Fiche TP 3 (Solution)

Activité 1 :

```
<RelativeLayout
 android:layout_width="match_parent"
 android:layout_height="match_parent">

<Button
   android:id="@+id/button"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:text="Un"
    />
 <Button
      android:id="@+id/button2"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="DEUX"
      android:layout_centerHorizontal="true"
       />

   <Button
      android:id="@+id/button3"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="TROIS"
      android:layout_alignParentRight="true"
      />

   <Button
      android:id="@+id/button4"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Quatre"
      android:layout_centerInParent="true"
      android:layout_alignParentLeft="true"
      />

   <Button
      android:id="@+id/button5"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Cinq"
      android:layout_centerVertical="true"
      android:layout_centerHorizontal="true"
      />

   <Button
      android:id="@+id/button6"
      android:layout_width="wrap_content"
```

```
        android:layout_height="wrap_content"
        android:text="Six"
        android:layout_centerInParent="true"
        android:layout_alignParentRight="true"
        />

    <Button
        android:id="@+id/button7"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Sept"
        android:layout_alignParentBottom="true"
        android:layout_alignParentLeft="true"
        />

    <Button
        android:id="@+id/button8"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Huit"
        android:layout_alignParentBottom="true"
        android:layout_centerHorizontal="true"
        />
    <Button
        android:id="@+id/button9"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Neuf"
        android:layout_alignParentBottom="true"
        android:layout_alignParentRight="true"
        />
        <ImageView
            android:id="@+id/imageView"
            android:layout_width="match_parent"
            android:layout_height="638dp"
            android:layout_below="@id/button2"
            app:srcCompat="@drawable/logo_univ"
/>
</RelativeLayout>
```

Remarque : il faut ajouter le fichier image nommé (logo_univ) dans le répertoire res->drawable de votre application

Attention : toute ressource ajoutée doit avoir un nom écrit en minuscule

Activité 2 :

```
<?xml version="1.0" encoding="utf-8"?>

<LinearLayout
 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"
   android:orientation="vertical"
   tools:context=".MainActivity"


   <TextView
      android:id="@+id/textView1"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Login" />

   <EditText
      android:id="@+id/editText1"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:hint="Entrer le login"


   <TextView
      android:id="@+id/textView2"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Password" />

   <EditText
      android:id="@+id/editTextTextPassword"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:ems="10"
      android:hint="Entrer le Password"
      android:inputType="textPassword"


   <LinearLayout
      android:layout_width= "match_parent"
      android:layout_height="wrap_content"
      android:weightSum="100"
      android:orientation="horizontal" >

      <Button
         android:id="@+id/button1"
         android:layout_width="wrap_content"
```

```
            android:layout_height="wrap_content"
            android:text="Envoyer"
            android:layout_weight="50" />

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

    </LinearLayout>

</LinearLayout>
```

Activité 3 :

Utilisez le même code de l'activité 2 avec les modifications suivantes :

- Dans le premier <LinearLayout> on ajoute l'attribut : android:background="@color/purple_200"
- Dans les deux **<EditText>** on ajoute l'attribut : android:background="@color/jaune"/>
- Sur le fichier colors.xml on ajoute cette ligne : <color name="jaune"> #FFFF00</color>

Activité 4 :

1. Les composants de cette application

Cette application contient les composants graphiques suivants :

- Une phrase dans le TextView « Entrez votre prénom »
- Un EditText, c'est ici qu'on donne le prénom.
- Un bouton « valider » pour faire apparaisse le prénom rentré
- De plus, on voudrait, de la clique sur valider, qu'un message s'affiche avec le prénom qui a été rentré

2. Les ressources de notre interface

Dans le dossier res/ values on va créer toutes les ressources de notre interface :

Couleurs, dimension, chaîne de caractères.

2.1. Chaine de caractères :

Ouvrir le fichier string.xml et ajouter les lignes suivantes :

<string name="app_name">Hello prenom</string>

Cette chaine de caractères « app_name » va être utilisée pour affichera le titre « Hello prenom » tout en haut de notre application.

<string name="prenom">Entrer votre prénom</string>

Cette chaine de caractères va être utilisée par le composant graphique «TextView », qui demande à l'utilisateur d'entrer son prénom

<string name="prenomHint">Tapez votre prénom</string>

Cette chaine de caractères « prenomHint » va être utilisé par l'« EditText », qui indique à l'utilisateur où taper son prénom

<string name="bouton">Valider</string>

Cette variable (chaine de caractères) bouton va être utilisé par le composant graphique « bouton » pour afficher le texte Valider.

Voilà pour les chaînes de caractères de notre interface, on va maintenant s'occuper des couleurs de notre application.

