# Les ressources, dispositions et vues

Hakim MOKEDDEM
Ecole nationale Supérieure d'Informatique

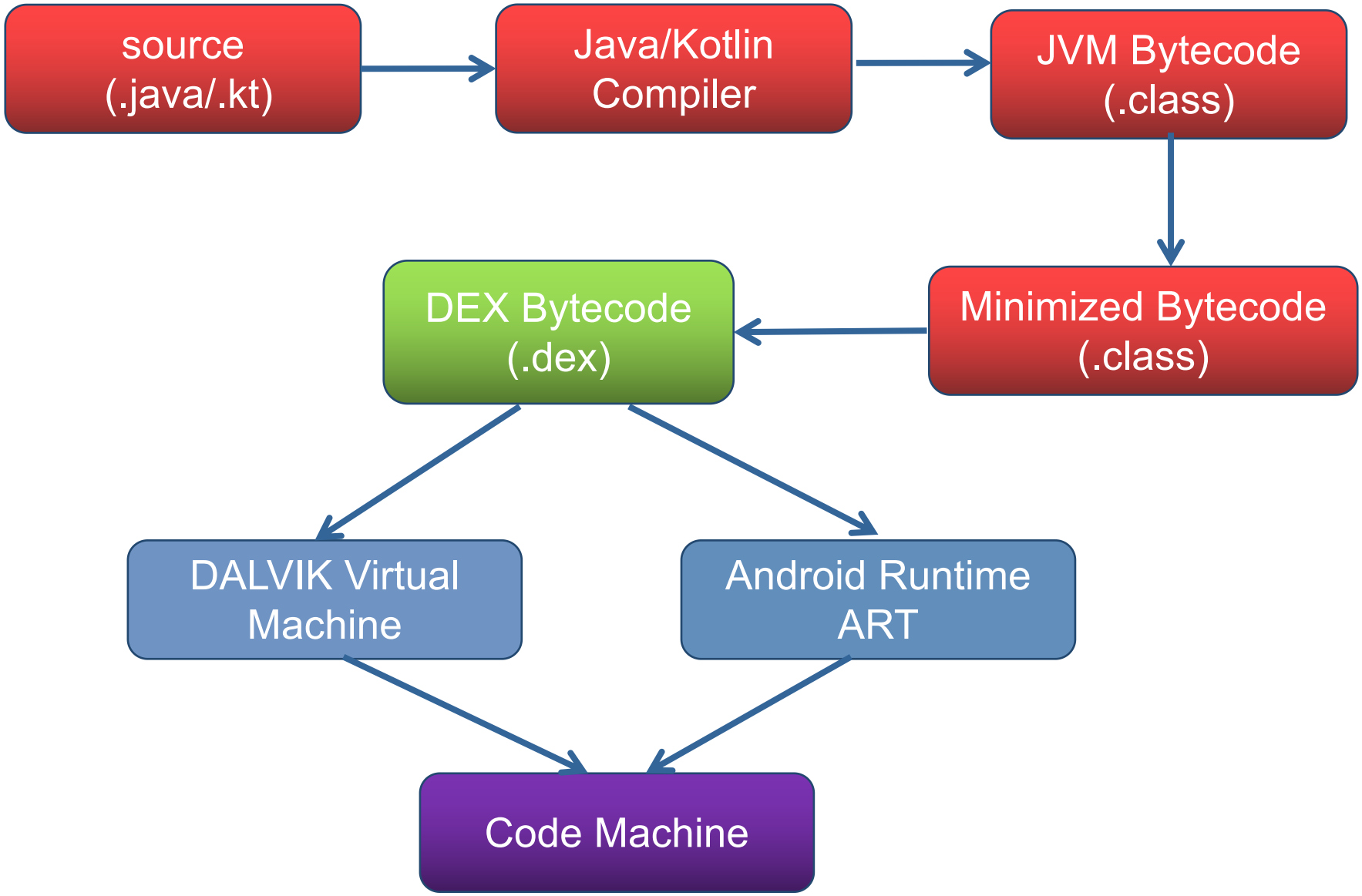
# Plan

- Le processus de compilation sous Android
- Développement Android: concepts-clés
- Les ressources
  - Les chaines de caractères
  - Les dimensions et images
  - Les dispositions
- Les vues

# Plan

- Le processus de compilation sous Android
- Développement Android: concepts-clés
- Les ressources
  - Les chaines de caractères
  - Les dimensions et images
  - Les dispositions
- Les vues

# Le processus de compilation sous Android



# DALVIK vs. ART

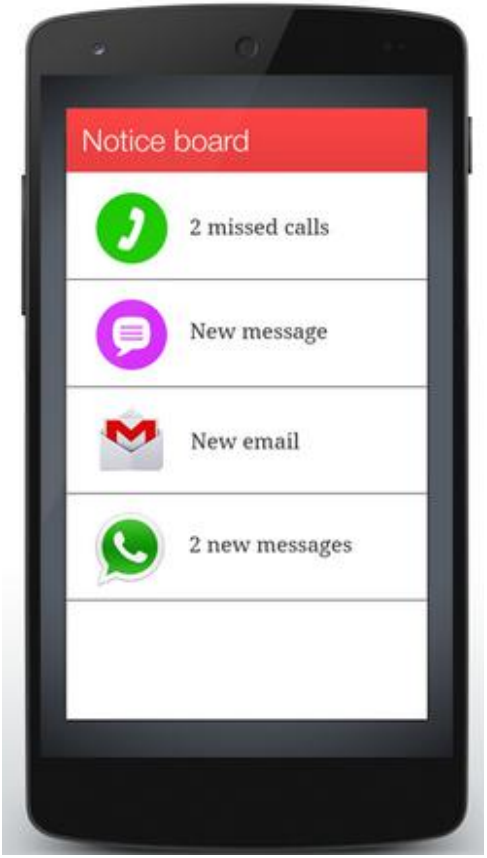
- Utilisation du mode de compilation JIT (Just in Time) sous DALVIK.
- ART remplace DALVIK dans la version Lollipop.
- Compilation avec ART pendant l'installation.
- Installation des applications sous ART plus lente que DALVIK.
- Exécution des applications sous ART plus rapide que DALVIK.

# Plan

- Le processus de compilation sous Android
- Développement Android: concepts-clés
- Les ressources
  - Les chaines de caractères
  - Les dimensions et images
  - Les dispositions
- Les vues

# Développement Android: concepts-clés

**Activité.** Un écran avec une seule interface utilisateur.



# Développement Android: concepts-clés

**App Manifest.** La configuration de l'application

- Il fournit des informations au dispositif mobile pour exécuter l'application.

