



# FORMATION ANDROID

# QUI SUIS-JE ?

- Anthony Monteiro
- Indépendant sur Toulouse
- Spécialisations
  - Architecture (Graphique, technique, projet).
  - User Experience
- Application sur le store :
  - UrbanPulse
  - Acadomia (iOS & Android)
- Site internet : [www.amonteiro.fr](http://www.amonteiro.fr)
- Contact : [contact@amonteiro.fr](mailto:contact@amonteiro.fr)

# PROJET

- Lyra Network
  - Payzen Mobile
  - SDK de paiement en ligne
- Fédération de Flying Disc France (Frisbee)
  - Ultimate Line Manager

Présentation et tour de table.

# PLAN

- Premiers pas
  - Présentation / Installation de l'IDE
- Rappels sur Java
- L'univers d'Android
- Architecture d'un projet
  - Les différents types de fichiers et classes
  - Les Activités
- IHM
  - Layouts, composants graphiques
  - Gestion des événements
  - RecyclerView/ListView
  - Communication entre activités

# PLAN

- AlerteDialog, Toast et Menu
- BroadCast
- Services
- Handler
- AsyncTask
- Les fragments
- Conseils d'architecture
- Ergonomie et User Expériences
- SQLite
  - Avec et sans GreenDAO

# PLAN

- Présentation de librairies existantes
  - GraphView
  - Zbar
  - Facebook
  - Scribe
- Notification et GCM
- Web
  - Requête HTTP
  - JSON
  - Webservice
- Google Map



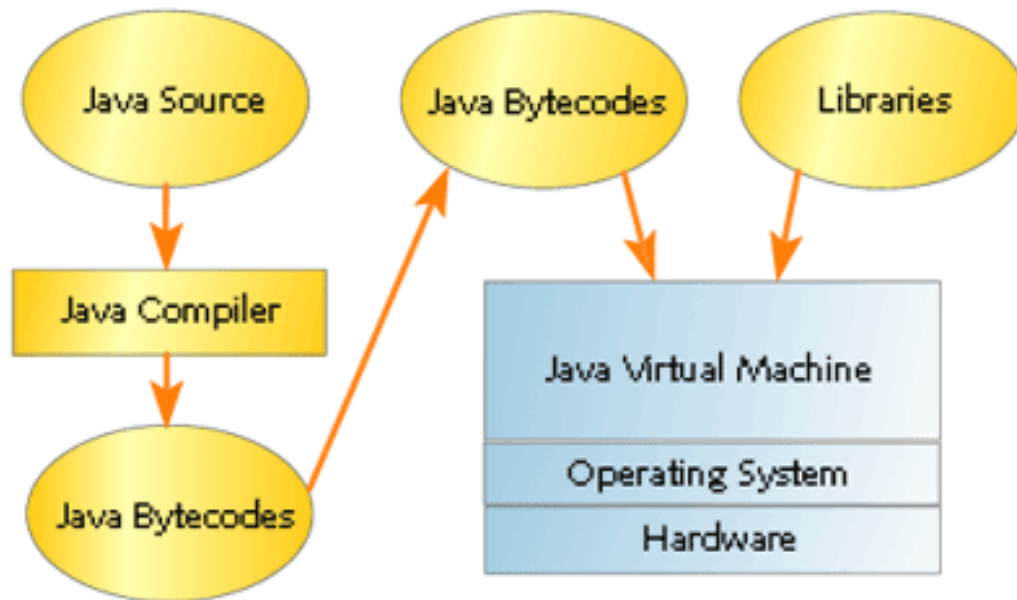
# RAPPELS JAVA

8



# RAPPELS JAVA

- La JVM



# RAPPELS JAVA

```
//Classe
public class Eleve {
    //Attribut
    private String nom;
    //constructeur
    public Eleve(String nom) {
        this.nom = nom;
    }
    //methode
    private void doSomething() {
        ...
    }

    //getter
    public String getNom() {
        return nom;
    }
    //setter
    public void setNom(String nom) {
        this.nom = nom;
    }
}
```

# RAPPELS JAVA

```
private void genocide() {  
  
    Eleve eleve = null;  
    //NullPointerException  
    eleve.getNom();  
  
    eleve = new Eleve("Bob"); //Allocation Mémoire pour créer Bob  
    eleve.setNom("John"); //Bob préfère qu'on l'appelle John  
  
    //On réassigne le pointeur, plus personne ne pointe sur l'espace mémoire de John  
    //il va être « garbage collecté ». Adieu John  
    eleve = new Eleve("Candy");  
  
    //Nous avons 2 pointeurs sur Candy  
    Eleve eleve2 = eleve;  
  
    //Nous n'avons plus qu'un pointeur sur Candy  
    eleve = null;  
  
    //Plus personne ne pointe sur Candy. Prépare toi à être recycler  
    eleve2 = null;  
}
```

# RAPPELS JAVA

- Conditions

```
boolean condition = true;
Eleve eleve = new Eleve("bob");

//Condition
if(eleve == null || condition) {
    //...
}
else if(eleve.getNom().equals("bob")) {
    //...
}
else {
    //...
}
```

# RAPPELS JAVA

- Les collections

	Utilisation générale
List	ArrayList LinkedList
Set	HashSet TreeSet LinkedHashSet
Map	HashMap TreeMap LinkedHashMap

- SparseArray HashMap<Integer, ?>

# RAPPELS JAVA

- List

```
//List
ArrayList<Eleve> eleveArrayList = new ArrayList<>();
eleveArrayList.add(eleve);

//Parcours de liste
for (Eleve e : eleveArrayList) {
    e.setNom(e.getNom() + "_eleve");
}

for(int i=0; i<eleveArrayList.size(); i++) {
    Eleve e = eleveArrayList.get(i);
    e.setNom(e.getNom() + "_eleve");
}

//While
int i = eleveArrayList.size() -1;
while(i>0) {
    Eleve e = eleveArrayList.get(i);
    e.setNom(e.getNom() + "_eleve");
    i--;
}
```

# EXERCICE

- Créez une liste de 10 élèves avec comme prénom « Eleve0, Eleve1, Eleve2... »
- Créez une méthode qui affiche une liste dans la console
- Créez une méthode qui ajouter « \_check » au prénom de chaque élève
- Créez une méthode qui retire de la liste 1 élève sur 2.

# RAPPELS JAVA

- Tableau

*//Déclaration d'un tableau de primitive ou d'objet*

```
float[] monTab;
```

```
Eleve[] mesEleves;
```

*//Instanciation du tableau, taille fixe obligatoire*

```
monTab = new float[10];
```

```
mesEleves = new Eleve[10];
```

*//Ecrire dans un tableau*

```
monTab[0] = 3;
```

```
mesEleves[1] = new Eleve();
```

*//Lire dans un tableau*

```
int i = monTab[0];
```

```
Eleve temp = meEleves[1];
```

*//Taille du tableau*

```
int size = monTab.length;
```



# EXERCICE

- Remplir un tableau de valeur aléatoire.
- Créez une méthode permettant de trier le tableau passé en paramètre à la main. (Sans utiliser la méthode de tri proposée par Java.)
- Créez une méthode triant le tableau passé en paramètre à l'aide de la méthode proposée par Java. Recherche Google
- A l'aide de la méthode, calculez le temps que prend de trier 100 fois un tableau de taille 10000. Ne pas recréer le tableau entre chaque trie, simplement y remettre des valeurs aléatoires.
  - `long debut = System.currentTimeMillis();`
- Faire de même avec le tri proposé par Java.

## EXERCICE : LE DRAPEAU FRANÇAIS

- Créez une énumération contenant 3 valeurs : Bleu, Blanc, Rouge
- Créer un tableau de taille 15 contenant aléatoirement ces 3 valeurs.
- Triez le tableau de manière à mettre les bleus en premier puis les blancs puis les rouges.
- Améliorez votre algorithme pour ne le faire qu'en 1 passage du tableau (Une seule boucle for).

# RAPPELS JAVA

- HashMap

```
//Déclaration
HashMap<Integer, Eleve> mesEleves;

//Instanciation de la Hashmap, taille fixe obligatoire
mesEleves = new HashMap<Integer, Eleve>();

//Ajout / remplace
mesEleves.put(3, new Eleve());

//Récupération
Eleve eleve = mesEleves.get(3);

//Taille de la hashmap
int size = mesEleves.size();

//Parcours
for (Map.Entry<Integer, Eleve> unEleve : mesEleves.entrySet()) {
    String cle = unEleve.getKey();
    Eleve temp = unEleve.getValue;
}
```

# EXERCICE

- Créez 50 élèves avec un prénom commençant par un chiffre aléatoire.
- Les positionner dans une `HashMap<Integer, ArrayList<Eleve>>` en fonction de leur chiffre de départ.
- Afficher dans la console la `HashMap` sous cette forme :  
    0 : 0Eleve32 0Eleve35 0Eleve42  
    1 : 1Eleve3 1Eleve33 1Eleve43  
    ...

# HÉRITAGE : UTILISATION DE LA CLASSE MÈRE

- Pour accéder aux attributs de la classe mère ceux-ci doivent avoir été déclarés avec la visibilité **protected** ou **public**
- L'équivalent de **this** pour la classe mère est **super**

```
1 public static class Personne {
2     protected String nom;
3     public Personne(String nom) {
4         this.nom = nom;
5     }
6 }
1 public static class Eleve extends Personne {
2     protected String section;
3     public Eleve(String nom, String section) {
4         super(nom); //Appel du constructeur de Personne
5         this.section = section;
6     }
7 }
1 public static void main(String[] args) {
2     Eleve e = new Eleve("Bob", "Math"); //ok
3     Personne p = new Personne("Bob2"); //ok
4     Personne p2 = new Eleve("Bob3", "Math"); //Ok Eleve est aussi de type personne
5     // KO, la classe personne n'est pas de type Eleve.
5     Eleve im1 = new Personne("Bob4");
6 }
```

- Ordre d'exécution des lignes :

• 1 - ... - 4 - 3 - 4 - 3 - 4 - 5 - 5 - 6 - 6

# HÉRITAGE : REDÉFINITION

- Une extension peut redéfinir une méthode de sa classe mère
  - Elle doit conserver la même signature

```
1 public class Personne {
2     protected String nom;
3     public void print() {
4         System.out.println("Je m'appelle " + nom);
5     }
6 }
```

```
1 public class Eleve extends Personne {
2     protected String section;
3
4     @Override
5     public void print() {
6         //Si je souhaite appeler la classe de la méthode mère
7         super.print();
8         System.out.println(" et je suis en classe de " + section);
9     }
10 }
```

```
1 public static void main(String[] args) {
2     Personne m1 = new Eleve();
3     m1.build();
4     Personne m2 = new Personne();
5     m2.build();
6 }
```

- Ordre d'exécution des lignes :

• 1 - 2 - 3 - 5 - 7 - 3 - 4 - 5 - 7 - 9 - 4 - 5 - 3 - 4 - 5 - 6

# RAPPELS JAVA

- Interface

//Déclaration

```
public interface OnClickOnEleve{  
    void onClick (Eleve eleve);  
}
```

//Implémentation

```
public classe MyClass implements OnClickOnEleve{  
    public void onClick (Eleve eleve) {  
        //...  
    }  
}
```

# RAPPELS JAVA

- Java.util
  - Arrays, Calendar, Date, Random, Timer...
- Transition List <-> Tableau
  - `Eleve[] elevTab = eleveList.toArray(new Eleve[0]);`
  - `Arrays.asList(elevTab);`
- Commons.lang (StringUtils, NumberUtils...)
  - `If(StringUtils.isNotBlank(var)) {`  
...  
`}`



# GESTION DES EXCEPTIONS

## ○ Définition

- Une exception est un signal qui indique que quelque chose d'exceptionnel (comme une erreur) s'est produit. Elle interrompt le flot d'exécution normal du programme.

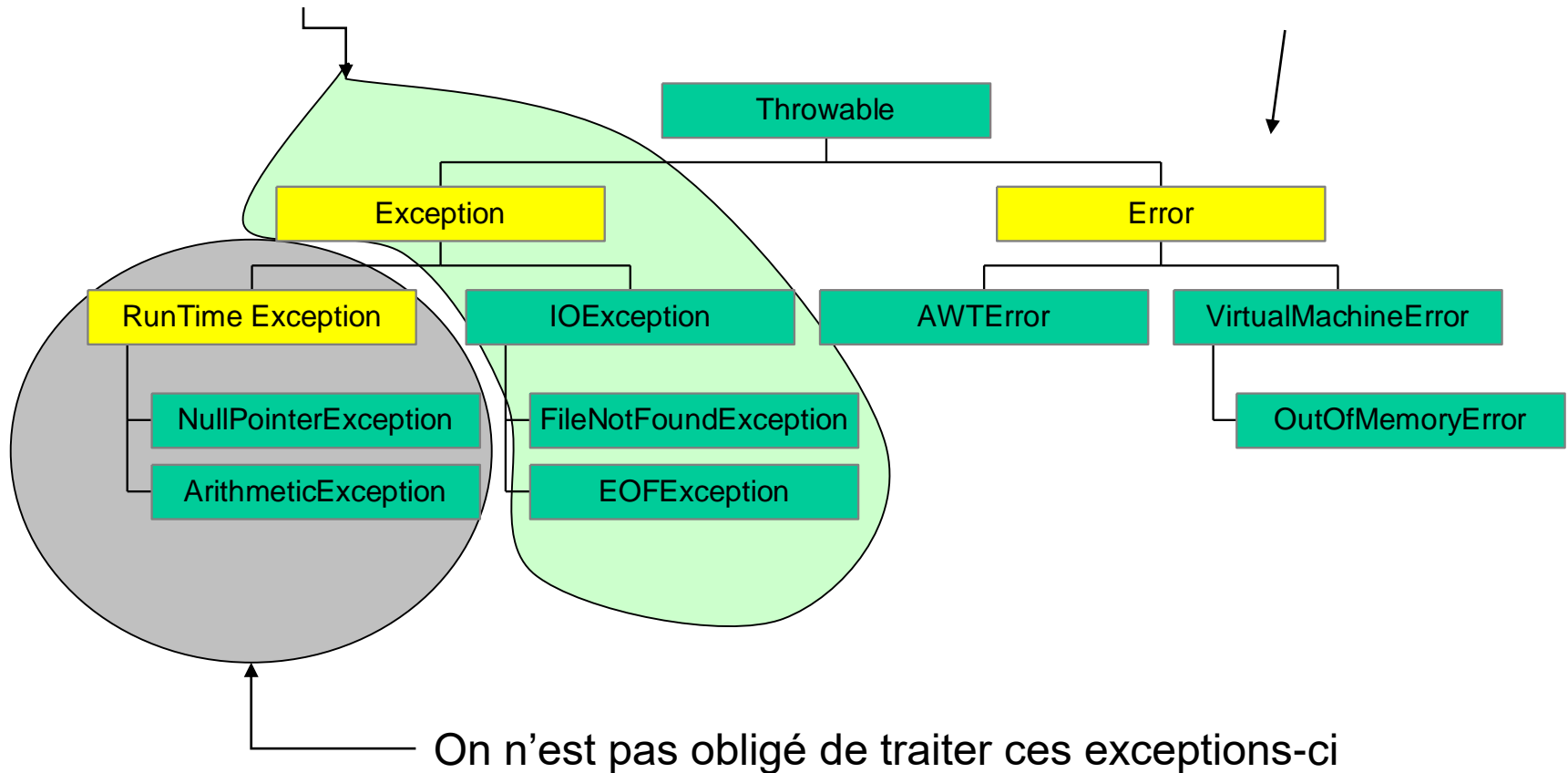
## ○ A quoi ça sert

- Gérer les erreurs est indispensable : Ariane 5
- Mécanisme simple et lisible
  - Regroupement du code de gestion de l'erreur
  - Possibilité de transmettre l'erreur.

