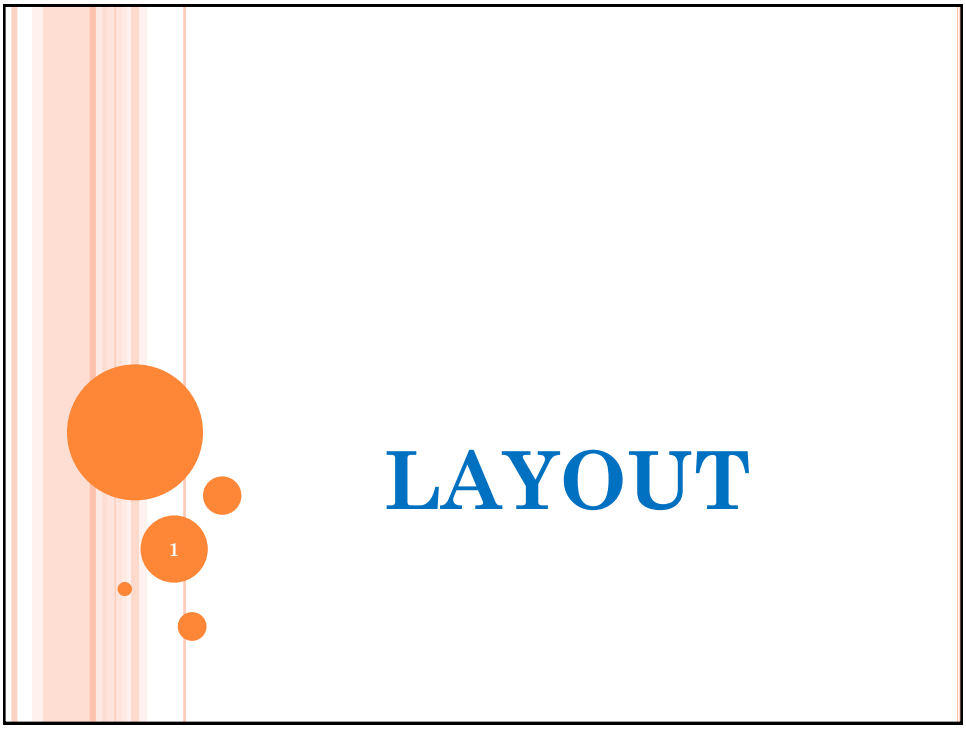
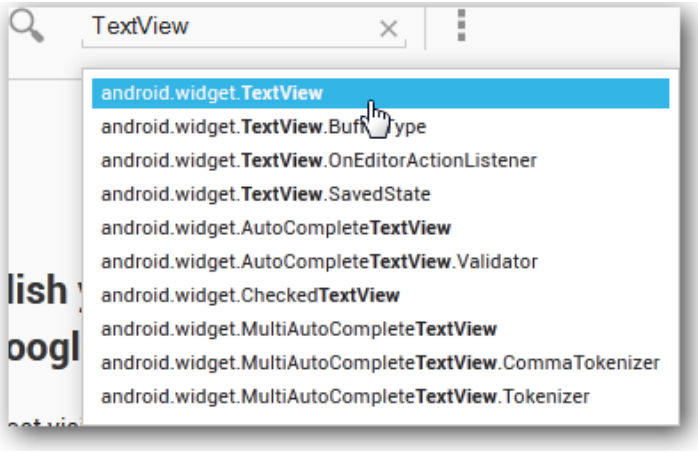


PLAN

Linear Layout
Relative Layout
Absolute Layout
Table Layout
Grid Layout (API >=14)

2

Je me demande comment faire pour changer la couleur du texte de ma TextView. Pour cela, je me dirige vers la documentation officielle : http://developer.android.com/.



3

LAYOUT (GABARIT)

- Un layout est un conteneur qui va accueillir les boutons, images, champs de texte,… tout ce que l'on appelle des widgets.
- Il existe plusieurs types de layout, mais le plus courant le «LinearLayout».
- Il existe plusieurs types de layout (LinearLayout, RelativeLayout, TableLayout,...), chacun avec des caractéristiques différentes.

4

Les éléments graphiques héritent de la classe **View**.

On peut regrouper des éléments graphiques dans une **ViewGroup**.

