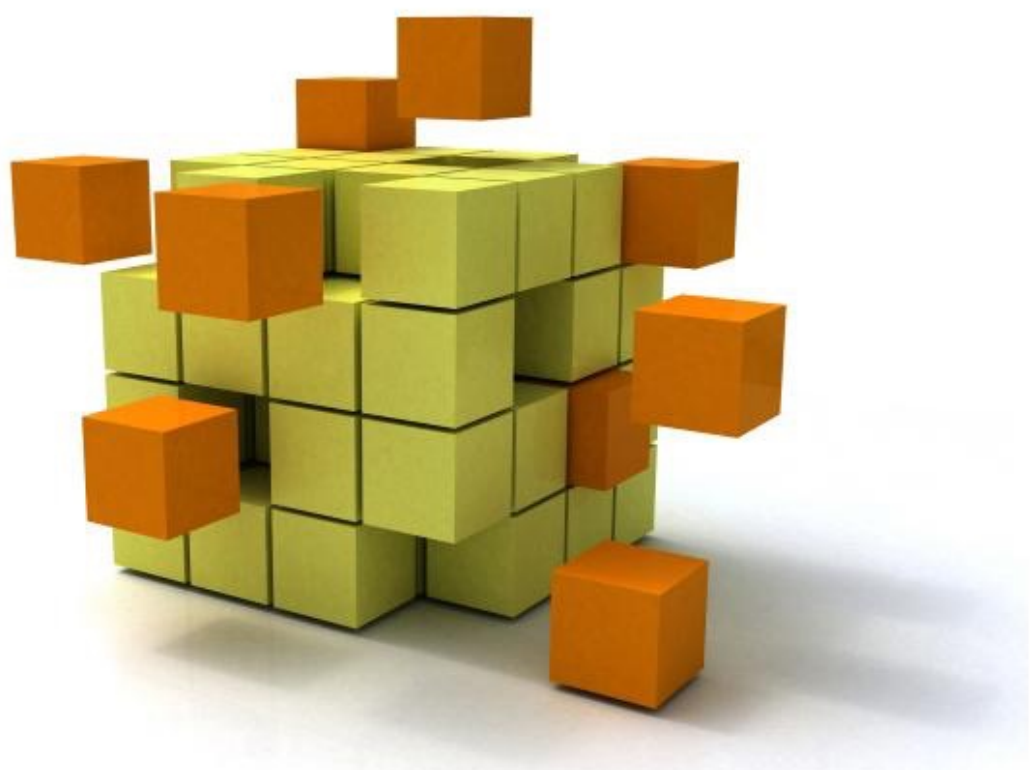


Applications concurrentes, mobiles et réparties en java

S7 Appro – Inf A3 EG

Stéphane Derrode, Alexandre Saidi, Bât E6, 2ième étage
stephane.derrode@ec-lyon.fr

Organisation de l'AF

- **Grands chapitres**
  1. **Apprentissage de Java** : 4h de cours, 4h de TP, 2h d'autonomie – Stéphane Derrode
  2. **IHM en Java** : 4h de cours, 8h de TP, 4h autonomie (BE noté #1) – Stéphane Derrode
  3. **Prog. concurrente et distribuée** : 4h de cours, 4h de TP, 4h d'autonomie (BE noté #2) – Alexandre Saïdi
  4. **Prog. mobile** : 4h de cours, 4h de TP, 2h d'autonomie (BE noté #3) – Stéphane Derrode

- **Évaluation**
  - Examen papier : 50 %
  - Moyenne de 3 BEs : 50%

Nous utiliserons Android Studio (à installer sur votre machine selon le tuto présent sur le site du cours)

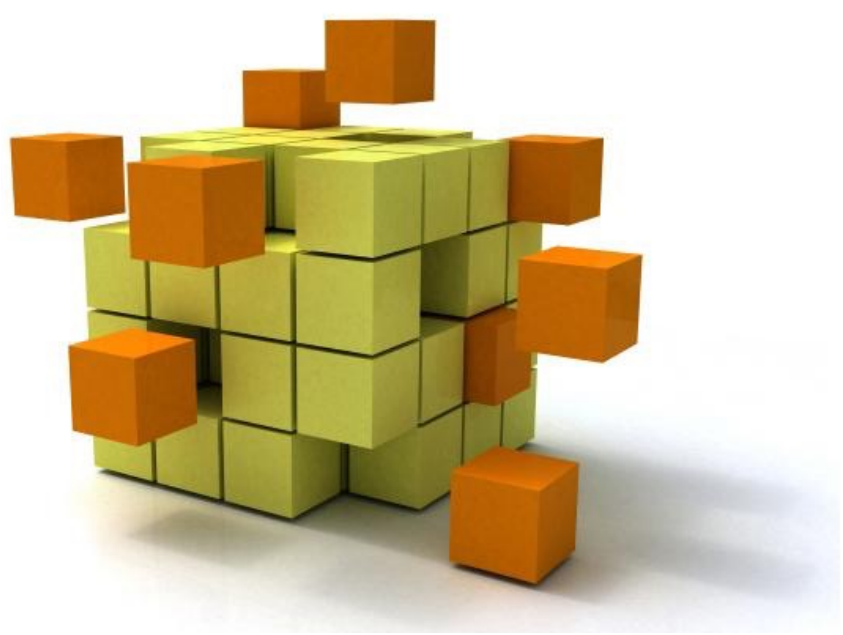
Distribution des créneaux

ActivitÈ	Intervenant	Date	Heure			
Cours	S. Derrode	02/09/2022	13:30-15:30			
BE	S. Derrode	05/09/2022	08:00-10:00			
Cours	S. Derrode	08/09/2022	08:00-10:00			
BE	S. Derrode	08/09/2022	10:15-12:15			
BE	S. Derrode	12/09/2022	08:00-10:00	Module 1 (D	Apprentissage Java	
Autonomie	S. Derrode	15/09/2022	08:00-10:00	Module 2 (D	IHM en Java	
Cours	S. Derrode	15/09/2022	10:15-12:15	Module 3 (Sa	Prog. Concurrente et distribuée	
BE	S. Derrode	19/09/2022	08:00-10:00	Module 4 (D	Prog. Mobile	
BE	S. Derrode	22/09/2022	08:00-10:00			
Cours	S. Derrode	22/09/2022	10:15-12:15			
Autonomie	S. Derrode	26/09/2022	08:00-10:00			
BE	S. Derrode	29/09/2022	08:00-10:00			
BE	S. Derrode	29/09/2022	10:15-12:15			
Cours	A. Saidi	03/10/2022	08:00-10:00			
BE	A. Saidi	06/10/2022	08:00-10:00			
BE	A. Saidi	06/10/2022	10:15-12:15			
Cours	A. Saidi	10/10/2022	08:00-10:00			
BE	A. Saidi	13/10/2022	08:00-10:00			
Autonomie	A. Saidi	13/10/2022	10:15-12:15			
Cours	S. Derrode	17/10/2022	08:00-10:00			
BE	S. Derrode	20/10/2022	08:00-10:00			
Cours	S. Derrode	20/10/2022	10:15-12:15			
BE	S. Derrode	27/10/2022	08:00-10:00			
Autonomie	S. Derrode	27/10/2022	10:15-12:15			

Sommaire

1. Introduction
2. Architecture d'une application Android
3. Les activités
4. Définir une interface graphique
5. Les intentions (une introduction)

Exercices / sujet du BE



1. Introduction

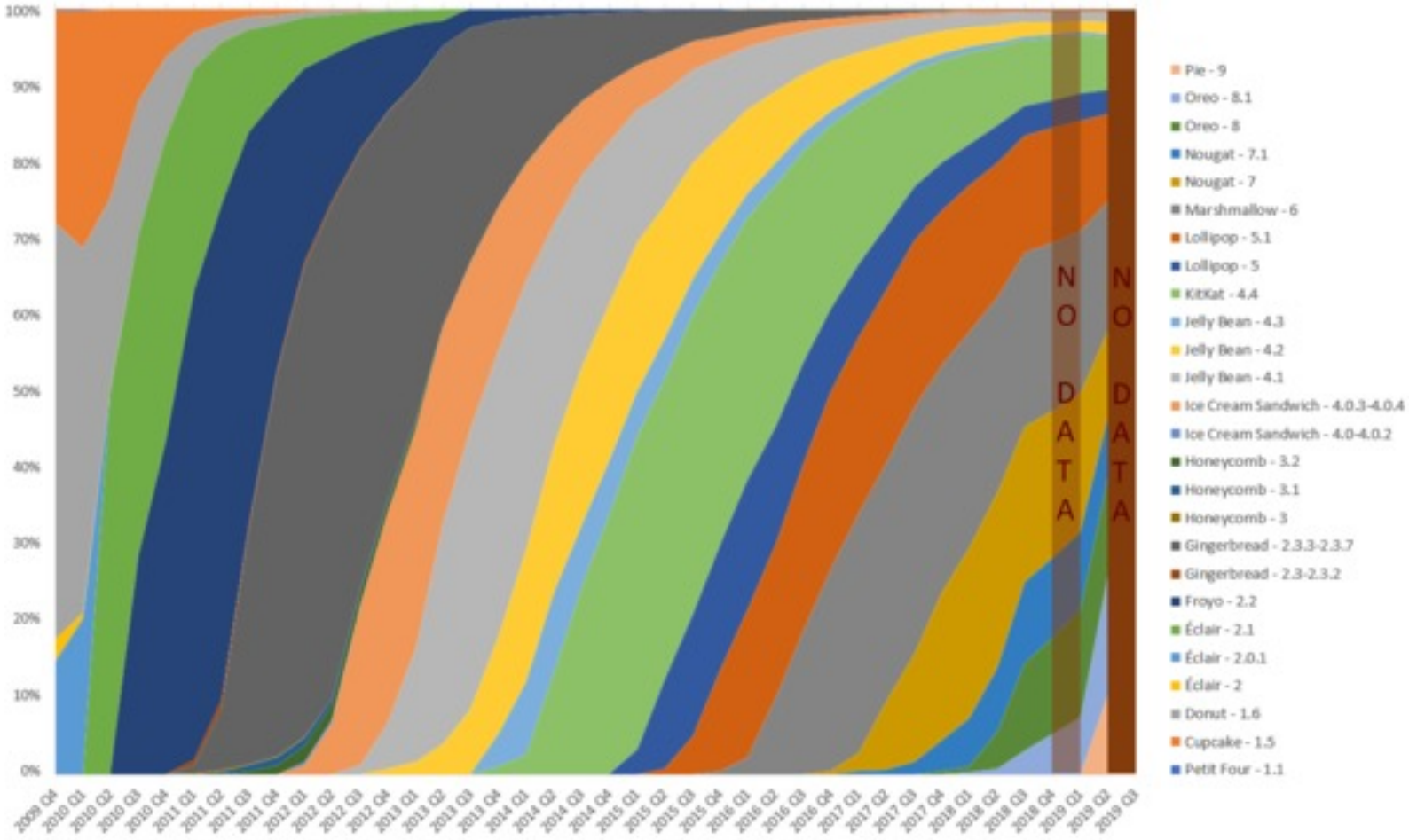
1- Introduction

- Système d'exploitation pour dispositifs mobiles
  - Téléphones, tablettes, téléviseurs, montres, lunettes, voitures…
- Caractéristiques :
  - Open Source (licence Apache), gratuit, flexible
  - Basé sur un noyau linux
  - Inclut les applications de base (téléphone, sms, carnet d'adresse, navigateur, ...)
  - Un ensemble important d'API (OpenGL, media, …)
  - Un SDK basé sur un sous-ensemble de JAVA (autres langages disponibles : C, C++, …)
  - Une machine virtuelle qui exécute la majorité des applications

1- Introduction

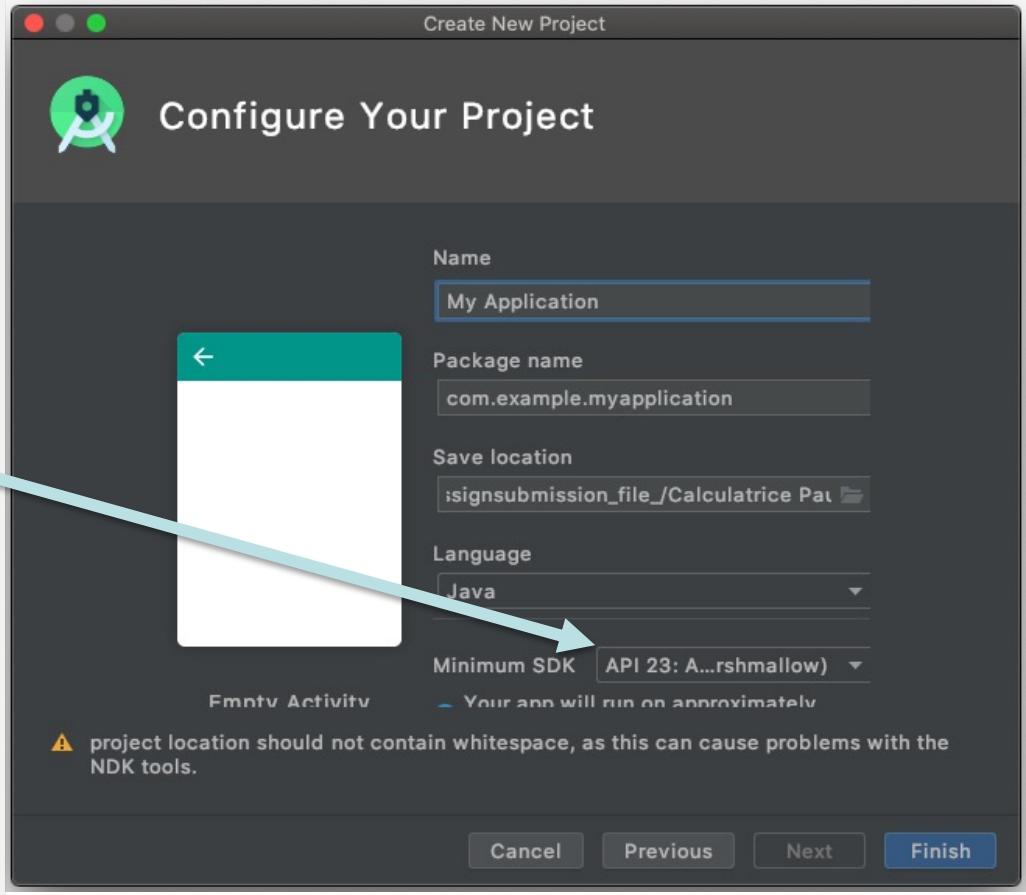
Historique :

- Créé en 2005 par la société Android
- Rachat en 2007 par Google
- Plus de 20 versions depuis la 1.0 (Apple Pie) en 2008.

