



Chapitre I V : Les Interfaces graphiques sous Android

Plan du Chapitre



Définitions



Les composants graphiques



Les conteneurs (Layout)



Les composants (Widget)

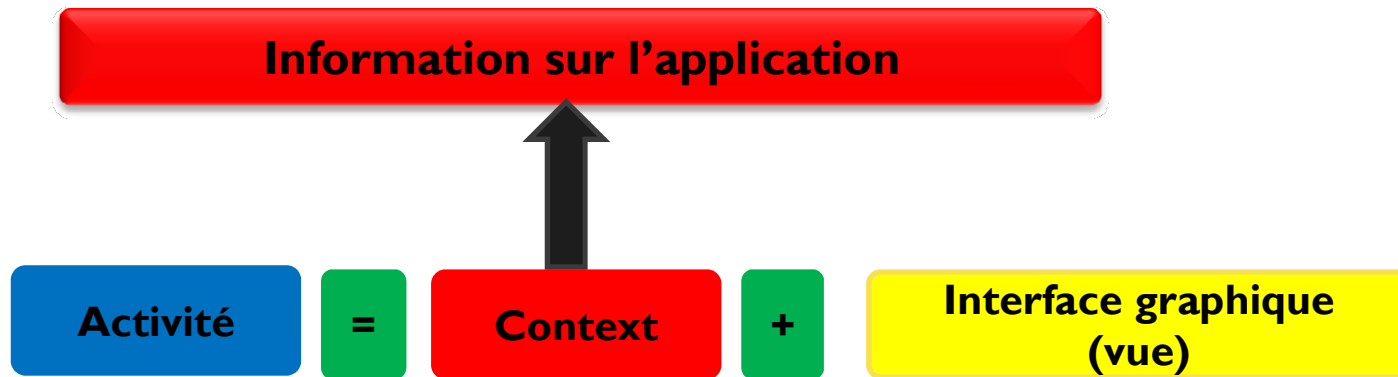


La gestion des événements

Définitions

- Une interface graphique est **un moyen de communication** entre un **utilisateur** et une **application**. Elle contient un ensemble de composants graphiques.
- Deux catégories de composants graphiques :
 - ✓ *android.view.**View***
 - ✓ *android.view.**ViewGroup***.
- **ViewGroup** réalise deux tâches :
 - ✓ *Regrouper un ensemble de composants*
 - ✓ *Responsable de la disposition des composants (*LayoutManager* de AWT).*

Définitions



- L'interface peut être déclarée :
 - ✓ En XML
 - ✓ Dans le code java de l'activité