Des **ViewGroup** particuliers sont prédéfinis: ce sont des gabarits (*layout*) qui proposent une prédispositions des objets graphiques:

Exemple :

- **LinearLayout**: dispose les éléments de gauche à droite ou du haut vers le bas

5

- **RelativeLayout**: les éléments enfants les uns par rapport aux autres

- **TableLayout**: disposition matricielle

et d'autres à voir …

Les déclarations se font principalement en XML, ce qui évite de passer par les instanciations Java.

6

Attributs des gabarits

Orientation

Sens de placement des vues dans un conteneur

android:orientation = vertical | horizontal

Taille

Surface prise par la vue

android:layout_width = ??px | fill_parent | wrap_content

android:layout_height = ??px | fill_parent | wrap_content

fill_parent : l'élément remplit tout l'élément parent

wrap_content : prend la place nécessaire à l'affichage

Gravité

Alignement d'une vue dans son conteneur

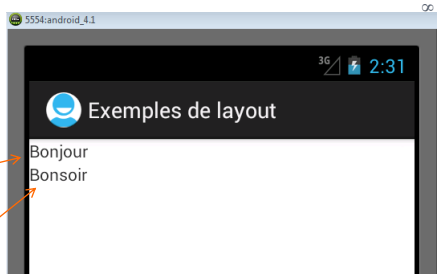
android:layout_gravity = left | center_horizontal | top | bottom | right

7

QUELQUES EXEMPLES

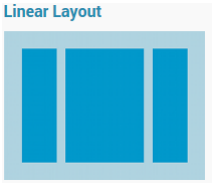
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:orientation="vertical"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
>
<TextView
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="Bonjour"
/>
<TextView
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="Bonsoir"
/>
</LinearLayout>
```

Le LinearLayout va aligner les widgets les uns après les autres à la verticale (en colonne) si vous choisissez «android:orientation="vertical" et à l'horizontale (en ligne) si vous choisissez «android:orientation="horizontal"

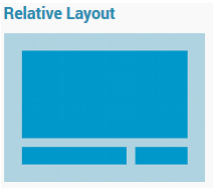


VUE D'ENSEMBLE

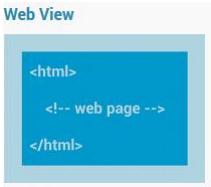
Éléments placés en horizontal ou vertical
Crée un «scrollbar» si la taille dépasse celle de la fenêtre



Éléments alignés un par rapport aux autres



Affiche des pages Web

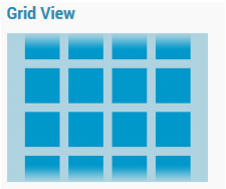


9

Layout soutenus par un adaptateur comprennent

Liste simple colonne avec «scrolling»

Affiche une grille avec des lignes et colonnes



Ces éléments sont remplis via un adaptateur (à voir plus tard).

10

Créer une View et la référencer via l'application:

Avec un id unique :

```
<Button android:id="@+id/mon_button"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="@string/mon_button_text“
      android:onClick="afficher"
/>
```

Créer une instance de la View, normalement dans la méthode onCreate() de l'pllication :

```
Button monButton = (Button) findViewById(R.id.mon_button);
```

11

Exemple : accès par java

```
public class MyFirstActivity extends Activity {
        private TextView element;
    @Override
    public void onCreate(Bundle savedInstanceState) {
       super.onCreate(savedInstanceState);
       setContentView(R.layout.main);
       element = (TextView) findViewById(R.id.idElement);
    }
    public void afficher(View v) {
        element.setText(R.string.hello);
    //      element.setText("Salut les amis! ");
    }
}
```

Contenu dans une variable

12

Layout Attributes

Each layout has a set of attributes which define the visual properties of that layout. There are few common attributes among all the layouts and their are other attributes which are specific to that layout. Following are common attributes and will be applied to all the layouts:

Attribute	Description
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: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.

Here width and height are the dimension of the layoutView which can be specified in terms of dp (Density-independent Pixels), sp (Scale-independent Pixels), pt (Points which is 1/72 of an inch), px(Pixels), mm (Millimeters) and finally in (inches).

You can specify width and height with exact measurements but more often, you will use one of these constants to set the width or height:

- **android:layout_width=wrap_content** tells your view to size itself to the dimensions required by its content.
- **android:layout_width=fill_parent** tells your view to become as big as its parent view.

13

EXEMPLES XML ET PAR PROGRAMMATION

Les étiquettes de texte

```
<TextView
android:id="@+id/le_texte"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="@string/salut"
android:layout_gravity="center"
/>
```