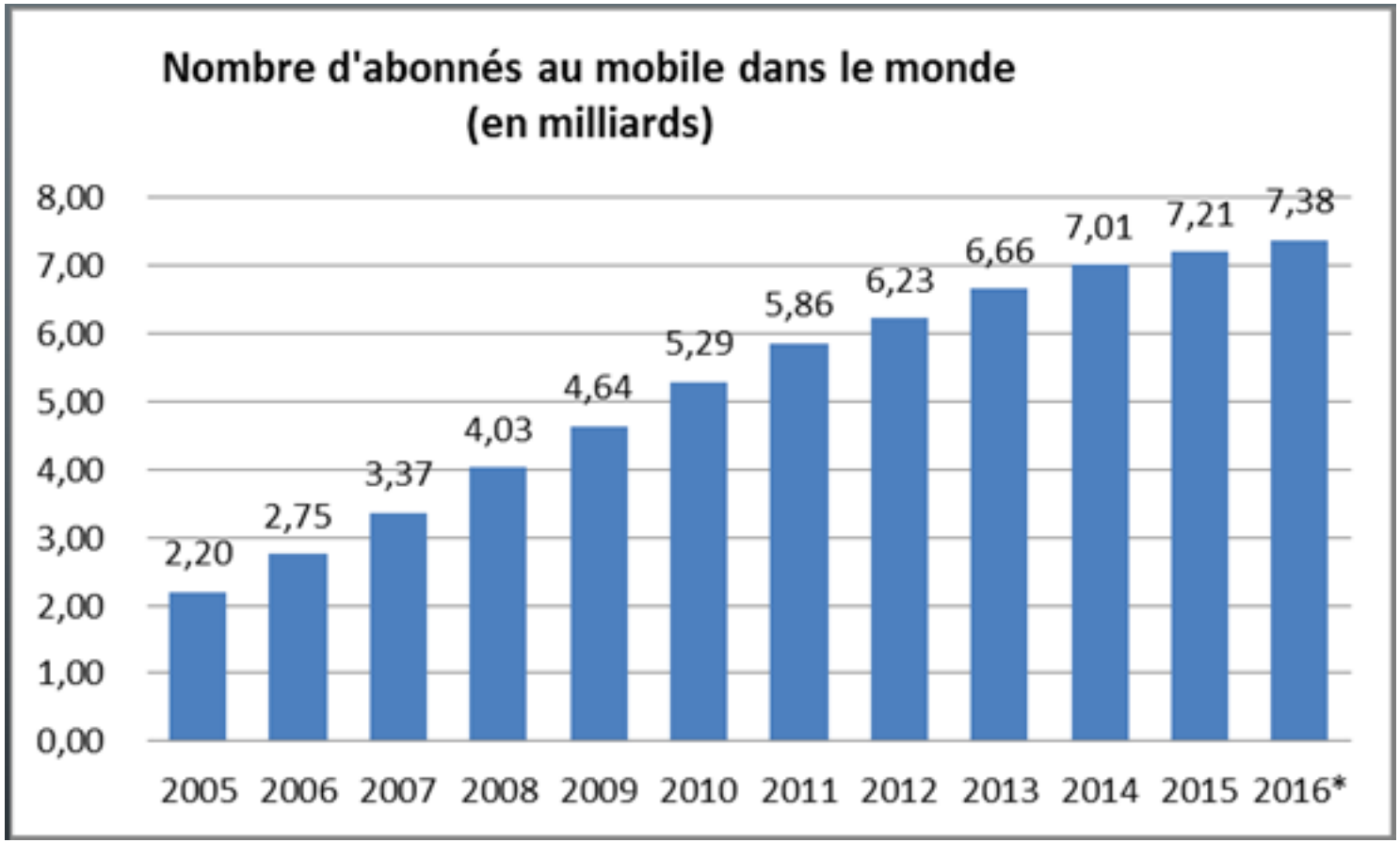


1- Introduction

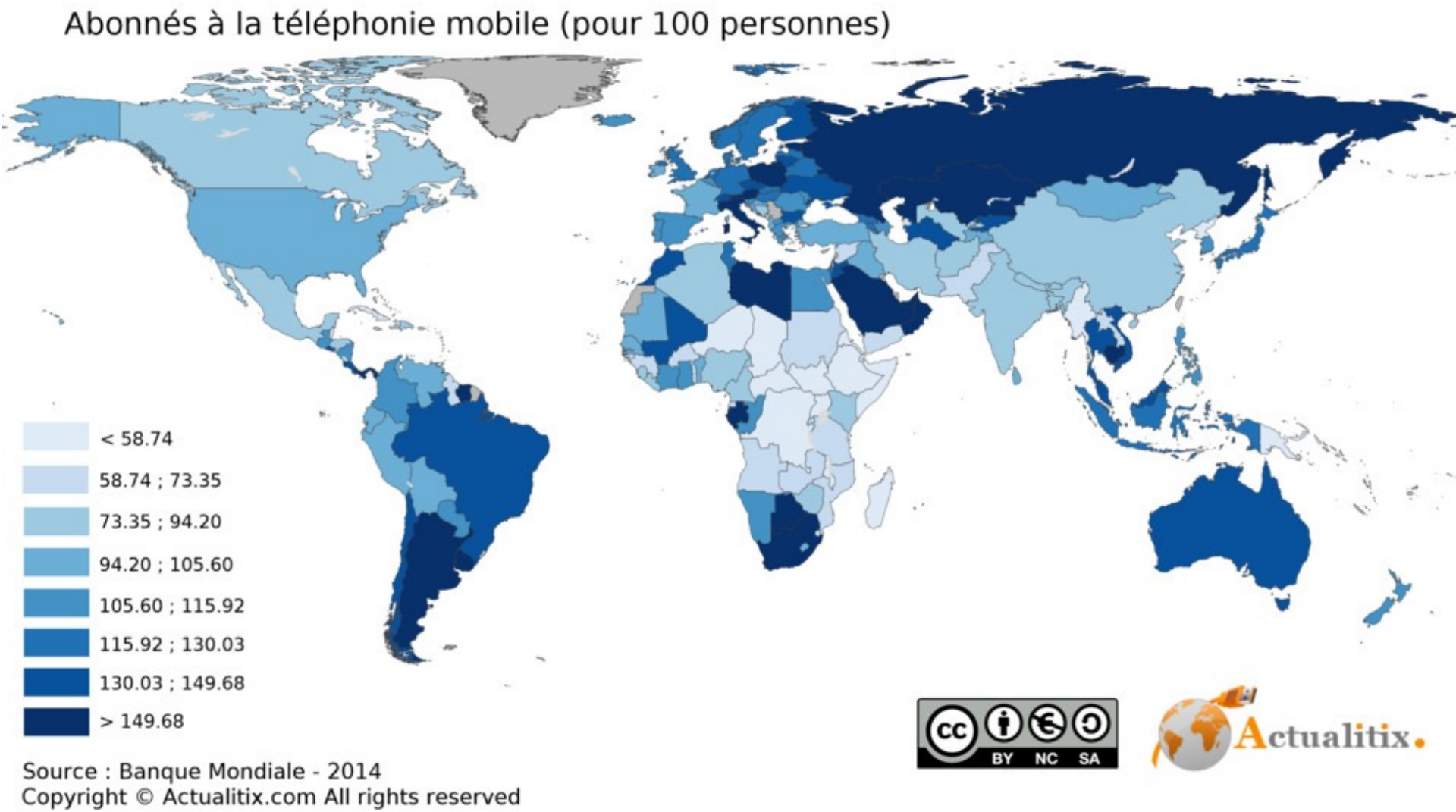
ANDROID PLATFORM VERSION	API LEVEL	CUMULATIVE DISTRIBUTION
4.0 Ice Cream Sandwich	15	
4.1 Jelly Bean	16	99.8%
4.2 Jelly Bean	17	99.2%
4.3 Jelly Bean	18	98.4%
4.4 KitKat	19	98.1%
5.0 Lollipop	21	94.1%
5.1 Lollipop	22	92.3%
6.0 Marshmallow	23	84.9%
7.0 Nougat	24	73.7%
7.1 Nougat	25	66.2%
8.0 Oreo	26	60.8%
8.1 Oreo	27	53.5%
9.0 Pie	28	39.5%
10. Android 10	29	8.2%



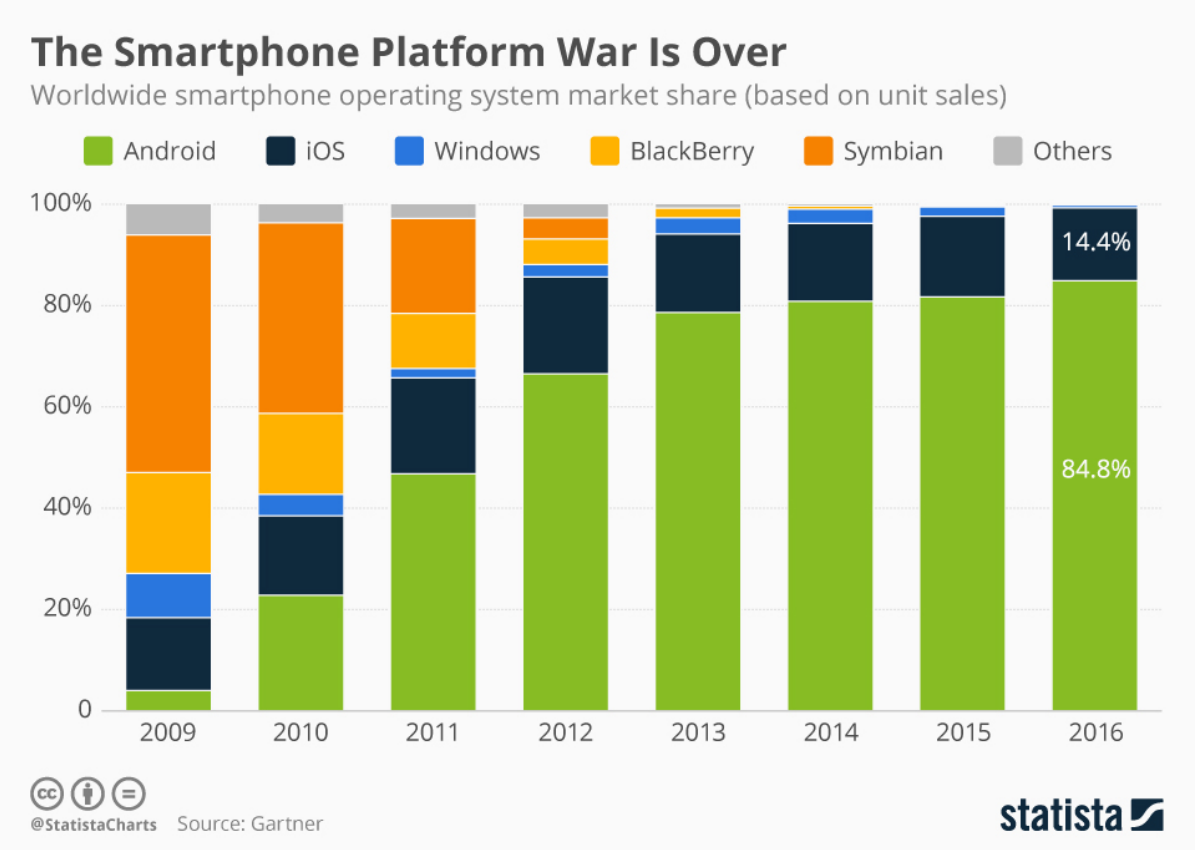
1- Introduction



1- Introduction



1- Introduction

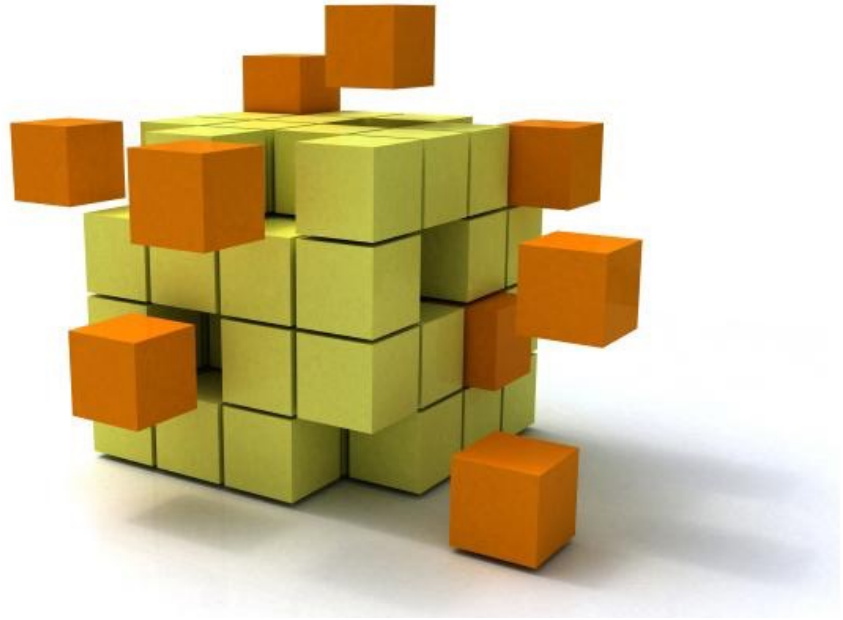


Year	2018	2019	2020	2021	2022	2023	2024
Android	85.1%	86.1%	85.0%	85.6%	86.0%	86.1%	86.2%
iOS	14.9%	13.9%	15.5%	14.4%	14.0%	13.9%	13.8%
Others	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
TOTAL	**100.0%**	**100.0%**	**100.0%**	**100.0%**	**100.0%**	**100.0%**	**100.0%**

1- Introduction

Contraintes

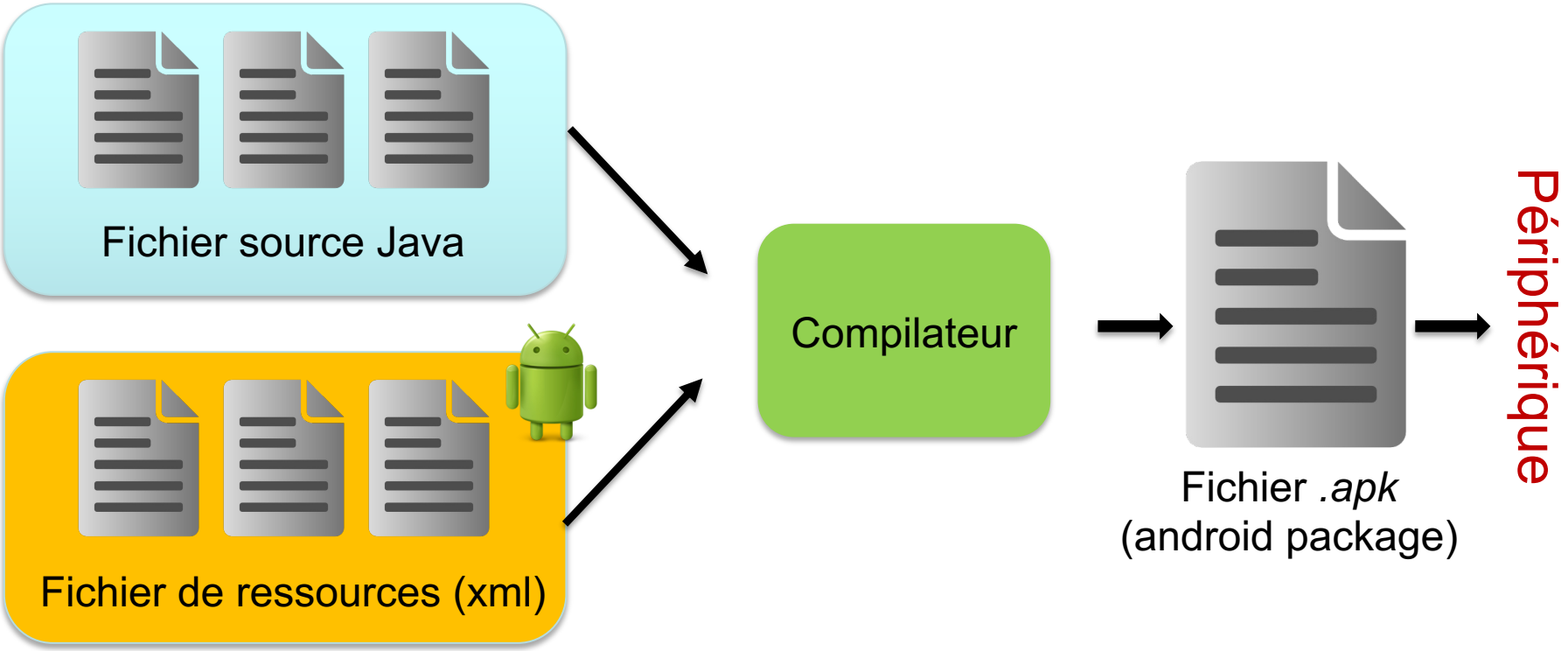
- Hétérogénéité du matériel
  - Processeurs, mémoire
  - Ecrans
  - Dispositifs spécialisés
- Puissance et mémoire limitées
- Interface tactile
- Connectivité à internet (disponibilité, rapidité, ...)
- Développement extérieur au périphérique



