IUT 2 Grenoble Université Pierre-Mendès-France Informatique

R4 Real 11
Programmation mobile sous Android

Introduction au développement Android

Introduction à la création interface utilisateur

Introduction à la gestion des événements

Android

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

Android 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)

- ✓ Depuis 2017, un second langage est supporté par Android : **Kotlin**

IDE : Android Studio

- ✓ Environnement de développement « officiel »
- ✓ Basé sur IntelliJ IDEA

- ✓ https://developer.android.com/studio/

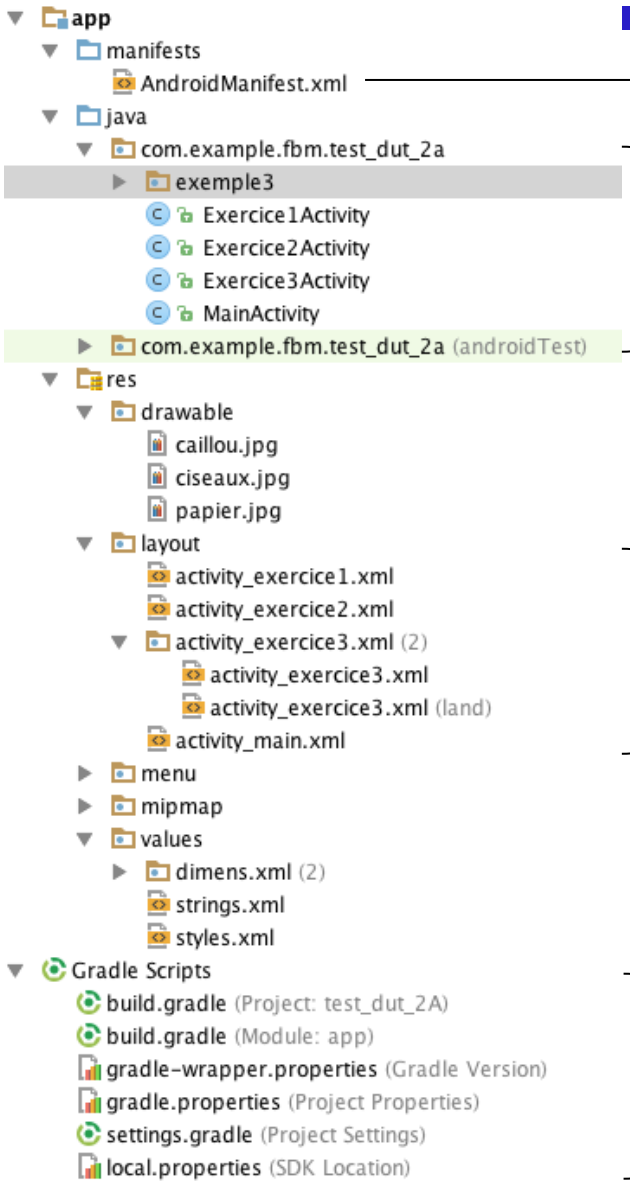
APPLICATION ANDROID

Application Android

- ✓ 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



Configuration de l'application : ses composants, ses exigences, etc.

Activités (~ contrôleurs)

Interfaces utilisateur

Ressources : images, vues, chaînes, styles, etc.

Peut y avoir beaucoup de vues pour chaque fenêtre (ex : portait, paysage)

Scripts de compilation

ACTIVITÉS

Activités

- ✓ Une application est constituée de plusieurs activités
- ✓ Une activité = une fenêtre d'une application (*pour le moment*)
- ✓ Une activité contient du code proposant une certaine fonctionnalité et qui affiche une interface graphique

Exemple d'activité (1/2)

- ✓ Une activité est une classe qui **étend la classe Activity** ou une de ses filles (`AppCompatActivity`, `FragmentActivity`, etc.)

  Gérer plusieurs fenêtres avec une seule activité.
  Gérer des systèmes d'onglets, si rapport entre les fenêtres.

```
public class MainActivity extends AppCompatActivity {

    …
```

La classe `AppCompatActivity` permet d'avoir une compatibilité avec les anciennes versions : actionBar API 11, material design API 21

Exemple d'activité (2/2)

- ✓ La méthode **onCreate** initialise l'activité : création de l'interface graphique, récupérer les vues, associer les événements, etc.

