

# SIL1

## Programmation mobile sous Android

---

Introduction à la création graphique

Introduction à la gestion des événements

Basé sur les transparents de Jérôme David

SIMO : Marc VASSOILLE [marc.vassoille@iut2.upmf-grenoble.fr](mailto:marc.vassoille@iut2.upmf-grenoble.fr)

MIAM : Francis Brunet-Manquat [Francis.Brunet-Manquat@iut2.upmf-grenoble.fr](mailto:Francis.Brunet-Manquat@iut2.upmf-grenoble.fr)

# Androïd

---

- ✓ Système d'exploitation + plateforme logicielle
  - OS Linux
  - Bibliothèques (système, SGBD, OpenGL, etc.)
  - Applications (navigateur, appareil photo, etc.)
  - Environnement de développement (SDK)

# Androïd et Java

---

- ✓ Environnement de programmation JAVA spécial :
  - NON compatible Java ME (application embarquée)
  - Machine virtuelle particulière : **Dalvik**
    - ◆ optimisée pour les mobiles (peu de mémoire, etc.)
    - ◆ Fichiers bytecode (.class) remplacés par des fichiers .dex
    - ◆ ... et aussi libre de la licence SUN (licence Apache)

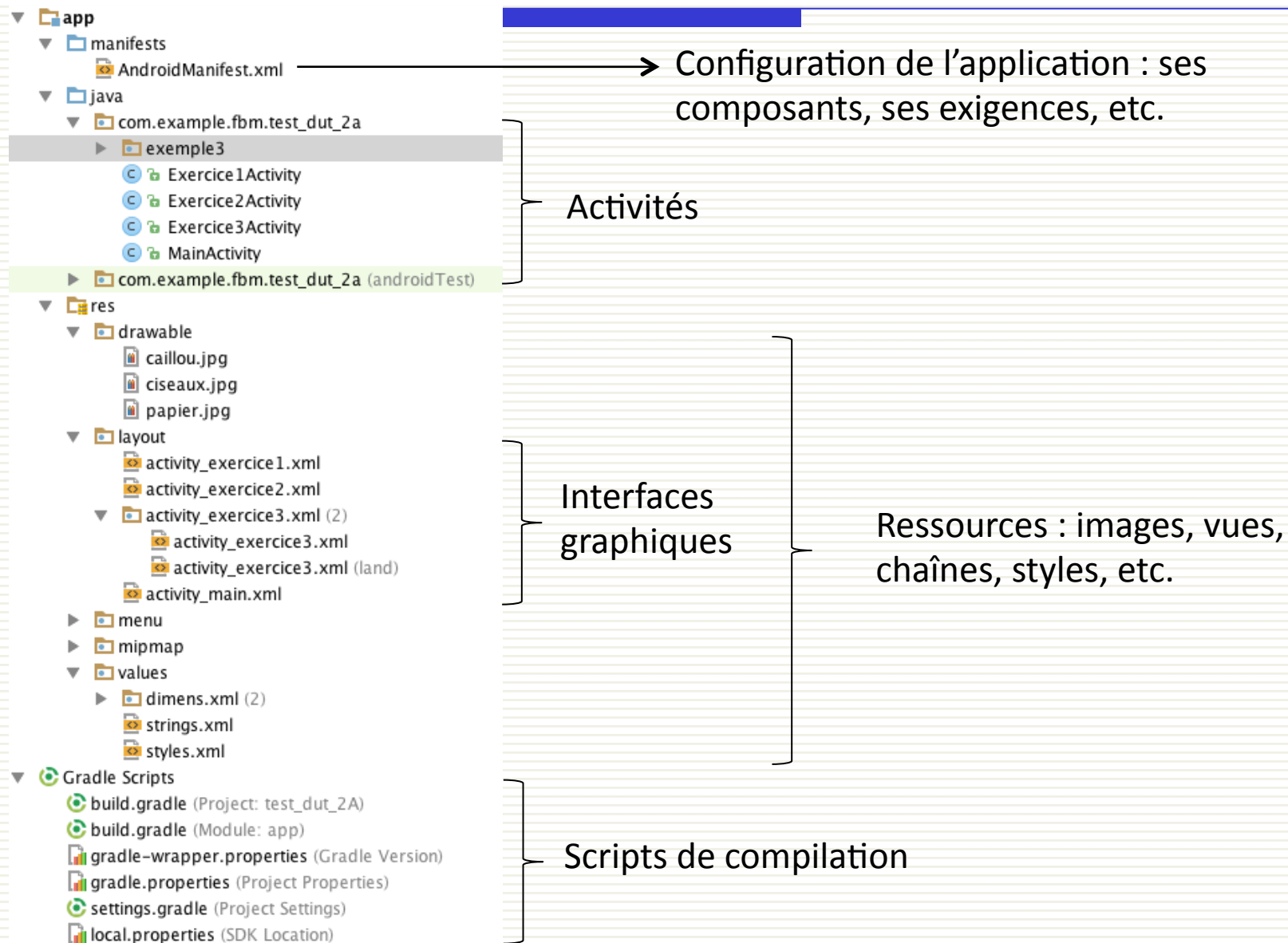
# APPLICATION ANDROÏD

# Application Androïd

---

- ✓ Application sous forme d'archive apk
  - Contient le code compilé ainsi que toutes les ressources nécessaires à l'application