2. Architecture d’une application Android

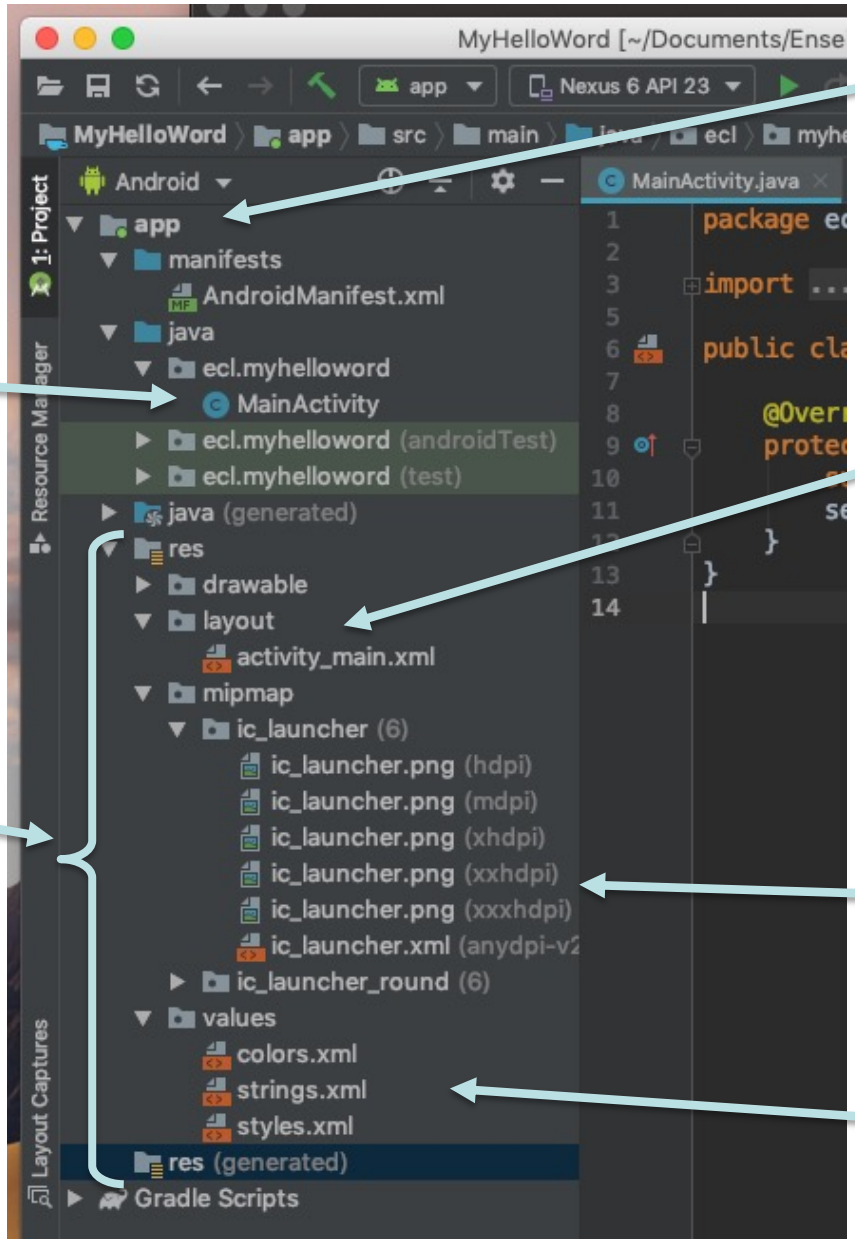
2- Architecture

Schéma de développement



2- Architecture

Android Studio



Dossier du projet

Sources Java

Mise en page

Ressources

Icônes

Constantes

2- Architecture

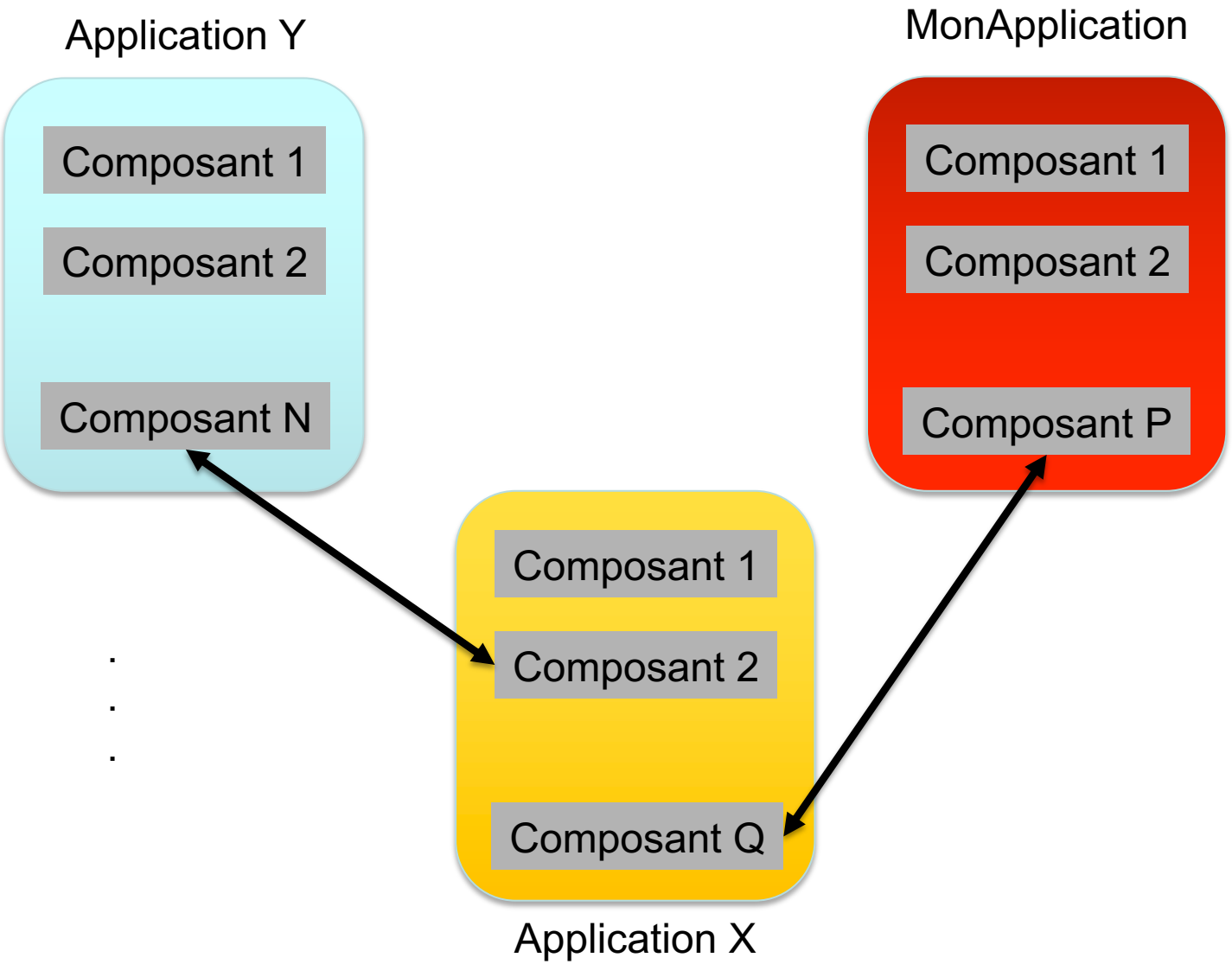
Les élts d'une application

- Une application = {composants}

- Les composants :
  - Existent de manière indépendante
  - Vus comme autant de points d'entrée par le système
    Pas de « main » dans une application

- Liés au design d'Android :
  - Toute application doit pouvoir démarrer un composant d'une autre application (sous réserve de droits) et récupérer ses « résultats »

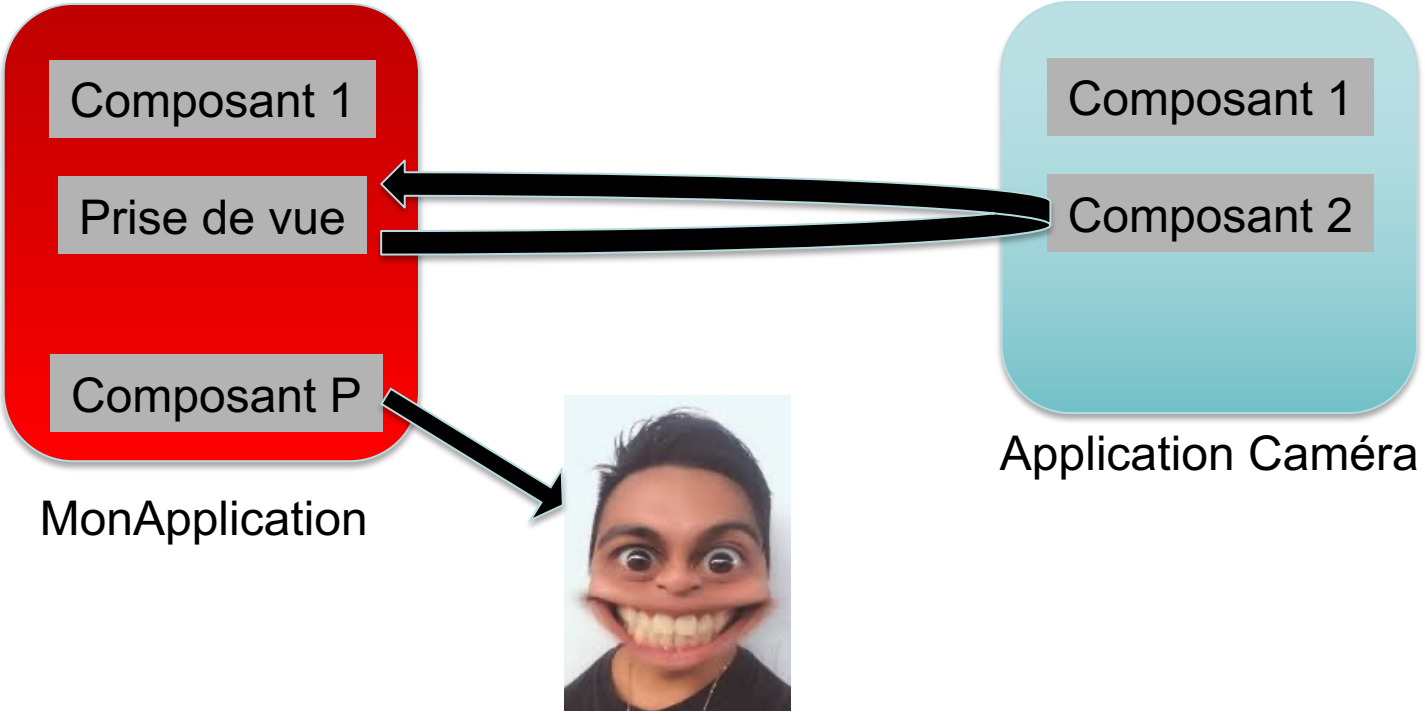
2- Architecture



2- Architecture

Exemple MonAppli : Effets visuels sur photo utilisateur

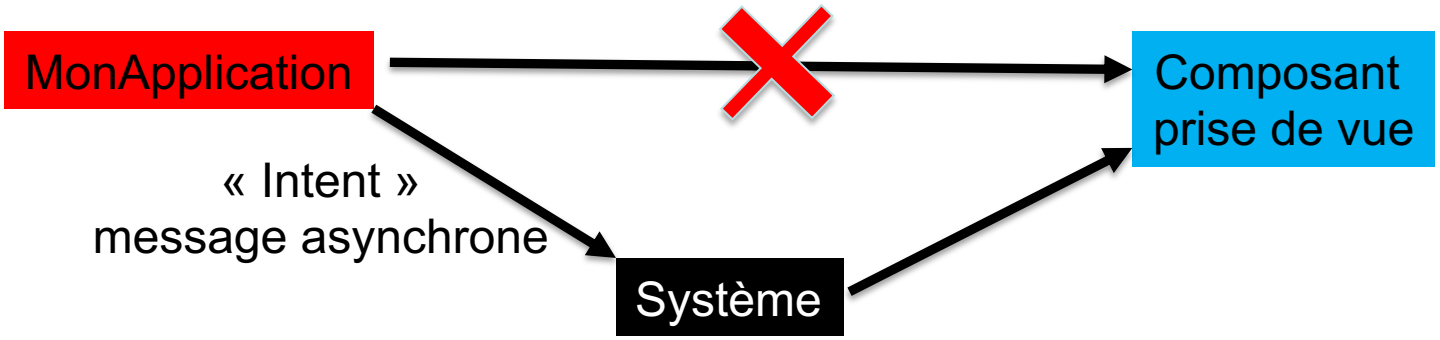
- Difficulté : écrire le code de gestion de l'appareil photo embarqué
- Android :
  - démarrage d'un composant existant permettant la prise de vue
  - récupération de l'image



2- Architecture

Remarques :

- Problèmes de droits :



- Problèmes d'information :

  Le système doit connaître le rôle particulier de certains composants

  Ce sont les applications qui enregistrent ces informations auprès du système

2- Architecture

Les composants :

- Les activités (*Activity*)
  – Un écran avec une interface utilisateur et un contexte

- Les services (*Service*)
  – Composant sans écran, qui tourne en fond de tâche (lecteur de musique, téléchargement, ...)

- Les fournisseurs de contenu (*ContentProvider*)
  – I/O sur des données gérées par le système ou par une autre application

- Des récepteurs d'intentions (*BroadcastReceiver*)
  – Récupération d'informations générales
    – arrivée d'un sms, batterie faible, ...