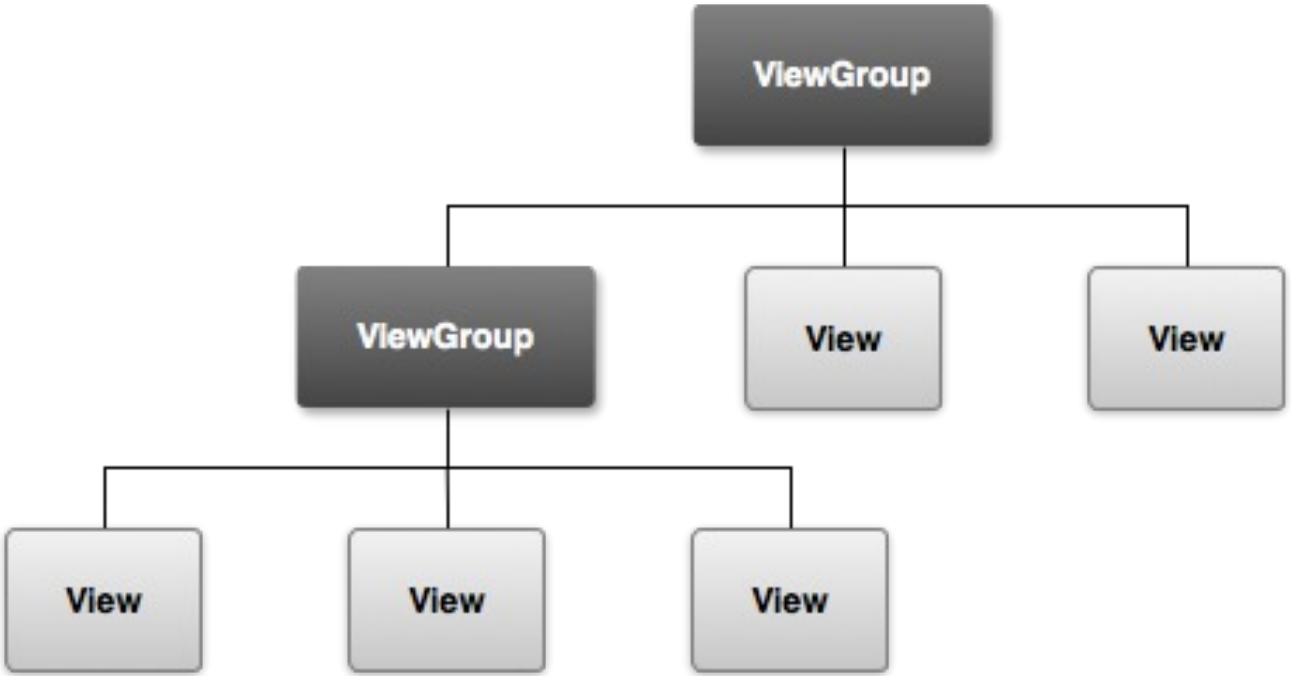
```
public class MainActivity extends AppCompatActivity {

    @Override
     protected void onCreate(Bundle savedInstanceState) {
         super.onCreate(savedInstanceState);

         // On charge le XML pour créer la hiérarchie
         // des composants graphiques
         setContentView(R.layout.activity_main);

         // Récupérer les vues
         …
         // Associer les événements
         …
```