```
public class Activity2 extends Activity {
public void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
  LinearLayout gabarit = new LinearLayout(this);
  gabarit.setGravity(Gravity.CENTER); // centrer les éléments graphiques
  gabarit.setOrientation(LinearLayout.VERTICAL); // empiler vers le bas !
  TextView texte = new TextView(this);
  texte.setText("Salut les amis!");
  gabarit.addView(texte);
  setContentView(gabarit);
} }
```

14

Les zones de texte :
http://developer.android.com/reference/android/widget/TextView.html

```
<EditText android:text=""
        android:id="@+id/EditText01"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content">
</EditText>
```

```
EditText edit = new EditText(this);
edit.setText("Mon texte");
gabarit.addView(edit);
```

Interception d'événements:

```
edit.addTextChangedListener(new TextWatcher() {
@Override
public void onTextChanged(CharSequence s, int start, int before, int count) {
        // faire quelque chose
}
);
```

abstract void onTextChanged(CharSequence s, int start, int before, int count)
This method is called to notify you that, within s, the count characters beginning at start have just replaced old text that had length before.

Les images

```
<ImageView
android:id="@+id/monLogo"
android:src="@drawable/logo"
android:layout_width="100px"
android:layout_height="wrap_content"
android:layout_gravity="center_horizontal"
></ImageView>
```

```
ImageView image = new ImageView(this);
image.setImageResource(R.drawable.logo);
gabarit.addView(image);
```

16

Les boutons

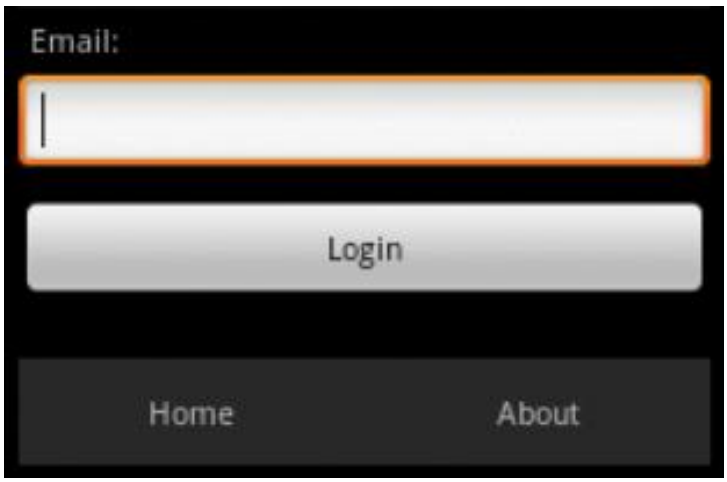
```
<Button
android:id="@+id/Button01"
android:text="Calculer !"
android:layout_width="wrap_content"
android:layout_height="wrap_content">
</Button>
```

La gestion des événements de *click* se font par l'intermédiaire d'un listener:

```
Button b = (Button)findViewById(R.id.Button01);

b.setOnClickListener(new OnClickListener() {
@Override
public void onClick(View v) {
    Toast.makeText(v.getContext(), "Stop !", Toast.LENGTH_LONG).show();
}
});
}
```

17

Linear Layout (plus utilisé)



18

```
<?xml version="1.0" encoding="utf-8"?>
<!– linear layout parent avec orientation vertical–>
<LinearLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:orientation="vertical"
  android:layout_width="match_parent"
  android:layout_height="match_parent">

  <TextView android:layout_width="fill_parent"
android:layout_height="wrap_content"
            android:text="Email:"
            android:padding="5dip"/>

  <EditText android:layout_width="fill_parent"
android:layout_height="wrap_content"
          android:layout_marginBottom="10dip"/>

  <Button android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="Login"/>
```

19

```
<!– linear layout enfant avec orientation  horizontal –>

 <LinearLayout android:layout_width="fill_parent"
          android:layout_height="wrap_content"
          android:orientation="horizontal"
          android:background="#2a2a2a"
          android:layout_marginTop="25dip">

 <TextView android:layout_width="fill_parent"
          android:layout_height="wrap_content"
           android:text="Home" android:padding="15dip"
           android:layout_weight="1"
           android:gravity="center"/>

 <TextView android:layout_width="fill_parent" android:layout_height="wrap_content"
      android:text="About" android:padding="15dip"
      android:layout_weight="1"
      android:gravity="center"/>