2- Architecture

Les interactions:

- Les intentions (*Intent*)

  Permet d'échanger des informations entre composants
  - Démarrage d'un composant en lui envoyant des données
  - Récupération de résultats depuis un composant
  - Recherche d'un composant en fonction d'un type d'action à réaliser

- Les filtres d'intentions *(<intent-filter>)*
  - Permet à un composant d'indiquer ce qu'il sait faire
  - Permet au système de sélectionner les composants susceptibles de répondre à une demande de savoir-faire d'une application

2- Architecture

Le Manifeste (*AndroidManifest.xml*) :

Description de l'application:

- Liste des composants
- Niveau minimum de l'API requise
- Liste des caractéristiques physiques nécessaires. Evite d'installer l'application sur du matériel non compatible (gestion de la visibilité sur Google Play)
- Liste des permissions dont l'application a besoin
- Liste des autres API nécessaires
  ex. Google Map
- …

Généré automatiquement par Android Studio

2- Architecture

Exemple (*AndroidManifest.xml*):

AndroidManifest.xml

AndroidManifest.xml

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3       package="ecl.myhelloword" >
4
5       <application
6           android:allowBackup="true"
7           android:icon="@mipmap/ic_launcher"
8           android:label="MyHelloWord"
9           android:roundIcon="@mipmap/ic_launcher_round"
10          android:supportsRtl="true"
11          android:theme="@style/AppTheme" >
12          <activity android:name=".MainActivity" >
13              <intent-filter>
14                  <action android:name="android.intent.action.MAIN" />
15
16                  <category android:name="android.intent.category.LAUNCHER" />
17              </intent-filter>
18          </activity>
19      </application>
20
21  </manifest>
```

2- Architecture

Les ressources

- **Ressources** = toutes les données (autres que le code) utilisées par l'application

- Rangées dans le dossier **res**, puis incluses dans **l'apk**
  - *res/drawable* et *res/mipmap* : images en différentes résolutions
  - *Layout* : description en XML des interfaces
  - *Menus* : description en XML des menus.
  - *Values* : définitions en XML des constantes utilisées par l'application : chaînes, tableaux, valeurs numériques, ...

2- Architecture

String.xml

- Fichier ressources, contenant toutes les chaînes constantes

Principalement utilisées pour l'interface

```
<resources>
    <string name="app_name">Fortune Ball</string>
    <string name="action_settings">Settings</string>
    <string name="fortune_description">Suggest the question, which you
        can answer “yes” or “no”, then click on the magic ball.</string>
</resources>
```

2- Architecture

Internationalisation

- **Objectif :**
  – Disposer de plusieurs versions des textes, libellés, … utilisés par l'application
  – Choix automatique des textes en fonction de la configuration du périphérique

- **Principe**
  – Dupliquer le fichier *strings.xml* : 1 version par langue supportée
  – Stocker chaque version dans un dossier spécifique
    - values-xx (ex. values-en, values-fr, …)
  – Géré via Android Studio

```
app/
  res/
    values/
      strings.xml
    values-en/
      strings.xml
    values-fr/
     strings.xml
```

2- Architecture

La classe R

- Classe générée par l'IDE
  - Permet l'accès aux ressources
  - Créée à partir de l'arborescence présente dans le dossier **res**
  - Elle contient des classes internes dont les noms correspondent aux différents types de ressources (*drawable*, *layout*, …)
  - Elle contient des propriétés permettant de représenter l'ensemble des ressources de l'application

- Utilisation en Java :
  - **R.type.identificateur**

R.string.app_name

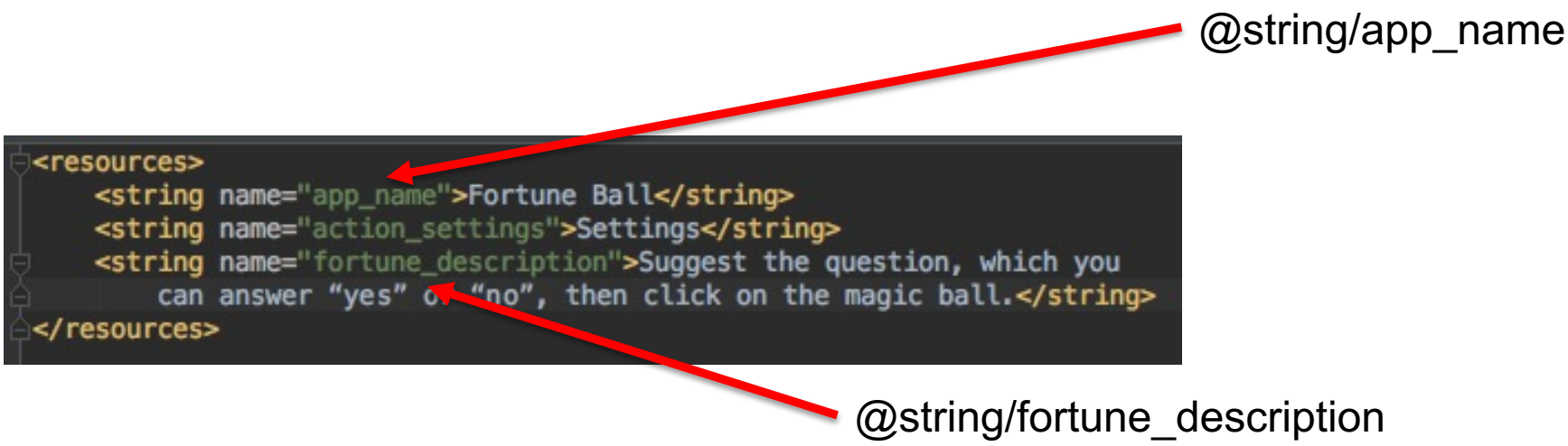
```
<resources>
    <string name="app_name">Fortune Ball</string>
    <string name="action_settings">Settings</string>
    <string name="fortune_description">Suggest the question, which you
        can answer “yes” o “no”, then click on the magic ball.</string>
</resources>
```

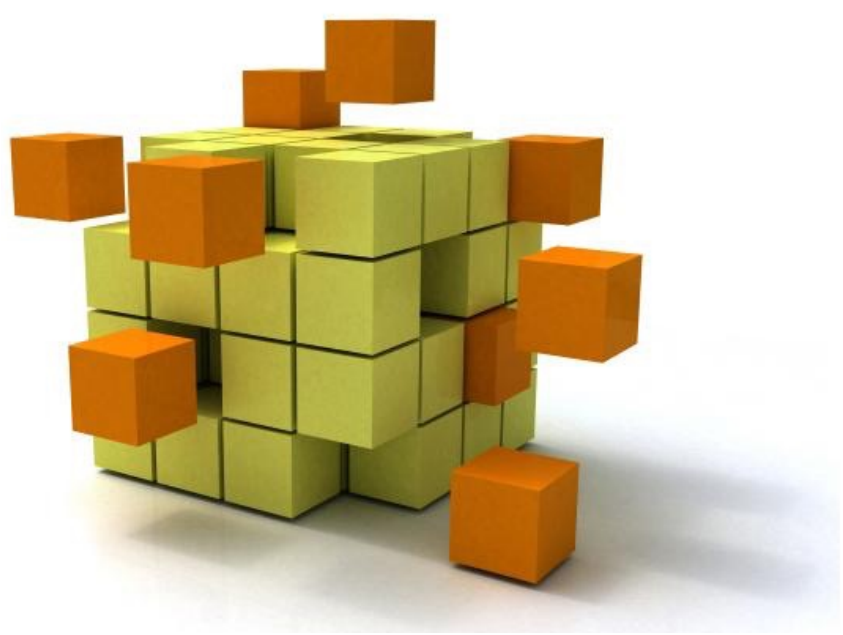
R.string.fortune_description

2- Architecture

Référencement des ressources en xml

- Forme générale : **@type/identificateur**





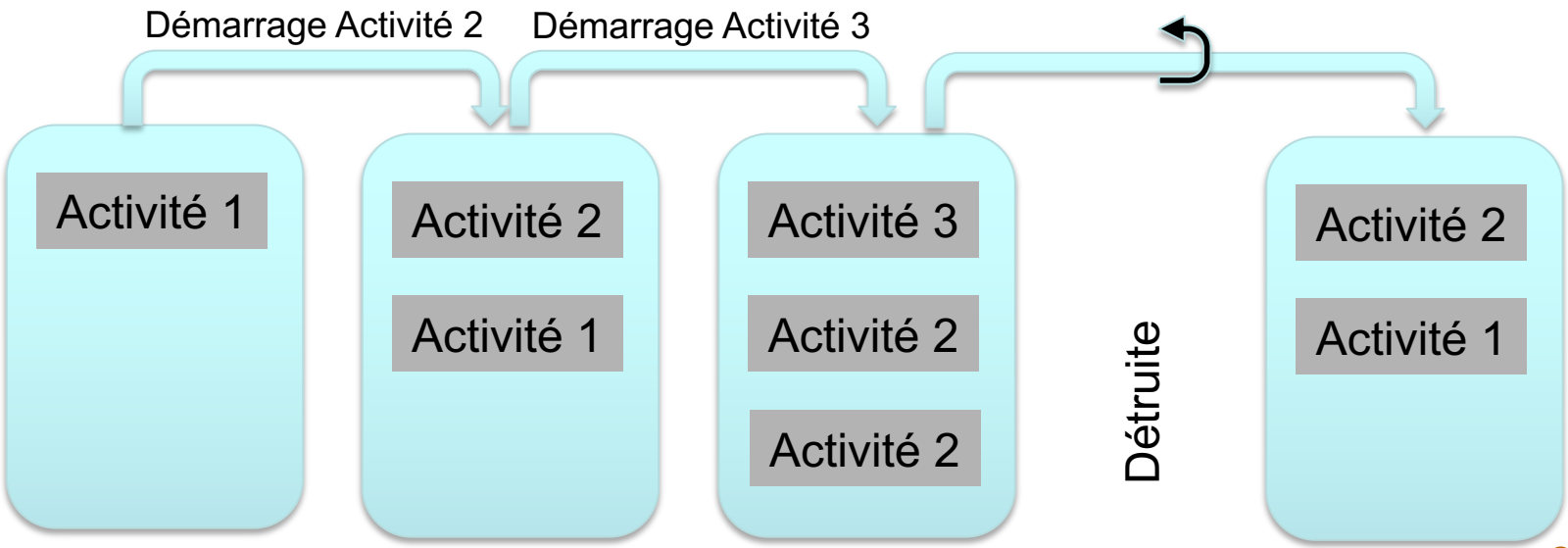
3. Les activités

3- Les activités

- Un composant d'une application, doté d'une interface graphique (IHM) et d'un contexte

- Une activité à la fois visible de l'utilisateur
  - Pour une même application
  - Pour des applications différentes

- Empilement des activités

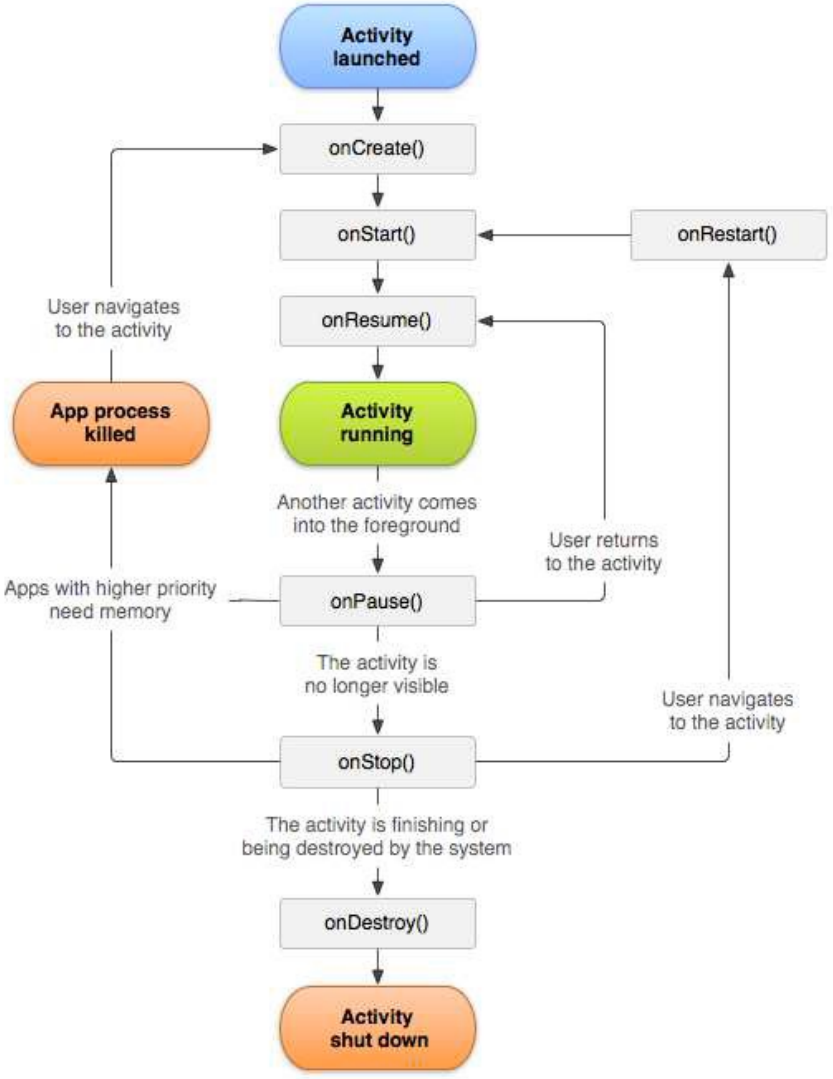


3- Les activités

Cycle de vie

Une activité peut se trouver dans différents états en fonction des actions du système et/ou de l'utilisateur :

- Active : après un appel à *onResume()* (reprise de la main)
- Suspendue : après un appel à *onPause()*
- Arrêtée : après un appel à *onStop()*
- Terminée ; après un appel à *onDestroy()*



3- Les activités

Développement

- Une classe java par activité ;
- Les ressources associées (*layout*, *menu*, ...) ;
- La classe hérite de la classe AppCompatActivity ;
- Génération d'un code minimum par défaut sous Android Studio.

```
package com.example.myhelloworld;

import ...

public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

3- Les activités

D'autres méthodes peuvent être surchargées, en précisant ce qui doit être fait quand :

- *protected void onDestroy()* : l'activité se termine
- *protected void onStart()* : l'activité démarre (ou redémarre)
- *protected void onPause()* : l'activité n'est plus au premier plan
- *protected void onResume()* : l'activité revient au premier plan
- *protected void onStop()* : l'activité n'est plus visible
- *protected void onRestart()* : l'activité redevient visible

3- Les activités

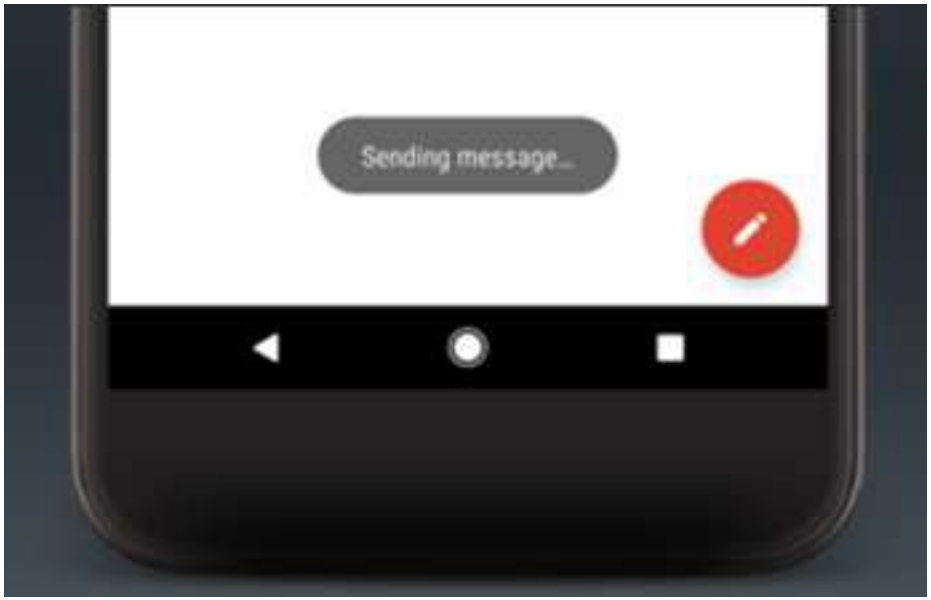
REMARQUE

Affichage de messages de mise au point

- *System.out.println(« texte »);*

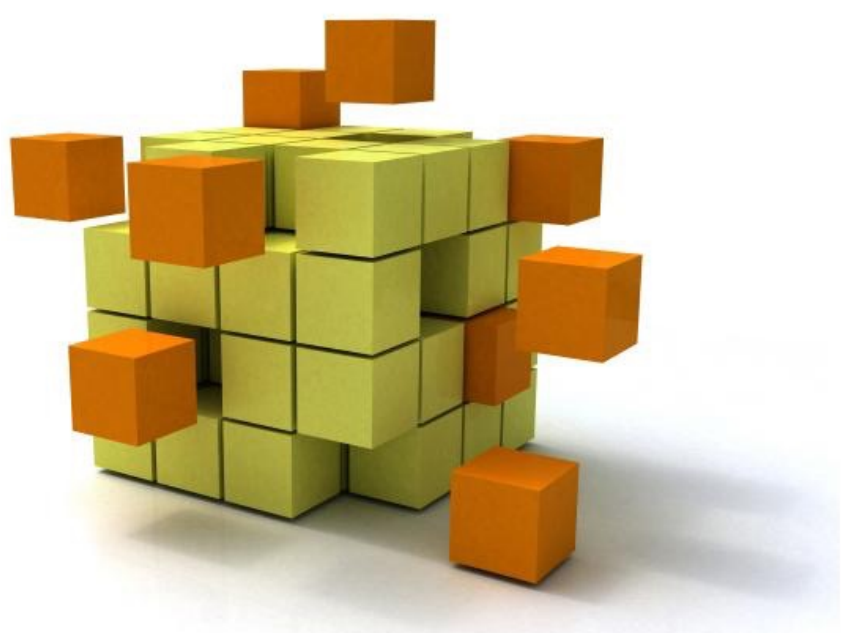
  → affichage dans la console d'android Studio