INTERFACES UTILISATEURS
USER INTERFACE UI

Hiérarchie de vues

/!\ pas au sens dans MVC

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



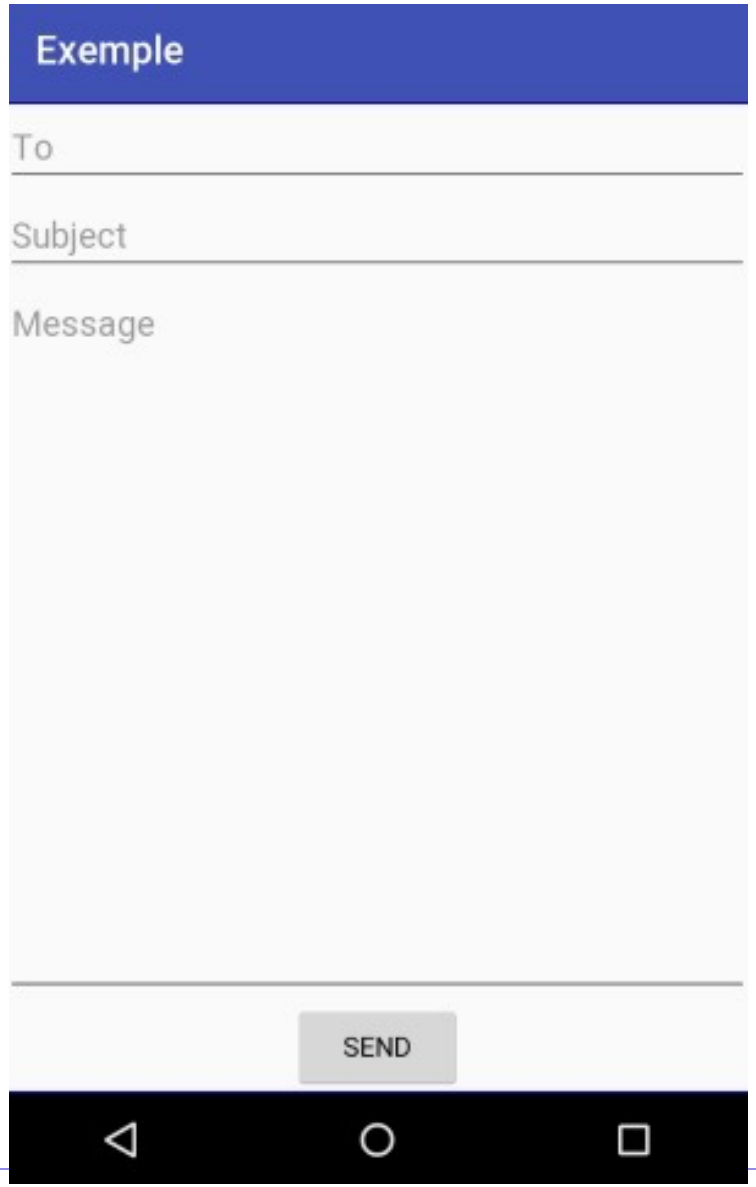
Briques graphiques (1/2)

- ✓ Classe **`View`**
  - Classe de base des UI (interfaces utilisateurs)
  - Tous les composants graphiques (bouton, liste déroulante, zone de texte) héritent de cette classe