Composants : XML VS Java

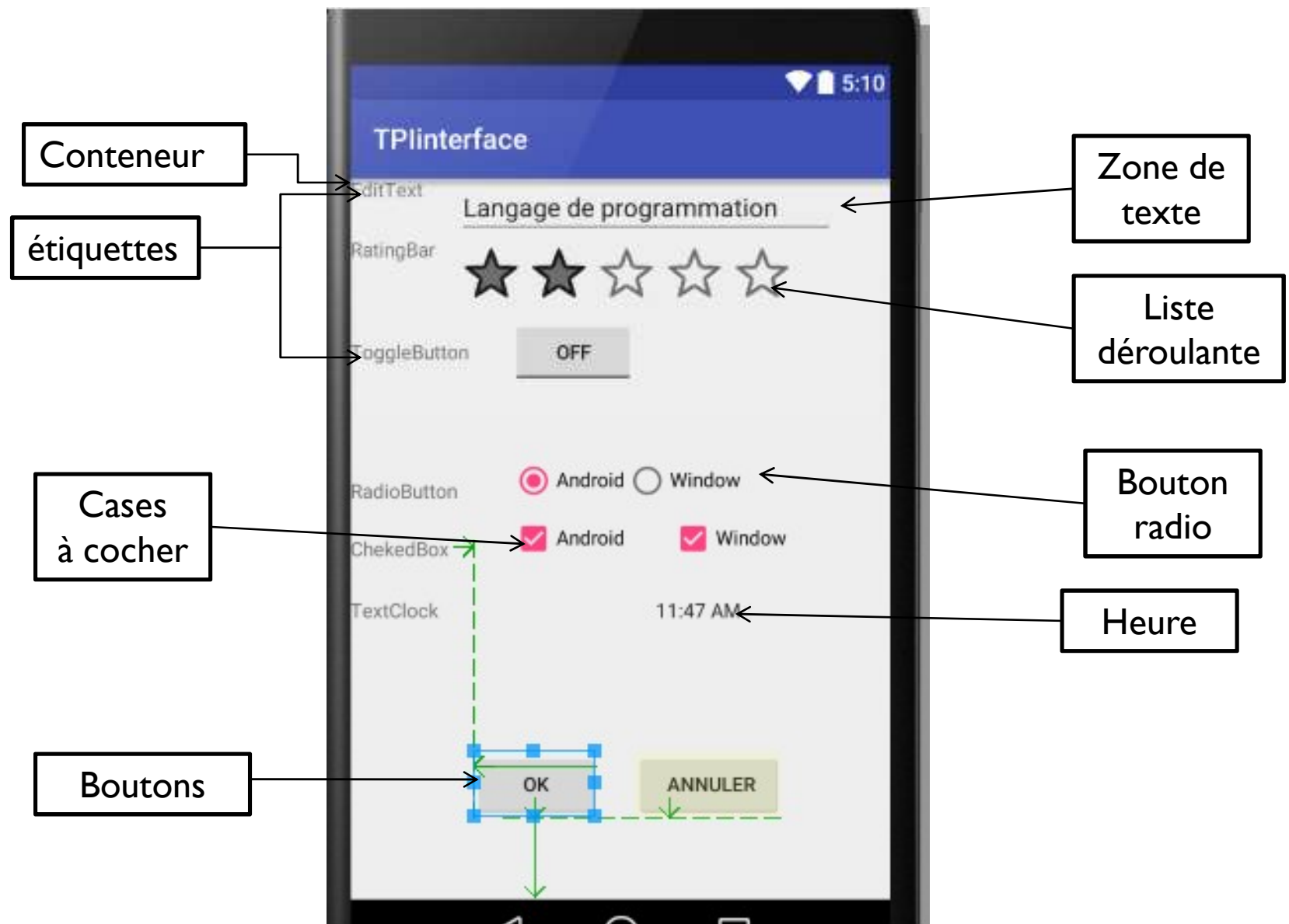
```
public class ExempleActivity extends Activity {  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        LinearLayout conteneur = new LinearLayout(this);  
        conteneur.setOrientation(LinearLayout.VERTICAL);  
        TextView texte = new TextView(this);  
        texte.setText(" une interface par un programme ");  
        conteneur.addView(texte);  
        setContentView(conteneur);  
    }  
}
```

JAVA

```
<?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">  
    <TextView  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        android:text="Une interface en XML"  
        android:id="@+id/text" />  
</LinearLayout>
```

XML

Exemple d'interface



Les conteneurs (ViewGroup)

- Un **viewGroup** est un conteneur
- Son rôle est de regrouper un ensemble de composants et de gérer leur disposition (Layout manager)
- Un conteneur doit étendre la classe ViewGroup
- Chaque vue doit spécifier :
 - *android:layout_height*
 - *android:layout_width*
 - Elles prennent l'une des valeurs suivantes : **match_parent**, **wrap_content** ou un **nombre exact** exprimé en (dp, sp, px, etc).

Les conteneurs (ViewGroup)

- Quelques conteneurs (layouts) pré-définis dans Android :
 - **LinearLayout**
 - **RelativeLayout**
 - **TableLayout**
 - **FrameLayout**
 - **AbsoluteLayout**
- Un conteneur peut être déclaré à l'intérieur d'un autre conteneur

Les conteneurs (LinearLayout)

- Le **LinearLayout** aligne les composants de haut en bas ou de gauche à droite.
- La propriété «*android:orientation*» spécifie cet alignement :
 - Vertical
 - Horizontal (par défaut)
- Ou par la méthode **setOrientation(int orientation) : HORIZONTAL ou VERTICAL**
- **LinearLayout** contient deux attributs supplémentaires :
 - *android:layout_gravity* (**setGravity(int)**)
 - *android:layout_weight* (**setWeight(int)**)

LinearLayout: exemple

```
<?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">

  <Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Bouton 1"
    android:id="@+id/Bouton1"
    android:layout_gravity="left|center"/>

  <Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Bouton 2"
    android:id="@+id/Bouton2"
    android:layout_weight="2" />

</LinearLayout>
```



Les conteneurs (RelativeLayout)

- Le **RelativeLayout** aligne les composants par rapport au conteneur parent et/ou par rapport aux autres composants frères.
- Parmi les propriétés utilisées par ce conteneur :
 - *android:layout_alignParentRight*
 - *android:layout_alignParentTop*
 - *android:layout_toRightOf*
 - *android:layout_above*
 - *android:layout_alignLeft*
 - *android:layout_alignBaseline*