- La classe *Toast*




```
Toast.makeText(this, "Sending message…", Toast.LENGTH_SHORT).show();
```

Le contexte — La durée d'affichage — Le déclenchement de l'affichage



4. Définir une interface graphique

4- Interface graphique

Quelques règles simples

Interface = seul contact de l'utilisateur

- Faire attirant
- Faire simple
  - L'application doit être intuitive
  - Eviter les trop longs messages

Faire ergonomique

- L'enchaînement des activités doit très rapide
- L'utilisateur doit toujours connaître l'état courant de l'activité

Conseils et « matériels » :

- http://developer.android.com/design/index.html

4- Interface graphique

Quelques règles simples

Définir les « interacteurs »

- Objets graphiques visibles par l'utilisateur pour :
  - L'affichage (texte, images, etc.)
  - L'interaction (boutons, cases, champs de saisie, etc.)

Définir leur mise en page

- Positions dans l'interface (fixes ou relatives)

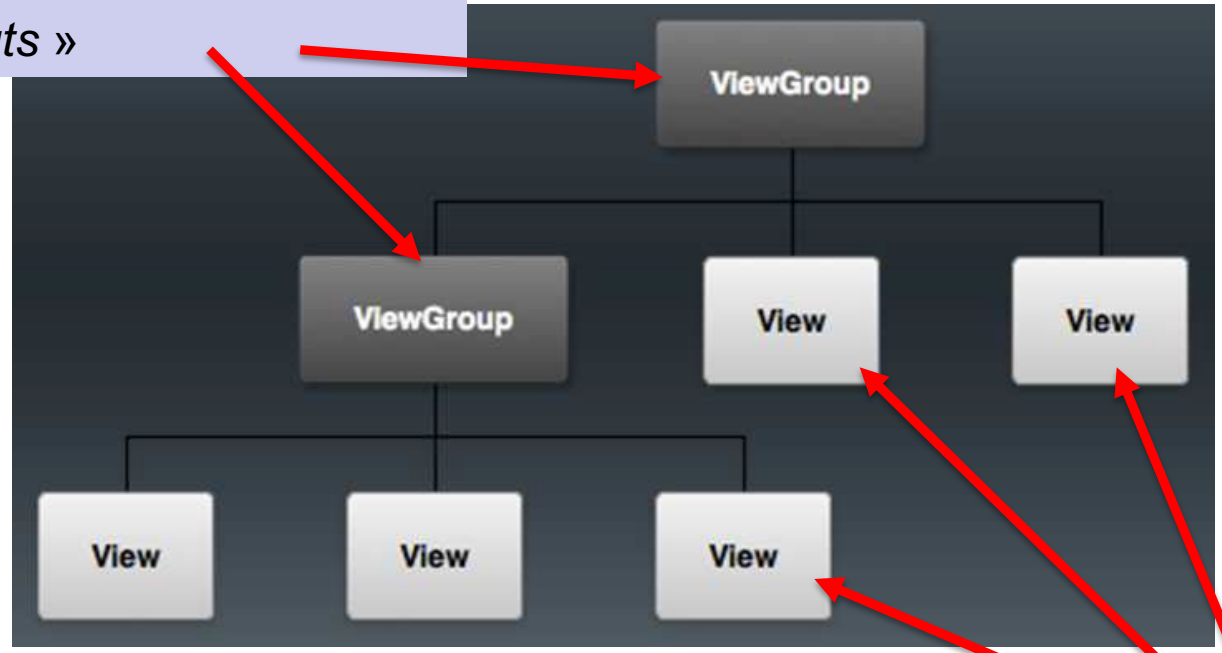
XML ou Java (sauf traitement de l'interaction : Java seul) - Privilégier XML

- Souplesse de mise à jour
- Permet la prise en compte simplifiée de différents types d'écran

4- Interface graphique

Représentation arborescente d'une IG

Conteneurs invisibles, précisant l'organisation de leurs nœuds fils : les « *Layouts* »



Objets graphiques permettant l'interaction (boutons, zones de texte, etc.) : les Widgets

4- Interface graphique

Les *Layouts*

- Zone invisible assurant l'organisation automatique des composants graphiques
  - Peuvent être déclarées en XML ou Java; privilégier XML
    - Séparation du code et de la mise en page
    - Souplesse d'adaptation à différents périphériques

- Possèdent des propriétés « intuitives » permettant l'organisation des composants

- Nombreux *layouts* différents
  - Peuvent être imbriqués (cf arborescence)

- Un *layout* doit être chargé dans onCreate()
  - *setContentView(R.layout.nom_du_layout)*

4- Interface graphique

Les *Layouts*

- **Gestion multi-écrans**
  - Différentes tailles
    - *small*, *normal*, *large*, *xlarge*
  - Différentes densités de pixels
    - *low* (ldpi), *medium* (mdpi), *high* (hdpi), *extra high* (xhdpi)
  - Prévoir un *layout* par taille (et orientation) de l'écran si nécessaire
    - effets de positionnements relatifs pouvant être gênants
  - Prévoir des images en différentes résolutions

4- Interface graphique

Les *Layouts – multi-écrans*

Fonctionnement similaire à l'internationalisation

- Un sous-dossier spécifique à chaque *layout* et/ou à chaque image

```
MyProject/
  res/
    layout/                 # default (portrait)
      main.xml
    layout-land/            # landscape
      main.xml
    layout-large/           # large (portrait)
      main.xml
    layout-large-land/      # large landscape
      main.xml
```

```
MyProject/
  res/
    drawable-xhdpi/
      awesomeimage.png
    drawable-hdpi/
      awesomeimage.png
    drawable-mdpi/
      awesomeimage.png
    drawable-ldpi/
      awesomeimage.png
```

4- Interface graphique

Relative *Layouts*

Positionnement des nœuds par rapport au parent ou les uns par rapport aux autres

activity_main_hello_android.xml

RelativeLayout | EditText

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activity_main_hello_android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:paddingTop="16dp"
    android:paddingBottom="16dp"
    tools:context="com.example.macbook_derrode.helloandroid.MainActivityH

    <EditText
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:inputType="textPersonName"
        android:text="Your Name..."
        android:ems="10"
        android:layout_alignParentTop="true"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"
        android:layout_marginLeft="36dp"
        android:layout_marginStart="36dp"
        android:layout_marginTop="36dp"
        android:id="@+id/editText2" />

    <Button
        android:text="@android:string/ok"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
```

- **match_parent :** S'adapte à la taille du conteneur parent (ici l'écran)
- **wrap_content :** s'adapte à la taille de ce qu'il contient (ici deux zones de texte)
- **dimension fixe**

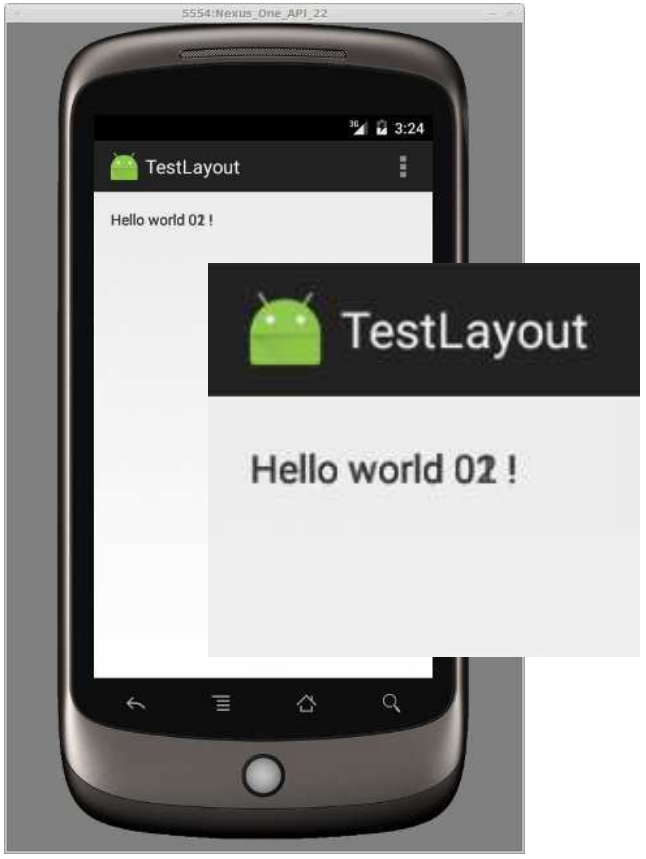
4- Interface graphique

Relative *Layouts*

Tous les nœuds sont positionnés à partir du coin supérieur gauche → Superposition !!

MainActivity.java × activity_main.xml × strings.xml ×

```
<RelativeLayout
    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"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin" tools:context=".MainActivity">

    <TextView
        android:text="@string/hello_world_01"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

    <TextView
        android:text="@string/hello_world_02"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        />

</RelativeLayout>
```



4- Interface graphique

Relative *Layouts* – Attributs de positionnement / parent

- android:layout_centerHorizontal
- android:layout_centerVertical
- android:centerInParent
- ... (cf RelativeLayout.LayoutParams)

MainActivity.java | activity_main.xml | strings.xml

```
<RelativeLayout
    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"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin" tools:context=".MainActivity">

    <TextView
        android:text="@string/hello_world_01"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        />

    <TextView
        android:text="@string/hello_world_02"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerVertical="true"
        />

</RelativeLayout>
```

4- Interface graphique

Relative *Layouts* – Attributs de positionnement / autres nœud

- android:layout_below
- android:layout_above
- android:layout_toLeftOf
- android:layout_toRightOf
- … (cf RelativeLayout.LayoutParams)

Nécessité de nommer les nœuds

Permet de préciser le nœud à partir duquel on se positionne


```
<TextView
    android:id="@+id/hw01"
    android:text="@string/hello_world_01"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_centerHorizontal="true"
    />
```

```
<TextView
    android:text="@string/hello_world_02"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_toLeftOf="@id/hw01"
    />
```

4- Interface graphique

Linear Layouts – Aligne les nœuds dans une seule direction

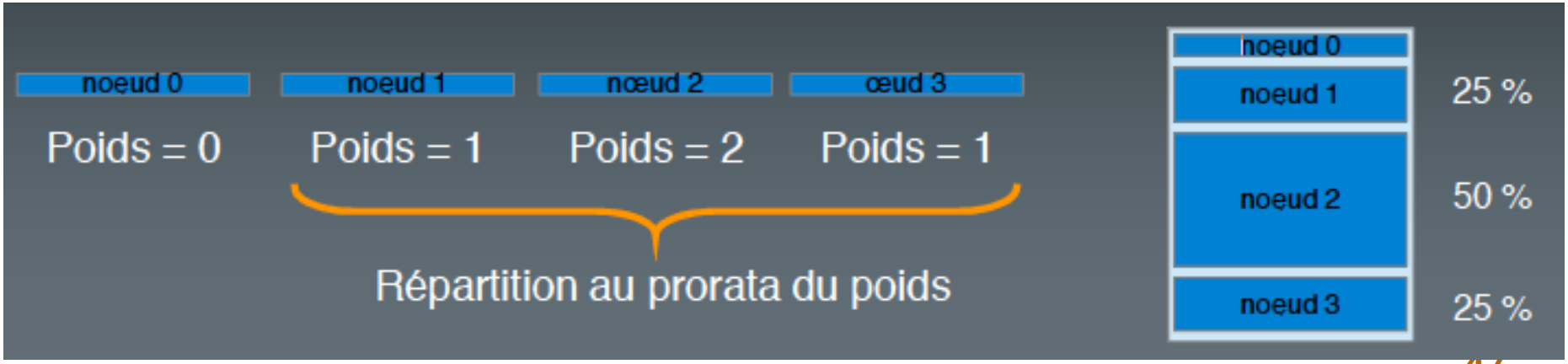
- horizontale (par défaut)
- verticale

4- Interface graphique

Linear Layouts – Modification du « poids » des nœuds

- Permet de changer la taille de la zone occupée par chaque nœud dans l'écran

- Ajout d'un attribut *android:layout_weight* à chaque nœud
  - 0 (défaut) : n'utilise que la zone nécessaire au nœud
  - n>0 : poids du nœud par rapport aux autres nœuds



4- Interface graphique

Linear Layouts – Exemple

```
<TextView
    android:text="@string/hello_world_01"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    />