*Exemples*. Activité principale, liste des activités, min SDK, le thème, l'icône de l'application, etc.

- Il contient la liste des permissions.

*Exemples*. Permission Internet, Appel, SMS, etc.

# Développement Android: concepts-clés

**App Manifest.** La configuration de l'application

- Il est utilisé par GooglePlay pour afficher les applications aux utilisateurs.

*Exemples*. Filtre en fonction de la version minimale du SDK, des tailles d'écrans supportées, etc.

# Développement Android: concepts-clés

**Gradle.** Outil de construction logicielle.

- Gestion des dépendances.
- Compilation.
- Génération de l'APK.
- Déploiement de l'APK.
- Build incrémental.

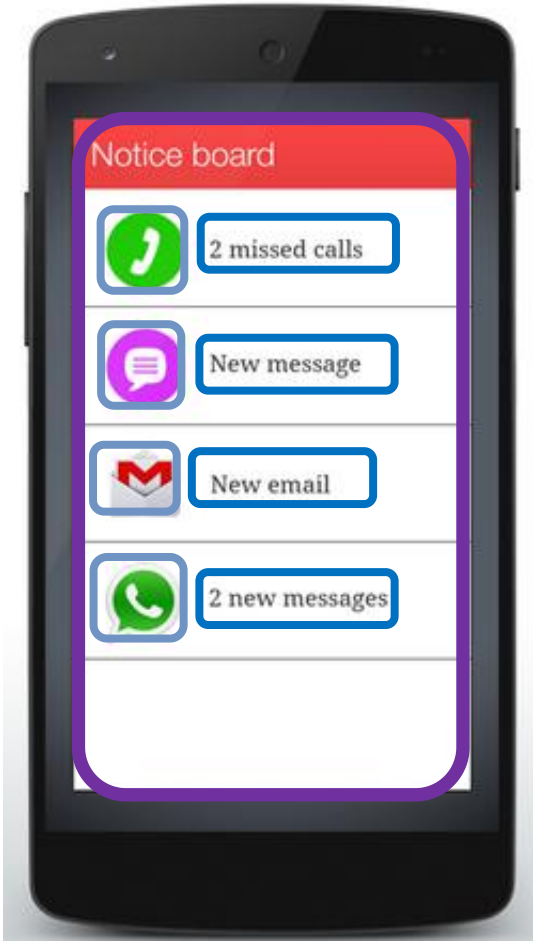
# Plan

- Le processus de compilation sous Android
- Développement Android: concepts-clés
- Les ressources
  - Les chaines de caractères
  - Les dimensions et images
  - Les dispositions
- Les vues

# Les ressources

**Ressource.** Un contenu statique utilisé dans l'application.

*Ressources images*




*Ressource Layout*

*Ressources*
*chaine de caractères*

# Exemples de ressources

Chaines de caractères

Dimensions

Images

Layout

# Utilité des ressources

- Séparation entre le code et son contenu
- Gérer les changements de configuration

**Exemple.** Changement de la langue

# Accès aux ressources

**A partir du code**

- L'accès à une ressource se fait par le référencement de l'identifiant de la ressource.
- Les identifiants des ressources sont auto-générés dans la classe R.

**Exemple**

*<string name="msg">Bonjour</string>*

*val message = resources. getString(R.string.msg)*

# Accès aux ressources

**A partir d'un fichier XML**

*@resource_type/resource_name*

**Exemple**

- Ressource chaine de caractères

  *<string name="msg">Envoyer</string>*

- Ressource Layout

  *<Button android:text="@string/msg" />*

# Exemples de ressources

Chaines de caractères

Dimensions

Images

Layout

# Les chaines de caractères

- **Les chaines de caractères statiques**

*<string name="msg">Bonjour</string>*

- **Accès**

Du Code

val msg = *getString(R.string.msg)*

De XML

*<Button android:text="@string/msg" />*

# Les chaines de caractères

- **Les chaines de caractères dynamiques**

*<string name="msg">Bonjour, %1$s! Vous avez %2$d*

*nouveaux messages.</string>*

- **Accès**

*val msg = String.format(getString(R.string.msg),"Mr  YY", 3)*

- **Résultat**

*"Bonjour, Mr  YY ;.Vous avez 3 nouveaux messages"*

# Les chaines de caractères

- **Les tableaux de chaines de caractères**

*<string-array name=“jours">*

*<item>Dimanche</item>*

*<item>Lundi</item>*

*<item>……</item>*

*</string-array>*

- **Accès**

val tab = resources.getStringArray(R.array. jours)