2.2. Les Couleurs :

Ouvrir le ficher colors.xml pour ajouter quelques couleurs en insérant les lignes suivantes :

<color name="couleurTitre"> #000000</color>

C'est la couleur qui va être utiliser pour le titre de notre application (Hello prenom)

<color name=" couleurFond"> #DDDDDD</color>
C'est la couleur qui va être utiliser pour le fond de notre application

<color name="couleurMessage">#221596</color>
C'est la couleur qui va être utiliser pour le texte qui affiche le prénom (ex :Hello Amine).

2.3 Les Dimension :

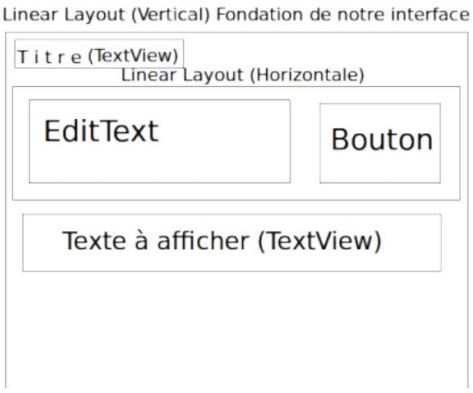
On va maintenant s'occuper de la dimension du texte qui affiche le prénom entré (qui est affiché en bleu). Donc on crée un nouveau fichier dans / res/ values, avec le nom : « dimension.xml »

Puis on ajoute sur ce fichier les lignes suivantes :

```
<resources>
    <dimension name=" dimMessage">30px</dimension>
</resources>
```

3. Interface

Maintenant on va créer l'interface de notre application en utilisant les ressources qu'on a créé précédemment :



3.1- **Linear Layout Vertical** :

On crée tout d'abord un « **Linear Layout** » **vertical** qui contient tous les autres composants graphiques :

```
<LinearLayout
android:orientation="vertical"
 android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:background="@color/ couleurFond" >
```

Pour définir la couleur du fond on a ajouté cette ligne :
android:background="@color/ couleurFond"

- « android:background » est la propriété de l'arrière-plan du Linear Layout.
- « @color/ couleurFond » est la ressource (couleur) que nous avons ajouté précédemment qui contient la couleur de fond.

TextView

Ensuite on place le TextView qui contiendra le titre :


```
<TextView
android:id="@+id/ TextViewPrenom"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="@string/ Prenom"
android:textColor="@color/ couleurTitre" / >
```

Là on est en train de créer un composant graphique « TextView »:

On lui donne aussi un identifiant unique avec lequel on l'appellera dans la partie java.

android:id="@+id/ TextViewPrenom"

La variable possède comme id : « TextViewPrenom »

Ensuite on lui donne des dimensions :

Largeur :
android:layout_width="fill_parent"

Avec « fill_parent », le TextView prend toute la place disponible en largeur.

Longueur :
android:layout_height="wrap_content"

Avec « wrap_content », le TextView prend seulement la place qu'il a besoin.

On lui affecte ensuite le texte qu'on a entré tout à l'heure dans la variable Prenom :

android:text="@string/ Prenom"

3.2- **Linear Layout Horizontal** :

<LinearLayout

android:orientation="horizontal"

android:layout_width="fill_parent"

android:layout_height="wrap_content"

>

…code …

</ LinearLayout>

On lui donne l'orientation horizontale :

android:orientation="horizontal"

Et on lui donne aussi des dimensions :

android:layout_width="fill_parent"

android:layout_height="wrap_content"

3.3 EditText

A l'intérieur de Linear Layout horizontal on insère un « EditText », là où l'utilisateur rentre son prénom :

<EditText

android:id="@+id/ EditTextPrenom"

android:layout_width="wrap_content"

android:layout_height="wrap_content"

android:layout_weight="1"

android:layout_gravity="bottom"

android:hint="@string/ prenomHint"

/ >

C'est le même principe que la TextView. On lui donne un identifiant

«EditTextPrenom » :

android:id="@+id/ EditTextPrenom"

Puis des dimensions.

android:layout_width="wrap_content"

android:layout_height="wrap_content"

On lui donne aussi une priorité par rapport au bouton qu'on va ajouter après.

L'EditText et le bouton étant sur la même ligne, on attribue un poids prioritaire a

l'EditText, pour qu'il prenne plus de la moitié de l'écran :

android:layout_weight="1"

Ensuite on le positionne en bas du Linear Layout :

android:layout_gravity="bottom"

Et on lui affiche une phrase par défaut (« Tapez votre prénom ... ») dont on a créé la valeur plus tôt :

android:hint="@string/ prenomHint"

La propriété « hint » permet de cacher cette chaine de caractères lorsque l'utilisateur tape quelque chose sur son clavier.