 </LinearLayout>

</LinearLayout>
```

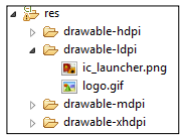
20

AUTRE EXEMPLE

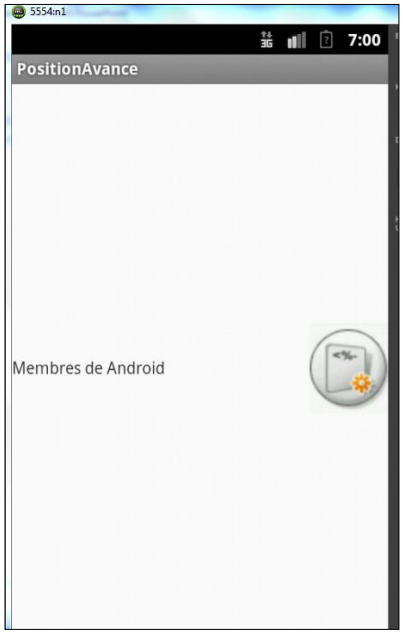
```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_height="fill_parent"
android:orientation="horizontal"
android:layout_width="fill_parent"
>

  <TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_centerHorizontal="true"
    android:layout_centerVertical="true"
    android:text="@string/hello_world" />
```

21

```
<LinearLayout android:layout_height="wrap_content"
android:orientation="horizontal"
android:layout_width="fill_parent"
android:gravity="right"
>
<ImageView
android:id="@+id/monLogo"
android:src="@drawable/logo"
android:layout_width="100px"
android:layout_height="wrap_content"
></ImageView>

</LinearLayout></LinearLayout>
```



22



23

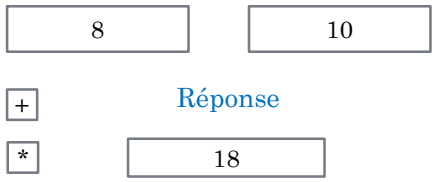
Faire l'exercice de pratique dont l'énoncé est dans LEA.

Le déposer lorsque fini.

24

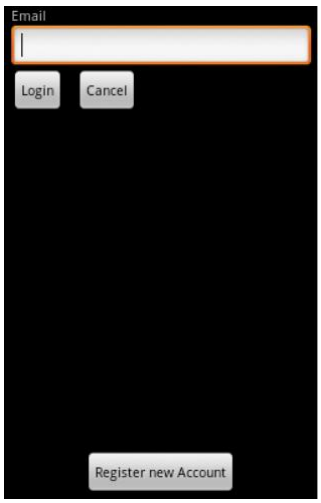
Exercice de pratique : layout, accès aux éléments, Événement

Créer le formulaire suivant :



25

Relative Layout



26

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content">

   <TextView android:id="@+id/label" android:layout_width="fill_parent"
          android:layout_height="wrap_content" android:text="Email" />

   <EditText android:id="@+id/inputEmail" android:layout_width="fill_parent"
          android:layout_height="wrap_content"
          android:layout_below="@id/label" />

   <Button android:id="@+id/btnLogin" android:layout_width="wrap_content"
        android:layout_height="wrap_content"
          android:layout_below="@id/inputEmail"
          android:layout_alignParentLeft="true"
          android:layout_marginRight="10px"
        android:text="Login" />
```

Propriétés propres à un RelativeLayout

27

```
<Button android:layout_width="wrap_content"
android:layout_height="wrap_content"
          android:layout_toRightOf="@id/btnLogin"
          android:layout_alignTop="@id/btnLogin"
          android:text="Cancel" />

   <Button android:layout_width="wrap_content"
android:layout_height="wrap_content"
          android:layout_alignParentBottom="true"
android:text="Register new Account"
          android:layout_centerHorizontal="true"/>
</RelativeLayout>
```

28

Modèle : haut,centre et bas

29

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >

    <!-- Entête aligné en haut -->

    <RelativeLayout
        android:id="@+id/header"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:background="#FC9"
        android:gravity="center" >