# Exemples de ressources

Chaines de caractères

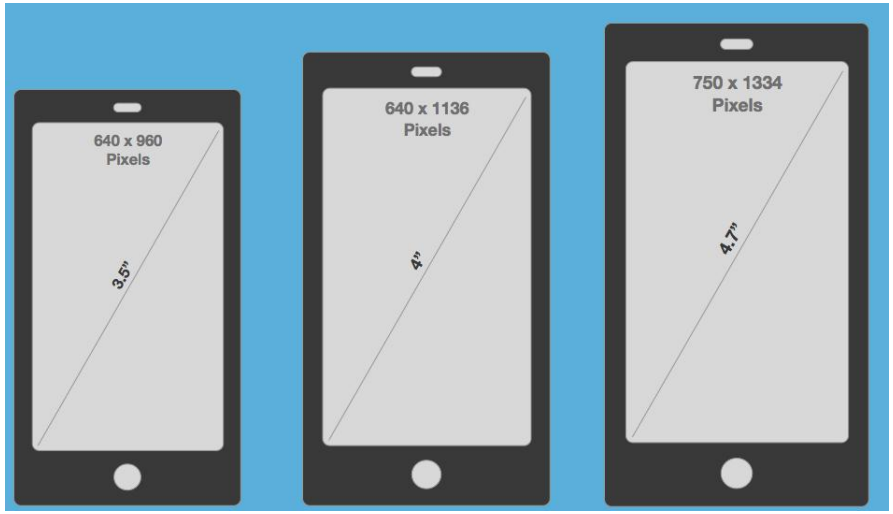
Dimensions

Images

Layout

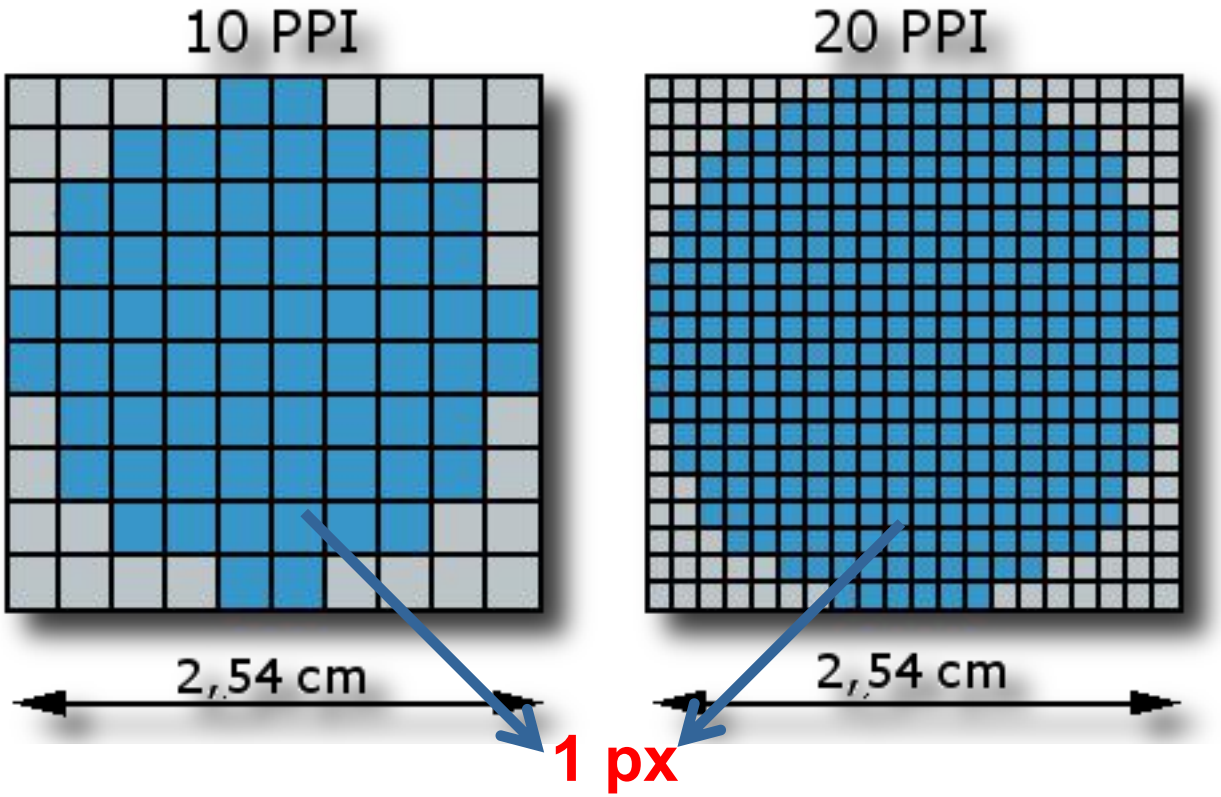
# Les dimensions

- **Pixel.** Le plus petit élément d'une image représenté sur un écran
- **Résolution de l'écran.** Le nombre total de pixels sur l'écran en largeur et hauteur



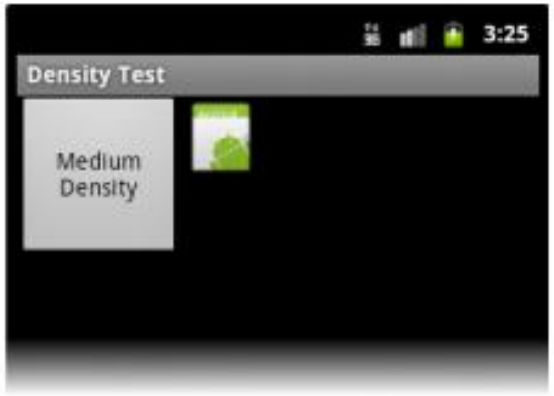
# Les dimensions

**Densité de l'écran.** Le nombre de pixels affichés par pouce.



# Les dimensions
# Problème d'utilisation des pixels

- La taille des pixels dépend de la densité de l'écran
- L'affichage sur des densités différentes n'est pas le



**Comment résoudre ce problème ?**

Utiliser une unité de mesure indépendante des pixels

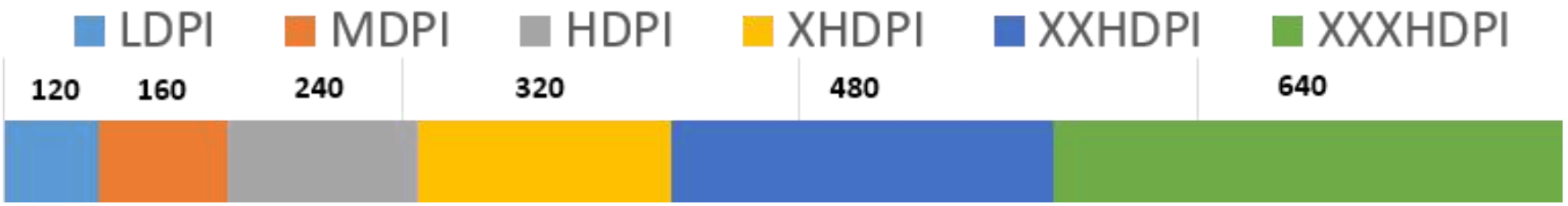
# Les dimensions
# Utilisation de l'unité DP

DP : Density-independent Pixels

Une unité abstraite relative à la densité de l'écran

$$dp = (width\ in\ pixels * 160) / screen\ density$$

**Les écrans Android**



# Les dimensions
# Utilisation de l'unité SP

- DP ne prend pas en compte les paramètres de l'utilisateur pour la taille d'un texte

**SP : Scaleable pixels**

- Utilisée pour définir la taille d'un texte.
- Prise en compte des préférences de l'utilisateur.