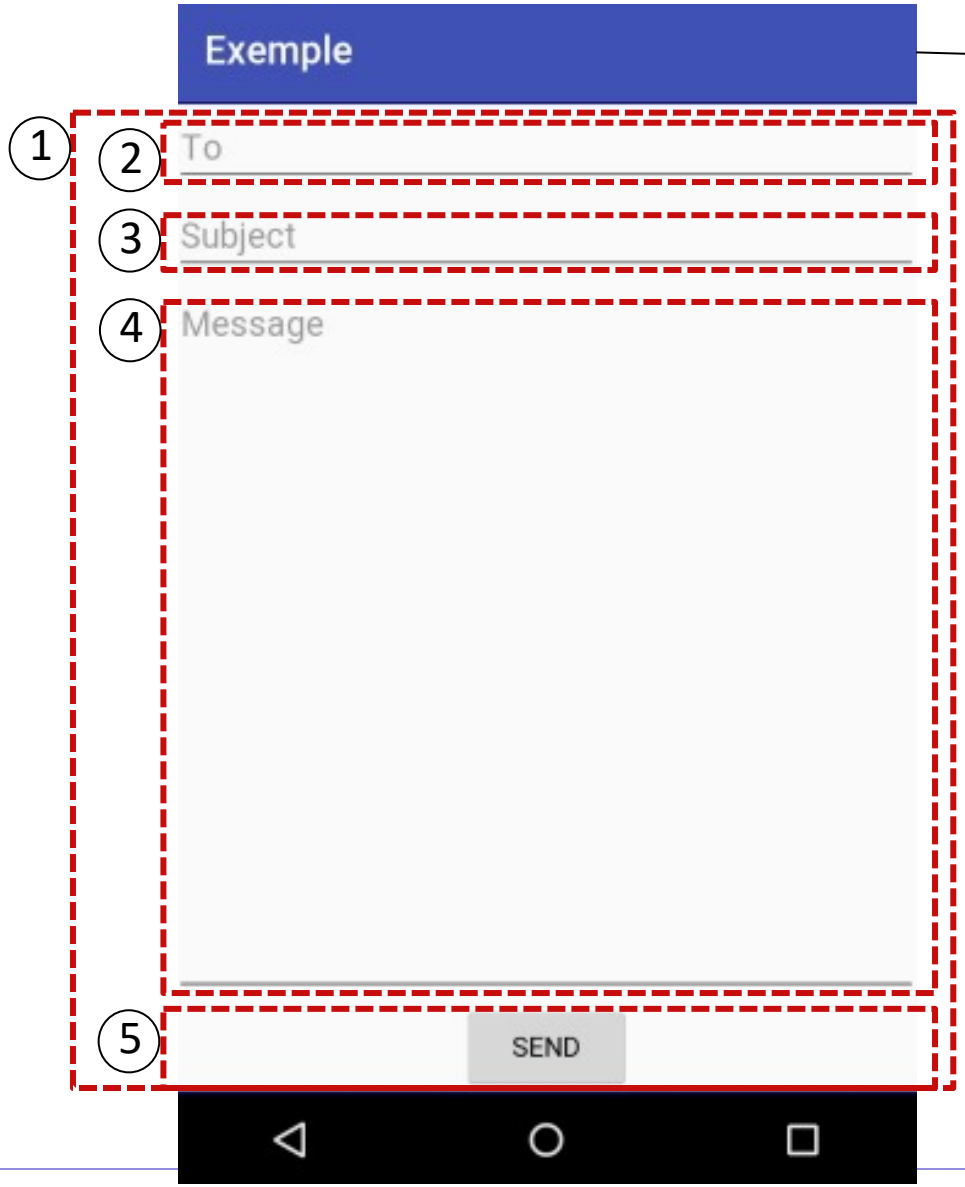
- ✓ Classe **`ViewGroup`**
  - Une « `View` » faite pour contenir des `View` et des `ViewGroup`
  - Super classe des Layouts (gestionnaires de placement de View)

Dans la librairie android.view

Exemple d'interface graphique (1/2)



Exemple d'interface graphique (2/2)



ActionBar fournit par le thème par défaut

-> pour modifier ça, faut modifier thèmes dans fichiers de config

1 **LinearLayout (ViewGroup)**

- 2 **EditText – « To » (View)**
- 3 **EditText – « Subject » (View)**
- 4 **EditText – « Message » (View)**
- 5 **Button – « Send » (View)**

Hiérarchie de vues

✓ 2 manières de créer une hiérarchie de vues :

- **dans le code de l'activité**
  - Comme en SWING (librairie Java) new... add...
- **dans un fichier XML lié à l'activité** --> à préférer
  - Comme vu en module IHM JavaFX
  - Dans le dossier `layout` en ressources

Exemple dans le code de l'activité

Démo Exemple_1_1_1Activity

```
// Gestionnaire d'agencement
LinearLayout linearLayout = new LinearLayout( context: this);
linearLayout.setOrientation(LinearLayout.VERTICAL);

// To
EditText to = new EditText( context: this);
to.setText(R.string.to);
linearLayout.addView(to);

// Subject
EditText subject = new EditText( context: this);
subject.setText(R.string.subject);
linearLayout.addView(subject);

// Message
EditText message = new EditText( context: this);
message.setText(R.string.message);
LinearLayout.LayoutParams layoutParamsMessage =
        new LinearLayout.LayoutParams(LinearLayout.LayoutParams.MATCH_PARENT,
                LinearLayout.LayoutParams.MATCH_PARENT, weight: 1);
linearLayout.addView(message, layoutParamsMessage);

// Send
Button send = new Button( context: this);
ViewGroup.LayoutParams layoutParamsSend =
        new ViewGroup.LayoutParams(ViewGroup.LayoutParams.WRAP_CONTENT,
                ViewGroup.LayoutParams.WRAP_CONTENT);
send.setLayoutParams(layoutParamsSend);
send.setText(R.string.send);
linearLayout.setHorizontalGravity(Gravity.CENTER);
linearLayout.addView(send);

setContentView(linearLayout);
```

① ② ③ ④ ⑤

Facile ?