# HIÉRARCHIE DES EXCEPTIONS

On doit traiter ces exceptions-ci

On ne peut pas traiter  
les « Error »



# GESTION DES EXCEPTIONS

- Créer sa propre classe d'exception

```
public class MyException extends Exception {  
    public MyException(String errorMessage) {  
        super(errorMessage);  
    }  
}
```

- Déclencher une exception

```
if (number > 10) {  
    throw new MyException("Bad number : " + number);  
}
```

# GESTION DES EXCEPTIONS

## ○ Propager une exception

```
public void test(int number) throws MyException {  
    if (number > 10) {  
        throw new MyException("Bad number : " + number);  
    }  
}
```

## ○ Traiter une exception

```
public void test2() {  
    try {  
        test(15);  
    }  
    catch (MyException e) {  
        e.printStackTrace();  
    }  
    catch (Exception e) {  
        e.printStackTrace();  
    }  
}
```

# GESTION DES EXCEPTIONS

```
public void test2() throws Exception {  
    try {  
        test(5);  
    }  
    catch (MyException e) {  
        e.printStackTrace();  
    }  
    catch (Exception e) {  
        throw e;  
    }  
    finally {  
        // Toujours faire ceci, quelle que soit l'exception  
    }  
}
```

# GESTION DES EXCEPTIONS

- Créer sa propre classe d'exception

```
public class MyException extends Exception {  
    public MyException(String errorMessage) {  
        super(errorMessage);  
    }  
  
    //un 2eme constructeur pour transmettre la StackTrace  
    public MyException(String message, Throwable e){  
        super(message, e);  
    }  
}
```

# EXERCICE

- Créez votre classe `InvalidNumberException`
- Créez une méthode « `generateException` » levant cette exception
- Créez une méthode « `throwException` » appelant `generateException` mais ne traitant pas l'exception.
- Créez une méthode « `catchException` » appelant `throwException`, catchant l'`InvalidNumberException` et retournant une `Exception`.

A decorative graphic on the left side of the slide. It features a series of vertical stripes in various shades of green and blue. Overlaid on these stripes are several circles of different sizes, also in shades of green and blue. One circle is particularly large and prominent.

# ANDROID

## Présentation

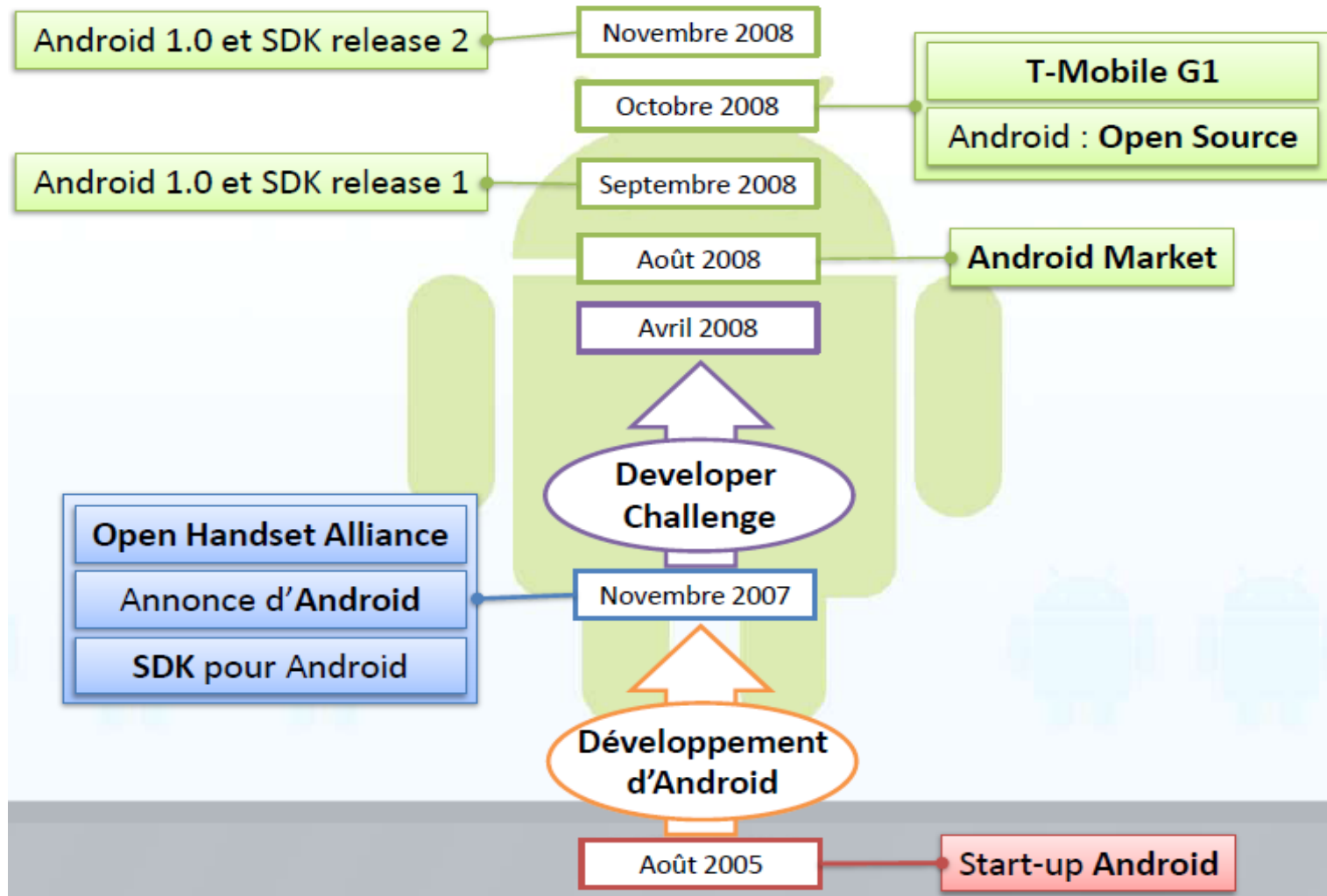
32



# INTRODUCTION

- Android est un système d'exploitation pour téléphone portable de nouvelle génération développé par Google. Celui ci met à disposition un kit de développement (SDK) basé sur le langage Java.
- OS complètement ouvert au développeur:
  - Lancer des appels, sms, emails
  - Accès au hardware (gps, appareil photo, wifi)
  - Accès à toutes les fonctionnalités du téléphone
  - => Applications plus riches

# EVOLUTION



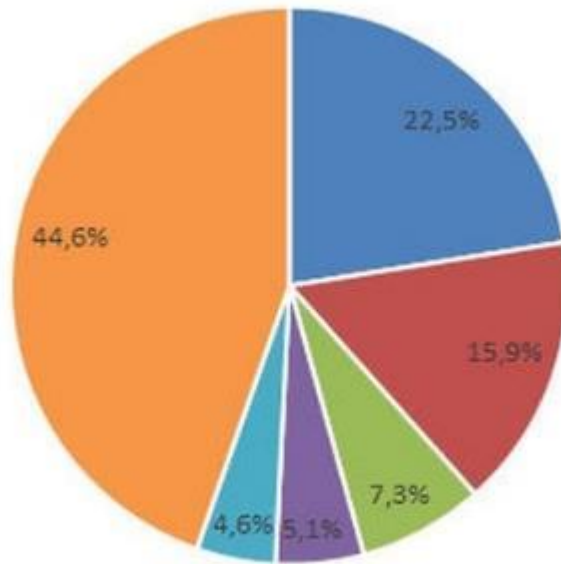
# EVOLUTION



# android 6.0 Marshmallow



# EN CHIFFRE



■ Samsung ■ Apple ■ Huawei ■ Lenovo ■ Xiaomi ■ Autres

- 1,4 milliard de smartphone vendu en 2015
- Supercell (Clash of Clans, Hey Day et Boom Beach)
- 2% des applications génèrent 90% des revenus

# EN CHIFFRE



- Supercell (Clash of Clans, Hay Day et Boom Beach)
  - 515M € de bénéfice en 2015
  - 402M€ en publicités
- 2% des applications génèrent 90% des revenus

# EN CHIFFRE

## Ventes de smartphone en France (janvier 2015)

■ Android ■ iOS ■ Windows ■ RIM ■ Autres

Android; 65,30%



iOS; 20,20%

**Microsoft**  
Windows; 13,00%



RIM; 1,10%

Autres; 0,30%

janv-15

*A propos des statistiques Kantar WorldPanel : Part de marché sur les ventes de smartphone en France. (Panel de consommateur)*

*Infographie : EcoConscient*

# ANDROID C' EST

- un noyau Linux qui lui confère notamment des caractéristiques multitâches
- des bibliothèques graphiques, multimédias
- une machine virtuelle Java adaptée : la Dalvik Virtual Machine
  - ART depuis KitKat (installation--, utilisation++)
- un framework applicatif proposant des fonctionnalités de gestion de fenêtres, de téléphonie, de gestion de contenu...
- des applications dont un navigateur web, une gestion des contacts, un calendrier...



# EN IMAGE...



# DÉVELOPPER POUR ANDROID

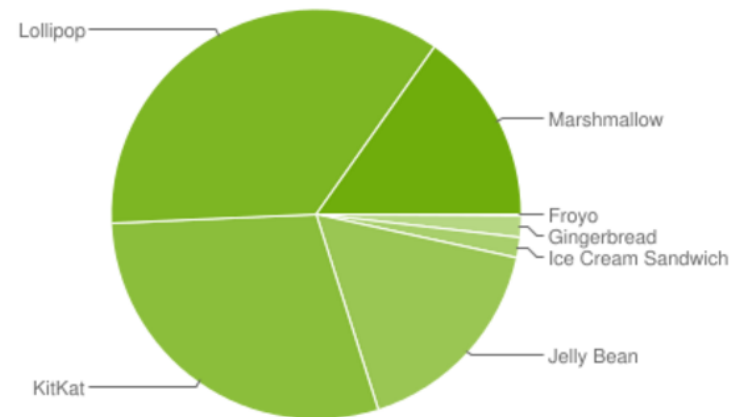
- Gratuit
- Application distribuable rapidement
  - Par mail
  - Par son propre «store»
  - Via Google Play (payant 25\$)
- Programmation en Java
  - Orientée Objet
  - Possibilité de réutiliser du code métier existant
- API Android / Google Service
  - Pas besoin d'écrire du code bas niveau

# VERSION DE L' API ANDROID

- Les API (classes et méthodes) android évoluent avec chaque nouvelle version de l' OS.
- Novembre 2014:
  - Les devices avec une version 4.0 ou supérieure représentent 89,6% du marché.

# RÉPARTITION DES OS PAR VERSION

Version	Codename	API	Distribution
2.2	Froyo	8	0.1%
2.3.3 - 2.3.7	Gingerbread	10	1.7%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	1.6%
4.1.x	Jelly Bean	16	6.0%
4.2.x		17	8.3%
4.3		18	2.4%
4.4	KitKat	19	29.2%
5.0	Lollipop	21	14.1%
5.1		22	21.4%
6.0	Marshmallow	23	15.2%



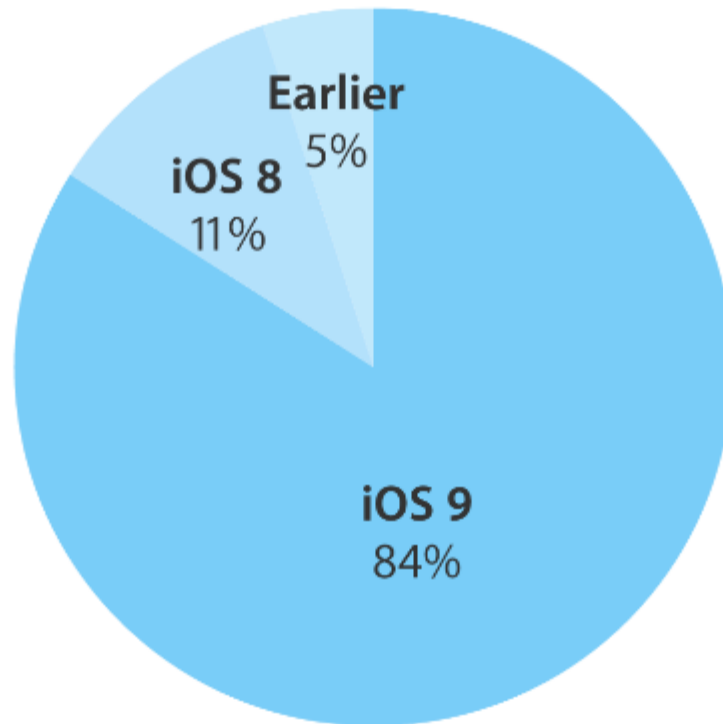
Data collected during a 7-day period ending on August 1, 2016.

Any versions with less than 0.1% distribution are not shown.

Source : <http://developer.android.com/about/dashboards/index.html>

# iOS

84% of devices are using iOS 9.



As measured by the App Store on May 9, 2016.