***SP= DP* Scale***

# Exemples de ressources

Chaines de caractères

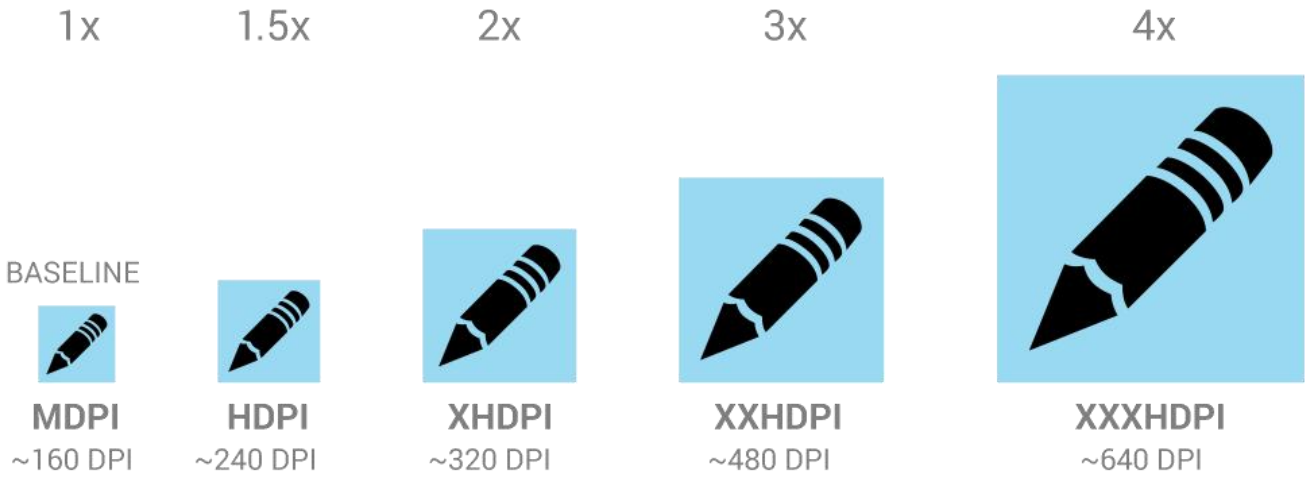
Dimensions

Images

Layout

# Les images

- Des images en fonciton de la densité de l'écran.

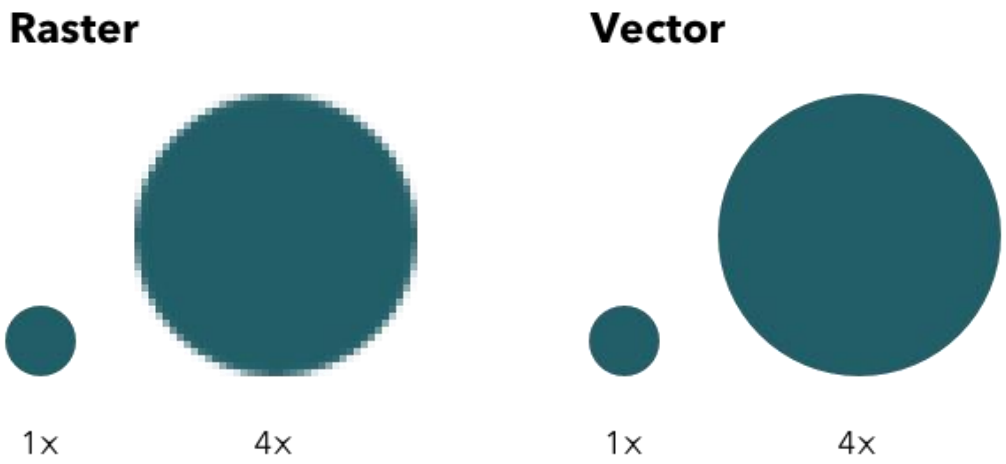



**Les outils**

- https://romannurik.github.io/AndroidAssetStudio/
- http://nsimage.brosteins.com/
- Plugin Android Studio : Android Drawable Importer

# Les images vectorielles

- Basées sur  des calculs mathématiques.
- Même qualité sur des densités différentes.
- Définies dans un fichier XML dans le dossier Drawable.



# Exemples de ressources

Chaines de caractères

Dimensions

Images

Layout

# Les dispositions

- **Qu'est ce qu'un layout ?**

Un layout est un gestionnaire de mise en place des éléments d'une interface.