RelativeLayout: exemple

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

    <EditText
        android:id="@+id/username"
```

```
        android:layout_alignParentRight="true"
        android:layout_toRightOf="@+id/usernameLabel"
```

```
    </EditText>
```

```
    <TextView
```

```
        android:layout_alignBaseline="@+id/username"
```

```
        android:id="@+id/password"
        android:text="password"
        android:inputType="textPassword"
```

```
        android:layout_below="@+id/username"
        android:layout_alignParentRight="true"
        android:layout_toRightOf="@+id/usernameLabel"
```

```
    </EditText>
```

```
    <TextView
```

```
        android:layout_alignBaseline="@+id/password"
```

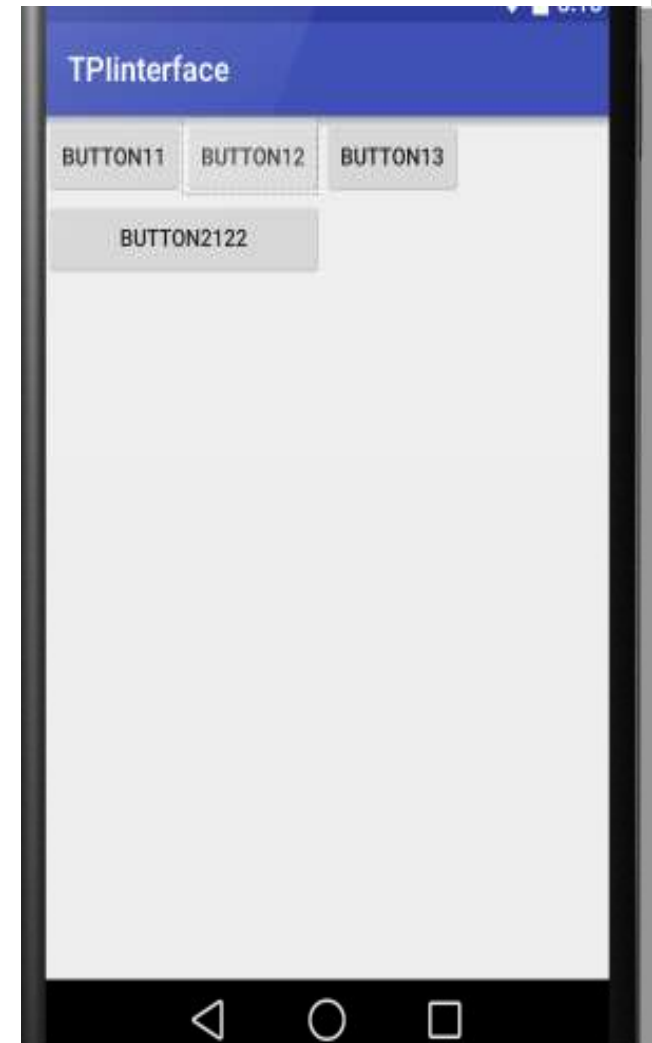


Les conteneurs (TableLayout)

- Le **TableLayout** aligne les composants sous forme d'une matrice (lignes et colonnes).
- Chaque ligne est déclarée à l'intérieur de l'élément **<TableRow>**
- Parmi les propriétés utilisées par ce conteneur :
 - *android:layout_column*
 - *android:layout_span*
 - *android:collapseColumns*
 - *android:stretchColumns*
 - *android:shrinkColumns*

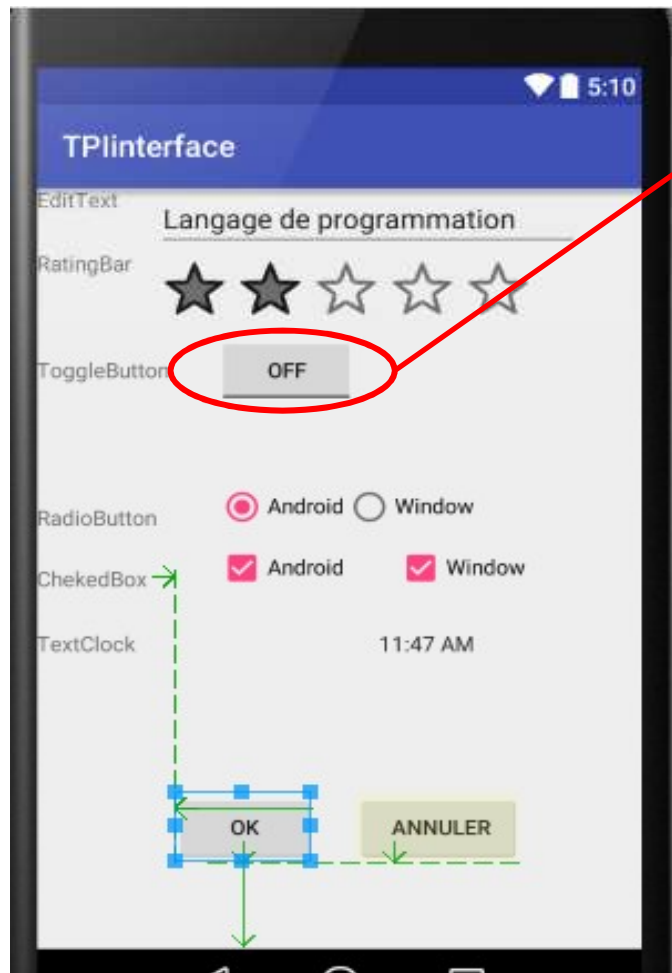
TableLayout: exemple