  - Créée par l'outil aapt
  
- ✓ Chaque application est, par défaut, indépendante :
  - Tourne dans **son propre processus**
  - Chaque processus a **sa propre machine virtuelle**
  - Chaque processus est assigné à **un unique utilisateur linux**. Les permissions par défaut sur les fichiers de l'application font que l'application a seulement accès à ses fichiers

# Structure d'une application



# ACTIVITÉS

# Activités

---

- ✓ Une activité = une fenêtre d'une application
- ✓ Une activité contient du code proposant une certaine fonctionnalité et qui affiche une interface graphique



# Exemple d'activité

---

- ✓ Une activité est une classe qui étend la classe Activity ou une de ses filles (ActionBarActivity, FragmentActivity, etc.)
- ✓ La méthode onCreate initialise l'activité : création de l'interface, chargement du context, etc.

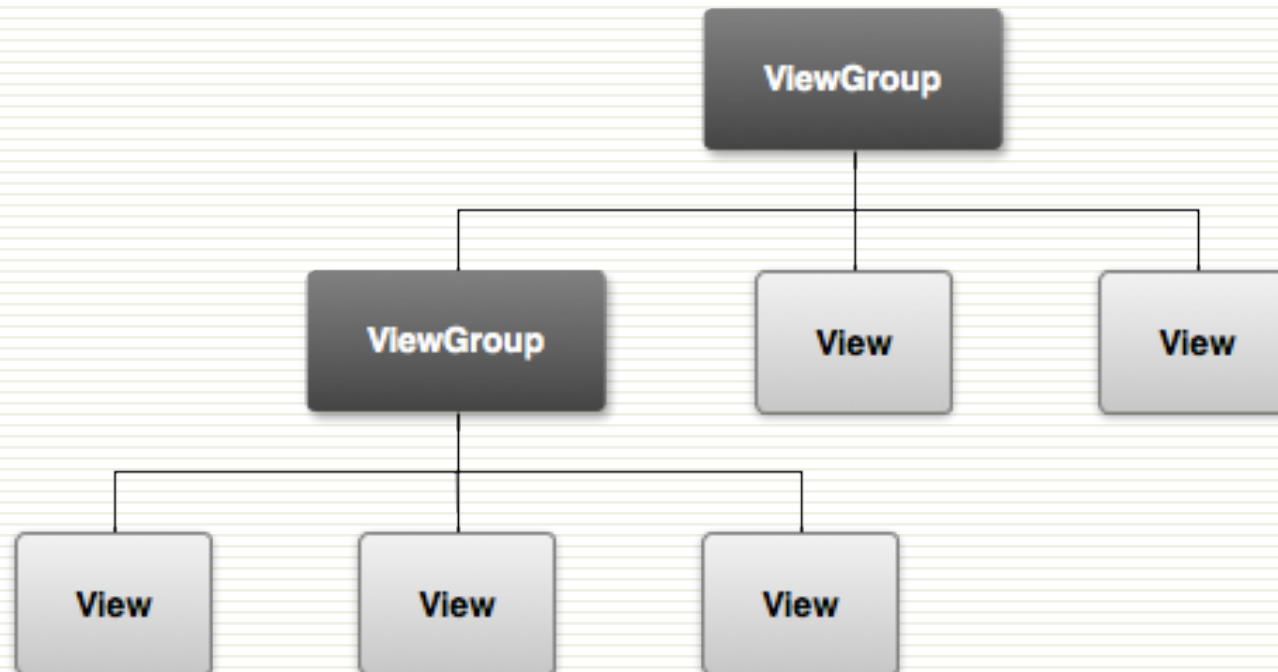
```
public class Exercice1Activity extends ActionBarActivity {  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
  
        // On charge le XML pour créer hiérarchie des composants graphique  
        setContentView(R.layout.activity_exercice1);  
    }  
}
```