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_margin="5dp"
            android:text="Entête fixe"
            android:textColor="#000"
            android:textSize="20sp" />
    </RelativeLayout>
```

30

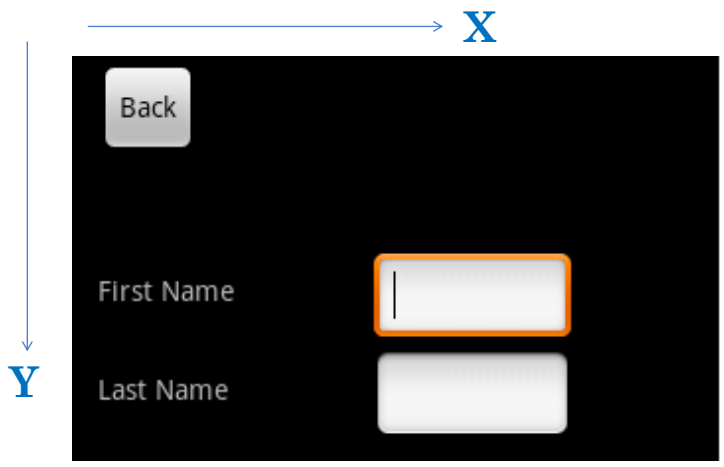
```
<!-- Pieds de page aligné en bas -->

   <RelativeLayout
      android:id="@+id/footer"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:layout_alignParentBottom="true"
      android:background="#FC0"
      android:gravity="center" >

      <TextView
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:layout_margin="5dp"
         android:text="Pieds de page fixe"
         android:textColor="#000"
         android:textSize="20sp" />
   </RelativeLayout>
```

31

```
<!-- Contenu -->

   <RelativeLayout
      android:id="@+id/content"
      android:layout_width="fill_parent"
      android:layout_height="fill_parent"
      android:layout_above="@id/footer"
      android:layout_below="@id/header"
      android:gravity="center" >

      <TextView
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:text="Contenu"
         android:textColor="#33E"
         android:textSize="20sp" />
   </RelativeLayout>

</RelativeLayout>
```

32

Absolute Layout



33

```
<AbsoluteLayout
xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent">
   <Button
         android:id="@+id/backbutton"
         android:text="Back"
         android:layout_x="10px"
         android:layout_y="5px"
         android:layout_width="wrap_content"
         android:layout_height="wrap_content" />
   <TextView
         android:layout_x="10px"
         android:layout_y="110px"
         android:text="First Name"
         android:layout_width="wrap_content"
         android:layout_height="wrap_content" />
```

34

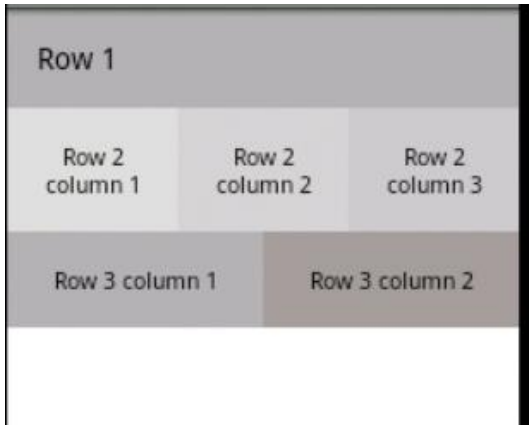
```
<EditText
        android:layout_x="150px"
        android:layout_y="100px"
        android:width="100px"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
  <TextView
        android:layout_x="10px"
        android:layout_y="160px"
        android:text="Last Name"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
        <EditText
        android:layout_x="150px"
        android:layout_y="150px"
        android:width="100px"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
</AbsoluteLayout>
```

FAIRE L'EXERCICE 2 QUI EST DANS LEA

35

Table Layout

http://developer.android.com/reference/android/widget/TableLayout.html



36

Operations Applicable to TableLayout

We can perform several operations on TableLayout columns, including stretching, shrinking, collapsing, and spanning columns.

Stretching Columns

The default width of a column is set equal to the width of the widest column, but we can stretch the column(s) to take up available free space using the android:stretchColumns attribute in the TableLayout. The value assigned to this attribute can be a single column number or a comma-delimited list of column numbers. The specified columns are stretched to take up any available space on the row.

Examples:

android:stretchColumns="1"—The second column (because the column numbers are 0-based) is stretched to take up any available space in the row.

android:stretchColumns="0,1"—Both the first and second columns are stretched to take up the available space in the row.

android:stretchColumns="*"—All columns are stretched to take up the available space.

37

Shrinking Columns

We can shrink or reduce the width of the column(s) using the android:shrinkColumns attribute in the TableLayout. We can specify either a single column or a comma delimited list of column numbers for this attribute. The content in the specified columns word-wraps to reduce their width.

Examples:

android:shrinkColumns="0"—The first column's width shrinks or reduces by word-wrapping its content.

android:shrinkColumns="*"—The content of all columns is word-wrapped to shrink their widths.

38