```
<?xml version="1.0" encoding="utf-8"?>
<TableLayout
    xmlns:android=http://schemas.android.com/apk/res/android
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:id="@+id/tableLayout" >
    <TableRow android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/firstRow">
        <Button android:id="@+id/button1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Button11" />
        <Button android:id="@+id/button2"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:text="Button12" />
        <Button android:id="@+id/button3"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:text="Button13"
        </Button>
    </TableRow>
    <TableRow android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/secondRow">
        <Button android:layout_column="0"
            android:layout_span="2"
            android:id="@+id/button4"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Button212">
        </Button>
    </TableRow>
</TableLayout>
```



Les composants graphiques (contrôle)

Composants: éléments de base de construction d'interface

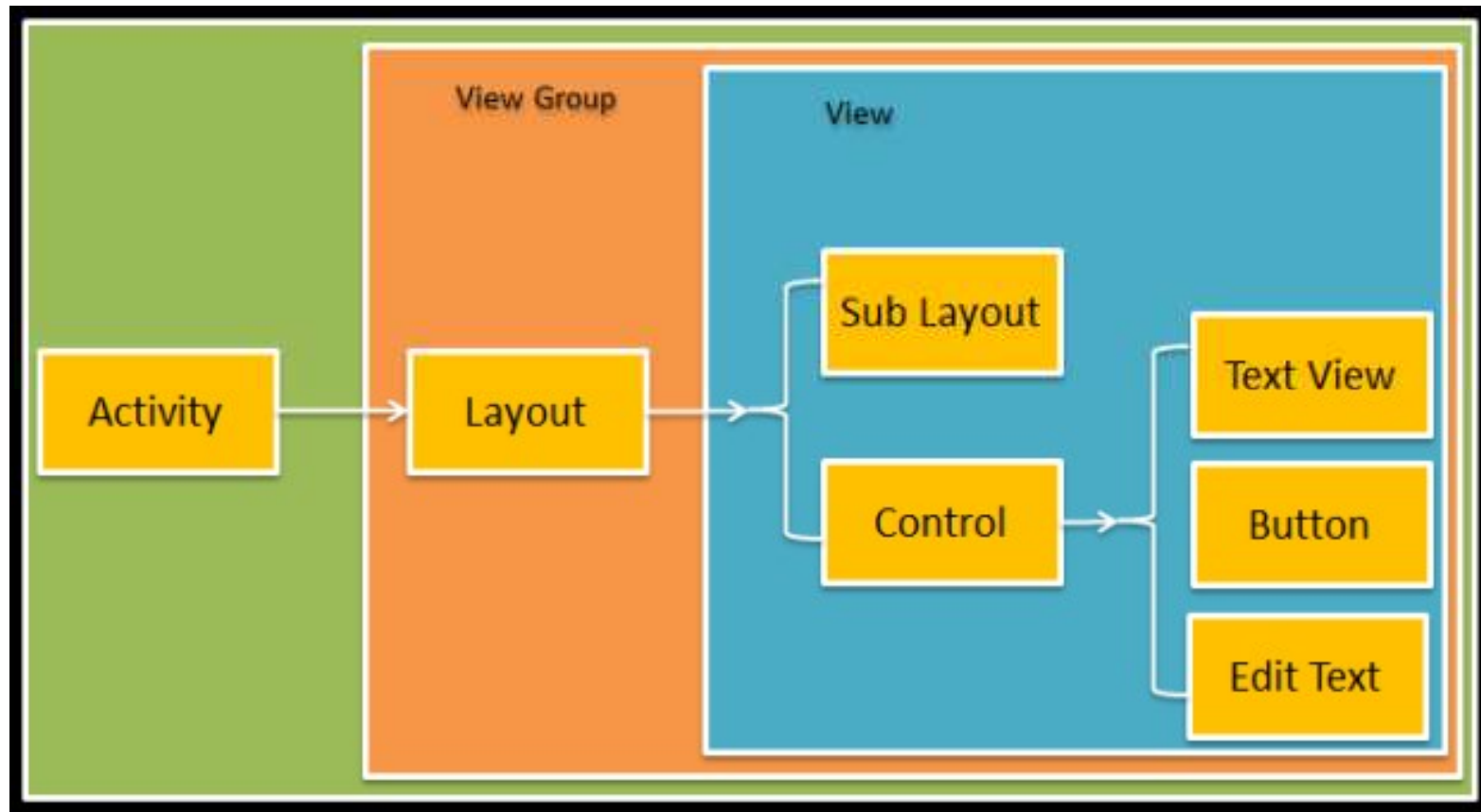