- **Quels sont les layout recommandés sous Android ?**

LinearLayout

FrameLayout

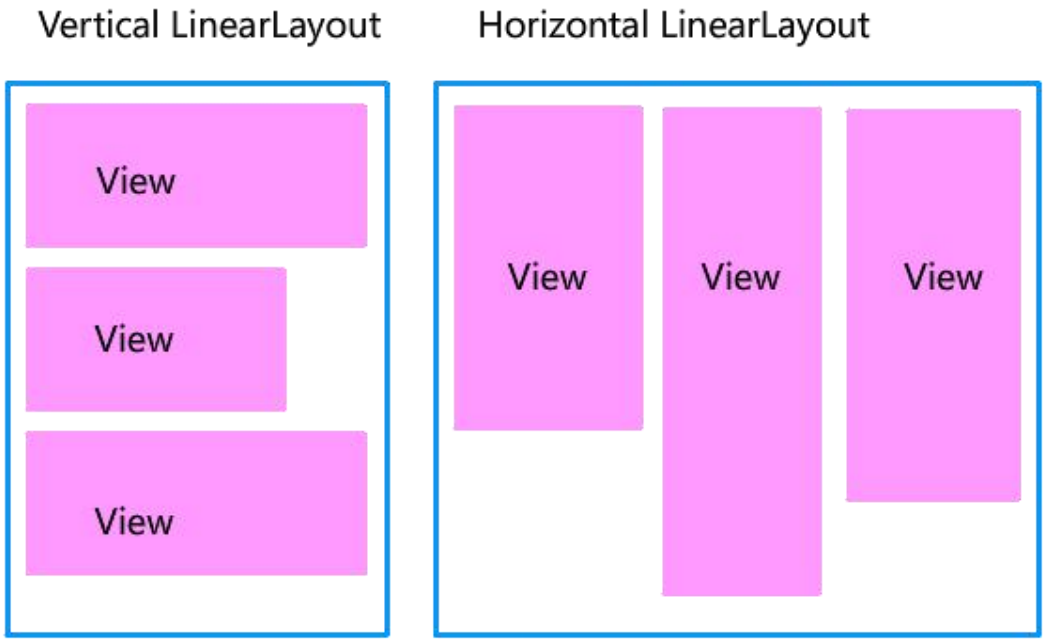
ConstraintLayout

# LinearLayout

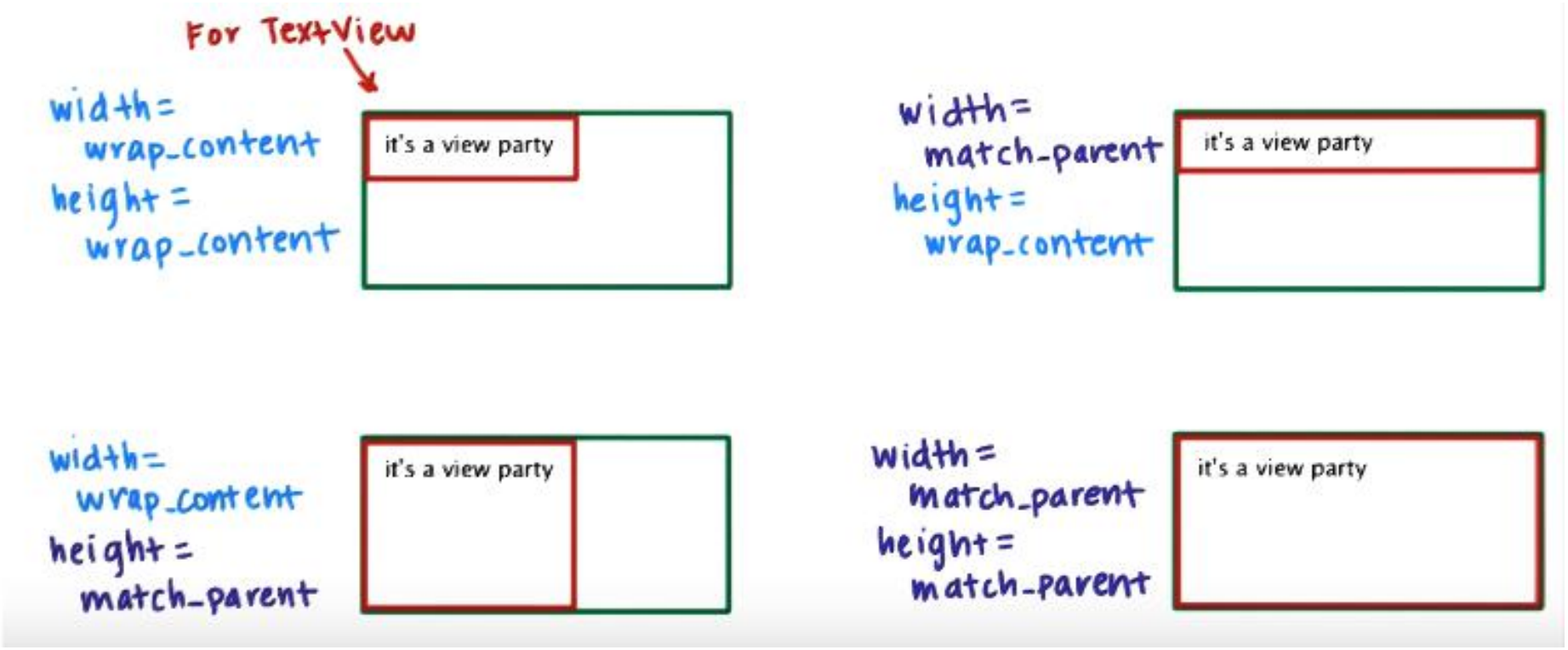
- **Quand utiliser LinearLayout ?**

Créer des interfaces simples.

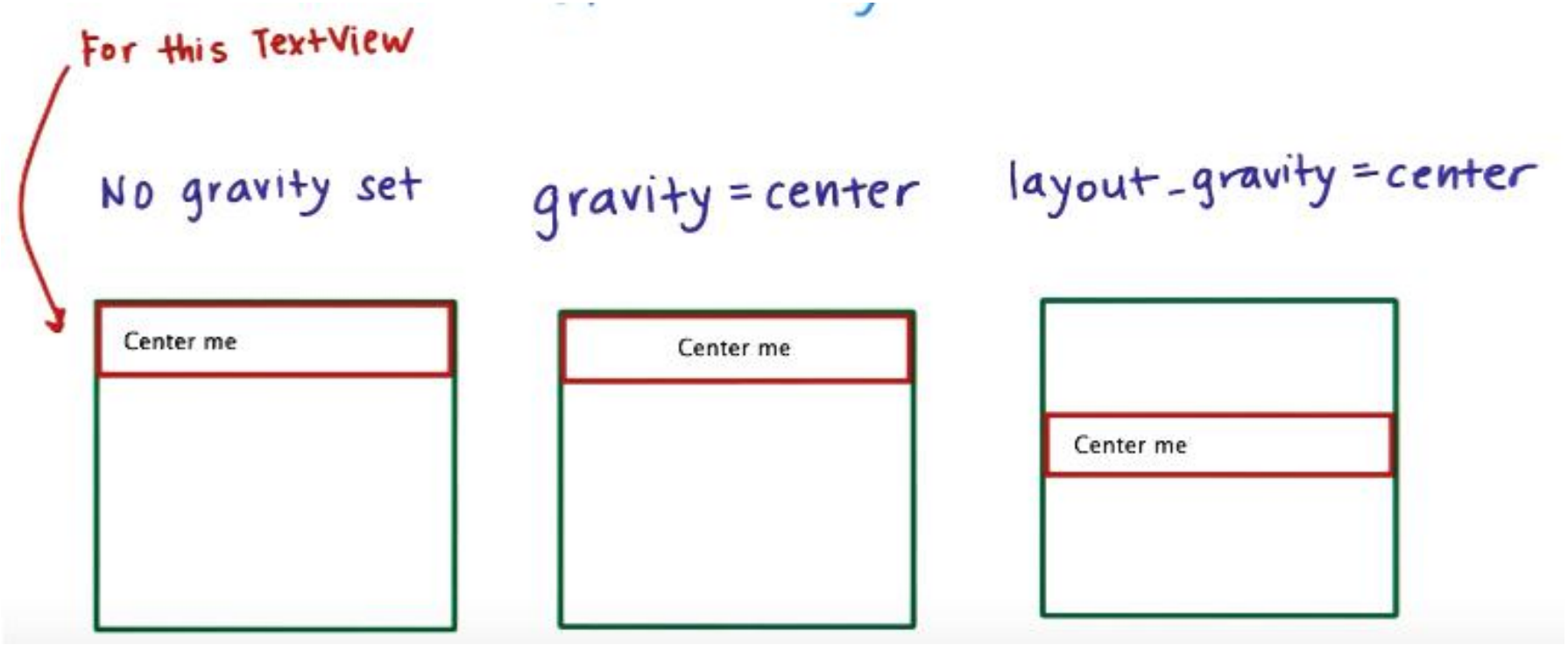
- **Quels sont les types de LinearLayout ?**