# INTERFACES GRAPHIQUES

# Hiérarchie de vues

---

- ✓ Tous les composants graphiques (vues) d'une activité sont organisés en 1 seule hiérarchie



# Briques graphiques

---

## ✓ Classe **View**

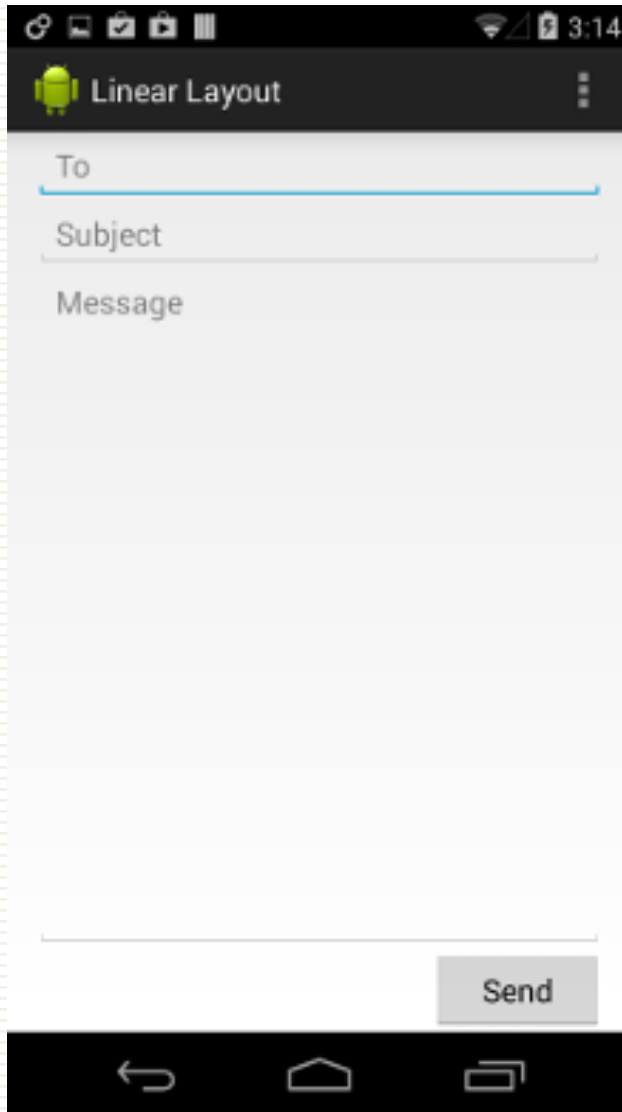
- Classe de base des interfaces graphiques
- Tous les composants graphiques (bouton, liste déroulante, zone de texte) héritent de cette classe

## ✓ Classe **Viewgroup**

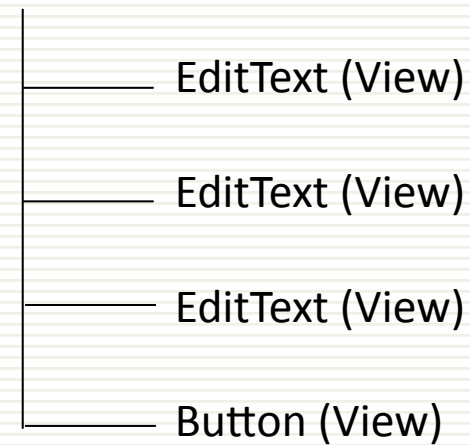
- Une « View » faite pour contenir des View et des ViewGroup
- Super classe des Layouts (gestionnaires de placement de fenêtres)

Dans la librairie android.view

# Exemple d'interface graphique



LinearLayout (ViewGroup)