```
<TableLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:shrinkColumns="*"  android:stretchColumns="*" android:background="#ffffff">
   <!– Ligne 1 avec 1 colonne–>
   <TableRow
      android:layout_height="wrap_content"
      android:layout_width="fill_parent"
      android:gravity="center_horizontal">
      <TextView
         android:layout_width="match_parent«
        android:layout_height="wrap_content"
         android:textSize="18dp"
         android:text="Row 1"
        android:layout_span="3"
         android:padding="18dip"
        android:background="#b0b0b0"
         android:textColor="#000"/>
   </TableRow>
```

39

```
<!-- Ligne 2 avec 3 colonnes-->
<TableRow
   android:id="@+id/tableRow1"
   android:layout_height="wrap_content"
   android:layout_width="match_parent">
   <TextView
      android:id="@+id/TextView04"
     android:text="Row 2 column 1"
      android:layout_weight="1"
     android:background="#dcdcdc"
      android:textColor="#000000"
      android:padding="20dip"
      android:gravity="center"/>
```

40

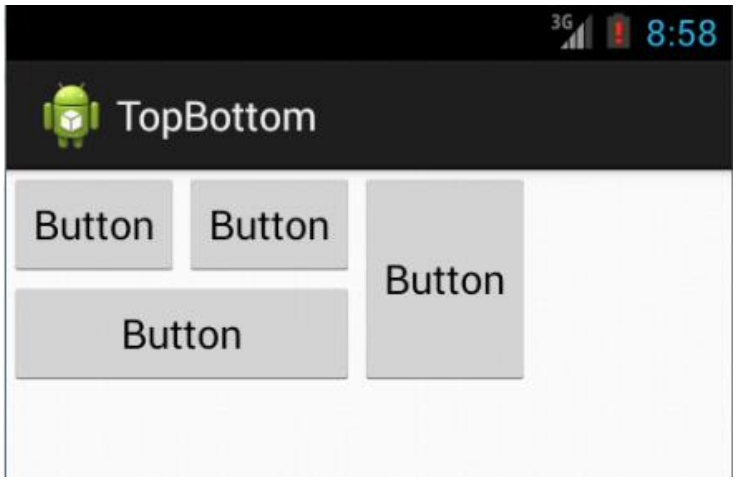
```
<TextView
        android:id="@+id/TextView04"
         android:text="Row 2 column 2"
        android:layout_weight="1"
        android:background="#d3d3d3"
        android:textColor="#000000"
        android:padding="20dip" android:gravity="center"/>

<TextView
        android:id="@+id/TextView04"
        android:text="Row 2 column 3"
        android:layout_weight="1"
        android:background="#cac9c9"
        android:textColor="#000000"
        android:padding="20dip" android:gravity="center"/>
  </TableRow>
```

41

```
<!—Ligne 3 avec 2 colonnes -->
  <TableRow
    android:layout_height="wrap_content"
    android:layout_width="fill_parent"
    android:gravity="center_horizontal">
    <TextView
      android:id="@+id/TextView04"
      android:text="Row 3 column 1"
      android:layout_weight="1" android:background="#b0b0b0"
      android:textColor="#000000"
      android:padding="20dip" android:gravity="center"/>

    <TextView
      android:id="@+id/TextView04" android:text="Row 3 column 2"
      android:layout_weight="1" android:background="#a09f9f"
      android:textColor="#000000"
      android:padding="20dip" android:gravity="center"/>
  </TableRow>
</TableLayout>
```

42

Grid Layout



43

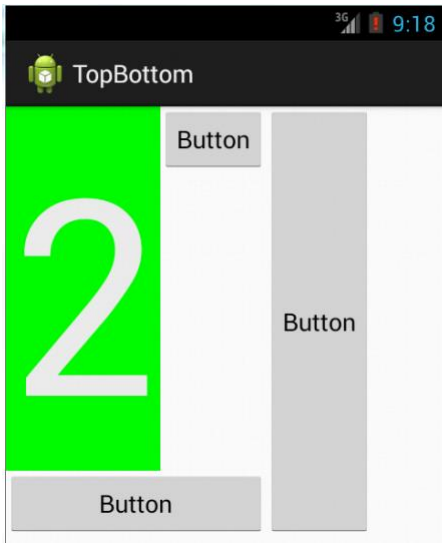
```
<GridLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/GridLayout1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:columnCount="3"
    tools:context=".GridLayoutActivity" >