# LinearLayout
# match_parent & wrap_content

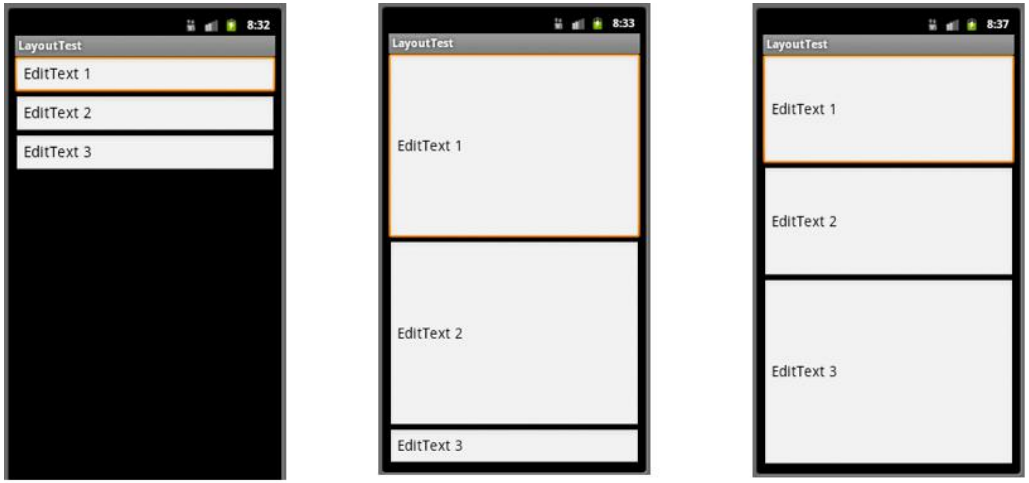


# LinearLayout
# Gravity vs. Layout_Gravity



# LinearLayout
# Weight

- Spécifier l'espace occupé d'un élément.
- La valeur par défaut de *layout_weight* est 0.
- Spécifier la valeur 0dp à *layout_width* ou *layout_ height* en fonction de l'orientation du Layout.

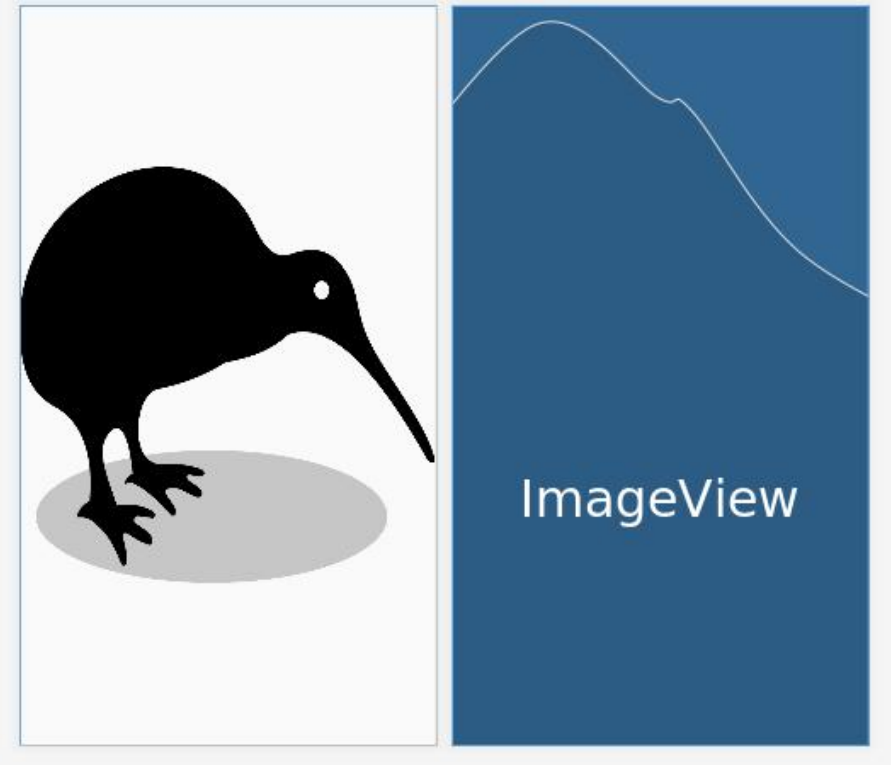



Default weights | Weights – 1,1,0 | Weights – 1,1,2

# FrameLyout

- **Quand utiliser FrameLayout ?**

Mettre une seule vue sur l'interface.



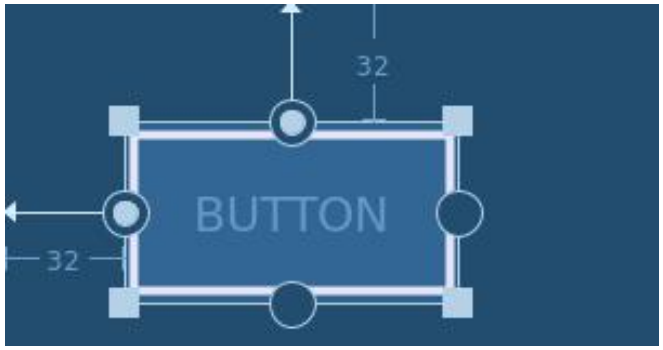
# ConstraintLayout

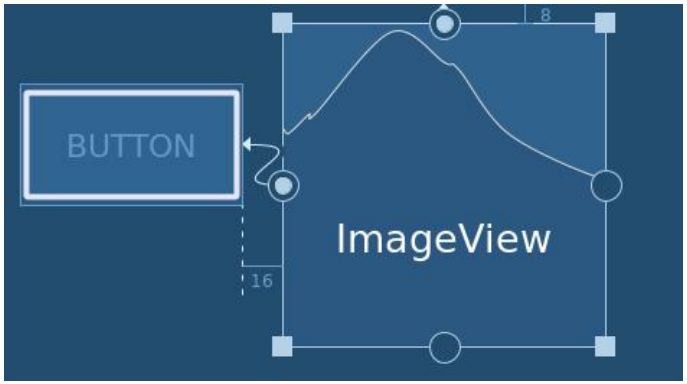
- **Quand utiliser ConstraintLayout ?**

Créer des interfaces complexes.

- **Quel est le principe de ConstraintLayout ?**

Chaque vue doit avoir au moins une contrainte de positionnement verticale et une autre horizontale.