# Hiérarchie de vues

---

- ✓ 2 manières de créer une hiérarchie de vues :
  - dans le code (classique – comme vu en SWING)
  - dans un fichier XML (fichier layout)
- ✓ Avantages du XML :
  - Séparation de la présentation et du contrôle (MVC)
  - Visualisation plus facile (XML adapté aux structures hiérarchiques)
- ◆ Un fichier XML doit être chargé dans la méthode onCreate de l'activité à l'aide de la méthode setContentView(...)  

```
// On charge le XML pour créer hiérarchie des composants graphique  
setContentView(R.layout.activity_exercice1);
```

R.layout.fichier\_de\_layout représente l'identifiant du fichier dans l'application (R est une classe référençant les ressources de l'application)

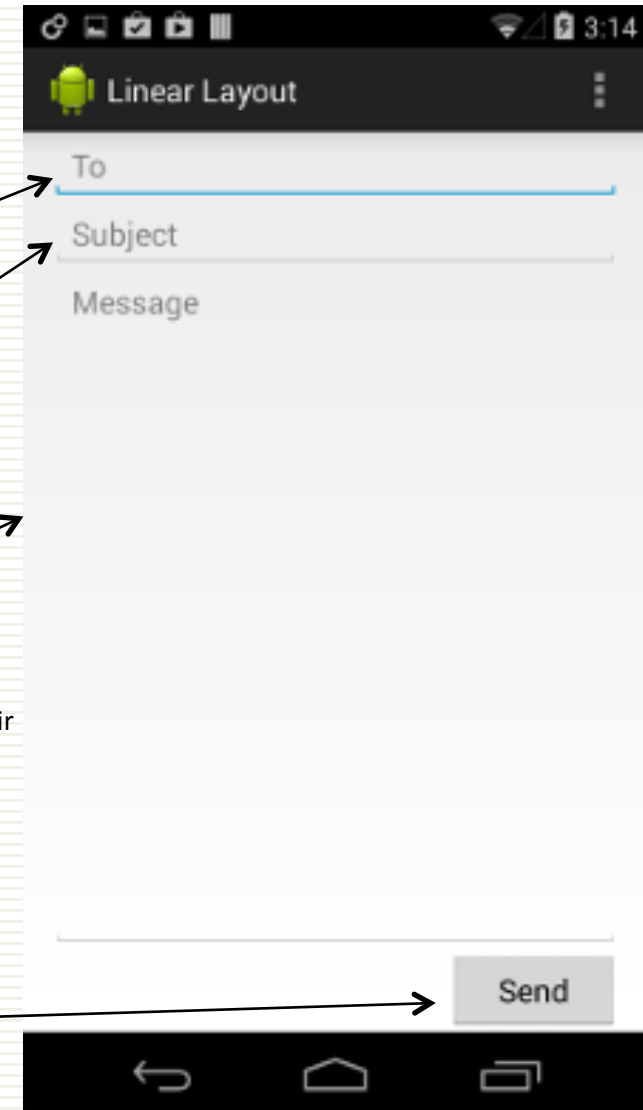
# Exemple sous forme XML

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/a
    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>
```

Prendre toute la  
place du parent

Un poids fort permet de remplir  
la place restante du parent.

Positionnement à  
droite du parent



# Identifié un composant graphique

- ✓ Une composant graphique dans un fichier XML peut être identifié par un entier
- ✓ Cet ID sert ensuite à référencer ce composant dans le code

■ Exemple :

```
<Button id="@+id/my_button"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text="@string/my_button_text"/>
```

Le @ signifie que c'est une ressource

Le + veut dire que c'est une nouvelle ressource et que ce nom doit être ajouté dans le fichier R.java

Dans le code Java