# BONNES PRATIQUES

- Respecter la charte d'Android.
- Respecter les bonnes pratiques du système.
- Android est différent d'iOS.
- Respecter l'utilisateur
  - Ses données
  - Sa confidentialité
- Respecter ses ressources
  - CPU
  - Batterie
  - Mémoire
- Prévenir l'utilisateur

# BONNES PRATIQUES

- Gestion des erreurs
  - Log
  - Messages techniques et logiques.
  - Donner la possibilité de recommencer
- Gérer les attentes augmente la fluidité
  - Un spinner ou message d'attente
  - Un préaffichage de l'écran
  - Chargement partiel des informations

# ABUS DE POUVOIR

- MÉFIEZ-VOUS DES APPLICATIONS LAMPE DE POCHE
  - [https://play.google.com/store/apps/details?id=goldenshorestechologies.brightestflashlight.free&hl=fr\\_FR](https://play.google.com/store/apps/details?id=goldenshorestechologies.brightestflashlight.free&hl=fr_FR)
  - <http://www.phonandroid.com/mefiez-vous-applications-lampe-poche-vous-etes-espionnes.html>



# ABUS DE POUVOIR

## ○ MÉFIEZ-VOUS DES APPLICATIONS LAMPE DE POCHE



### Privacy Flashlight

SnoopWall

La version 1.1.0 peut accéder aux éléments suivants :



Appareil photo/Micro

- prendre des photos et filmer des vidéos



Autre


- contrôler la lampe de poche

Des fonctionnalités peuvent être automatiquement ajoutées au sein de chaque groupe en cas de mise à jour de l'application "Privacy Flashlight". [En savoir plus](#)



Fermer

# ABUS DE POUVOIR

## ○ MÉFIEZ-VOUS DES APPLICATIONS LAMPE DE POCHE



**Brightest Lampe de Poche**  
GoldenShores Technologies, LLC

- afficher les connexions Wi-Fi
-  **Identifiant de l'appareil et informations relatives aux appels**
  - voir l'état et l'identité du téléphone
-  **Autre**
  - désactiver ou modifier la barre d'état
  - Lire les paramètres et les raccourcis de la page d'accueil
  - contrôler la lampe de poche
  - empêcher la mise en veille de l'appareil
  - afficher les connexions réseau
  - bénéficier d'un accès complet au réseau
  - Installer des raccourcis

Des fonctionnalités peuvent être automatiquement ajoutées au sein de chaque groupe en cas de mise à jour de l'application "Brightest Lampe de Poche". [En savoir plus](#)

Fermer

# ABUS DE POUVOIR

## ○ MÉFIEZ-VOUS DES APPLICATIONS LAMPE DE POCHE

La version 2.4.2 peut accéder aux éléments suivants :

### Données de localisation

- position approximative (réseau)
- position précise (GPS et réseau)

### Photos/Contenus multimédias/Fichiers

- Modifier ou supprimer le contenu de la mémoire de stockage USB
- Tester l'accès à la mémoire de stockage protégée

### Appareil photo/Micro

- prendre des photos et filmer des vidéos

### Informations relatives à la connexion Wi-Fi

- afficher les connexions Wi-Fi

### Identifiant de l'appareil et informations relatives aux appels

- voir l'état et l'identité du téléphone

### Autre

- désactiver ou modifier la barre d'état
- Lire les paramètres et les raccourcis de la page d'accueil
- contrôler la lampe de poche
- empêcher la mise en veille de l'appareil
- afficher les connexions réseau
- bénéficier d'un accès complet au réseau
- Installer des raccourcis
- Désinstaller les raccourcis



# ANDROID

Outils de développement

52

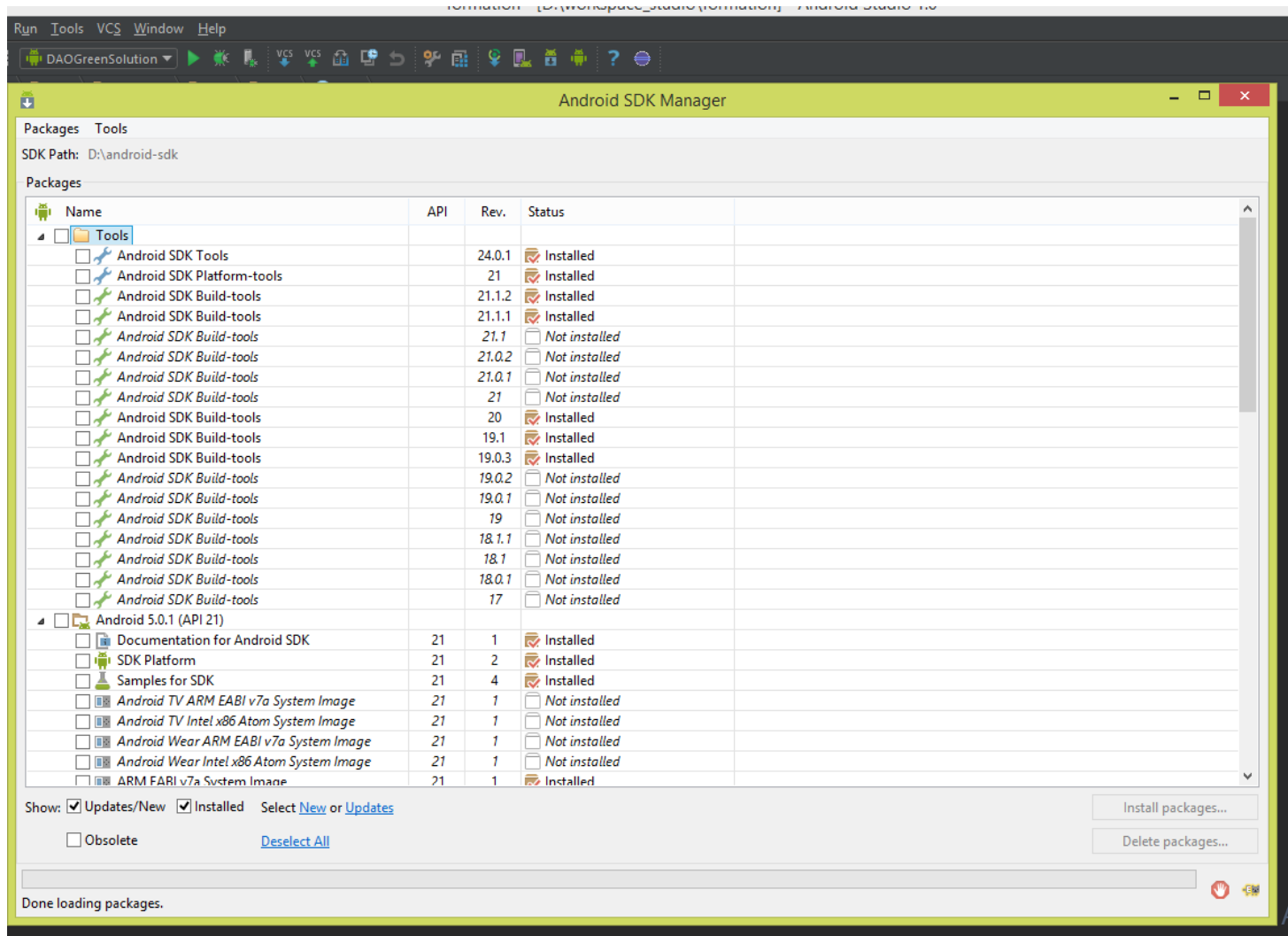
# LE KIT DE DÉVELOPPEMENT

- De quoi avons-nous besoin?
  - Un Environnement de développement
    - Android Studio (basé sur IntelliJ)
    - Eclipse (avec plugin Android)
  - Le framework Android
    - Téléchargeable via le SDK Manager
  - Un Device
    - Réel de test (préférable)
    - Simulateur créé depuis le AVD Manager ou depuis Genymotion

# ANDROID STUDIO

- IDE développé par Google (2013)
- Basé sur IntelliJ Community Edition
- Multiplateforme (MacOS, Windows, Linux)
- Installe automatiquement le SDK Manager et le AVD Manager
- NECESSITE d' avoir le JDK installé !!!
- <http://developer.android.com/sdk/installing/studio.html>

# SDK ANDROID



# RÉGLAGES ANDROID STUDIO

## ○ Importer mes Settings

- File -> import settings -> settingAndroidStudio.jar

## ○ Contenu

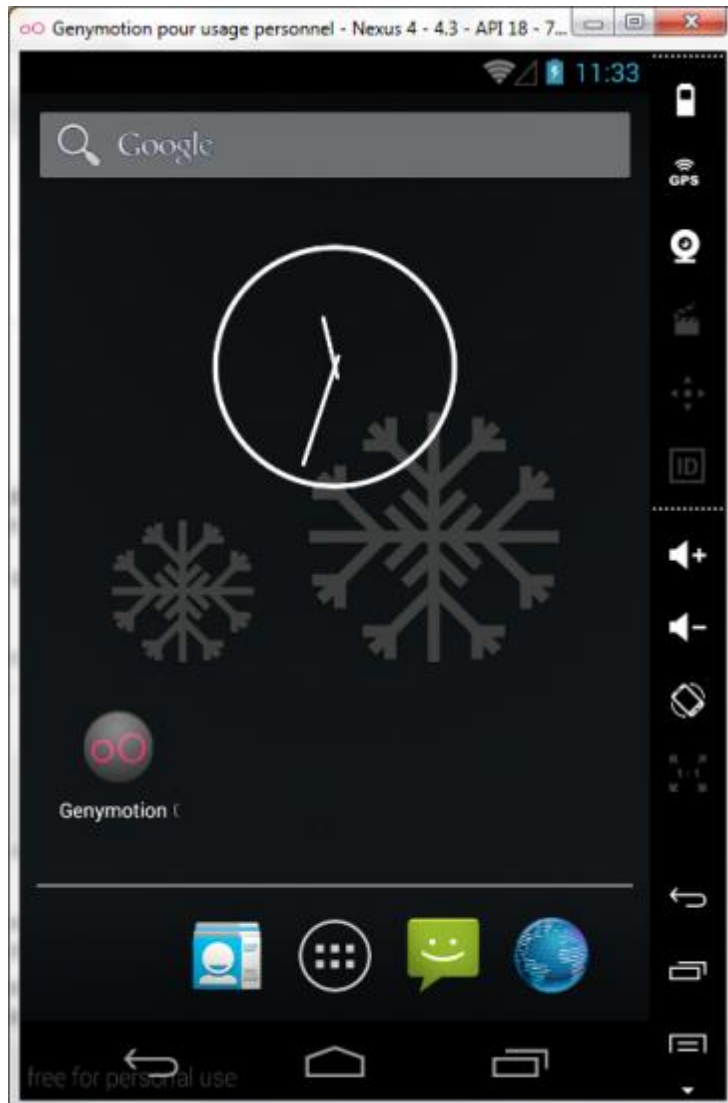
- Raccourci clavier Eclipse
- Save Action (Import Automatique, Indent code).



# RÉGLAGES ANDROID STUDIO

- Quelques raccourcis clavier
  - Ctrl + espace : proposition de code
  - Alt + entrée : Proposition de solution
  - Ctrl + d : Supprimer ligne
  - Ctrl + shift + d : Dupliquer ligne
  - Alt + a : Revenir en arrière
  - Ctrl + g : Rechercher dans tous le projet
  - Ctrl + shift + o : Rechercher fichier dans le projet

# GENYMOTION



- L'émulateur plus rapide que le device !!!
- Les Google services ne sont plus installés.
  - [http://wiki.cyanogenmod.org/w/Google\\_Apps#Downloads](http://wiki.cyanogenmod.org/w/Google_Apps#Downloads)
- La compatibilité ARM non plus.
  - <http://filetrip.net/dl?4SUOrdcmRv>

# DALVIK DEBUG MONITOR SERVER (DDMS)

- Fonctionnalité de l'IDE que l'on ouvre via une perspective
- Liste les devices avec la possibilité d'avoir des informations sur les processus créés (thread, mémoire), fichiers systèmes
- Possibilité d'interagir avec l'émulateur, en simulant des appels ou un envoi de sms, changer de position de GPS
- Contient aussi la journalisation de toutes les activités de l'émulateur : le logCat.

# TESTER SON APPLICATION

- Impossible sur chaque type de téléphone.
- Minimum :
  - Device réel
  - 3 tailles d'écran (petite, normale (nexus 4), tablette).
  - 3 densités (mdpi, hdpi, xhdpi).
  - Avec réseau faible
  - Sur un Samsung
  - Portrait / paysage

# UNE FOIS SUR LE PLAYSTORE

- Librairie d'Analytics d'application
  - Google Analytics
  - Capptain
  - CrashLitycs
  - Accra
- Serveur Push
  - java-apns
  - Capptain
- Ils existent des centaines de librairies :
  - <http://android-arsenal.com/free>

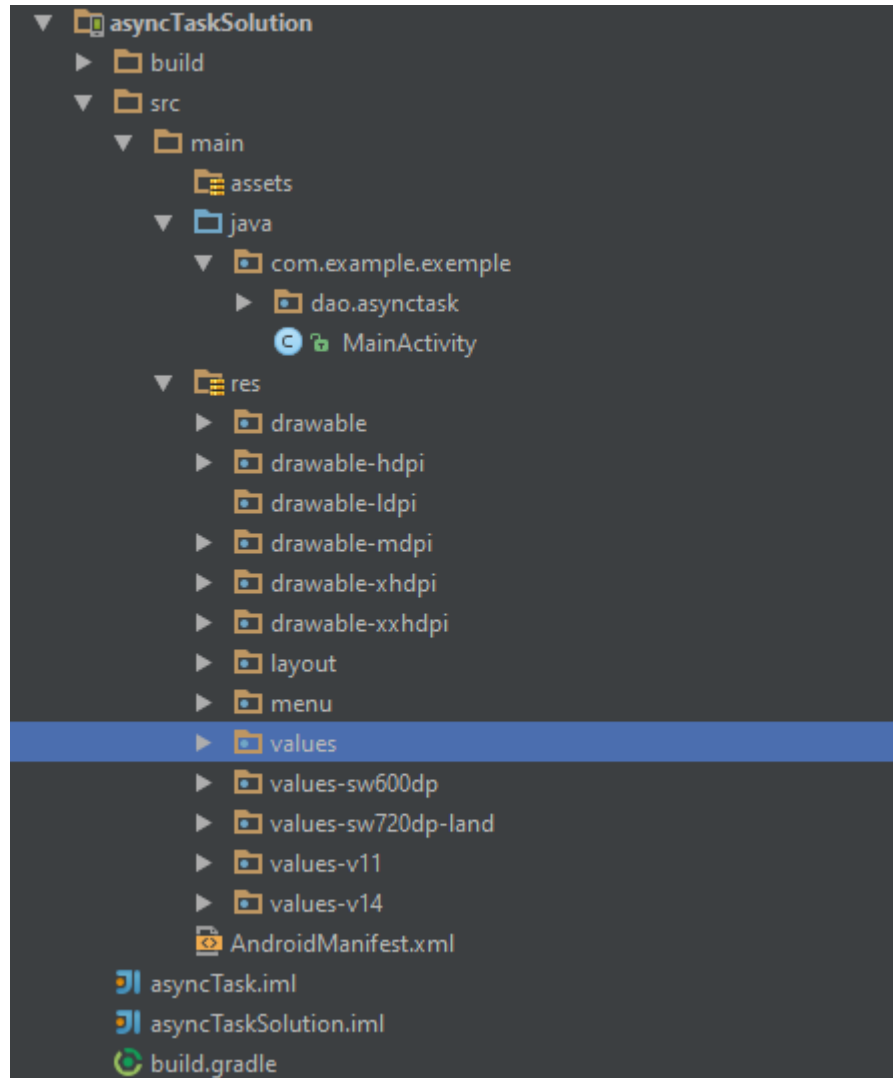


# ANDROID

## Architecture d' une application

62

# QUE CONTIENT MON PROJET ?



# LE RÉPERTOIRE GEN

- La classe R (générée)

```
public final class R{  
    public static final class drawable {  
        public static final int icon=0x7f020000;  
    }  
    public static final class layout {  
        public static final int main=0x7f030000;  
    }  
}
```