# ConstraintLayout
# Fonctionnalités sous Android Studio

Biais

MatchConstraint

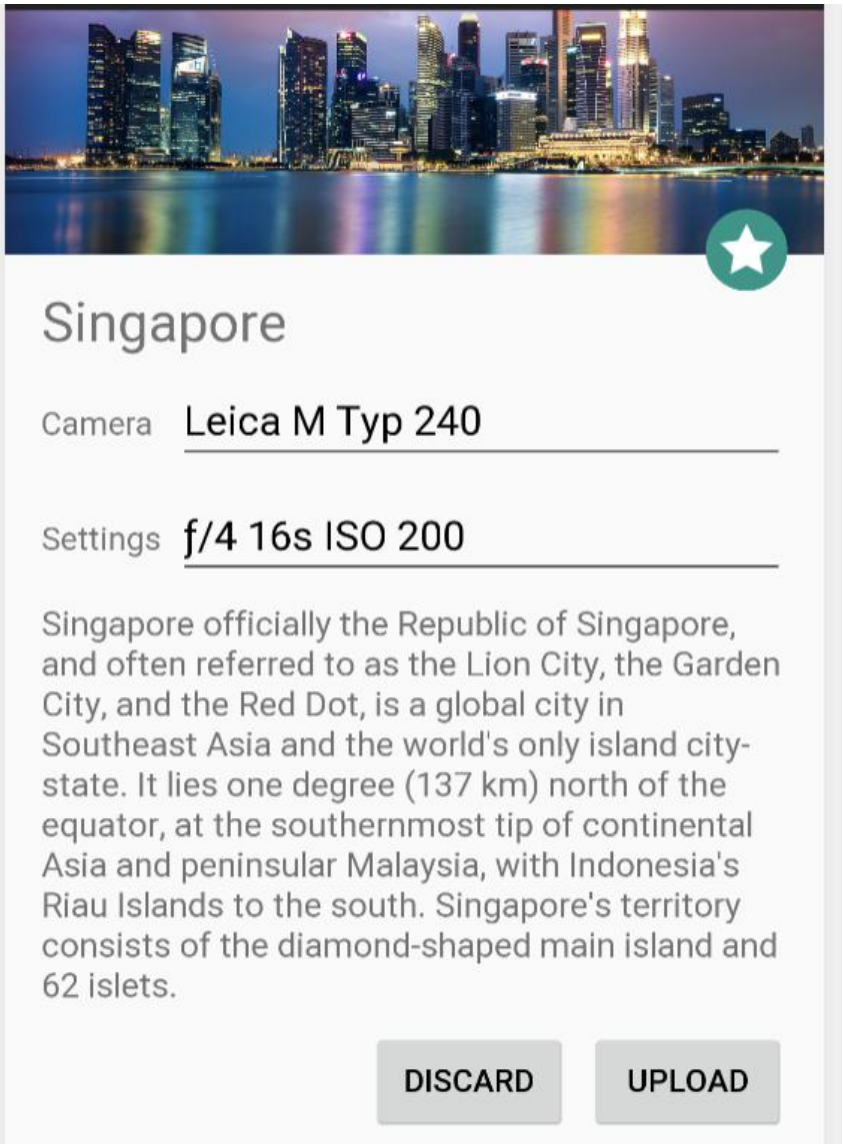
Baseline

Chains

Guidelines

Barriers

# ConstraintLayout

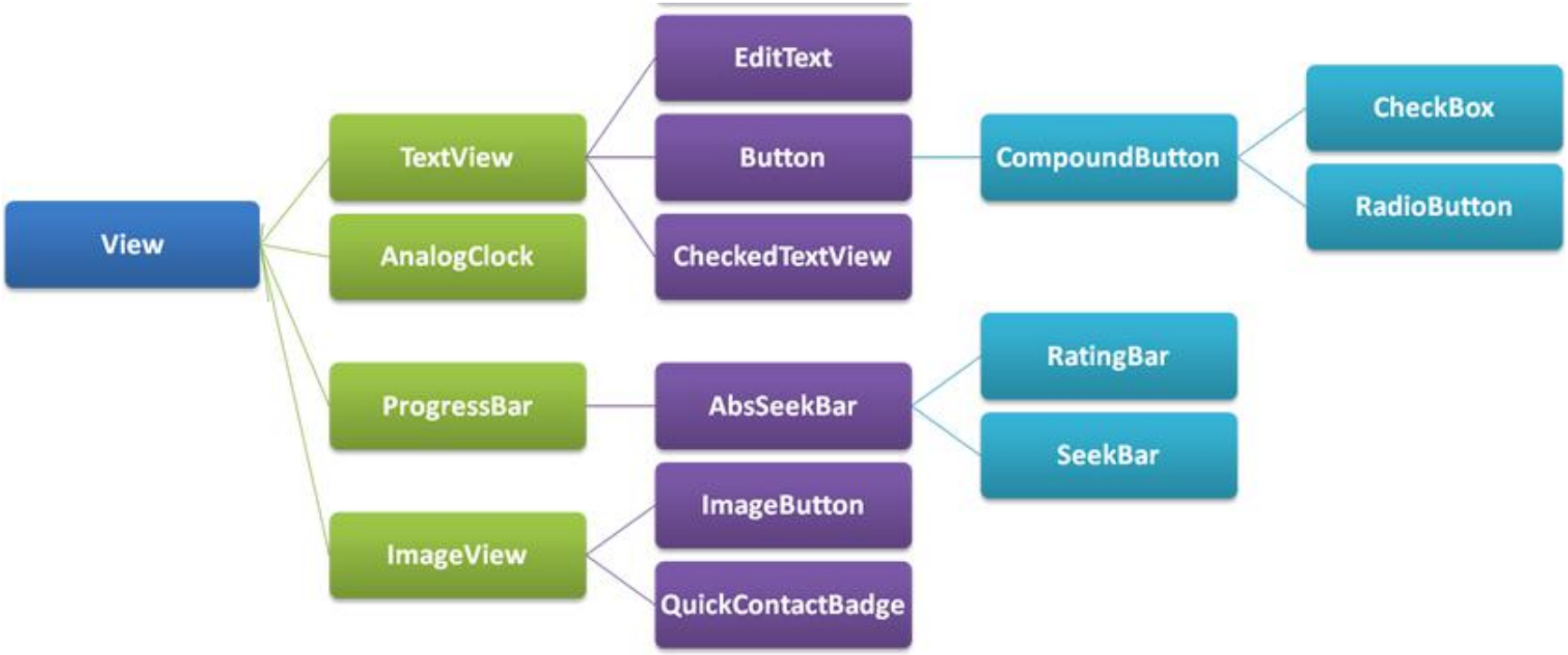


# Plan

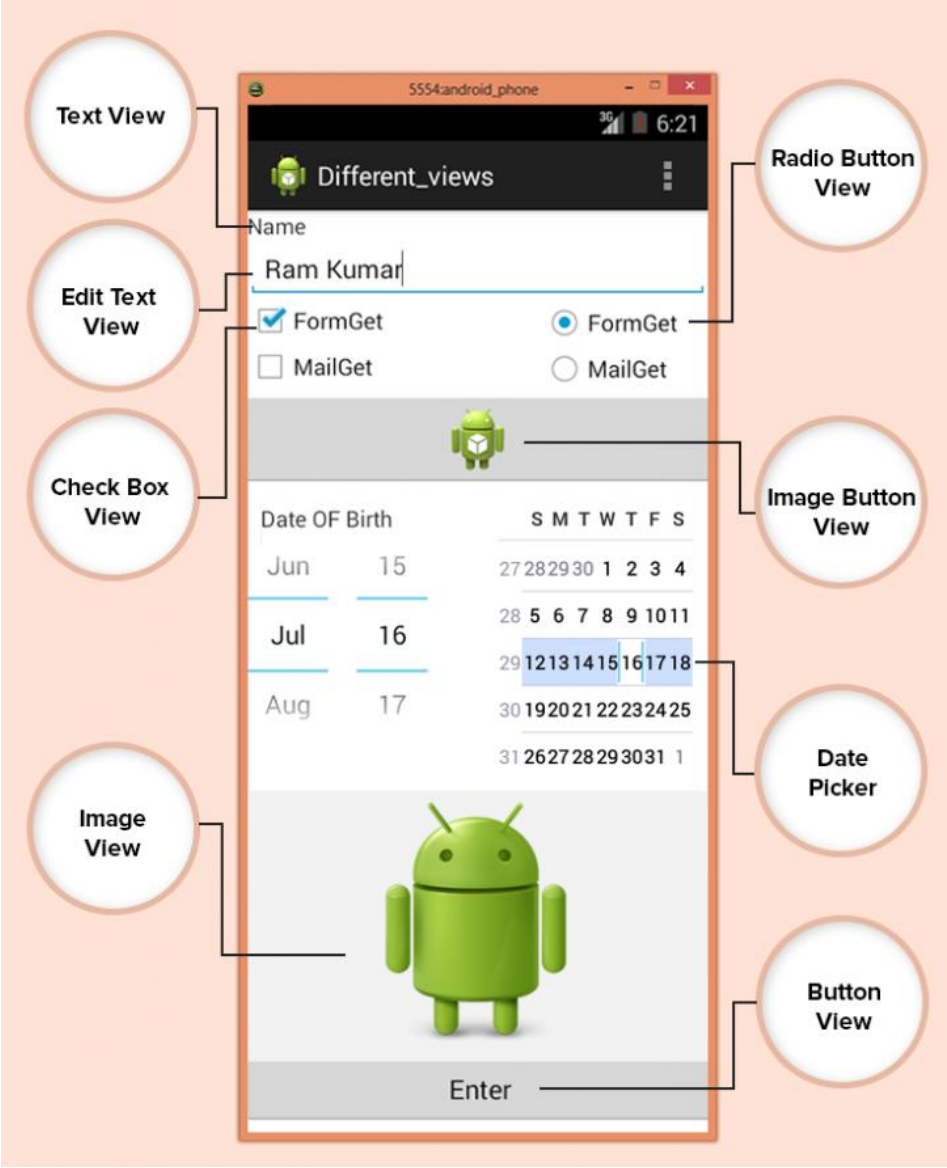
- Le processus de compilation sous Android
- Développement Android: concepts-clés
- Les ressources
  - Les chaines de caractères
  - Les dimensions et images
  - Les dispositions
- Les vues

# Les vues

- Une vue représente un élément de l'interface.
- Une vue est une instance d'une sous classe de View



