Institut Supérieur d'Informatique et de Mathématiques de Monastir

Année universitaire : 2023-2024

SERIE DE TP N° 2

Matière : Programmation Mobile

Sections : Ing Info 1

Enseignant : M. BEN SALEM

ÉLEMENTS GRAPHIQUES

Objectif :

Le but de ce TP est créer et manipuler des éléments graphiques de base dans Android.

I. View et ViewGroup :

I.1 Le Composant View

La classe View représente la classe de base pour la création d'une interface graphique en Android. C'est la classe mère de tous les *widgets* (éléments graphiques), utilisés pour créer des composants graphiques interactifs (boutons, champs texte, champs de saisie…).

Une vue occupe un espace rectangulaire sur l'écran, responsable de la gestion des évènements initiés par l'utilisateur.

Une sous-classe ViewGroup est définie par Android, représentant la classe de base pour tous les *layouts* (dispositions), qui sont des conteneurs invisibles qui rassemblent plusieurs View ou ViewGroup, et définissent leur emplacement dans l'écran (détaillés dans la section *I.2 Les Layouts*).

I.2 Les Layouts

Chaque élément d'une interface graphique est soit un objet View, soit ViewGroup. Les dispositions de ces éléments dans l'écran sont précisées principalement par des *Layouts*.

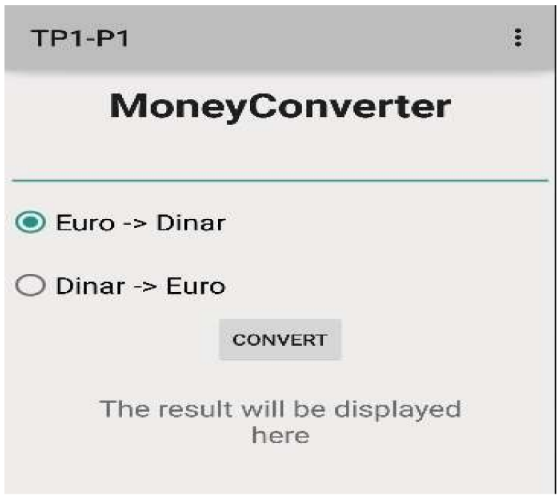
Un Layout est un ViewGroup qui regroupe un ensemble de widgets ou d'autres layouts, permettant ainsi de définir leur disposition sur l'écran de la fenêtre principale. Parmi les

layouts existants, on cite le *FrameLayout*, *LinearLayout*, *RelativeLayout*…

II. Exercice :

II.1 Objectif

L'objectif de cet exercice est de réaliser une application simple de conversion de monnaie, ayant une interface semblable à ce qui suit (veiller à respecter les tailles et emplacements des éléments graphiques):



Activité 1. Commencer par créer l'interface graphique, identique à la figure qui précède. Premièrement **la vue est créée en XML**, ensuite **elle est créée en Java**. Puis, lancer l'émulateur, et vérifier que le rendu correspond à ce qui est demandé.

Dans ce qui suit, nous allons montrer les étapes à suivre pour définir le comportement de cette interface.

II.2 Comportement du bouton

Soit le bouton défini dans le fichier main.xml, dont l'identifiant est *b_convert*. Pour manipuler ce bouton, trois méthodes principales sont en général utilisées :

II.2.1 Méthode 1 : Surcharge du Listener du bouton

En Java, un Listener (écouteur) est un objet permettant au programmeur de réagir suite aux actions de l'utilisateur (clic de souris, touche du clavier, etc.). Dans notre cas, nous avons besoin d'un écouteur pour le clic sur le bouton, appelé *onClickListener*. *Listener* est une

interface, qui fournit des méthodes qui doivent être implémentées par le programmeur. Par exemple, le onClickListener contient une méthode *onClick*, qu'on doit implémenter pour définir le comportement de notre bouton.

1. Créer un attribut dans votre activité de type Button :

```
private Button bConvert;
```

2. Dans la méthode onCreate(), initialiser l'attribut bAfficher en lui associant le bouton créé dans le main.xml :

```
this.bConvert = (Button) this.findViewById(R.id.b_convert) ;
```

3. Utiliser le code suivant pour définir le comportement du bouton bConvert (de préférence à l'intérieur de la méthode onCreate).

```
this.bConvert.setOnClickListener(new OnClickListener() {
        @Override
        public void onClick(View v) {
        //comportement de votre bouton
        }
});
```

II.2.2 Méthode 2 : Définition d'une méthode propre au bouton

Il existe une manière moins encombrante de définir le comportement du bouton, mais qui ne peut pas s'appliquer à tous les évènements, seulement au clic.

- Dans l'élément XML du bouton, ajouter l'attribut :

```
android:onClick="convert"
```

Cet attribut indique qu'il existe une méthode dans le code de l'activité, appelée *convert*, qui définit le comportement du bouton au clic.