# LE RÉPERTOIRE JAVA

- Contient votre code source
  - Classes java
  - Différents packages

# LE RÉPERTOIRE RES

- Contient les ressources externes de l'application
  - **values** : Valeurs simples (chaines, couleurs, dimensions)
  - **drawable** : ressources images (conseil : utiliser le png)
  - **layout** : fenêtre correspondant à l'interface graphique; entièrement paramétrable en XML
  - **anim** : associées aux composants graphiques (translation, rotation)
- Possibilité de fournir des ressources pour différentes langues et tailles d'écran

# LE RÉPERTOIRE RES






















```
//strings.xml
<string name="today">Aujourd'hui</string>
<string name="annonce">Le petit %1$s est attend à l'accueil par %2$s</string>

//dimens.xml
<dimen name="standard_height">48dp</dimen>

// Retrouver le texte
Resources resources = context.getResources();
String today= resources.getString(R.string.today);
String annonce = resources.getString(R.string.annonce, "Timmy", "sa maman");

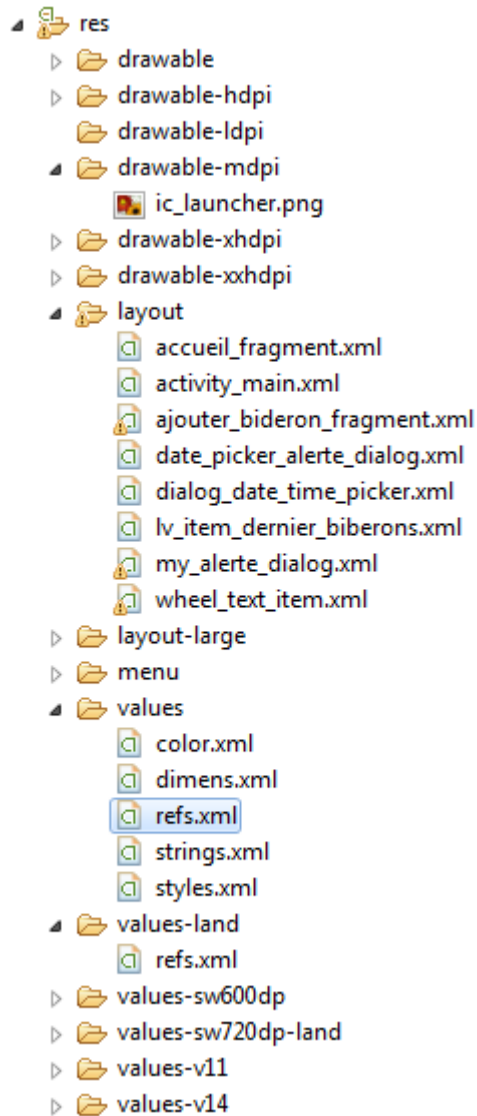
//Les chiffres
int standard_height = getResources().getDimensionPixelSize(R.dimen.standard_height);
```

# LE RÉPERTOIRE RES

Available Qualifiers	
 125	Country Code
	Network Code
	Language
 US	Region
	Bidi Layout Direction
	Smallest Screen Width
	Screen Width
	Screen Height
	Size
	Ratio
	Orientation
	UI Mode
	Night Mode
	Density
	Touch Screen
	Keyboard
	Text Input
	Navigation State
	Navigation Method
	Dimension
	Version

- On ne peut pas tout prendre tous en charge.
- Recommandation :
  - La langue
  - ~~3 tailles d'écrans (medium, large, xLarge) sw600dp~~
  - Orientation
  - Density (mdpi, ldpi, hdpi, xhdpi, )
  - Version Android

# LE RÉPERTOIRE RES



- Création de sous répertoire impossible.

# LE RÉPERTOIRE RES

- Les valeurs de bases sont dans les répertoires /res/values et peuvent contenir.
  - String : Vous pouvez utiliser les HTML tags <b>, <i> and <u>.
    - Tags compatibles : <http://daniel-codes.blogspot.fr/2011/04/html-in-textviews.html>
  - Colors : #RGB, #ARGB, #RRGGBB and #AARRGGBB
  - Dimensions: In pixels (px), inches (in), millimeters (mm), points (pt) , density-independent pixel (dp) or scale-independent pixel (sp)

# ANDROIDMANIFEST.XML

- Fichier permettant de configurer votre application.
- Plusieurs sortes de configuration:
  - Informations générales (version, packages)
  - Informations concernant l'application : activity, attributs de l'application
  - Permission : pour autoriser l'application à avoir accès à certaines ressources (géolocalisation, internet)
  - Instrumentation : correspond aux classes de test associées.

# ANDROIDMANIFEST.XML

```
<?xml version="1.0" encoding="utf-8"?>
<manifest package="com.httpexemple"
    xmlns:android="http://schemas.android.com/apk/res/android">

    <uses-permission android:name="android.permission.INTERNET"/>

    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme">
        <activity
            android:name=".MainActivity"
            >
            <intent-filter>
                <action android:name="android.intent.action.MAIN"/>

                <category android:name="android.intent.category.LAUNCHER"/>
            </intent-filter>
        </activity>
    </application>
```



# GRADLE

- Equivalent de maven et Ant

```
android {  
    compileSdkVersion 20  
    buildToolsVersion "20.0.0"  
  
    defaultConfig {  
        applicationId "com.facebooklogin"  
        minSdkVersion 15  
        targetSdkVersion 19  
        versionCode 1  
        versionName "1.0"  
    }  
}  
  
dependencies {  
    compile fileTree(include: ['*.jar'], dir: 'libs')  
    compile project(':Simple Facebook')  
    compile files('libs/commons-lang3-3.2.1.jar')  
}
```

# TP

- Créez un nouveau projet
- Le lancer dans un émulateur
- Mettre le HelloWorld dans strings.xml
- Changer la langue du helloworld en fonction de la langue du téléphone.



**IHM**



**75**



# ANDROID

- Construire une interface graphique
- Pas de traitement lourd sur l'UIThread. (Service, AsyncTask...)
- Les modifications d'IHM uniquement sur l'UIThread

```
//On peut afficher les info
runOnUiThread(new Runnable() {

    @Override
    public void run() {
        //traitement IHM
        tv_prenom.setText("Bob");
    }
});
```

# CONSTRUIRE SON INTERFACE GRAPHIQUE

- 2 possibilités
  - En code
  - En XML
- Privilégier autant que possible le XML
  - Séparation comportement et visuel (MVC)
  - Outil de rendu graphique en XML
- Utilisation du code pour des changements dynamiques

# LAYOUTS

- Les layouts sont des composants permettant de positionner les différents composants graphiques.
- Ce sont des conteneurs d'éléments visuels (ils peuvent contenir d'autre vues et même d'autres layouts) représentés sous forme de fichier xml ou créés directement dans le code.
- Plusieurs types de layouts :
  - Linear
  - Relatif
  - Table
  - ...

# LES LAYOUTS ET LE XML

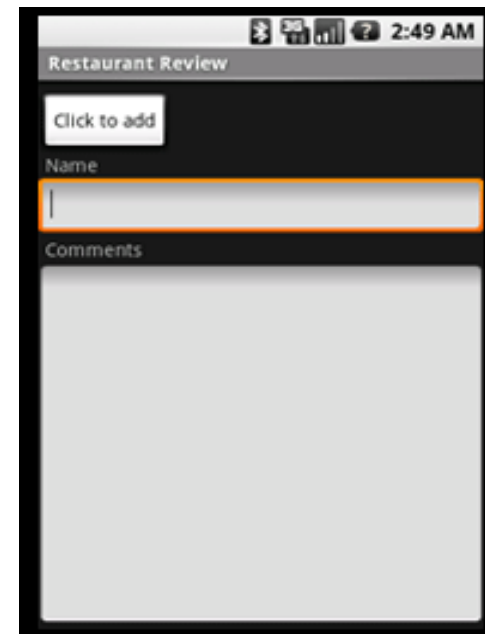
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    // Attributs du layout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

    <!-- Contenu du layout -->

</LinearLayout>
```

# LINEAR LAYOUT

- Layout le plus utilisé
- Container permettant de placer les éléments en ligne
- Constructeur :
  - `LinearLayout(context,[object arg])`
- Méthodes :
  - `setOrientation(LinearLayout.VERTICAL)` : changer l'orientation des lignes du container
  - `setGravity(Gravity.RIGHT)` : place les éléments selon un côté.
  - `setPadding(left,top,right,bottom)` : marge des composants
  - `addView(View)` : ajouter des composants graphiques
  - `setBackgroundDrawable(BitmapDrawable)`: definit un arrière plan



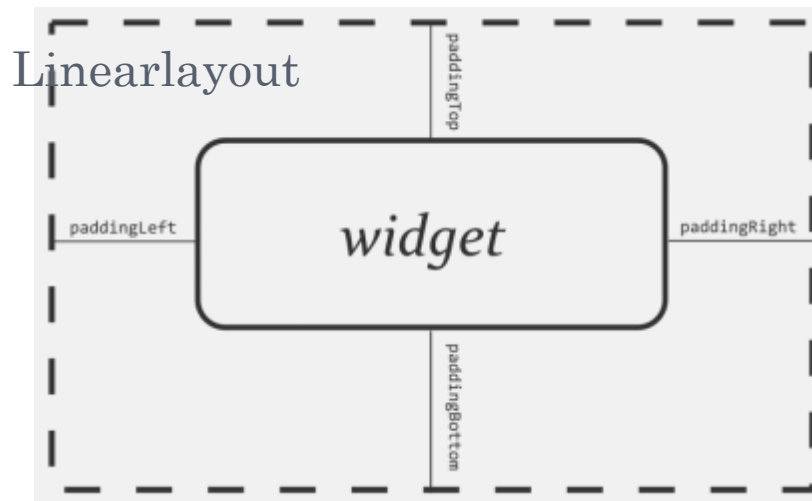


# ATTRIBUTS DU LINEARLAYOUT

- Orientation : indique si le layout ajoute les composants par ligne ou par colonne
  - android:orientation
- Width et height : 2 propriétés
  - "wrap-content" et "fill\_parent" (match\_parent)
- Gravity : par défaut, l'alignement se fait de haut en bas
- Weight : définit l'espace que prendra la vue associée

# LINEAR LAYOUT

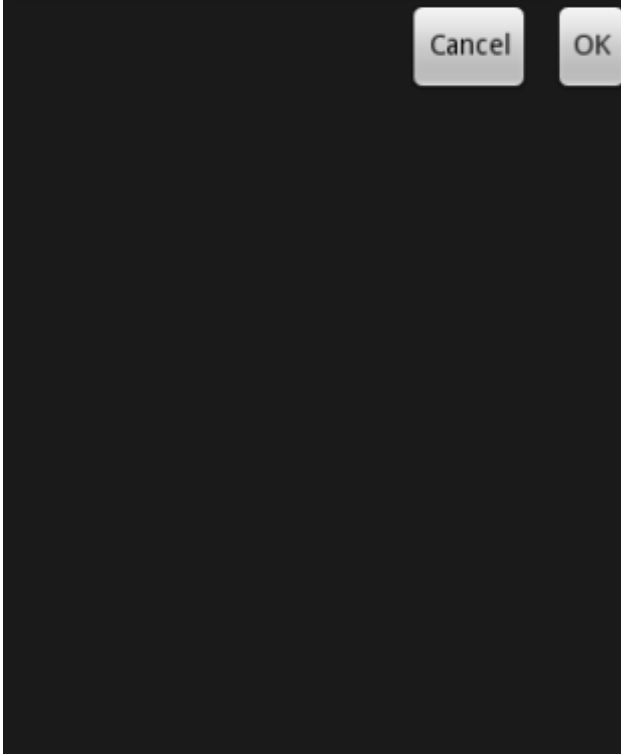
- Padding : spécifie l'espace entre le composant et son wrapper



# Exemple

Type here:

Cancel OK



# TEXTVIEW

- Champ texte non modifiable par l'utilisateur
- En code :
  - new TextView(Context context)
  - utilisation de setText() et getText() pour gérer le contenu
- Code Html accepté
  - <http://daniel-codes.blogspot.fr/2011/04/html-in-textviews.html>

```
<TextView  
    android:layout_width="fill_parent"  
    android:layout_height="wrap_content"/>
```

# EDITTEXT

- Champ texte modifiable par l'utilisateur
- En code :
  - new EditText(Context context)
  - utilisation de setText() et getText() pour gérer le contenu
  - setHint() pour afficher un texte grisé quand il n'y a pas de contenu

```
<EditText  
    android:layout_width="fill_parent"  
    android:layout_height="wrap_content"/>
```

# BUTTON

- Un simple bouton à appuyer.
  - setText(var) : ajouter un texte.