- ✧ Surface rectangulaire à l'écran
- ✧ Responsable de dessin
- ✧ Responsable de **gestion d'événement**

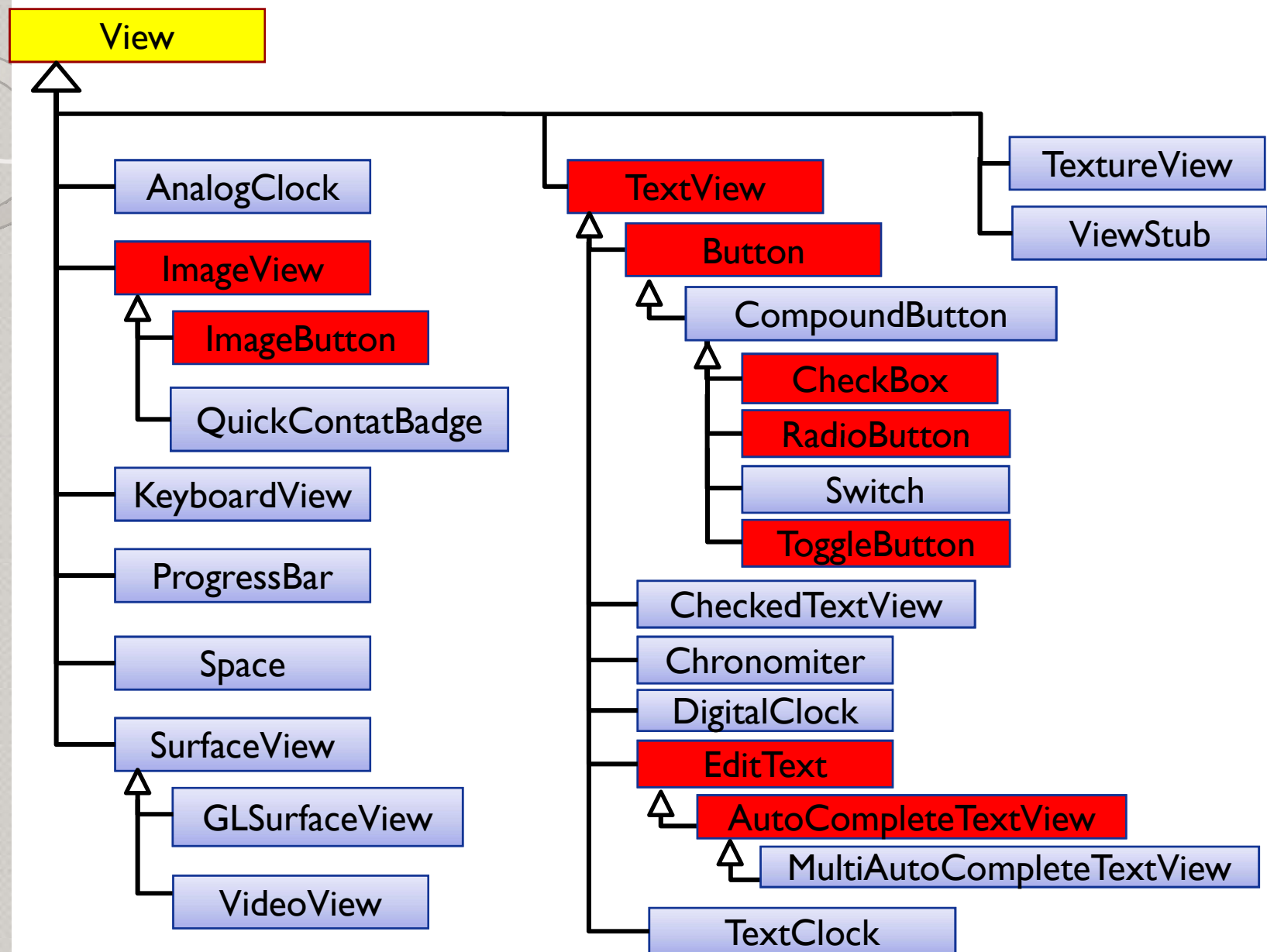
EXEMPLES de composants:

- GoogleMap
- WebView
- **Widgets** → Voir Ci-après
- ...
- Vues définies par l'utilisateur

Les views (Composants)



Hiérarchie des composants



La classe View

- La classe **View** est la classe mère de toutes les classes des composants graphiques.
- parmi les propriétés et méthodes que contient cette classe :

attributs	Signification
android:visibility	Contrôle la visibilité initiale du widget
android:background (android:src)	Permet de spécifier la couleur de fond au format RGB ou une image

Méthodes	Signification
setEnabled()	Permet de basculer entre l'état actif et inactif d'un widget
isEnabled()	Permet de tester est ce qu'un widget est actif
requestFocus()	Permet de demander le focus pour le widget
isFocused()	Permet de tester est ce que le widget a le focus
getParent()	Renvoie le widget ou le conteneur parent
getRootView()	Renvoie la racine de l'arborescence

Composants : XML VS Java