<TextView
    android:text="@string/hello_world_02"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_weight="2"
    />
<TextView
    android:text="@string/hello_world_03"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_weight="1"
    />
<TextView
    android:text="@string/hello_world_04"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_weight="2"
    />
<TextView
    android:text="@string/hello_world_05"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    />
```

4- Interface graphique

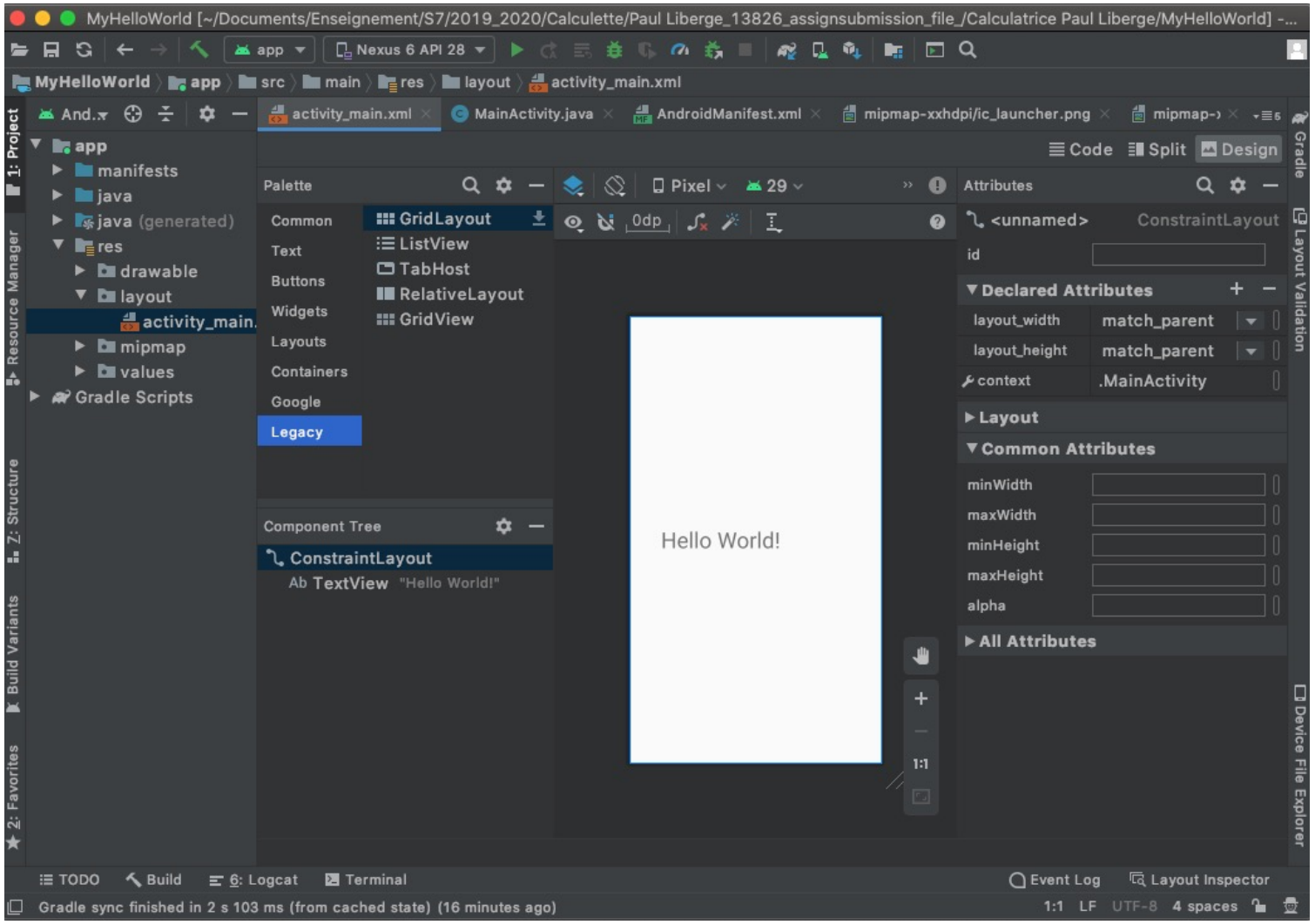
Linear Layouts – Alignement de chaque nœud dans sa zone

- Ajout d'un attribut
  android:layout_gravity

- Nombreuses valeurs possibles :
  - center, center_vertical, center_horizontal
  - left, right, top, bottom
  - (cf LinearLayout.LayoutParams)

4- Interface graphique

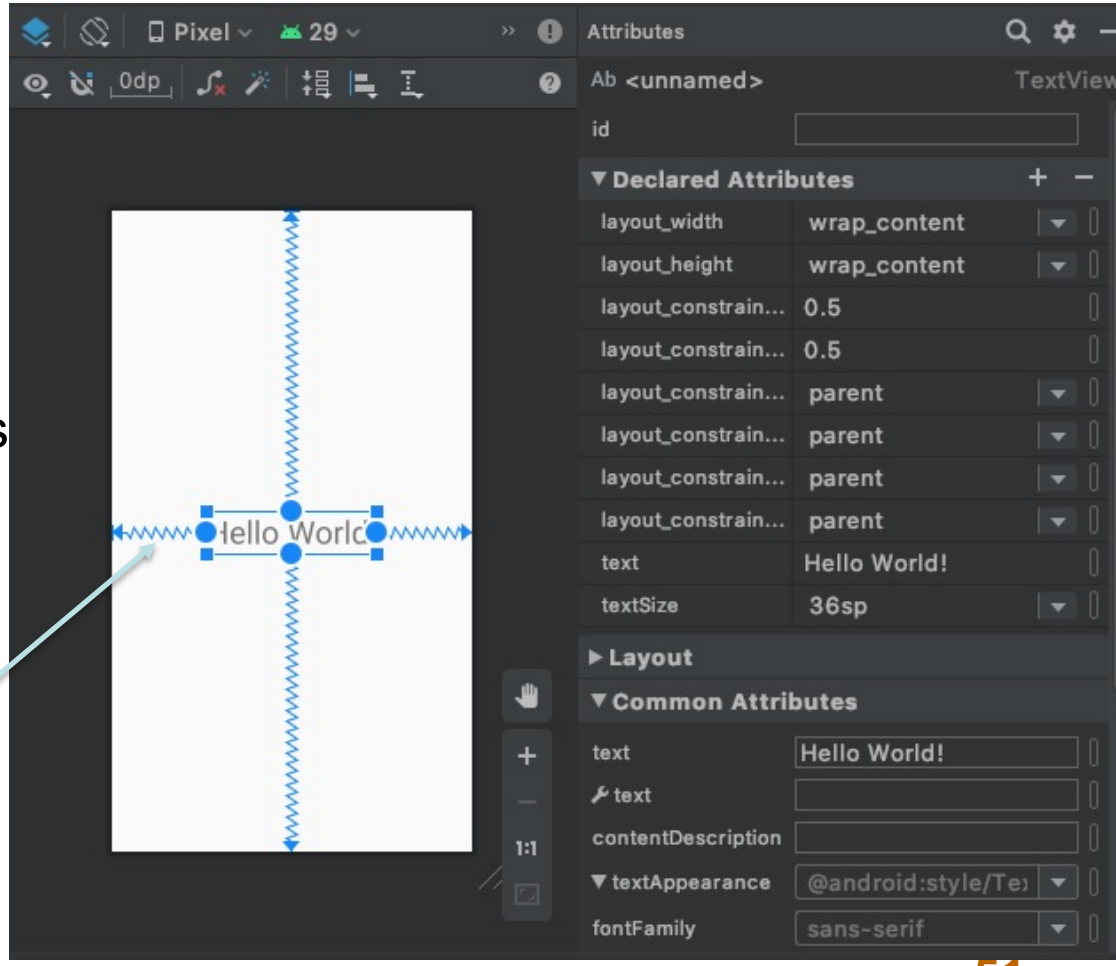
Remarque : passage en mode graphique



4- Interface graphique

Constraint Layout – Faciliter la création de layouts complexes

- Philosophie similaire au *RelativeLayout* :
  - Création de relations entre les composants graphiques et avec leur layout parent
    - Notion de contraintes
  - Plus flexible
  - Mieux adapté à l'éditeur graphique

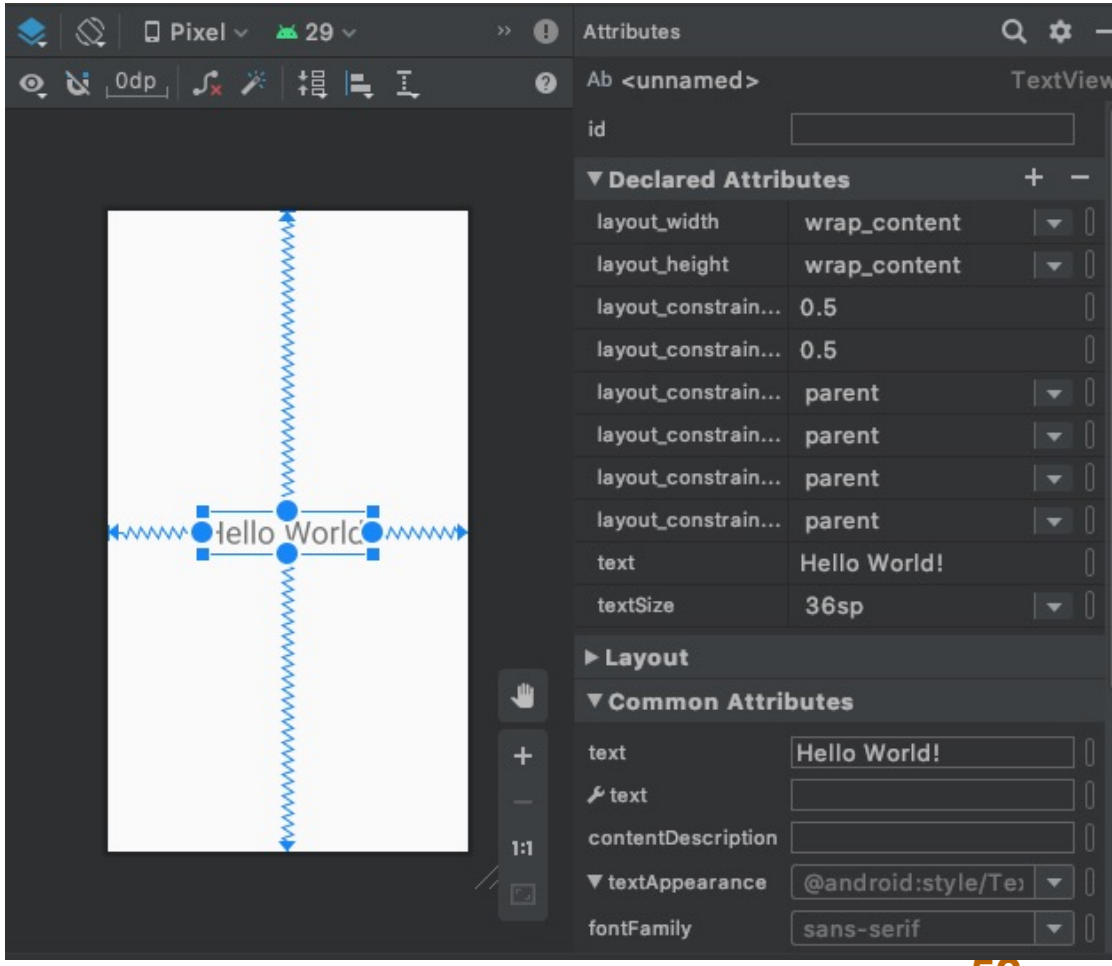


Contrainte gauche

4- Interface graphique

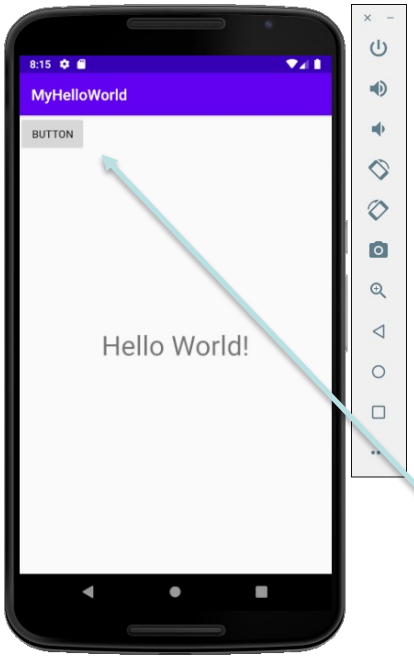
Constraint Layout — Faciliter la création de layouts complexes

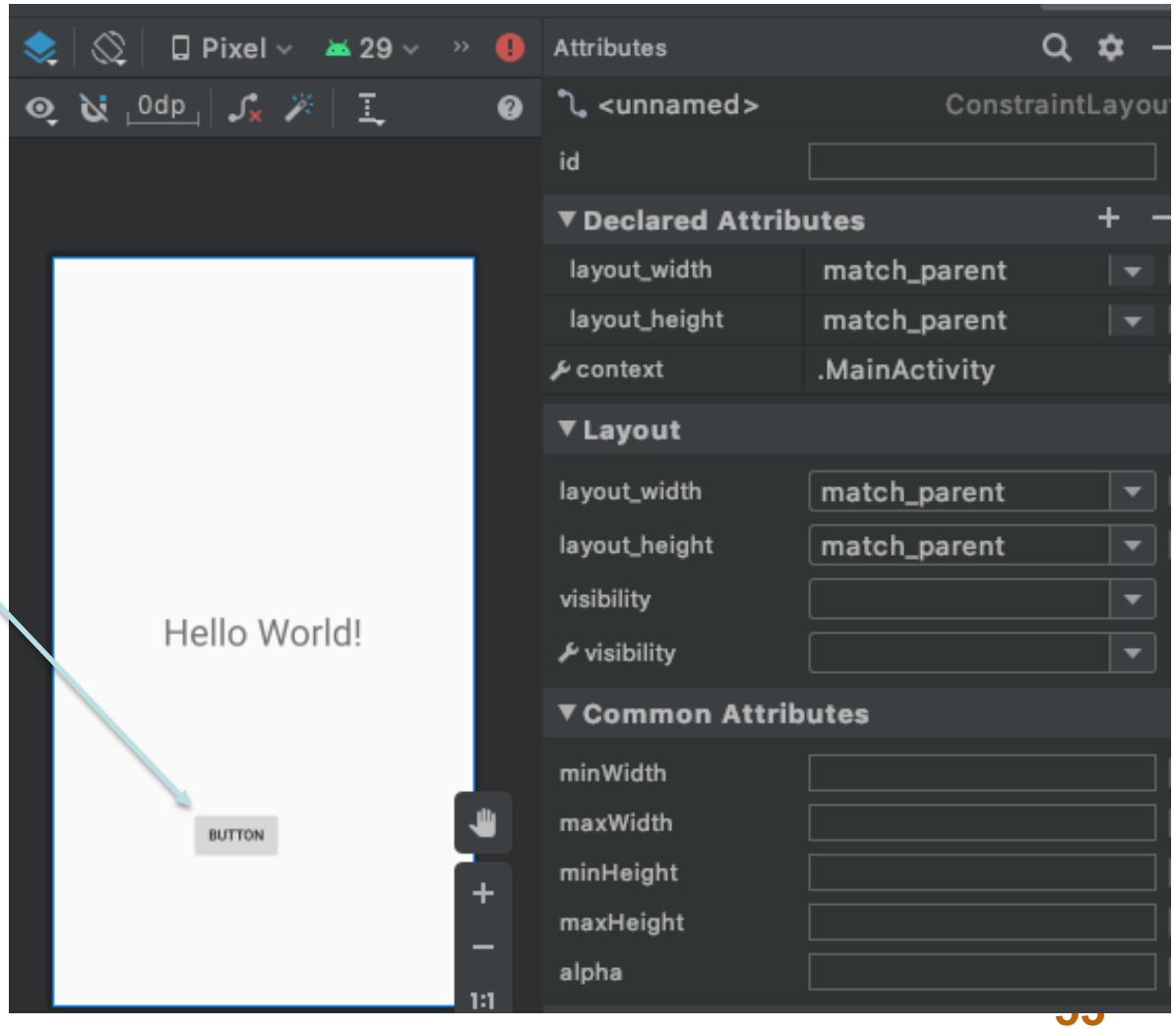
```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmln
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/text_name"
        android:textSize="36sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintHorizontal_bias="0.5"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.5" />