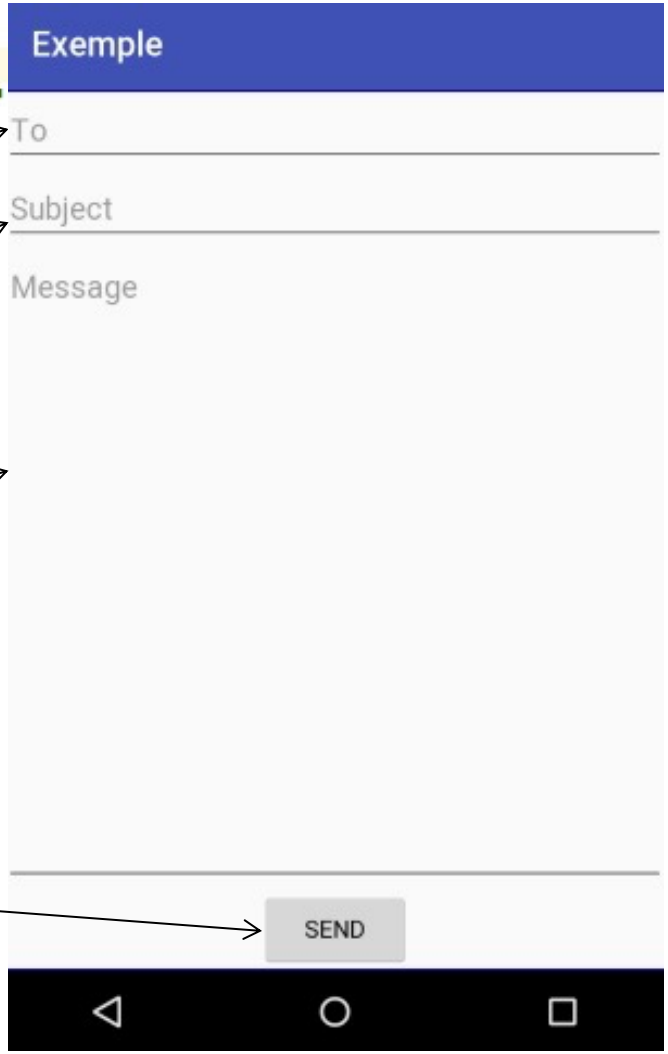
MAIS surcharge de l'activité -> on met de la vue alors qu'il s'agit du traitement de l'activité

Exemple **sous forme XML**

Démo Exemple_1_1_2Activity

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout 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"
    tools:context="m4104C.exemples.exemple_1_1.Exemple_1_1_1Activity"
    android:orientation="vertical">

    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="to"/>

    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="subject"/>

    <EditText
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:gravity="top"
        android:hint="message"/>

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:text="send"/>

</LinearLayout>
```

-> permet de gérer hauteur (prend toute la place ?)



Hiérarchie de vues

- ✓ **Avantages du XML** :
  - Séparation de la présentation et du contrôle (MVC)
  - Visualisation plus facile (XML adapté aux structures hiérarchiques)

    - Chargement à l'aide de la méthode **`setContentView(…)`** dans la méthode `onCreate` de l'activité

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    // On charge le XML pour créer l'UI
    setContentView(R.layout.activity_exemple_1_1_2);
}
```

R.layout.NOM_DU_FICHIER représente l'identifiant du fichier XML dans l'application
R est une classe référençant les ressources de l'application

Identifier un objet View

- ✓ Un objet View dans un fichier XML peut être identifié par un identifiant ID
- ✓ Cet ID sert ensuite à récupérer cet objet dans le code de l'activité