```
public class ExempleActivity extends Activity {  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        LinearLayout conteneur = new LinearLayout(this);  
        TextView texte = new TextView(this);  
        texte.setText(" une interface par un programme ");  
        conteneur.addView(texte);  
        setContentView(conteneur);  
    }  
}
```

JAVA

```
<?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">  
    <TextView  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        android:text="Une interface en XML"  
        android:id="@+id/text1" />  
</LinearLayout>
```

XML

```
public TextView text;  
text=(TextView)findViewById(R.id.text1);
```

Label (TextView)

Appelé aussi **Label** est une zone de texte non modifiable

attributs importants	Méthodes Java
android:text : texte du label	setText(CharSequence)
android:textStyle : (bold, italic , bold_italic)	setTextStyle(Style)
android:textSize : (Taille en sp)	setTextSize(float)
android:textColor : (en RGB :#FF0000)	setTextColor(ColorStateList)

```
<?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">
  <TextView
    android:text="Exemple de TextView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"

    android:textStyle="bold"
    android:textColor="#FF0000"
    android:textSize="30sp"/>
</LinearLayout>
```

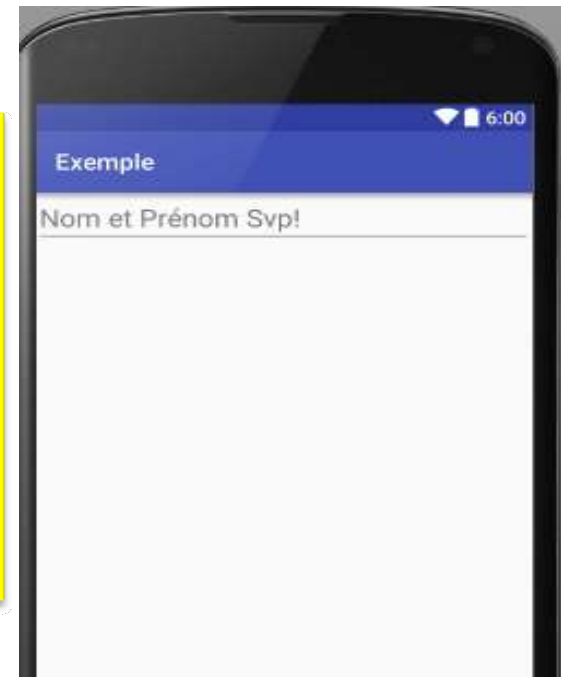


Champ de Saisie (EditText)

- Permet à l'utilisateur de saisir un texte, représenté par **EditText**
- C'est une sous classe **TextView**

Attributs	Signification
android:autoText	Indique si le champ effectue une correction automatique de l'orthographe
android:capitalize	Mettre la première lettre en majuscule
android:digits	Le champ n'accepte que certains chiffres
android:singleLine	Indique si le champ est sur une seule ligne ou plusieurs
android:password	Contrôle la visibilité du champ
android:phoneNumber	Formater le champ pour des numéro du téléphone