    <Button
        android:id="@+id/button3"
        android:layout_column="0"
        android:layout_gravity="left|top"
        android:layout_row="0"
        android:text="Button" />

    <Button
        android:id="@+id/button1"
        android:layout_column="1"
        android:layout_gravity="left|top"
        android:layout_row="0"
        android:text="Button" />
```

44

```
<Button
        android:id="@+id/button2"
        android:layout_column="2"
        android:layout_gravity="fill_vertical"
        android:layout_row="0"
        android:layout_rowSpan="2"
        android:text="Button" />

    <Button
        android:id="@+id/button4"
        android:layout_column="0"
        android:layout_columnSpan="2"
        android:layout_gravity="fill_horizontal"
        android:layout_row="1"
        android:text="Button" />

</GridLayout>
```

45

En remplaçant premier bouton par un TexView (formaté):



46

```
<TextView
        android:text="2"
        android:textSize="200sp"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:id="@+id/textView2"
        android:background="#00FF00"
        android:gravity="center"
        android:textColor="#EEEEEE"
/>
```

47

Deux boutons dans un layout :

Vous remarquerez que l'espace est toujours divisé entre les deux boutons, soit de manière égale, soit un bouton écrase complètement l'autre.

Et si on voulait que le bouton de droite prenne deux fois plus de place que celui de gauche par exemple ?

Pour cela, il faut attribuer un poids au composant. Ce poids peut être défini grâce à l'attribut android:layout_weight.

Pour faire en sorte que le bouton de droite prenne deux fois plus de place, on peut lui mettre android:layout_weight="1" et mettre au bouton de gauche android:layout_weight="2".

48

C'est alors le composant qui a la plus faible pondération qui a la priorité.

Et si, dans l'exemple précédent où un bouton en écrasait un autre, les deux boutons avaient eu un poids identique, par exemple android:layout_weight="1" pour les deux, ils auraient eu la même priorité et auraient pris la même place.

Par défaut, ce poids est à 0.

Une astuce consiste à faire en sorte que la somme des poids dans un même layout fasse 100. C'est une manière plus évidente pour répartir les poids.

49

```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" >

    <LinearLayout
      android:layout_width="match_parent"
      android:layout_height="match_parent"
       >

      <Button
        android:id="@+id/button4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="50"
        android:text="Button" />
```

50

```
<Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="50"
        android:text="Button" />
    <Button
    android:id="@+id/button2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_weight="50"
    android:text="Button" />

  </LinearLayout>


</LinearLayout>
```

51

Layout et Widgets

52

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent"
  android:orientation="vertical">
  <LinearLayout
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
  >
    <TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Poids : "
      android:textStyle="bold"
      android:textColor="#FF0000"
      android:gravity="center"
    />
    <EditText
      android:id="@+id/poids"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:hint="Poids"
      android:inputType="numberDecimal"
      android:layout_weight="1"
    />
  </LinearLayout>
```

53

```
<LinearLayout
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
  >
    <TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Taille : "
      android:textStyle="bold"
      android:textColor="#FF0000"
      android:gravity="center"
    />
    <EditText
      android:id="@+id/taille"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:hint="Taille"
      android:inputType="numberDecimal"
      android:layout_weight="1"
    />
  </LinearLayout>
```

54

```
<RadioGroup
  android:id="@+id/group"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:checkedButton="@+id/radio2"
  android:orientation="horizontal"
 >
  <RadioButton
   android:id="@+id/radio1"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:text="Mètre"
  />
  <RadioButton
   android:id="@+id/radio2"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:text="Centimètre"
  />
</RadioGroup>
```

55

```
<CheckBox
  android:id="@+id/mega"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:text="Mega fonction !"
 />
```

56

```
<LinearLayout
   android:layout_width="fill_parent"
   android:layout_height="wrap_content"
   android:orientation="horizontal"
 >
   <Button
    android:id="@+id/calcul"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Calculer l'IMC"
    android:layout_weight="1"
    android:layout_marginLeft="25dip"
    android:layout_marginRight="25dip"
   />
   <Button
    android:id="@+id/raz"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="RAZ"
    android:layout_weight="1"
    android:layout_marginLeft="25dip"
    android:layout_marginRight="25dip"
   />
 </LinearLayout>
```