- Exemple :

Dans le fichier XML

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

Le @ signifie que c'est une ressource

Le + veut dire que c'est une nouvelle ressource et que cet ID doit être ajouté à la classe R

Dans le code Java

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

Comment trouver les vues ? (1/3)

- ✓ En utilisant la **javadoc**

https://developer.android.com/reference/android/view/View

https://developer.android.com/reference/android/view/ViewGroup

View

added in API level 1 ☆☆☆☆☆

```
public class View
extends Object implements Drawable.Callback, KeyEvent.Callback, AccessibilityEventSource
```

java.lang.Object
↳ android.view.View

Known direct subclasses
AnalogClock, ImageView, KeyboardView, MediaRouteButton, ProgressBar, Space, SurfaceView, TextView, TextureView, ViewGroup, ViewStub

Known indirect subclasses
AbsListView, AbsSeekBar, AbsSpinner, AbsoluteLayout, ActionMenuView, AdapterView<T extends Adapter>, AdapterViewAnimator, AdapterViewFlipper, AppWidgetHostView, AutoCompleteTextView, Button, CalendarView, and 52 others.

Comment trouver les vues ? (2/3)

Démo Exemple_1_1_3Activity

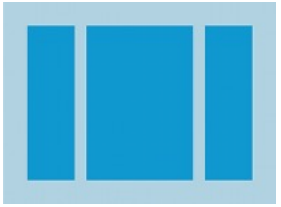
✓ En utilisant les **guides** (documentation android)

https://developer.android.com/guide/

https://developer.android.com/guide/topics/ui/

✓ Par exemple, pour les layouts (viewgroup)
http://developer.android.com/guide/topics/ui/declaring-layout.html

- **LinearLayout** : agencement sur une seule dimension
- 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

Comment trouver les vues ? (3/3)

✓ En utilisant l'**éditeur** de l'IDE

https://developer.android.com/studio/write/layout-editor

- **ConstraintLayout**
- https://developer.android.com/training/constraint-layout
- https://www.youtube.com/watch?v=XamMbnzI5vE

✓ **ATTENTION** il faut comprendre ce que l'on produit !!!

Vous pouvez toujours voir le fichier XML généré

Multiple devices

- ✓ Comment faire si plusieurs devices envisagés ?
  - Plusieurs vues (layout)
    - Une vue pour le mode portrait
    - Une vue pour le mode paysage
    - Voir plus !
      - Une vue pour le mode portrait HD
      - Une vue pour le mode portrait SD
      - Une vue pour le mode paysage HD
      - Une vue pour le mode paysage SD
    - Voir encore plus !
      - Un vue pour le mode portrait HD 5''
      - 7'', 10'', etc.