```
<EditText
    android:id="@+id/txtUserName"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:textSize="22sp"
    android:autoText="true"
    android:capitalize="words"
    android:hint="Nom et Prénom Svp!"
/>
```



Bouton (Button)

- Un bouton permet la simulation de l'action de clic sur une interface
- Hérite de la classe TextView

```
<Button  
    android:id="@+id/btnExit"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text="Quitter Application"  
    android:textSize="24sp"  
    android:textStyle="bold"  
    android:gravity="center_horizontal"  
  
</Button>
```



CompoundButton (CheckBox)

- Est un bouton avec deux états (**checked/unchecked**)
- Les composants : **CheckBox**, RadioButton, ToggleButton

isChecked()	pour savoir si la case est cochée
setChecked()	pour forcer la case dans l'état coché ou décoché
toggle()	pour inverser l'état de la case

<CheckBox

```
android:layout_width="wrap_content"  
android:layout_height="wrap_content"  
android:id="@+id/buttonCheck"  
android:text="CheckBox"  
android:checked="true" />
```

<CheckBox>



CompoundBouton (RadioButton)

- Un bouton Radio est généralement placé dans un **RadioGroup** (où un seul bouton puisse être sélectionné à un instant donné)
- Hérite de la classe CompoundButton

```
<RadioGroup
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="vertical">
    <RadioButton
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/buttonRadio"
        android:text="Button Radio 1"
        android:checked="true"/>
    <RadioButton
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/buttonRadio"
        android:text="Button Radio 2"
        android:checked="false"/>
</RadioGroup>
```



CompoundBouton (ToggleButton)

- Hérite de la classe CompoundButton

android:textOn	Label pour l'état activé
android:textOff	Label pour l'état non activé

```
<ToggleButton
  android:layout_width="264dp"
  android:layout_height="wrap_content"
  android:id="@+id/toggleButtonId"
  android:textOn="Button ON"
  android:textOff="Button OFF"
  android:checked="true"
  android:layout_gravity=
"center_horizontal">
</ToggleButton>
```



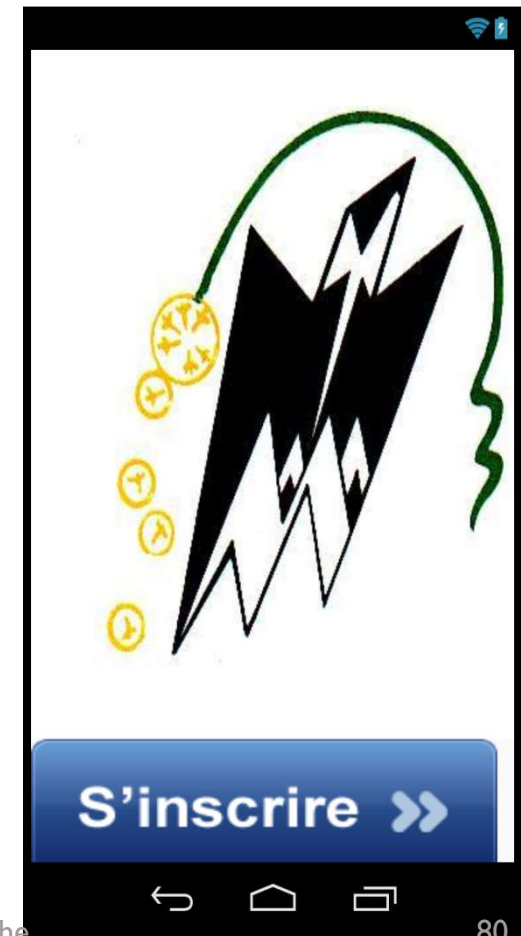
Images (ImageView et ImageButton)

- Héritent de la classe View. Permet d'insérer des images
- Sont analogues à TextView et Button respectivement

android:src ; spécifie l'image utilisée (de type drawable généralement); on peut utiliser l'attribut **android:background**

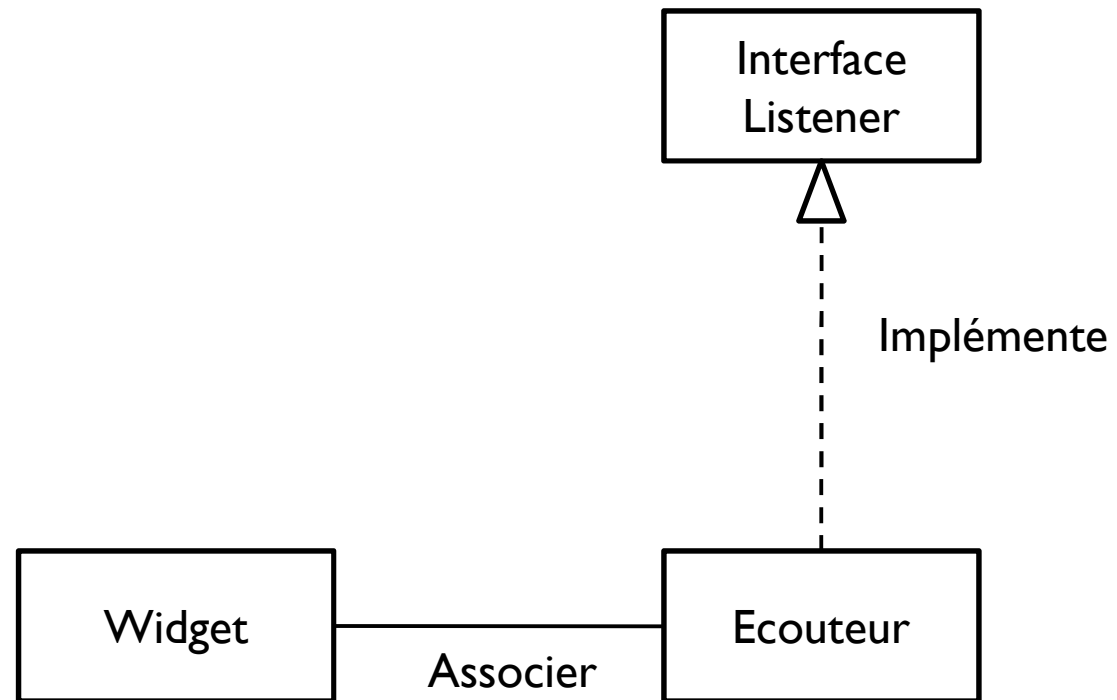
setImageURI().