```
Button myButton = (Button) findViewById(R.id.my_button);
```



# Layouts : gestionnaires d'agencement

---

<http://developer.android.com/guide/topics/ui/declaring-layout.html>

## ✓ Exemples :

- LinearLayout : agencement sur une seule dimension (horizontale ou verticale)

<http://developer.android.com/guide/topics/ui/layout/linear.html>

- RelativeLayout : agencement relatif aux autres vues

<http://developer.android.com/guide/topics/ui/layout/relative.html>

- Voir aussi ListView, GridView, WebView, etc.



# Documentations

---

- ✓ Pour être guider dans le développement des interfaces graphiques :

<http://developer.android.com/guide/topics/ui/index.html>

- ✓ Pour connaître les propriétés des composants graphiques que vous utilisez :

<http://developer.android.com/reference/packages.html>

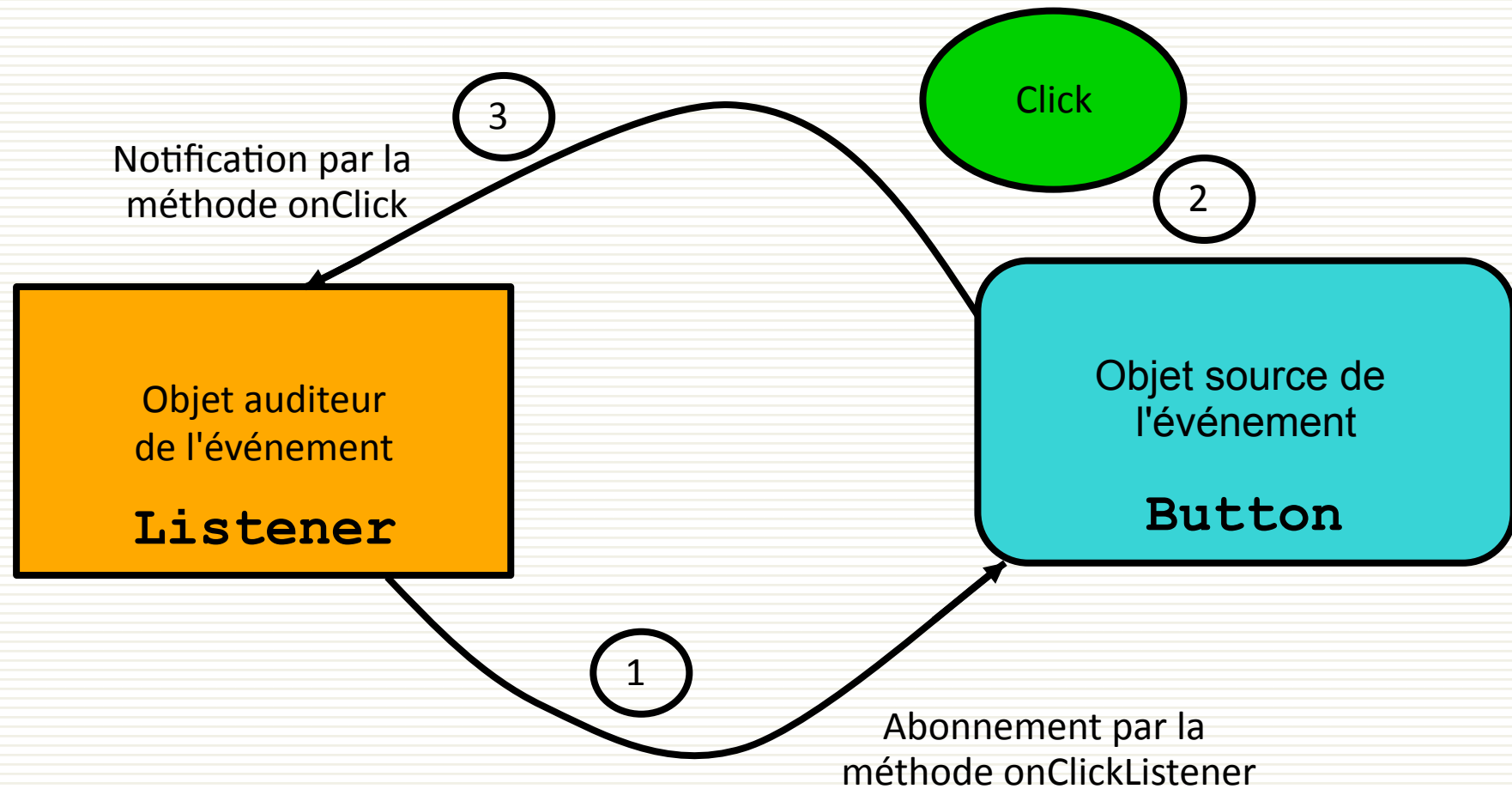
- Exemple la classe Button :

<http://developer.android.com/reference/android/widget/Button.html>

# ÉVÉNEMENTS

# Gestion des événements (1/2)

- ✓ Comme en SWING vu au semestre 2



# Gestion des événements (2/2)

---

- ✓ Les méthodes d'abonnement et de notification :
  - View.OnClickListener : onClick()
  - View.OnLongClickListener : onLongClick()
  - View.OnFocusChangeListener : onFocusChange()
  - View.OnKeyListener : onKey()
  - View.OnTouchListener : onTouch()
  - View.OnCreateContextMenuListener : onCreateContextMenu()

<http://developer.android.com/guide/topics/ui/ui-events.html>

# Exemple (1/2)

## ✓ Création d'un listener spécifique au bouton