```
<Button  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"/>
```

# Checkbox

- Sélectionne des informations

☒ New CheckBox

```
<CheckBox android:id="@+id/checkbox"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:text="check it out" />
```

# RADIOBUTTON

## ○ Boutons à choix exclusifs

● New RadioButton

```
<RadioGroup
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical">

    <RadioButton android:id="@+id/radio_red"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Red" />

    <RadioButton android:id="@+id/radio_blue"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Blue" />

</RadioGroup>
```



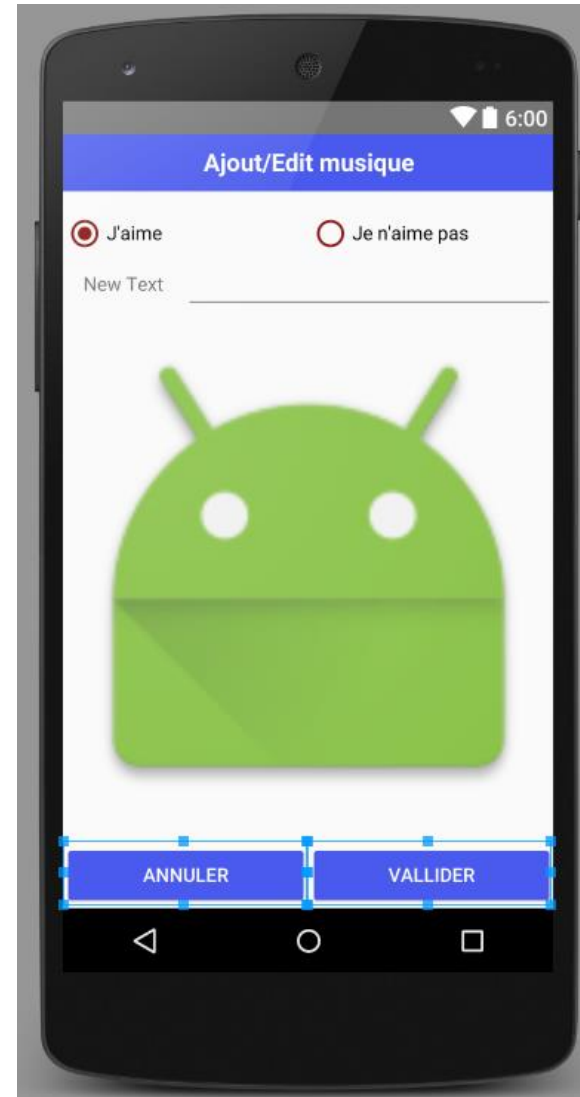
# ImageView

- Affiche une image simple
  - [setImageResource\(int\)](#)

```
<ImageView  
    android:layout_width="wrap_content"  
    android:layout_height="match_parent"  
    android:src="@drawable/icon"/>
```

# TP : construction d'une vue

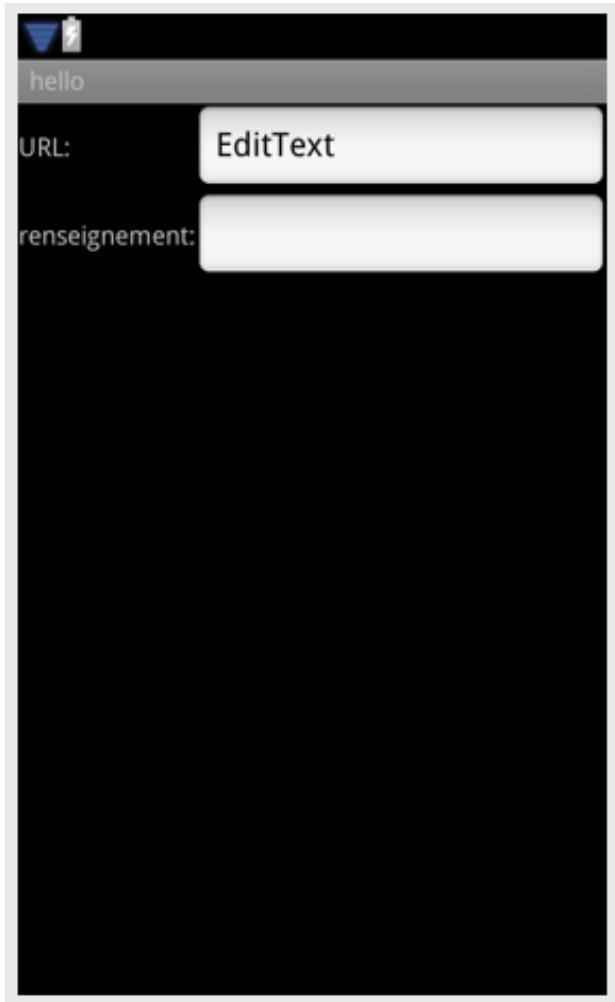
Réaliser et afficher le layout suivant



# TABLELAYOUT

- Le fonctionnement du `tableLayout` ressemble beaucoup à celle des tables en HTML
- Les éléments enfants sont des `tableRow` où on ajoute nos composants
- On les utilise pour aligner les formulaires

# Exemple



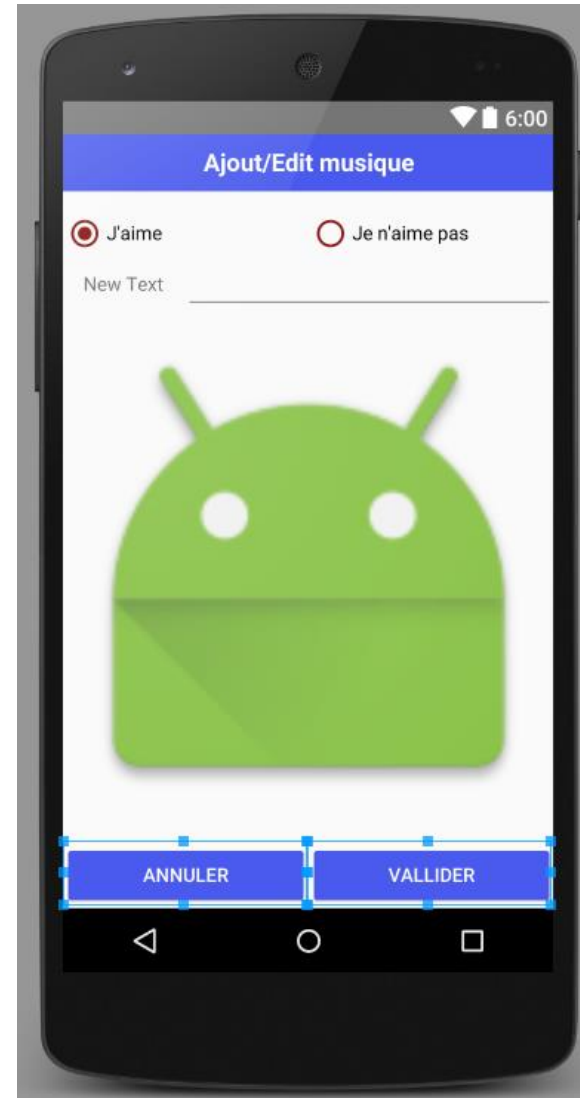
```
<TableLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:stretchColumns="1"
android:layout_height="fill_parent"
android:layout_width="fill_parent">
<TableRow android:id="@+id/ligne1">
  <TextView
    android:text="URL:"
    android:id="@+id/textView1"></TextView>
  <EditText android:layout_height="wrap_content"
    android:text="EditText"
    android:layout_width="wrap_content"
    android:id="@+id/EditText01"></EditText>
</TableRow>
<TableRow android:id="@+id/ligne2">
  <TextView android:text="renseignement:"/>
  <EditText android:id="@+id/entry"
    android:layout_height="wrap_content"
    android:layout_width="wrap_content"/>
</TableRow>
</TableLayout>
```

# RELATIVE LAYOUT

- Positionner les composants de façon relative entre eux (leurs positions dépendent d'eux-mêmes)
- Position relative à un container
  - android:layout alignParentTop: le composant est aligné par rapport au début du container
  - android:layout alignParentBottom: le composant est aligné par rapport à la fin du container
  - 
  - android:layout alignParentLeft: le composant est aligné par rapport à la partie gauche du container
  - 
  - android:layout alignParentRight: le composant est aligné par rapport à la partie droite du container

# TP : construction d'une vue

Réaliser et afficher le layout suivant à l'aide d'un RelativeLayout



# Du XML au code

- Comment

- Mettre du texte dans le label?
- Etre informé du click sur un bouton?

- Solution:

- Récupérer les instances qui nous intéressent
- Appeler les méthodes des composants
- Ajouter des « écouteurs d'événements »

# Du XML au code

## ○ Dans le XML

- On ajoute un identifiant au composant

```
<Button  
    android:id="@+id/am_bt_next"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"/>
```

## ○ Dans le code de l'activity

- On récupère le composant via son identifiant

```
Button bt_next= (Button)findViewById(R.id.am_bt_next) ;
```



# Du XML au code

```
public class MainActivity extends AppCompatActivity {  
  
    //Déclaration d'un pointeur vers un Button  
    private Button bt_valider;  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        //Chargement de l'interface graphique  
        setContentView(R.layout.activity_main);  
  
        //On récupère le Button qui a été créé par le fichier XML  
        bt_valider = (Button) findViewById(R.id.bt_valider);  
    }  
}
```

# GESTION DES ÉVÉNEMENTS

- Un écouteur d'événements est appelé en java un « Listener ».
- Certains composants peuvent réagir à des événements
- Ces composants proposent une interface pour écouter leurs événements
  - `public void setOnXListner(OnXListener listener)`

```
Button myButton = ...;
// Méthode 1 ; Objet anonyme
myButton.setOnClickListener(new View.OnClickListener() {
    public void onClick(View view){
        // Exécuter quand on clique sur myButton
    }
});
// Méthode 2 : Que l'activité implémente l'interface :
myButton.setOnClickListener(this);
```

# QUELQUES APIs

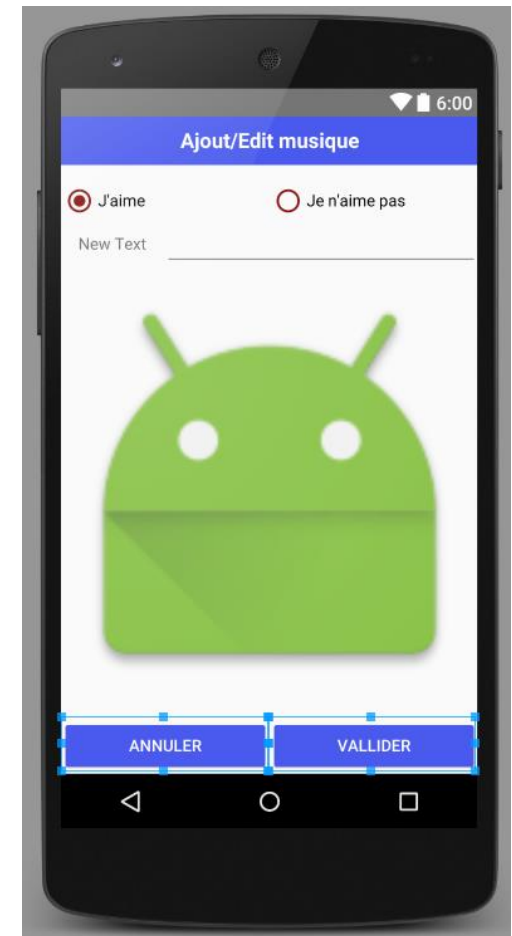
- Button (clic)
  - `setOnClickListener(...)`
  - `setOnLongClickListener(...)`
- EditText (Chaque touche appuyée)
  - `setKeyListener(...)`
- CheckBox / RadioButton (Coché ou décoché)
  - `setOnClickListener(...)`
  - `setOnCheckedChangeListener(...)`

# XML TO CODE

- Un site permettant de générer le code à partir du xml
  - <https://www.buzzingandroid.com/tools/android-layout-finder/>

# TP : construction d'une vue

- **Etape 1 : Gérer les évènements**
  - Un clic sur **Valider**, affiche le radioButton sélectionné.
  - Un clic sur **Annuler**, efface l'editText et désélectionne les radioButton.
- **Etape 2 : Gestion d'images**
  - Valider : affiche l'image 'done'
  - Annuler affiche l'image 'delete forever'
  - Base d'icônes Material Design :  
<https://design.google.com/icons/>
  - Prendre sur fond noir, taille 48dp, Png
- **Etape 3 : changer la couleur de l'image dynamiquement grâce à la méthode ColorFilter**
  - `imageView.setColorFilter(Color.CYAN);`



# ANDROID - LES COMPOSANTS

- Principaux composants d'une application Android:
  - Les activités (activities)
  - Les services (services)
  - Les broadcast receivers

# LES COMPOSANTS - LES SERVICES

- Effectuent une tâche en arrière-plan
  - Téléchargement de données, jouer de la musique
- Ne présentent pas d'interface graphique
  - Se contrôlent via le code
- Emettent une / des notification(s) quand elles ont des informations à communiquer
  - «Téléchargement terminé»
- Tournent sur le processus de l'application

# LES COMPOSANTS – BROADCAST RECEIVER

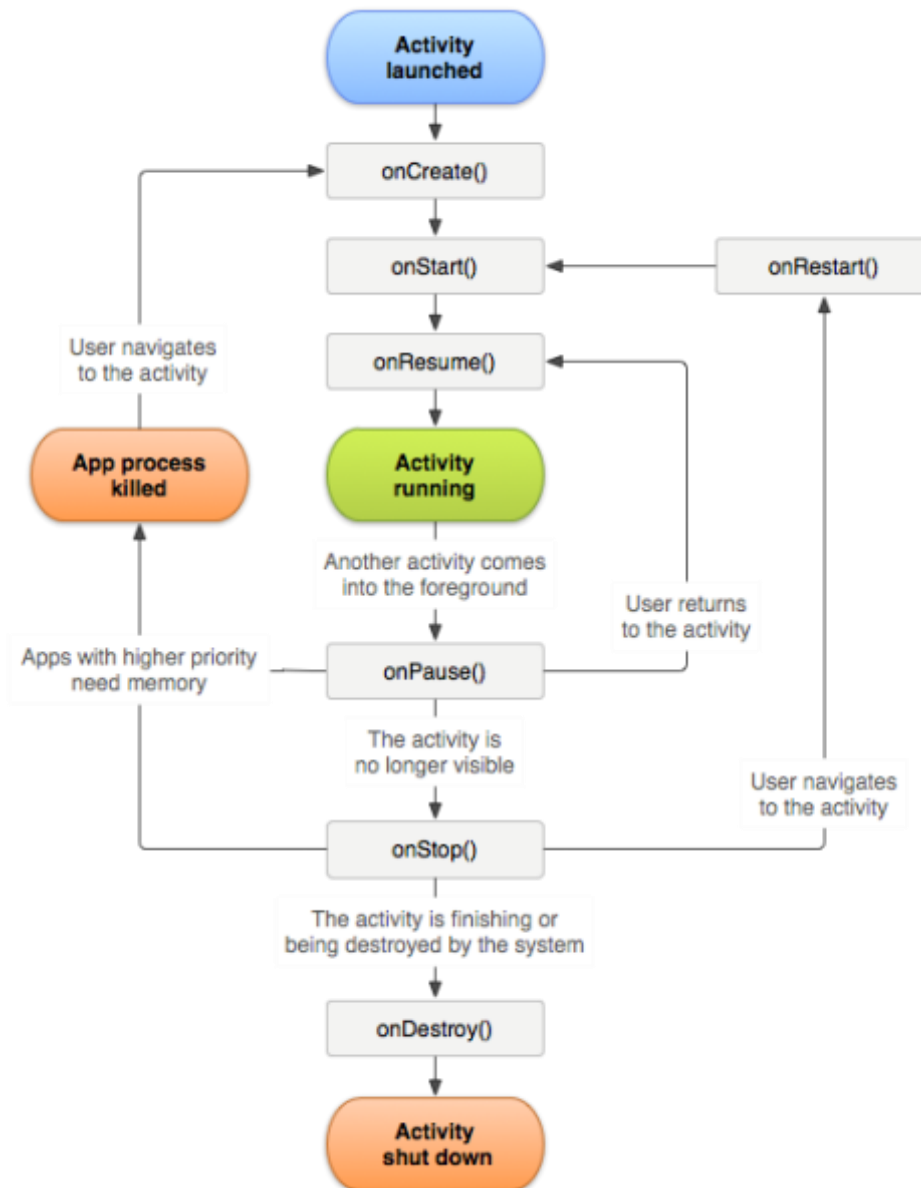
- «Ecoutent» une notification donnée
  - Notification Système
    - batterie faible, sms reçu
  - Notification Interne
    - notification émise par un service
  - Notification Google
    - émise par le Google Cloud Messaging
- Exécute un code à la réception de cette notification
  - Affichage d'une notification à l'écran, code métier, arrêt d'un service...
- Possibilité d'avoir plusieurs BR par application



# LES ACTIVITÉS

- Composant de base d'une application
- Représente un écran (une tâche) de l'application
- **DOIT** être déclarée dans le fichier manifest
- Programmation événementielle
  - Pas de main()
  - Répondre à des événements
    - Activité démarrée, click sur le bouton menu...
  - possède un cycle de vie géré par le systeme
    - reçoit des événements à différents moments de sa vie

# LES ACTIVITÉS



## ○ 4 états :

- **Active** : Au 1<sup>er</sup> plan
- **En pause** : Visible mais elle n'a plus la main, une notification ou une autre activité est active.
- **Stoppée** : Existe, mais n'est plus visible. L'activité ne peut interagir avec l'utilisateur que par notification.
- **Morte** : L'activité n'est pas lancée.

# CLASSE ACTIVITY