</androidx.constraintlayout.widget.ConstraintLayout>
```



4- Interface graphique

Constraint Layout – Placement sans contrainte

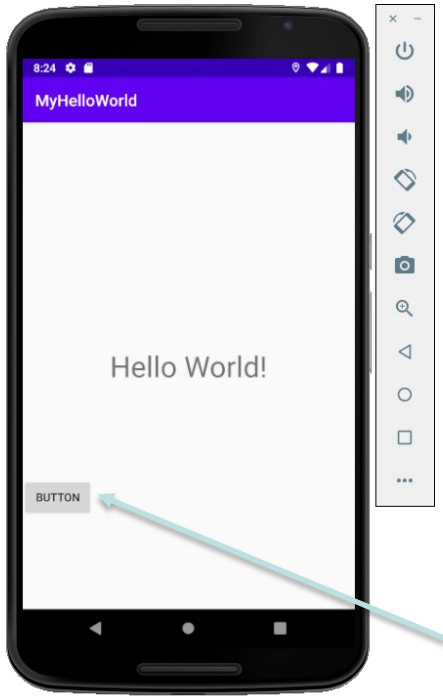


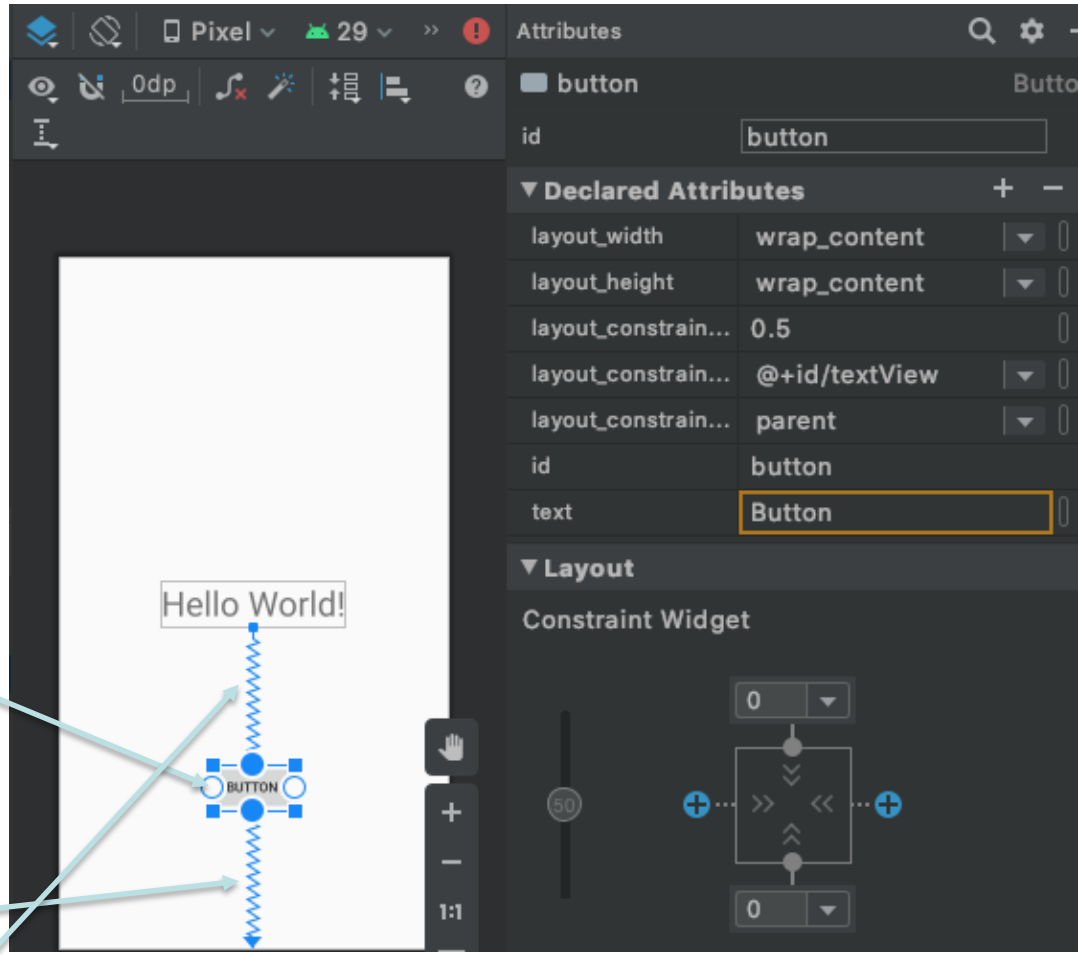



```
<Button
    android:id="@+id/button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Button"
    tools:layout_editor_absoluteX="136dp"
    tools:layout_editor_absoluteY="541dp" />
```

4- Interface graphique

Constraint Layout – Placement avec contrainte verticale

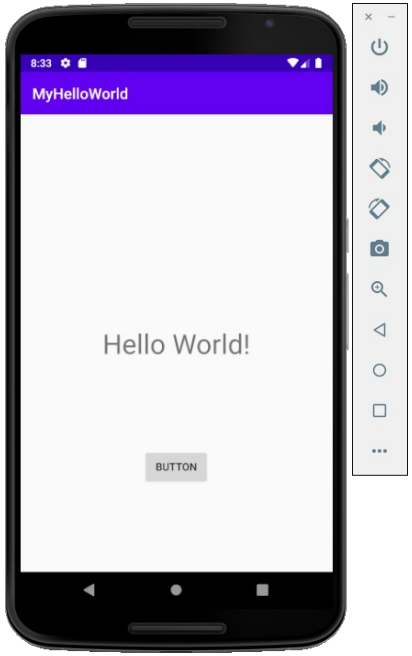




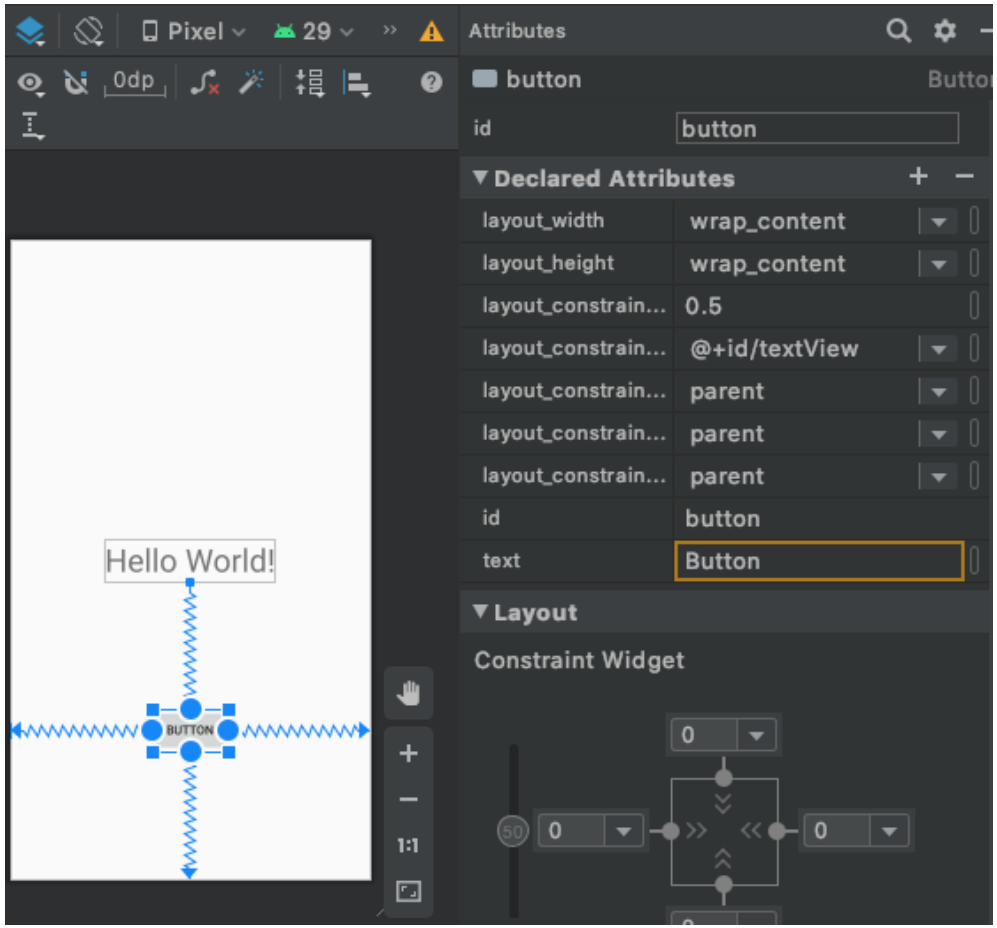

```
<Button
    android:id="@+id/button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Button"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/textView"
    app:layout_constraintVertical_bias="0.5"
    tools:layout_editor_absoluteX="161dp" />
```

4- Interface graphique

Constraint Layout — Placement avec contraintes



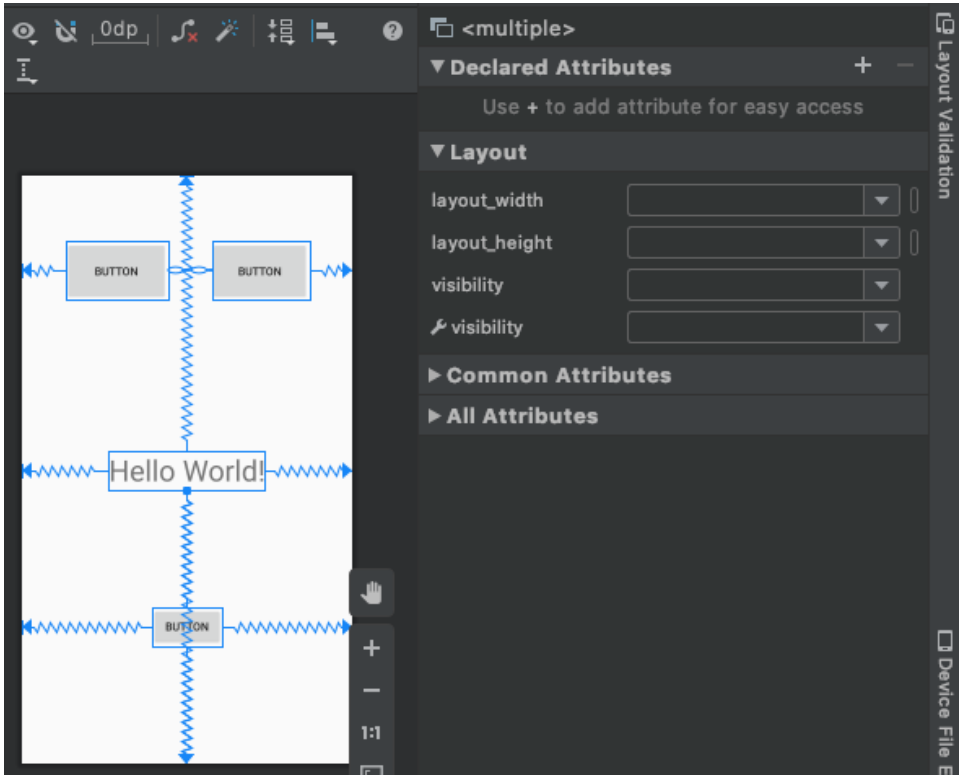

```
<Button
    android:id="@+id/button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Button"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/textView"
    app:layout_constraintVertical_bias="0.5" />
```



4- Interface graphique

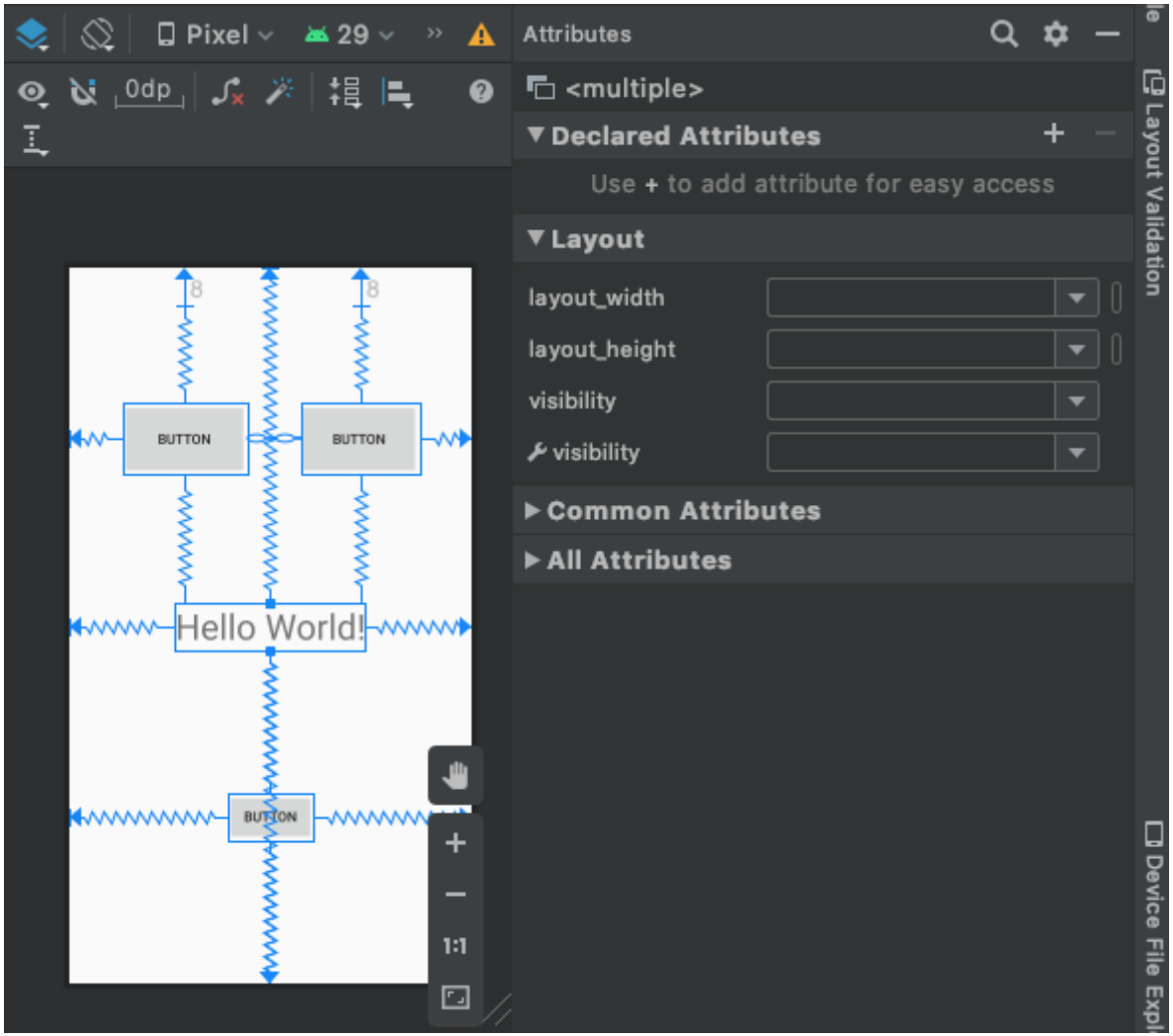
Constraint Layout – Chainage

```
<Button
    android:id="@+id/button2"
    android:layout_width="127dp"
    android:layout_height="74dp"
    android:text="Button"
    app:layout_constraintEnd_toStartOf="@+id/button3"
    app:layout_constraintHorizontal_bias="0.5"
    app:layout_constraintStart_toStartOf="parent"
    tools:layout_editor_absoluteY="82dp" />

<Button
    android:id="@+id/button3"
    android:layout_width="121dp"
    android:layout_height="74dp"
    android:text="Button"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.5"
    app:layout_constraintStart_toEndOf="@+id/button2"
    tools:layout_editor_absoluteY="82dp" />
```



4- Interface graphique

Constraint Layout – Tout est relatif!



4- Interface graphique

Les Widgets

- Composants graphiques visibles par l'utilisateur
  - Widgets simples : zones de texte, boutons, listes
  - Widgets plus complexes : horloges, barres de progression, ...
- Héritent de la classe View
- Utilisation :
  - Définition en XML (type, taille, centrage, position, ...)
  - Comportement en Java
  - Peuvent également être créés dynamiquement en Java

4- Interface graphique

Les TextView

- Widget permettant l'affichage d'un texte
  - Normalement non éditable
- Exemple :