Bouton

Pour le composant Button on retrouve le même principe :

<Button android:id="@+id/ ButtonEnvoyer "

android:layout_width="wrap_content"

android:layout_height="wrap_content"

android:text="@string/ bouton"

/ >

Assignation d'un ID :

android:id="@+id/ ButtonEnvoyer

Des dimensions :

android:layout_width="wrap_content"

android:layout_height="wrap_content"

Et un texte sur le bouton qu'on a créé précédemment (« Valider ») :

android:text="@string/ bouton"

TextView prénom

Après avoir fermé le Linear Layout horizontal :

</ LinearLayout>

On ajoute un TextView qui va afficher le prénom qu'on aura rentré dans l'EditText lorsqu'on aura cliqué sur le bouton.

C'est toujours le même procédé :

<TextView android:id="@+id/ TextViewHello"

android:layout_width="wrap_content"

android:layout_height="fill_parent"

android:layout_gravity="center_horizontal"

android:textSize="@dimen/ dimMessage"

android:textColor="@color/ couleurMessage"

/ >

Un ID :

android:id="@+id/ TextViewHello"

Des dimensions :

android:layout_width="wrap_content"

android:layout_height="fill_parent"

On le positionne au centre :

android:layout_gravity="center_horizontal"
Ici on applique une dimension en utilisant la variable qu'on a créé précédemment

(30px) : android:textSize="@dimen/ dimMessage"

Et on donne une couleur au texte qui s'affiche (« #221596 ») :

android:textColor="@color/ couleurMessage"

Et enfin, la dernière ligne doit fermer le Linear Layout Vertical.

</ LinearLayout>

Dans cette partie XML, on a vu comment créer des variables (dimensions, textes, couleur) dans « res/ values » et comment créé l'interface principal de notre application (le fichier xml dans « res/ layout ») à l'aide de LinearLayout et d'autre composants (Bouton, EditText, TextView).

4- JAVA :

Après avoir réalisé la partie interface (XML), on va réaliser la partie JAVA de l'application.

4.1. Composants

On déclare d'abord nos composants dynamiques (Bouton, EditText et chaine de caractères qui affichera le prénom) :

```
private EditText editText;
private Button button;
private String prenom;
```

Ce qui donne :

```
public class MainActivity extends AppCompatActivity {
private EditText editText;
private Button button;
private String prenom;
…
}
```

4.2 Evenement onCreate()

On va ensuite créer la méthode qui s'exécutera à la création de l'activity MainActivity (On met en paramètre une variable de type Bundle qui est utile en cas de fermeture non prévu de l'activity) :

```
protected void onCreate(Bundle savedInstanceState) {
```

Ensuite on initialise la création :

```
 super.onCreate(savedInstanceState);
```

Puis on affiche le Layout qu'on a créé en XML :

```
setContentView(R.layout.activity_main);
```

4.3 Instanciation des composants

Ensuite on va instancier l'EditText et le bouton :

```
editText = (EditText) findViewById(R.id.EditTextPrenom);
button = (Button) findViewById(R.id.ButtonEnvoyer);
```

La méthode findViewById va permettre d'instancier un composant par rapport a celui créer dans le fichier XML (rappelez-vous : <EditText android:id="@+id/ EditTextPrenom", l'ID EditTextPrenom est réutilisé dans cette méthode).

À noter que l'on a réalisé un « cast » :(EditText), (Bouton). Ceci oblige la méthode précédente a retourné un objet de type EditText ou Bouton.

4.4 Listener

Ensuite on crée un « écouteur » sur le bouton :

```
button.setOnClickListener(
new OnClickListener() {
public void onClick(View v) {
```

A l'intérieur de la fonction onClick, on placera le code qui permet d'afficher le texte remplie dans l'EditText.

La chaîne rentrée par l'utilisateur :

Ensuite on retourne ce que l'utilisateur a rentré dans l'EditText dans la variable de type String crée précédemment :

```
prenom = editText.getText().toString();
```

4.5. Pop-up

Avec la fonction suivante, on va pouvoir afficher dans un pop-up le nom rentré dans l'EditText :

```
Toast.makeText(mainActivity.this, "Bonjour " + prenom + " !",
Toast.LENGTH_LONG).show();
```

Pour afficher un pop-up, on utilise cette fonction :

Toast.makeText(Context context, CharSequence text, int duration) ;

Le 1er paramètre (Context context) :

Ici on place le pop-up dans l'activity actuelle (MainActivity.this).

Le 2ème (CharSequence text) :

Ici on rentre le texte que l'on veut afficher ("Bonjour " + prenom + " !").

Donc on concatène le texte « Bonjour » avec la phrase dans l'EditText puis un " !"

Donc tout d'abord, on choisit la TextView qui affichera la phrase puis on définit le texte.