```
public class MainActivity extends Activity {  
    private TextView tv;  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
        tv = (TextView) findViewById(R.id.tv);  
    }  
  
    @Override  
    protected void onResume() {  
        super.onResume();  
    }  
  
    @Override  
    protected void onPause() {  
        super.onPause();  
    }  
  
    @Override  
    protected void onDestroy() {  
        super.onDestroy();  
    }  
}
```

# CYCLE DE VIE

- **onCreate** initialise l'activité (création d'un objet à partir du layout xml, affectation des variables, chargement des données sur les composants)
- **onDestroy** est appelée à la destruction et se charge de fermer toutes les connexions
- **onStop** aura la charge de stopper les threads, animations, et plus généralement les process qui vont agir sur l'interface
- Utilisation de **onStart** pour lancer/relancer ces process
- **onPause/Resume** doit être léger de façon à ne pas ralentir la machine lors du redémarrage (on peut s'en servir pour s'abonner à des broadcasts)

# LANCER UNE ACTIVITY

```
Intent intent = new Intent(this, SecondActivity.class);  
startActivity(intent);  
// Tuer l'activité courante  
finish()
```

# AFFICHER DES MESSAGES EN CONSOLE

- Plusieurs niveaux de
  - Verbose
  - Debug
  - Information
  - Warning
  - Error
- Utilisation des méthodes statiques de la classe Log( respectivement les fonctions v(),d(),i(),w(), et e())
- Chaque message est associée a un tag facilitant le filtre des messages dans la console

# EXEMPLE

- Bonne Pratique :
  - Définir dans le fichier de constante les tags pour être sûr d'avoir toujours le même dans la console.
  - Définir une variable permettant de savoir si on est en prod.

```
protected void onResume() {  
    //Log.e("tag", "text");  
    if (DEV_MODE) {  
        Log.v("MON_TAG", "activity: main_activity ");  
    }  
}
```

# TP

- Ajouter une seconde Activity
- Créer un bouton sur la 1<sup>er</sup> qui redirige sur la seconde
- Surchargez chacun des événements (OnCreate, onResume...) correspondant au changement d'état dans le cycle de vie de l'activity et y mettre des logs.
- «Jouer» avec le simulateur pour comprendre à quel moment ces messages sont reçus
- Même chose en ajoutant un `SystemClock.sleep` dans chacun des états.
- Ajouter un point d'arrêt pour utiliser le mode debug





# MENUS ET BOÎTES DE DIALOGUE

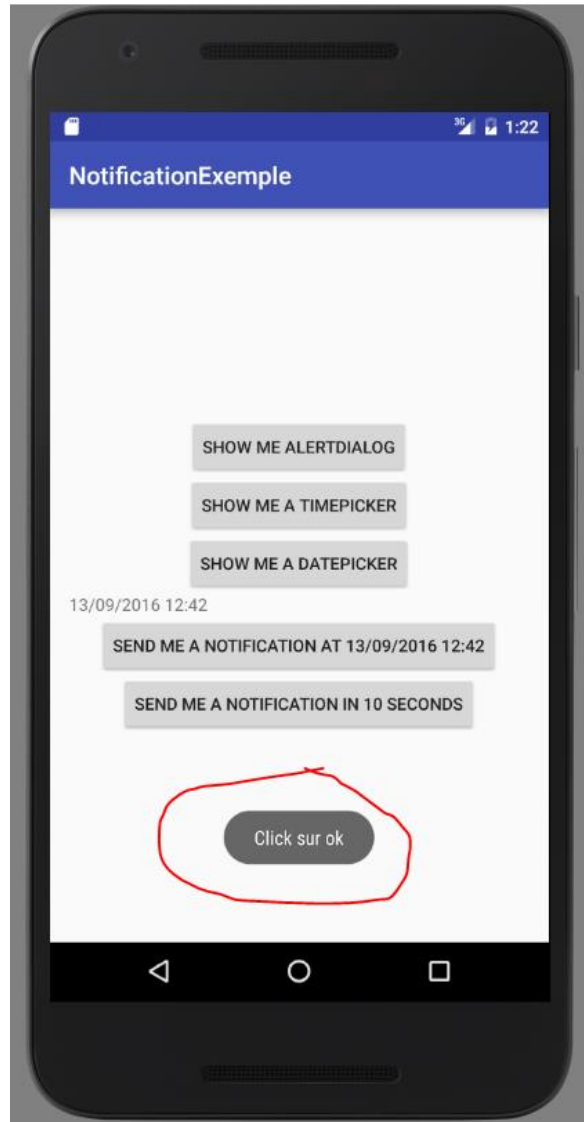
113

# TOAST

- Moyen simple et rapide pour afficher une information à l'utilisateur
- Affichage limité à quelques secondes
- Disparition automatique

```
Toast.makeText(context, "Ceci est un Toast", Toast.LENGTH_LONG).show();
```

# TOAST



# LES BOÎTES DE DIALOGUES

- Les boîtes de dialogue sont des éléments visuels flottants
- Il existe plusieurs sortes de dialogue :
  - alertDialog
  - DatePicker
  - TimePicker..
- Héritent de la classe “Dialog”
- Permet à l'utilisateur de faire des actions rapides comme répondre à des questions, voir les messages...

# TIMEPICKER



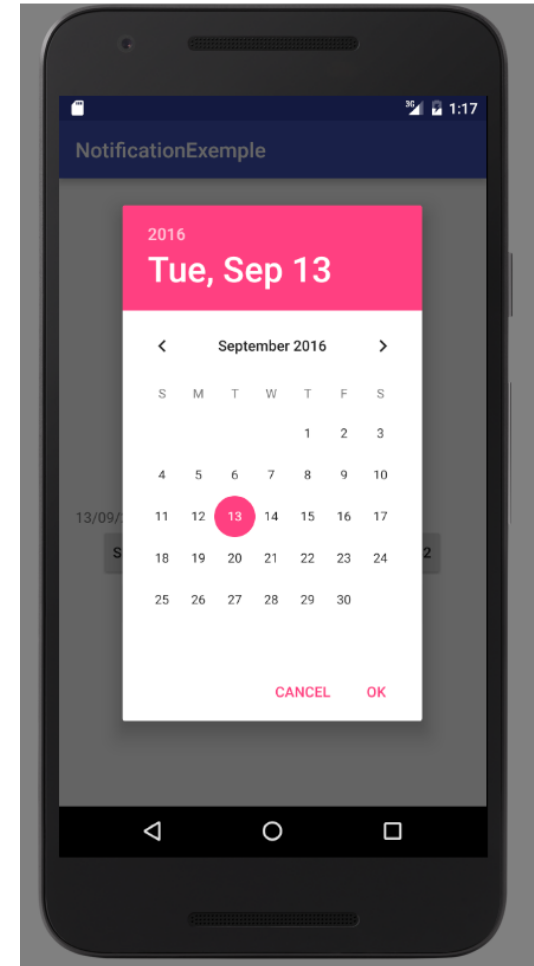
```
//(Context, callback, heure, minute, 24h format)
TimePickerDialog timePickerDialog = new TimePickerDialog(this, this, 14, 33, true);
timePickerDialog.show();
```

# DATEPICKER

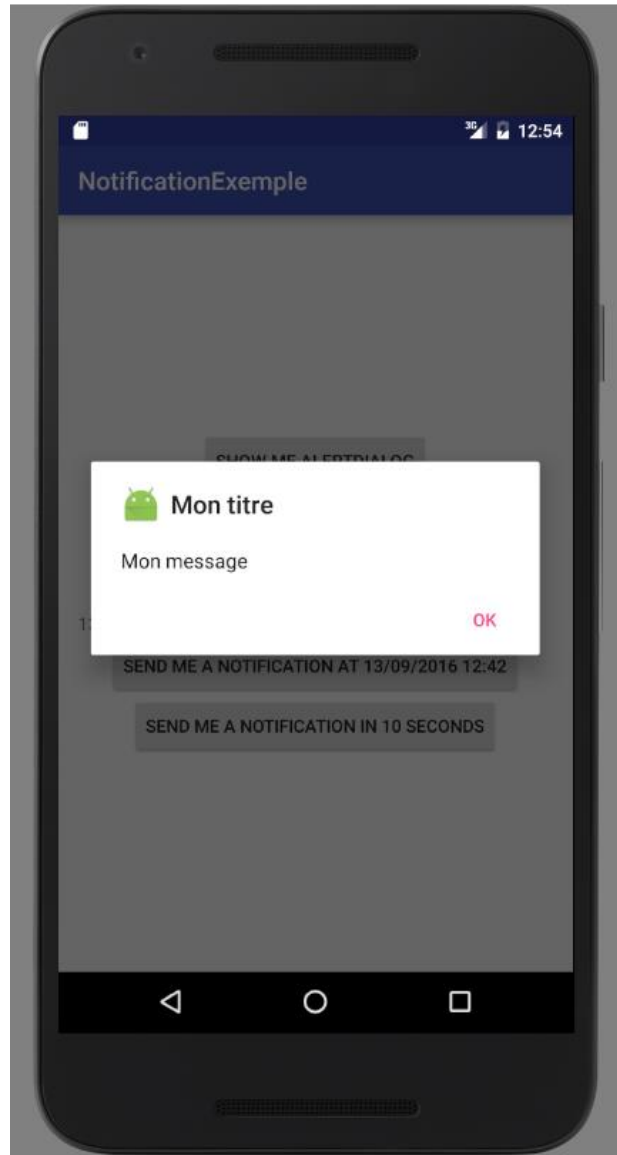
```
//Gestion de la date
Calendar calendar = Calendar.getInstance();

//Création de la fenêtre
DatePickerDialog datePickerDialog = new DatePickerDialog(this, this,
    calendar.get(Calendar.YEAR), calendar.get(Calendar.MONTH),
    calendar.get(Calendar.DAY_OF_MONTH));

//Afficher la fenêtre
datePickerDialog.show();
```



# ALERTDIALOG

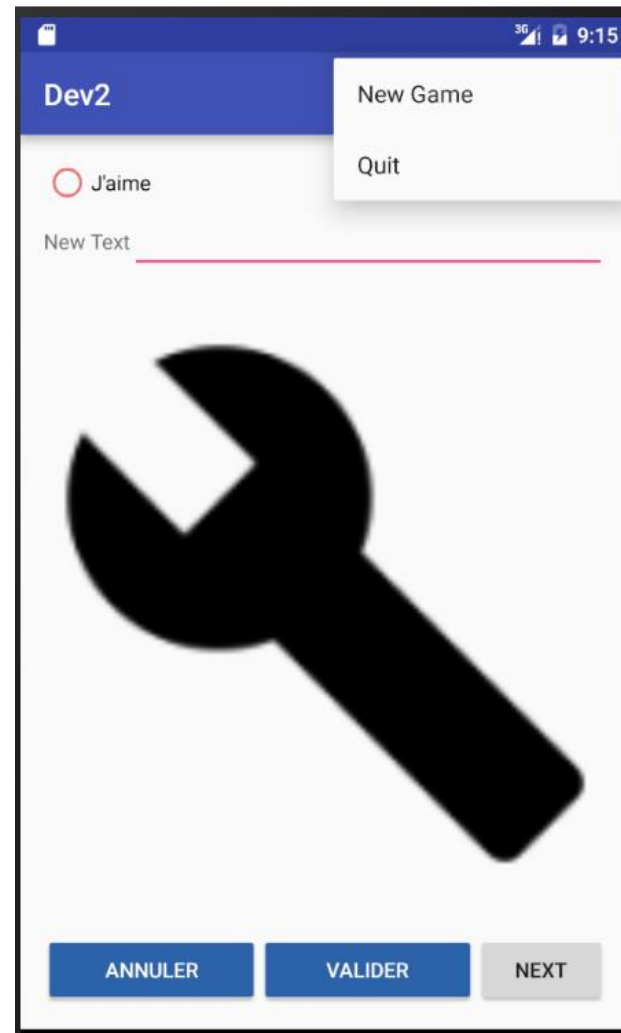
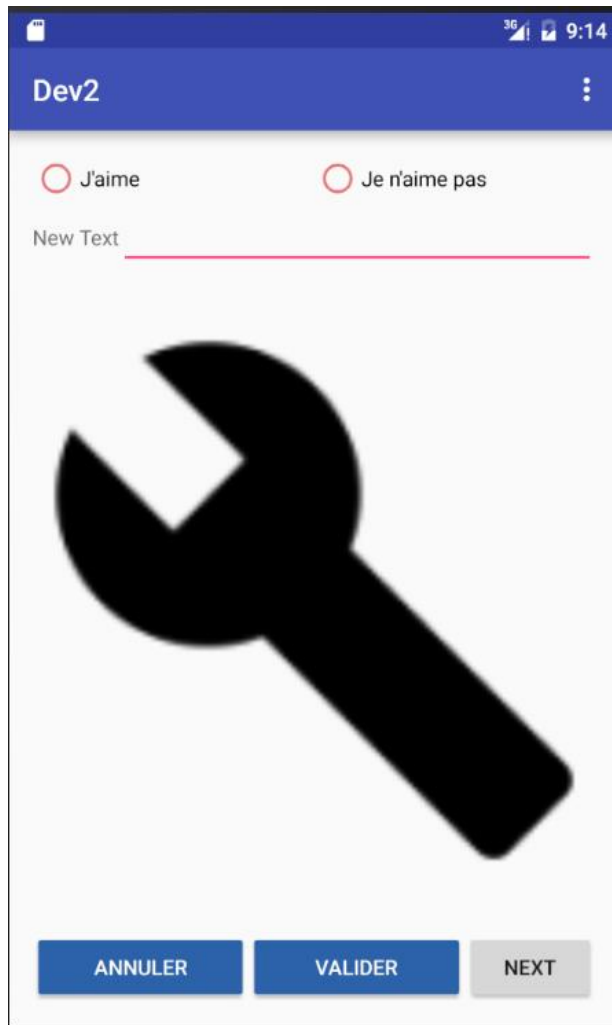


# AlertDialog