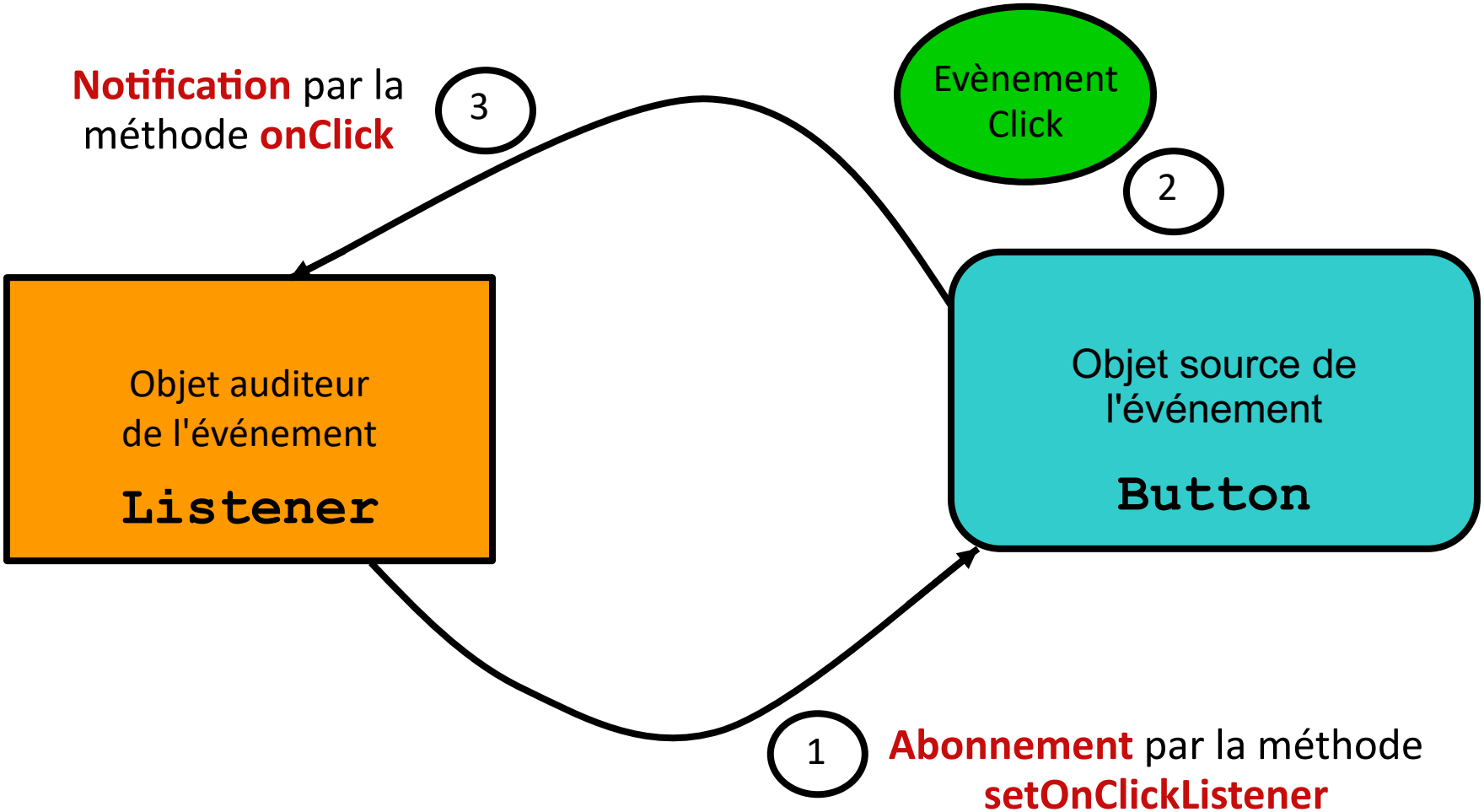
Android natif pas forcément bonne solution pour petites entreprises : trop de choses à défninir.
Peut être plus intéressant d'utiliser du cross-platform.

On simplifiera de notre côté

ÉVÉNEMENTS

Gestion des événements (1/2)

- ✓ Comme en ~~SWING~~ JavaFX vu au semestre 2



Gestion des événements (2/2)

- ✓ Un couple composé
  - D'une méthode d'**abonnement**
  - ET d'une **interface** de **notification**

  - Exemples de couples :

setOnClickListener(View.OnClickListener) / View.OnClickListener contenant onClick()

setOnLongClickListener(View.OnLongClickListener) / View.OnLongClickListener contenant onLongClick()

setOnFocusChangeListener(View.OnFocusChangeListener) / View.OnFocusChangeListener contenant onFocusChange()

setOnKeyListener(View.OnKeyListener) / View.OnKeyListener contenant onKey()

setOnTouchListener(View.OnTouchListener) / View.OnTouchListener contenant onTouch()

Etc…

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

Listener spécifique (classe anonyme) (1/2)

Démo Exemple_1_1_4Activity

-> "spécifique" car il n'est associé qu'à cet objet

-> "anonyme" --> on peut donc créer objets directement dans view

✓ Création d'un listener spécifique au bouton

```
// Récupération de l'objet graphique
Button monPremierBouton = (Button)findViewById(R.id.monPremierBouton);

// Associer un événement à cet objet
monPremierBouton.setOnClickListener(new View.OnClickListener() {

    @Override
    public void onClick(View view) {
        Toast toast = Toast.makeText( context: Exemple_1_1_4Activity.this,
                text: "Clique effectué sur le premier bouton", Toast.LENGTH_LONG);
        toast.show();
    }
});
```

Abonnement

Notification

Le paramètre de type **`View`** de la méthode `onClick(…)` est l'objet **source** de l'événement. Par exemple, un bouton, une image, etc.