57

```
<TextView
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:text="Résultat:"
 />
 <TextView
   android:id="@+id/result"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   android:text="Vous devez cliquer sur le bouton « Calculer
l'IMC » pour obtenir un résultat."
 />
</LinearLayout>
```

58

XML	JAVA

TextView
<TextView
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="@string/textView"
android:textSize="8sp"
android:textColor="#112233" />
	TextView textView = new TextView(this);
	textView.setText(R.string.textView);
	textView.setTextSize(8);
	textView.setTextColor(0x112233);

EditText
<EditText
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:hint="@string/editText"
android:inputType="textMultiLine"
android:lines="5" />
	EditText editText = new EditText(this);
	editText.setHint(R.string.editText);
	editText.setInputType(InputType.TYPE_TEXT_FLAG_MULTI_LINE);
	editText.setLines(5);

Button <Button android:layout_width="fill_parent" android:layout_height="wrap_content" android:text="@string/button" />	Button button = new Button(this); editText.setText(R.string.button);

59

CheckBox
<CheckBox
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="@string/checkBox"
android:checked="true" />
	CheckBox checkBox = new CheckBox(this);
	checkBox.setText(R.string.checkBox);
	checkBox.setChecked(true)
	if(checkBox.isChecked())
	// Faire quelque chose si le bouton est coché

RadioButton et RadioGroup
<RadioGroup
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:orientation="horizontal" >
<RadioButton
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:checked="true" />
<RadioButton
android:layout_width="wrap_content"
android:layout_height="wrap_content" />
<RadioButton
android:layout_width="wrap_content"
android:layout_height="wrap_content" />
</RadioGroup>
	RadioGroup radioGroup = new RadioGroup(this);
	RadioButton radioButton1 = new RadioButton(this);
	RadioButton radioButton2 = new RadioButton(this);
	RadioButton radioButton3 = new RadioButton(this);
	// On ajoute les boutons au RadioGroup
	radioGroup.addView(radioButton1, 0);
	radioGroup.addView(radioButton2, 1);
	radioGroup.addView(radioButton3, 2);
	// On sélectionne le premier bouton
	radioGroup.check(0);
	// On récupère l'identifiant du bouton qui est coché
	int id = radioGroup.getCheckedRadioButtonId();

60

ScrollView

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView
xmlns:android="http://schemas.android.com/apk/res/android
"
android:layout_width="fill_parent"
android:layout_height="wrap_content">
  <LinearLayout>
    <!-- contenu du layout -->
  </LinearLayout>
</ScrollView>
```

Attention cependant, il ne faut pas mettre de widgets qui peuvent déjà défiler dans une ScrollView

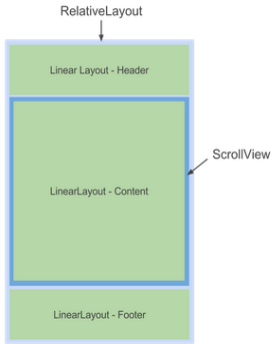
61

Exercice de laboratoire (à remettre 2 étudiants max.)

Exercice sur les Layouts

Partie 1

Le but de cette exercice est de vous faire travailler avec les layouts pour obtenir une vue finale qui ressemble à la capture ci-contre, soit un formulaire avec 2 champs et des contrôles en bas de l'écran. Cet écran représente le formulaire de connexion de notre application.



Indications:

- **TextView** permet d'afficher un texte
- **EditText** permet d'afficher un champ de saisie
- **ScrollView** permet d'avoir un contenu scrollable
- **Button** permet d'afficher un bouton

Une bonne façon de commencer est de mettre en place des layouts avec des couleurs de fond différentes pour s'assurer que tout est bien à sa place et réagit correctement au changement d'orientation.

Pour changer la couleur de fond d'un élément :

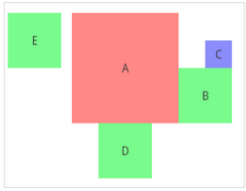
android:background="#FFFF0000"

En rouge la transparence de 00 à FF

En vert le code couleur en hexadécimal (ici du rouge)

62

Partie 2



Créez un nouveau projet et en utilisant simplement un RelativeLayout et des TextViews, essayez de reproduire l'exemple. Le résultat devrait être comme ci-contre.

La correction des exercices sera faite sur votre poste de travail.

Vous devez créer un dossier 186NomPrénom(exercice1) et y mettre vos 2 projets. Copier votre dossier dans DEPOT.

63