```
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
  
    // On charge le XML pour créer hiérarchie des composants graphique  
    setContentView(R.layout.activity_exercice1);  
  
    // VERSION 1 : dans la méthode onCreate  
    // On récupère les objets de l'arbre graphique (à l'aide de leur id)  
    Button valider = (Button) findViewById(R.id.exercice1_valider);  
    final TextView hello = (TextView) findViewById(R.id.exercice1_hello);  
    final TextView prenom = (TextView) findViewById(R.id.exercice1_prenom);  
  
    //  
    valider.setOnClickListener(new View.OnClickListener() {  
        @Override  
        public void onClick(View v) {  
            if (!TextUtils.isEmpty(prenom.getText())) {  
                hello.setText("Hello " + prenom.getText() + " !");  
            }  
        }  
    });  
}
```

# Exemple (2/2)

✓ L'activité devient listener

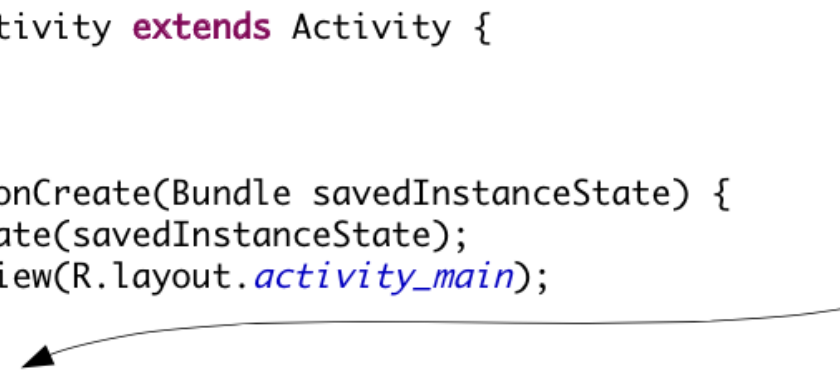
```
public class Exercice1Activity extends ActionBarActivity implements View.OnClickListener {  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
  
        // On charge le XML pour créer hiérarchie des composants graphique  
        setContentView(R.layout.activity_exercice1);  
  
        // On récupère les objets de l'arbre graphique (à l'aide de leur id)  
        Button valider = (Button) findViewById(R.id.exercice1_valider);  
  
        // VERSION 2 : la classe Exercice1Activity devient listener  
        valider.setOnClickListener(this);  
    }  
  
    @Override  
    public void onClick(View v) {  
        final TextView hello = (TextView) findViewById(R.id.exercice1_hello);  
        final TextView prenom = (TextView) findViewById(R.id.exercice1_prenom);  
  
        if (!TextUtils.isEmpty(prenom.getText())) {  
            hello.setText("Hello " + prenom.getText() + " !");  
        }  
    }  
}
```

# L'événement OnClick simplifié

- ✓ Ajouter l'événement OnClick directement sur le composant graphique dans le XML

```
<Button  
    android:id="@+id/button1"  
    android:text="Un joli Bouton"  
    android:onClick="onClickSurJoliBouton"/>
```

```
public class MainActivity extends Activity {  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
    }  
  
    public void onClickSurJoliBouton(View v) {  
        Context context = getApplicationContext();  
        CharSequence text = "On m'a cliqué !!!";  
        int duration = Toast.LENGTH_SHORT;  
        Toast toast = Toast.makeText(context, text, duration);  
        toast.show();  
    }  
}
```

A curved arrow originates from the `onClick="onClickSurJoliBouton"` attribute in the XML snippet and points to the `onClickSurJoliBouton` method in the Java code, illustrating the link between the UI component and its event handler.