```
//Préparation de la fenêtre
AlertDialog.Builder alertDialogBuilder = new AlertDialog.Builder(this);
//Message
alertDialogBuilder.setMessage("Mon message");
//titre
alertDialogBuilder.setTitle("Mon titre");
//bouton ok
alertDialogBuilder.setPositiveButton("ok",
    new DialogInterface.OnClickListener() {
        public void onClick(DialogInterface dialog, int which) {
            //Affiche un toast apres le click sur le bouton ok
            Toast.makeText(MainActivity.this, "Click sur ok",
                Toast.LENGTH_SHORT).show();
        }
    });
//Icône
alertDialogBuilder.setIcon(R.mipmap.ic_launcher);
//Afficher la fenêtre
alertDialogBuilder.show();
```



# MENU



# MENU

- Pré construit par le système
- La classe Activity met à disposition 2 méthodes à surcharger

```
//Remplir le menu
```

```
@Override
```

```
public boolean onCreateOptionsMenu(Menu menu) {...}
```

```
//Intercepter un click sur le menu
```

```
@Override
```

```
public boolean onOptionsItemSelected(MenuItem item) {...}
```

# Exemple

```
/* Crée le menu */
public boolean onCreateOptionsMenu(Menu menu)
{
    menu.add(0, 25, 0, "New Game");
    menu.add(0, 26, 0, "Quit");
    return super.onCreateOptionsMenu(menu);
}

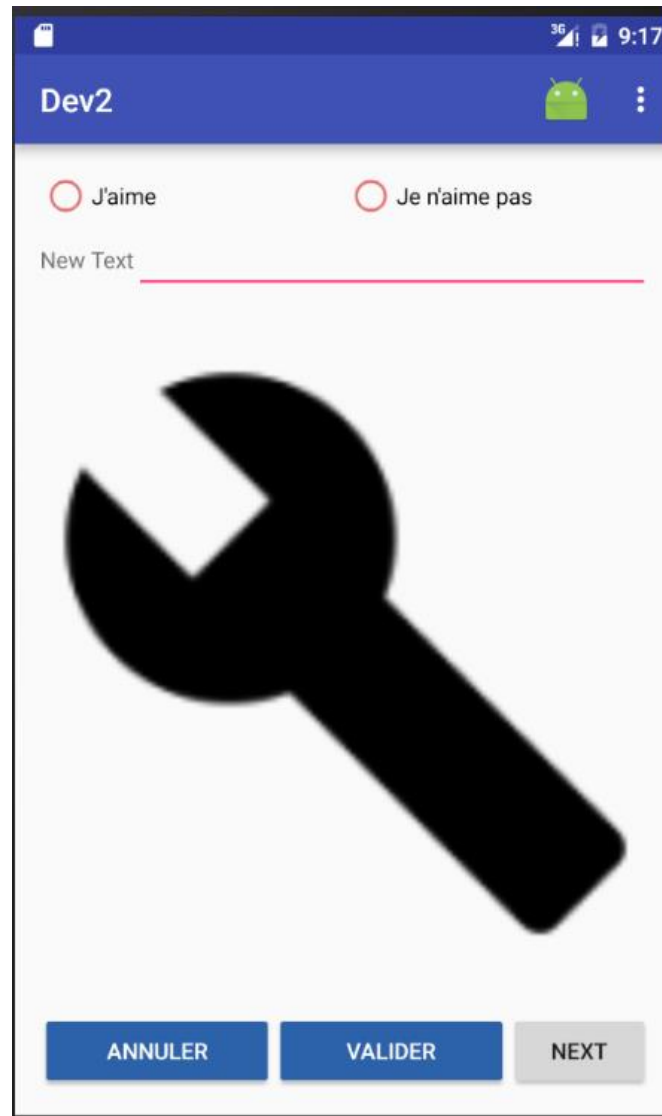
/* Handles item selections */
public boolean onOptionsItemSelected(MenuItem item)
{
    switch (item.getItemId())
    {
        case 25:
            newGame();
            break;
        case 26:
            quit();
            break;
    }
    return super.onOptionsItemSelected(item);
}
```

# MENUITEM AVEC ICÔNES

- Possibilité d'ajouter une icône à un menuitem grâce à la méthode
  - `setIcon(Drawable icon)`
- Possibilité d'utiliser nos propres drawables ou ceux d'Android
  - accessible via `android.R.drawable.ic_*`

```
menu.add(...).setIcon(R.drawable.myIcon);  
menu.add(...).setIcon(android.R.drawable.ic_menu_agenda)
```

# MENU



# MENUITEM AVEC ICÔNES VERSION XML

## ○ Chargement du menu depuis l'XML

```
public boolean onCreateOptionsMenu(Menu menu) {  
    //Version chargement du fichier menu à parti d'un XML  
    getMenuInflater().inflate(R.menu.menu_main_activity, menu);  
}
```

## ○ Menu\_main\_activity.xml

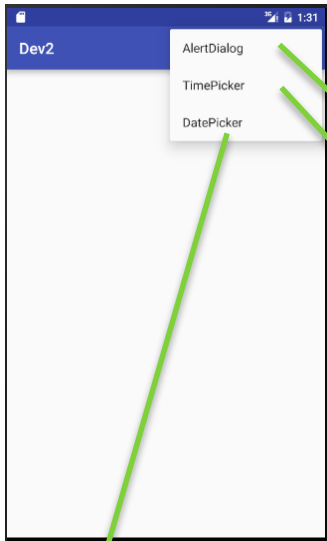
```
<menu xmlns:android="http://schemas.android.com/apk/res/android"  
    xmlns:app="http://schemas.android.com/apk/res-auto">  
    <item  
        android:id="@+id/menu_add"  
        android:icon="@drawable/ic_action_add_group"  
        android:orderInCategory="100"  
        android:title="@string/st_menu_add"  
        app:showAsAction="ifRoom"/>  
</menu>
```

# DÉTAILS

- MenuItem.SHOW\_AS\_ACTION\_ALWAYS :
  - Nous permet d'afficher l'icone même s'il y a peu de place
- MenuItem.SHOW\_AS\_ACTION\_IF\_ROOM :
  - Affiche l'icone seulement s'il y a de la place dans la barre
- MenuItem.SHOW\_AS\_ACTION\_WITH\_TEXT :
  - Affiche le texte à côté de l'icone

# TP MENU – DIALOG - TOAST

## Menu



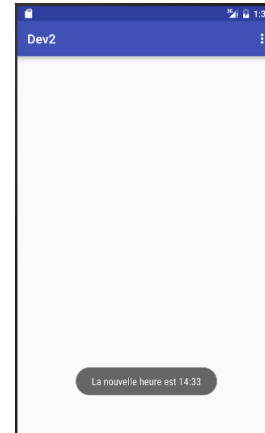
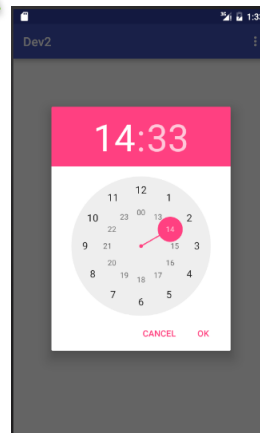
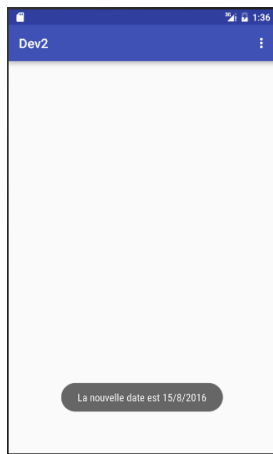
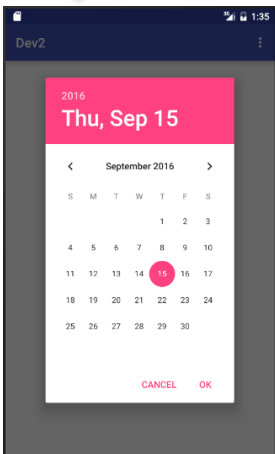
## AlertDialog



## Toast



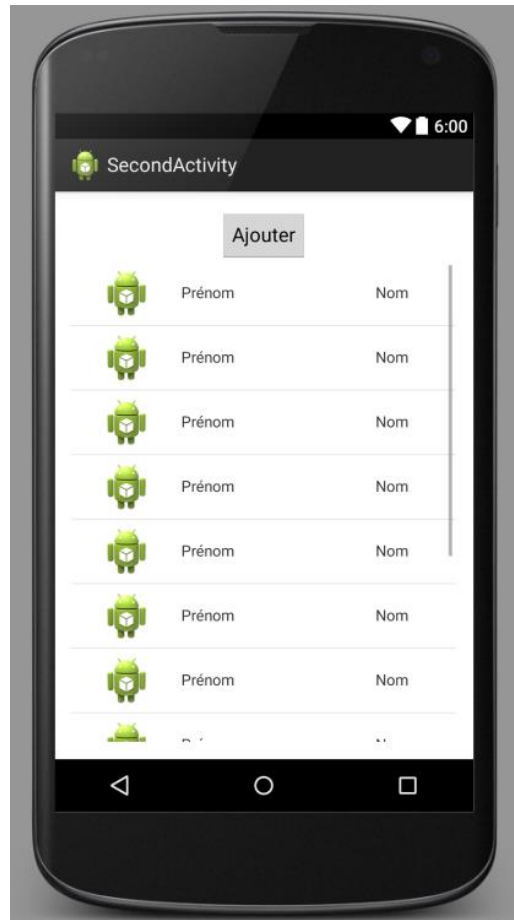
- Pour les plus rapides faire le TimePicke et le DatePicker





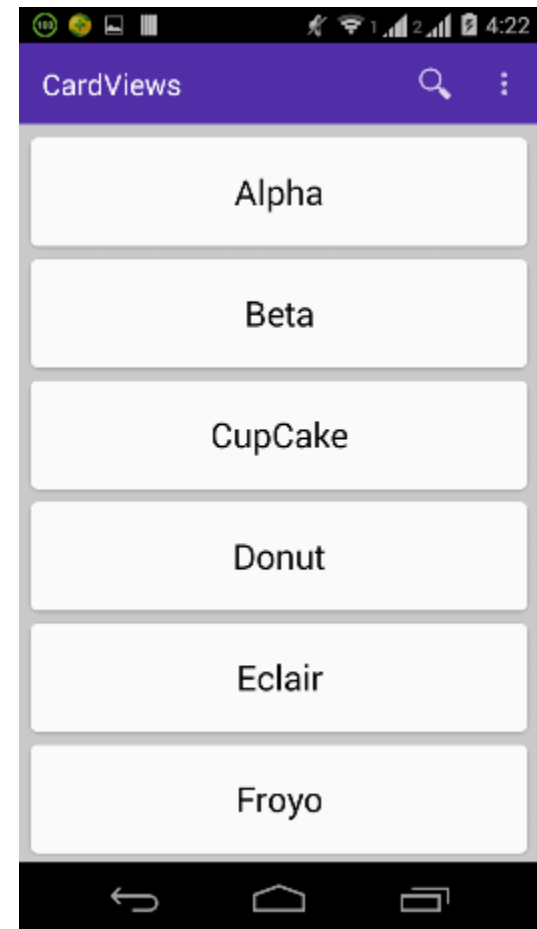
# ANDROID

- Afficher les données sous forme de liste



# NOUVEAU COMPOSANT DE LOLLIPOP

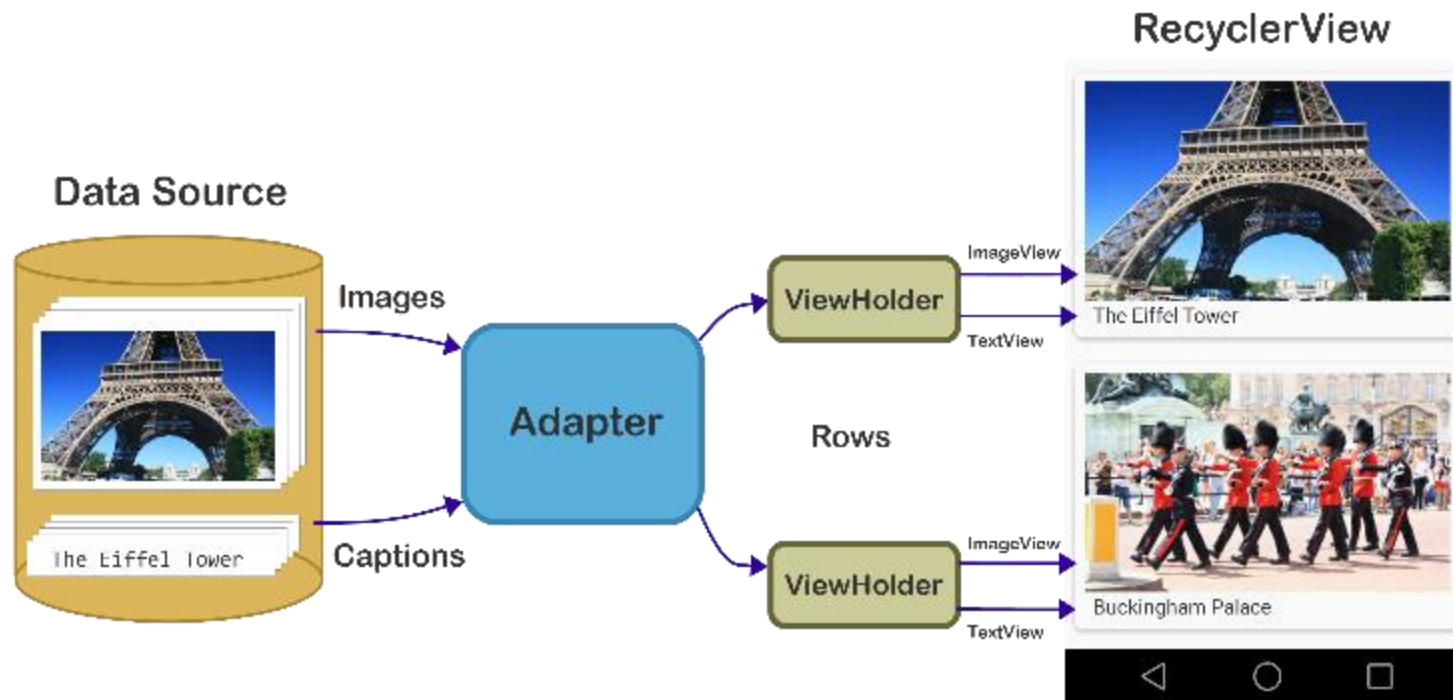
- RecyclerView
  - La nouvelle ListView de Lollipop
  - Permet des animations à l'utilisation
- CardView
  - Composant XML
  - Créer un bloc avec une ombre.



# LISTVIEW / RECYCLEVIEW

- Recycler les cellules, pour le bien de votre device.
- Créer un fichier XML représentant le layout **d'une ligne**
- Créer un adapter indiquant comment afficher chaque ligne
- Utiliser un composant d'affichage d'adapter:
  - RecyclerView
  - ListView

# RECYCLEVIEW



# RECYCLEVIEW : LE COMPOSANT

- Ajouter la librairie dans dependencies dans build.gradle

- `//Recycleview`  
`compile 'com.android.support:recyclerview-v7:23.+'`

- Utiliser le composant dans les layouts :

