscroll“
        android:layout_width="match_parent“
        android:layout_height="50dp“
        app:layout_constraintTop_toBottomOf="@+id/linearLayout">
        <LinearLayout
                android:layout_width="wrap_content“
                android:layout_height="match_parent“
                android:orientation="horizontal">
                <TextView
                        android:id="@+id/main_page“
                        android:layout_width="150dp“
                        android:layout_height="50dp“
                        android:clickable="true“
                        android:focusable="true“
                        android:text="@string/main_page“
                        android:textAlignment="center“
                        android:textSize="30sp" />
                <TextView
                        android:id="@+id/set_page“
                        android:layout_width="150dp“
                        android:layout_height="50dp“
                        android:clickable="true“
                        android:focusable="true“
                        android:text="@string/set_page“
                        android:textAlignment="center“
                        android:textSize="30sp" />
                <TextView
                        android:id="@+id/my_page“
                        android:layout_width="150dp“
                        android:layout_height="50dp“
                        android:clickable="true“
                        android:focusable="true“
                        android:text="@string/my_page“
                        android:textAlignment="center“
                        android:textSize="30sp" />
        </LinearLayout>
</HorizontalScrollView>
```



# Density-independent Pixels (DP)

- Density independent pixel(dp , dip) is an Android mechanism which provides compatibility with any possible display density.
- In real world 1dp is approximately 0.2mm

# Exercise

- Implementing a basic layouts
  - Add more than 5 views
    - ex) TextView, Button, EditText, RadioButton, Switch, …