- Dans le code de l'activité, ajouter la méthode suivante :

```
public void convert(View v){
        // comportement du bouton
}
```

La méthode *convert* ainsi définie a une signature particulière : elle doit obligatoirement être publique, retourner void et prendre comme paramètre un objet de type android.view.View (vous remarquerez que cette méthode a la même signature que la méthode onClick, surchargée dans la première méthode). Il faut noter également que la vue *v* passée en paramètre, correspond à l'objet cliqué.

Remarque : Si vous utilisez cette solution, il est inutile de définir une variable de type Button en Java, et de l'associer au bouton défini dans le fichier XML.

II.2.3 Méthode 3 : Implémentation de l'interface OnClickListener

Il est possible d'utiliser l'héritage pour surcharger la méthode *onClick*, sans passer par l'appel à la méthode *setOnClickListener*. Il suffit de suivre les étapes suivantes :

1. Votre activity doit implémenter l'interface *OnClickListener*. Ceci est réalisé en transformant la signature de votre classe activité comme suit, par exemple :

```
public class MoneyConverter extends Activity implements OnClickListener {…}
```

2. Créer l'attribut *bConvert* de type Button et l'associer à l'élément XML *b_convert*, comme dans la méthode 1.

3. Définir l'activité courante comme étant l'écouteur du clic sur le bouton bConvert :

```
bConvert.setOnClickListener(this);
```

4. Ajouter la méthode onClick dans votre activité, comme suit :

```
public void onClick(View v) {
        if (v.getId()==R.id.b_convert){
          //Comportement du bouton b_convert
          }
}
```

Attention, cette méthode sera commune à tous les éléments cliquables, il faut donc distinguer le comportement selon l'identifiant de l'élément cliqué.

Cette méthode peut s'avérer utile dans le cas où on voudrait par exemple regrouper les implémentations de tous les boutons de l'interface dans la même méthode, ou si plusieurs boutons partagaient une partie de leur comportement.

II.3 Comportement d'un EditText

Un EditText est un objet graphique qui permet à l'utilisateur de saisir une chaîne de caractères, utilisable par l'application.

De même que pour le bouton (ainsi que tous les éléments graphiques que nous désirons utiliser dans notre application), nous devons définir et initialiser un objet Java associé au champs EditText :

1. Créer un attribut dans votre activité de type EditText:

```
private EditText eEntry;
```

2. Dans la méthode onCreate(), initialiser l'attribut eEntree en lui associant le champs de saisie créé dans le main.xml :

```
this.eEntry = (EditText) this.findViewById(R.id.e_entry) ;
```

3. Définir son comportement. Pour un champs de saisie, les principales fonctionnalités sont :
    - La lecture du contenu : pour cela on utilise la méthode :

```
String s = eEntry.getText().toString();
```

    - La modification du contenu :

```
eEntry.setText("Nouveau texte");
```

II.4 Comportement d'un TextView

Un TextView est un objet graphique qui permet d'afficher une chaîne de caractères non-éditable par l'utilisateur.

Le TextView peut être utilisé par l'application exactement de la même manière qu'un EditText.

II.5 Comportement d'un Bouton Radio

Un bouton radio est un bouton à deux états qui peut être soit coché (*checked*) ou décoché (*unchecked*). Les boutons radios sont en général utilisés dans un groupe *RadioGroup*. Au sein d'un même groupe, un seul bouton radio peut être coché.

Pour gérer l'état d'un bouton radio, il faut suivre les étapes suivantes :

1. Créer un attribut de type RadioButton dans votre activité (par exemple rDinarEuro).
2. L'associer au bouton radio approprié de votre interface en utilisant la méthode *findViewById*.
3. Pour tester l'état de votre bouton radio, appeler la méthode isChecked(). Par exemple :

```
if (rDinarEuro.isChecked() ){

//traitement
}
```

4. Pour mettre à jour l'état du bouton radio, utiliser la méthode *setChecked(boolean etat)*. Par exemple, si on veut cocher l'élément radio1 et décocher radio2, on peut faire comme suit :

```
radio1.setChecked(true)
radio2.setChecked(false);
```

Remarques :

1. Pour utiliser les boutons radios, il est inutile de définir des variables en Java pour le RadioGroup.
2. Dans l'exemple précédent (4), si radio1 et radio2 se trouvent dans un RadioGroup, il est inutile de changer leurs deux états : le changement de l'état de l'un va automatiquement changer l'autre, puisqu'un seul peut être coché à la fois.

Activité 2. Définir maintenant le comportement des différents composants de votre interface en utilisant la méthode de votre choix. Actuellement (15 Septembre 2023), 1 € = 3.36 TND