```
<android.support.v7.widget.RecyclerView  
    android:id="@+id/rv"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"/>
```

- Récupérer le composant dans l'activité :

```
RecyclerView rv = (RecyclerView) findViewById(R.id.rv);
```

# RECYCLEVIEWADAPTER

```
public class RVAdapter
    extends RecyclerView.Adapter<RVAdapter.ViewHolder> {

    //Classe qui stocke les composants graphiques d'1 ligne
    protected static class ViewHolder extends RecyclerView.ViewHolder {

        public TextView ec_tv_nom, ec_tv_prenom;
        public ImageView ec_iv;

        public ViewHolder(View itemView) {
            super(itemView);

            ec_tv_nom = (TextView) itemView.findViewById(R.id.ec_tv_nom);
            ec_tv_prenom = (TextView) itemView.findViewById(R.id.ec_tv_prenom);
            ec_iv = (ImageView) itemView.findViewById(R.id.ec_iv);
        }
    }

    public RVAdapter.ViewHolder onCreateViewHolder(ViewGroup vg, int viewType) {...}
    public void onBindViewHolder(RVAdapter.ViewHolder holder, int position) {...}
    public int getItemCount() {...}
}
```

# RECYCLEVIEWADAPTER

```
public class RVAdapter extends RecyclerView.Adapter<RVAdapter.ViewHolder> {  
  
    protected static class ViewHolder extends RecyclerView.ViewHolder {...}  
  
    //Détermine quel fichier XML on utilise pour représanter une cellule  
    @Override  
    public RVAdapter.ViewHolder onCreateViewHolder(ViewGroup vg, int viewType) {  
        View v=LayoutInflater.from(vg.getContext()).inflate(R.layout.eleve_cellule, vg, false);  
        return new RVAdapter.ViewHolder(v);  
    }  
  
    public void onBindViewHolder(RVAdapter.ViewHolder holder, int position) {...}  
    public int getItemCount() {...}  
}
```

# RECYCLEVIEWADAPTER

```
public class RVAdapter extends RecyclerView.Adapter<RVAdapter.ViewHolder> {  
  
    private ArrayList<Eleve> eleveBeanList;  
    //Constructeur  
    public RVAdapter(ArrayList<Eleve> eleveBeanList) {  
        this.eleveBeanList = eleveBeanList;  
    }  
  
    protected static class ViewHolder extends RecyclerView.ViewHolder {...}  
    public RVAdapter.ViewHolder onCreateViewHolder(ViewGroup vg, int viewType) {...}  
  
    //Remplir les composants graphique de chaque cellule  
    @Override  
    public void onBindViewHolder(RVAdapter.ViewHolder holder, int position) {  
        //L'élève correspondant à la ligne  
        Eleve eleve = eleveBeanList.get(position);  
        holder.ec_tv_nom.setText(eleve.getNom());  
        holder.ec_tv_nom.setText(eleve.getNom());  
        holder.ec_tv_prenom.setText(eleve.getPrenom());  
    }  
  
    public int getItemCount() {...}  
}
```



# RECYCLEVIEWADAPTER

```
public class RVAdapter extends RecyclerView.Adapter<RVAdapter.ViewHolder> {  
  
    private ArrayList<Eleve> eleveBeanList;  
    public RVAdapter(ArrayList<Eleve> eleveBeanList) {...}  
    protected static class ViewHolder extends RecyclerView.ViewHolder {...}  
    public RVAdapter.ViewHolder onCreateViewHolder(ViewGroup vg, int viewType) {...}  
    public void onBindViewHolder(RVAdapter.ViewHolder holder, int position) {...}  
  
    //Combien de cellule on affiche  
    @Override  
    public int getItemCount() {  
        return eleveBeanList.size();  
    }  
}
```

# RecyclerView (Coté Activity)

```
//Données
private ArrayList<Eleve> eleveList;
private RVAdapter rVAdapter;
//Composant graphique afficheur de RecyclerView.Adapter
private RecyclerView recycleView;

//Création de la liste
eleveList = new ArrayList<Eleve>();

//Instanciation d'un RVAdapter
rVAdapter = new RVAdapter(eleveList);

recycleView = (RecyclerView) findViewById(R.id.recycleView);
// L'adapter que l'on souhaite afficher
recycleView.setAdapter(rVAdapter);
//Réglage : Est ce qu'on affiche ligne par ligne ou
recycleView.setLayoutManager(new LinearLayoutManager(this));
//new GridLayoutManager(this, 2) //Sous forme de tableau à 2 colonnes
// Réglage : type d'animation qu'on utilise
recycleView.setItemAnimator(new DefaultItemAnimator());
```

# RecyclerView Actualisation et animation

//Indiquer que ma source de donnée à changé et qu'il faut rafraichir

```
rVAdapter.notifyDataSetChanged();
```

//Si on sait ce qui a changé, on l'indique pour une animation d'insertion

```
rVAdapter.notifyItemInserted(0);
```

//Animation de suppression

```
rVAdapter.notifyItemRemoved(0);
```

//Animation de déplacement

```
rVAdapter.notifyItemMoved(0,3);
```

# CARDVIEW: LE COMPOSANT

- Ajouter la librairie dans dependencies dans build.gradle

```
//CardView
```

```
compile 'com.android.support:cardview-v7:23.+'
```

- Utiliser le composant dans les layouts (Il ne peut contenir qu'un seul enfant):

```
<android.support.v7.widget.CardView  
    xmlns:android="http://schemas.android.com/apk/res/android"  
    xmlns:card_view="http://schemas.android.com/apk/res-auto"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:layout_margin="6dp"  
    card_view:cardCornerRadius="4dp" >
```

# TP – RECYCLEVIEW

- **Etape 1 : Créez une RecyclerView d'élève**
  - Un bouton « ajouter » ajoute un élève à la liste
  - Pour le nom et le prénom de l'élève : eleve0, eleve1, eleve2...
  - Pas d'image pour le moment
- **Etape 2 : Ajouter une animation d'insertion.**
- **Etape 3 : Ajouter un CardView à votre layout de cellule**
- **Etape 4 (pour les plus avancés) : Gestion d'images en ligne**
  - Utiliser la librairie Glide pour gérer le chargement des images depuis une URL
  - <http://www.tutos-android.com/tag/glide-tuto>
  - Quelques url d'images d'élèves :
    - [https://pixabay.com/static/uploads/photo/2014/04/02/17/02/girl-307747\\_960\\_720.png](https://pixabay.com/static/uploads/photo/2014/04/02/17/02/girl-307747_960_720.png)
    - <http://www.coloriage.tv/dessincolo/ecolier.png>
    - <http://clamart-lafontaine-blog.e-monsite.com/medias/images/eleve.png>
    - <http://ekladata.com/2NvmX2GdczA71ZMewxFwyR9CesE@350x586.png>

# LISTVIEW

## ○ Etapes simples

- Créer un fichier XML représentant 1 ligne
- Créer un adaptateur qui utilisera ce fichier pour chaque ligne.
- Ajouter et récupérer le ListView dans l'activité
- Relier l'adapter à la ListView et l'utiliser dans l'activité.

## ○ Cela se complique

- Récupérer les pointeurs vers les composants de ce XML
- Les remplir avec l'élément correspondant à la ligne.

## ○ C'est compliqués mais il faut comprendre

- On réutilise nos lignes plutôt que d'en créer de nouvelle.
- On fait de même avec nos pointeurs

# Ce qu'on veut faire

```
LinearLayout ll = (LinearLayout) findViewById(R.id.ll);
ArrayList<Eleve> eleves = new ArrayList<>();
LayoutInflater inflater = (LayoutInflater) getSystemService(Context.LAYOUT_INFLATER_SERVICE);

//Ce qu'on veut faire
for (int i = 0; i < eleves.size(); i++) {
    //je crée une ligne par élève
    View v = inflater.inflate(R.layout.cellule_eleve, null);
    //Je l'ajoute au layout
    ll.addView(v);
    //Je récupère le pointeur vers le composant graphique
    TextView tv_prenom = (TextView) v.findViewById(R.id.tv_prenom);
    //je mets le prénom dedans.
    tv_prenom.setText(eleves.get(i).getPrenom());
}
```

# Exemple : BaseAdapter 1/2

```
public class EleveAdapter extends BaseAdapter {
    private LayoutInflater mInflater;
    private List<Eleve> eleveList;

    public EleveAdapter(Context context, List<Eleve> eleveList) {
        mInflater = (LayoutInflater) context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
        this.eleveList = eleveList;
    }

    @Override
    public int getCount() {
        return eleveList.size();
    }

    @Override
    public Eleve getItem(int position) {
        return eleveList.get(position);
    }

    @Override
    public long getItemId(int position) {
        return position;
    }

    @Override
    public View getView(int position, View rowView, ViewGroup parent) {...}
}
```



# Ce qu'on veut faire

```
LinearLayout ll = (LinearLayout) findViewById(R.id.ll);  
ArrayList<Eleve> eleves = new ArrayList<>();  
LayoutInflater inflater = (LayoutInflater) getSystemService(Context.LAYOUT_INFLATER_SERVICE);
```

```
//Ce qu'on veut faire
```

```
for (int i = 0; i < eleves.size(); i++) {  
    //je crée une ligne par élève  
    View v = inflater.inflate(R.layout.cellule_eleve, null);  
    //Je l'ajoute au layout  
    ll.addView(v);  
    //Je récupère le pointeur vers le composant graphique  
    TextView tv_prenom = (TextView) v.findViewById(R.id.tv_prenom);  
    //je mets le prénom dedans.  
    tv_prenom.setText(eleves.get(i).getPrenom());  
}
```

```
//Ce que le système fait avec l'adaptateur (en très très simple)
```

```
for (int i = 0; i < eleveAdapter.getCount(); i++) {  
    ll.addView(eleveAdapter.getView(i, null, null));  
}
```

# Exemple : BaseAdapter 2/2

```
@Override
public View getView(int position, View rowView, ViewGroup parent) {
    //création
    View temp = mInflater.inflate(R.layout.eleve_cellule, null);
    TextView tv_nom = (TextView) temp.findViewById(R.id.tv_nom);
    //on remplit avec l'objet voulu
    Eleve eleve = getItem(position);
    tv_nom.setText(eleve.getNom());

    return temp;
}
```

# BaseAdapter (Coté Activity)

```
//Création
private ListView lv;
private List<Eleve> eleveList;
private EleveAdapter eleveAdapter;

eleveList = new ArrayList<Eleve>();
eleveAdapter = new EleveAdapter(this, eleveList);
lv = (ListView) findViewById(R.id.lv);
lv.setAdapter(eleveAdapter);

//on prévient la liste que les données ont changés
eleveAdapter.notifyDataSetChanged();
```

# TP – LISTVIEW

- Créez une ListView d'élève à l'aide d'un baseAdapter.
  - Une cellule sera définit par un prénom et un nom
- Prévoir un bouton pour ajouter un élève à la liste.

//Pour prévenir l'adapter que les données ont changées

```
eleveAdapter.notifyDataSetChanged();
```

# BaseAdapter : Recyclage

```
@Override
public View getView(int position, View rowView, ViewGroup parent) {
    if (rowView == null) {
        //création
        rowView = mInflater.inflate(R.layout.eleve_cellule, null);
    }

    TextView tv_nom = (TextView) rowView.findViewById(R.id.tv_nom);
    //on remplit avec l'objet voulu
    Eleve eleve = getItem(position);
    tv_nom.setText(eleve.getNom());

    return rowView;
}
```

# TP – LISTVIEW

- Recyclez vos cellules dans votre adapter
- Faire en sorte que pour les filles le nom/prénom apparaissent en violet, et ne rien faire pour les garçons. M'appeler une fois que c'est fait.

# BaseAdapter : Optimiser encore plus

- Le View Holder represente les composants à modifier entre chaque cellule.
- Le but est de reutiliser une cellule existante et de ne faire que modifier les valeurs.

```
//-----  
// View Holder  
//-----  
public static class ViewHolder {  
    public TextView tv_nom;  
}
```

# BaseAdapter / Recyclage

@Override

```
public View getView(int position, View rowView, ViewGroup parent) {  
  
    ViewHolder viewHolder;  
    if (rowView == null) {  
        //création  
        rowView = mInflater.inflate(R.layout.eleve_cellule, null);  
        viewHolder = new ViewHolder();  
        viewHolder.tv_nom = (TextView) rowView.findViewById(R.id.tv_nom);  
        rowView.setTag(viewHolder);  
    }  
    else {  
        //recyclage  
        viewHolder = (ViewHolder) rowView.getTag();  
    }  
    //on remplit avec l'objet voulu  
    Eleve eleve = getItem(position);  
    viewHolder.tv_nom.setText(eleve.getNom());  
    return rowView;  
}
```



# TP – LISTVIEW

- Continuer d'optimiser votre liste
- Passer le device en mode paysage, qu'est ce qu'il se passe?
- Solution TP : Module listViewRecyclage