Listener spécifique (classe anonyme) (2/2)

Démo Exemple_1_1_4Activity

- ✓ Création d'un listener spécifique au bouton

```
// Récupération de l'objet graphique
Button monPremierBouton = (Button)findViewById(R.id.monPremierBouton);

// Associer un événement à cet objet
monPremierBouton.setOnClickListener(new View.OnClickListener() {

    @Override
    public void onClick(View view) {
        Toast toast = Toast.makeText( context: Exemple_1_1_4Activity.this,
                text: "Clique effectué sur le premier bouton", Toast.LENGTH_LONG);
        toast.show();
    }
});
```

Pour avoir accès à l'objet `this` de la classe contenant le listener, on doit faire `NomDeLActivity.this` dans le listener.

-> pas dans le même domaine, donc il faudra rappeler la classe

Activité comme listener

Démo Exemple_1_1_5Activity

```
public class Exemple_1_1_5Activity extends AppCompatActivity implements View.OnClickListener

    // VIEW
    Button monPremierBouton;
    Button monDeuxiemeBouton;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_exemple_1_1_5);

        // Récupération des objets graphiques
        monPremierBouton = (Button) findViewById(R.id.monPremierBouton);
        monDeuxiemeBouton = (Button) findViewById(R.id.monDeuxiemeBouton);

        // Associé l'activité comme listener
        monPremierBouton.setOnClickListener(this);
        monDeuxiemeBouton.setOnClickListener(this);

    }

    @Override
    public void onClick(View view) {

        //
        String message = null;
        if (view == monPremierBouton) {
            message = "Clique effectué sur le premier bouton";
        } else {
            message = "Clique effectué sur le deuxième bouton";
        }
```

-> activité principale peut écouter objets source

Abonnement

Notification

On pourrait également créer une classe spécifique

Simplification d'écriture pour onClick

Démo Exemple_1_1_6Activity

- ✓ Revient au même que l'activité comme listener

Il vaut mieux mettre les événements dans les activités que dans les présentations, mais c'est tout de mêm possible :

```
<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 clicqué !!!";
        int duration = Toast.LENGTH_SHORT;
        Toast toast = Toast.makeText(context, text, duration);
        toast.show();
    }
}
```

Seulement pour les onClick !

pas pour les autres évènements

Deprecated c'est-à-dire que cette méthode doit être évitée dans un nouveau code (et enlevée dans les anciens)

Comment développer proprement ?

Démo Exemple_1_1_7Activity

✓ Dans l'activité

Données n'est pas une brique graphique !

- **Déclarer** des données
- **Déclarer** les briques graphiques utilisées

- Dans la méthode onCreation()
  - **initialiser** des données
  - **Récupérer** les briques graphiques de la vue
  - **Associer** les événements
  - Mettre à jour graphiquement
    - Créer pour ce faire **une méthode ou plusieurs de mise à jour**

Exemple de code propre 1/2

```
public class Exemple_1_1_7Activity extends AppCompatActivity {

    // DATA
    private int compteur = 0;

    // VIEW
    private TextView compteurView;
    private Button boutonAjoutView;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_exemple_1_1_7);

        // Récupération des briques graphiques
        compteurView = findViewById(R.id.compteurView);
        boutonAjoutView = findViewById(R.id.boutonAjoutView);
```

Exemple donné dans le Projet "R4Real11Exemples" sur Chamilo avec les autres exemples présentés dans les cours

Exemple de code propre (2/2)

```
    // Associer un événement au bouton
    boutonAjoutView.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            compteur++;                          // Mise à jour de la donnée
            miseAJourGraphique();
        }
    });

    // Mise à jour graphique
    miseAJourGraphique();
}


// Mise à jour graphique de l'activité
private void miseAJourGraphique() {

    compteurView.setText(String.valueOf(compteur));
}
}
```