```
<TextView
    android:id="@+id/text"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/letexte"
    android:hint="texte initial"
    android:layout_gravity="center"
    android:gravity="center"/>
```

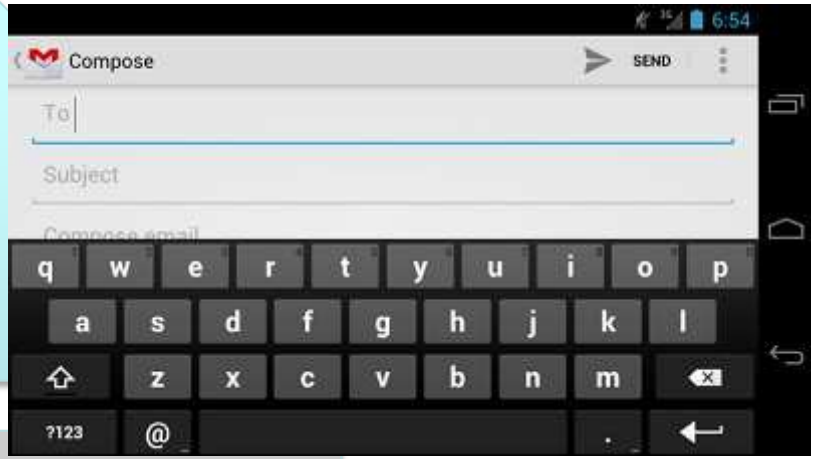
- Nombreux autres attributs: Cf classe TextView

4- Interface graphique

Les EditText

- Widget permettant la saisie d'un texte (TextFields)
  - – Accès : ouverture d'un clavier pour la saisie
  - – nombreux attributs permettant l'aide à la saisie

```
<EditText
    android:id="@+id/email_address"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:hint="@string/email_hint"
    android:inputType="textEmailAddress"
    />
```



"text" : Normal text.
"textEmailAddress" : Normal text with the @ character.
"textUri" : Normal text with the / character.
"number" : Basic number keypad.
"phone" : Phone-style keypad.

4- Interface graphique

Les Button

- Widget représentant un bouton d'action
  - Renvoie un évènement lors de l'appui
  - Peut contenir un texte, une image ou les deux
- Exemples :

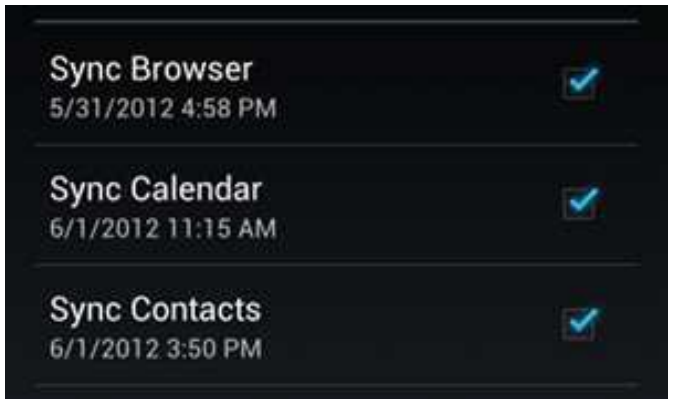
```
<Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/button_text"
... />
```

```
<Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/button_text"
        android:drawableLeft="@drawable/button_icon"
... />
```

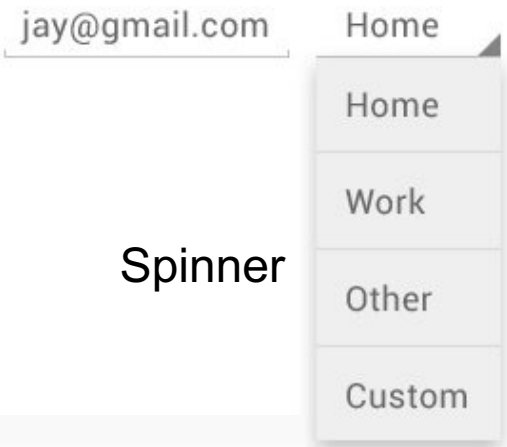

```
<ImageButton
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@drawable/button_icon"
... />
```

4- Interface graphique

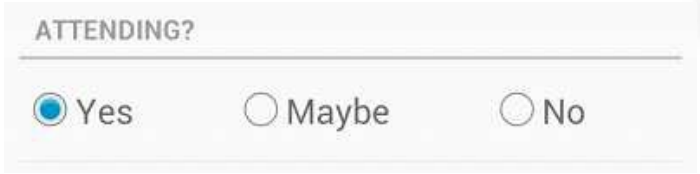
Quelques widgets sup



CheckBox



Spinner



RadioButton

ToggleButton

Switch

4- Interface graphique

Implémentation du comportement

- Les fichiers XML ne permettent que de :
  – positionner les composants ;
  – définir leurs caractéristiques.

- Nécessité de :
  – définir leur comportement
    - type d'interaction (clic court, clic long, ...)
    - code de prise en compte (Java)
  – lier composant et code
    - XML : attribut *android:onClick*
    - Java : instancier un *event listener*

4- Interface graphique

Implémentation du comportement

- Attribut android:onClick
  - – Doit être suivi du nom de la méthode à appeler en cas de déclenchement
  - – Prototype :
    - *public void nomDeLaMethode(View maVue)*

```
<Button
        android:id="@+id/monBouton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/monTexte"
        android:onClick="onBoutonClique"
/>
```

```
public void onBoutonClique(View maVue) {
        System.out.println( "Clique") ;
}
```

Permet de récupérer des informations sur le composant graphique qui a généré l'évènement

Récupération :

maVue.getId() → **R.id.monBouton**

4- Interface graphique

Implémentation du comportement

- Les *event listener*
  – interfaces de la classe *View*
  – ne disposent que d'une seule méthode à implémenter
  – méthode appelée quand le composant associé est déclenché par l'utilisateur

- Exemples :

Interface	**Méthode**
View.OnClickListener	abstract void onClick(View v)
View.OnLongClickListener	abstract boolean onLongClick(View v)
View.OnFocusChangeListener	abstract void onFocusChange(View v, boolean hasFocus)

4- Interface graphique

Implémentation du comportement

- Exemple : l'interface View.onClickListener
  - public void onClick(View v)

Retrouver un widget à partir du nom qui lui a été associé dans le xml

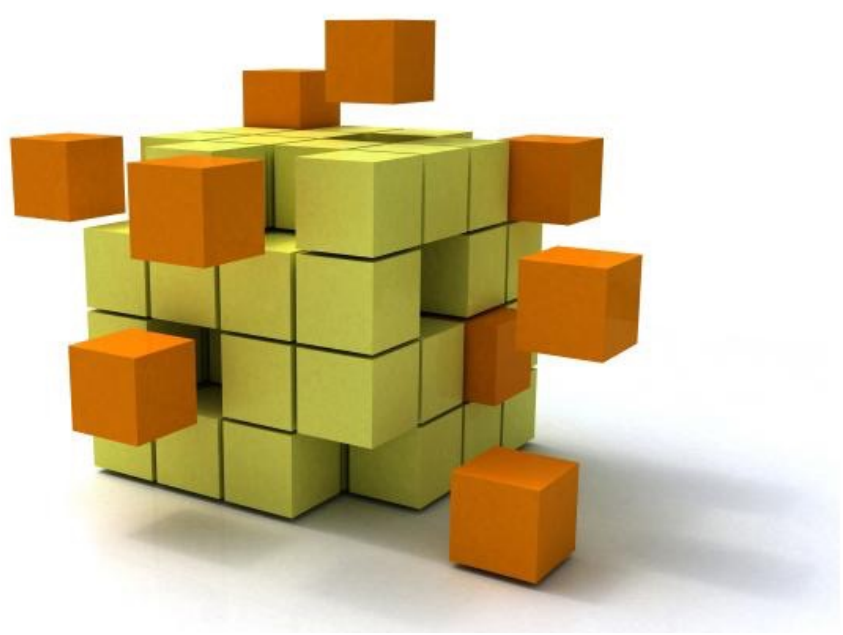
```
...
Button button = (Button) findViewById(R.id.button_name);

button.setOnClickListener(new View.OnClickListener() {

    public void onClick(View v) {
        // Do something in response to button click
    }

});
...
```

Création d'un « OnClickListener »

Source : developer.android.com

Associer un « onClickListener » au bouton

Surcharge de la méthode « onClick » de l'interface « onClickListener »



5. Les intentions (une introduction)

5- Les intentions

Classe représentant un message échangé entre une activité et un composant présent sur le système

- Une autre activité
- Un service
- Un diffuseur d'événements

Deux types de messages

- Explicite : on nomme le composant à démarrer
- Implicite : on demande au système de trouver un composant adéquat, en fonction d'une action à effectuer

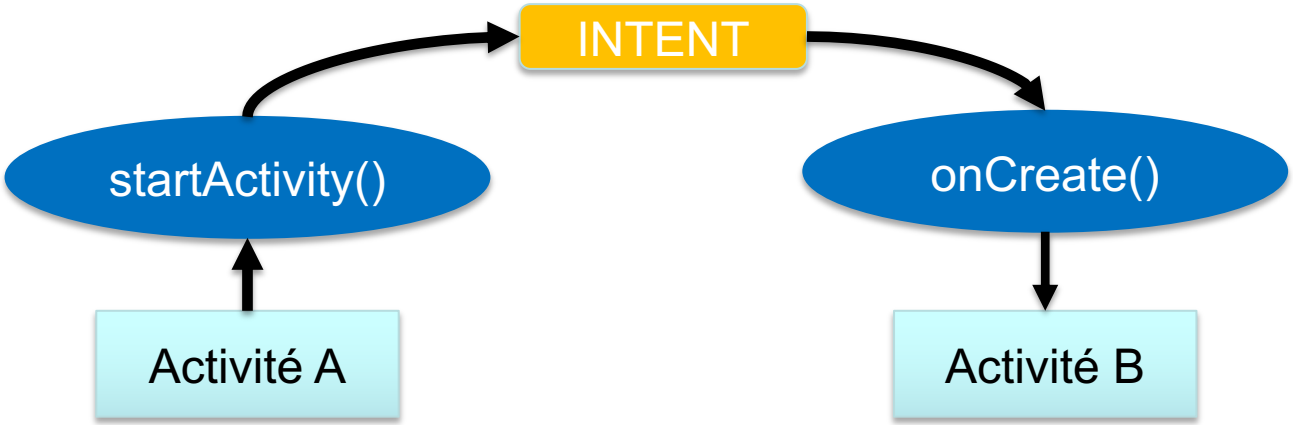
5- Les intentions

Les intentions explicites:

Message adressé à un composant connu

- On donne le nom de la classe correspondante
- Généralement réservé aux composants appartenant à la même application …

Le composant est démarré immédiatement

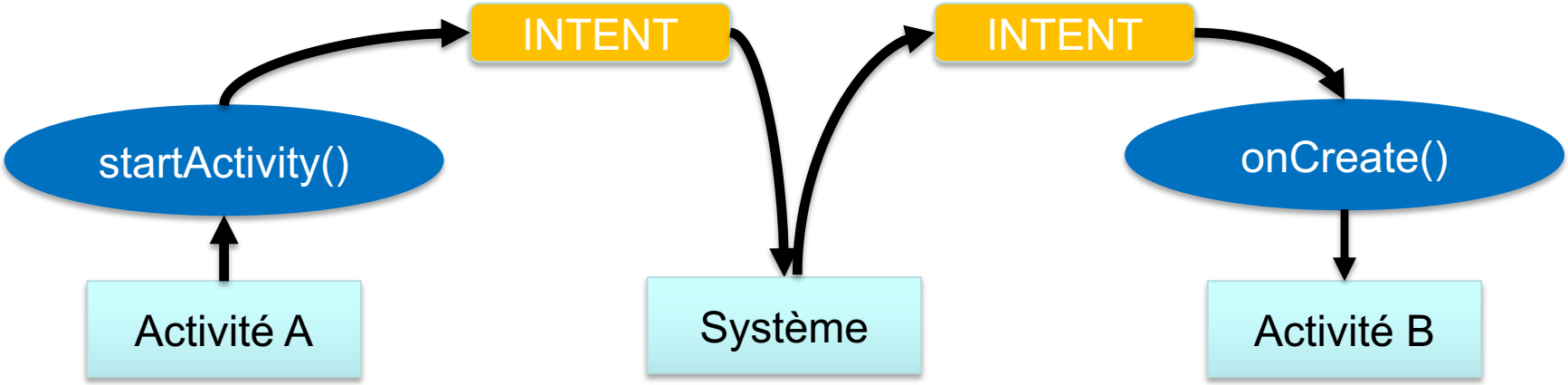


5- Les intentions

Les intentions implicite:

Message à destination d'un composant inconnu

- Le système se charge de trouver le composant adéquat et le démarre



- En cas de possibilités multiples, affichage des choix à l'utilisateur

5- Les intentions

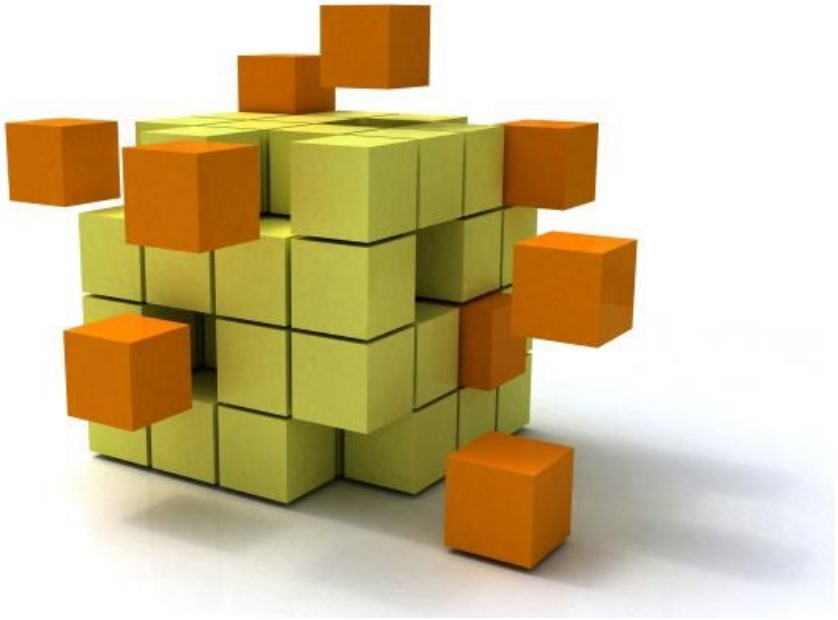
Les Intent Filters

- Utilisation d'intents implicites :
  - Le système recherche le(s) composant(s) possible(s)
  - Le système doit connaître les types d'intents que peut recevoir chaque composant

    Nécessité de préciser pour chaque composant les intents réceptionnables

    → Les intent filters

- Présents dans le fichier manifest.xml pour chaque composant d'un application.



Travaux Pratiques et BE

Travaux pratiques

- Finir le tuto sur Pédagogie, intitulé *TutorialActivity.pdf*
- Faire le BE "calculette" ou un "convertisseur de monnaies".
- Utiliser un émulateur Nexus 6, API 23.

Toute amélioration sera appréciée : bouton graphiques, multilingues, design original, plusieurs activités, gestion de la rotation du périphérique, menu déroulant ou contextuel, le logo de l'application original…