```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="vertical" >
    <ImageView
        android:id="@+id/myImageView1"
        android:layout_width="313dp"
        android:layout_height="349dp"
        android:background="@drawable/ummto" />
    <ImageButton
        android:id="@+id/monImageBtn1"
        android:layout_width="254dp"
        android:layout_height="131px"
        android:background="@drawable/inscription" />
</LinearLayout>
```



La gestion des événements

- Une interface (IHM) **interagit** avec l'utilisateur en réponse aux **actions** de ce dernier.
- Les **événements peuvent être** : clic, long clic, selection d'un item, cocher un item , etc.
- La gestion des événements peut être réalisée :
 - Via l'utilisation d'XML
 - **Par l'utilisation des écouteurs d'événements (java : recommandée)**





Gestion des événements

1. Un événement se produit sur un composant
2. La source d'événement dans laquelle il se produit génère un objet de type événement
3. La source transmet l'événement à son écouteur
4. L'écouteur appelle la méthode correspondant au type d'événement
5. La méthode en question spécifie les traitements à réaliser lorsqu'un événement du type correspondant se produit
6. Dans ses traitements, la méthodes peut examiner les caractéristiques de l'événement et adapter son comportement en fonction de ces caractéristiques

ONCLICK event

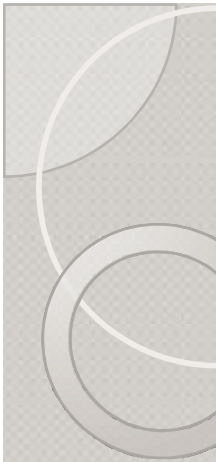


Le code Java qui
gère l'événement **onClick**



- 1 Implémenter l'interface nécessaire (**OnClickListener**) dans l'activité courante
- 2 Implémenter la méthode (onClick(View v))
- 3 Associer l'écouteur au bouton en utilisant la méthode **setOnClickListener()**

```
public class ExempleActivity extends Activity implements OnClickListener {  
    ...  
    Button bouton=(Button)findViewById(R.id.buttonNext);  
    bouton.setOnClickListener(this);  
    ...  
    public void onClick(View v) {  
        // code à executer lors de clic sur le bouton  
    }  
}
```



Catégorie	Interface	Méthode
	OnClickListener	<i>onClick()</i>
	OnLongClickListener	<i>onLongClick()</i>
	OnFocusChangeListener	<i>onFocusChange()</i>
	KeyListener	<i>onKey()</i>
	OnCheckedChangeListener	<i>onCheckedChanged()</i>
	OnItemSelectedListener	<i>onItemSelected()</i>
	TouchListener	<i>onTouch()</i>