# Les vues
# Exemples



# Les vues
# Accés à partir du code

- Avec la méthode *findViewById*.

```
val button =  findViewById(R.id.button) as Button
```

- Directement à partir de son identifiant.

**Exemple**

```
button.setOnClickListener {
    // implémentation
}
```

# Liste


9:22
ListViewImages
Pizza
Spicy Chiken Pizza
R$ 310.00
Burger
Beef Burger
R$ 350.00
Pizza
Chiken Pizza
R$ 250.00
Burger
Chicken Burger
R$ 350.00
Burger
Fish Burger
R$ 310.00
Mango
Mango Juice
R$ 250.00

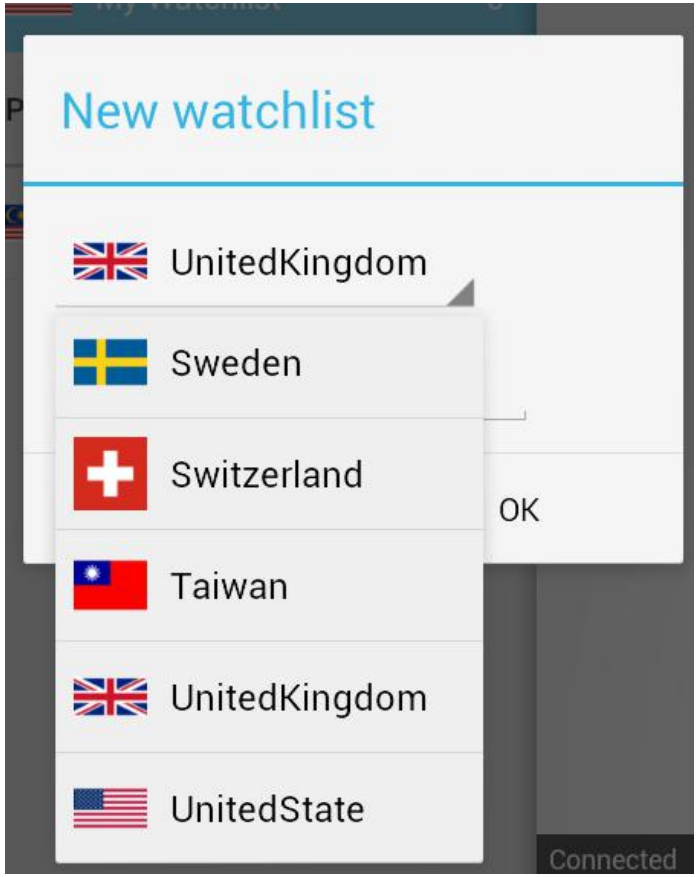

16:52
Test
London
http://en.wikipedia.org/wiki/London
Rome
http://en.wikipedia.org/wiki/Rome
Paris
http://en.wikipedia.org/wiki/Paris

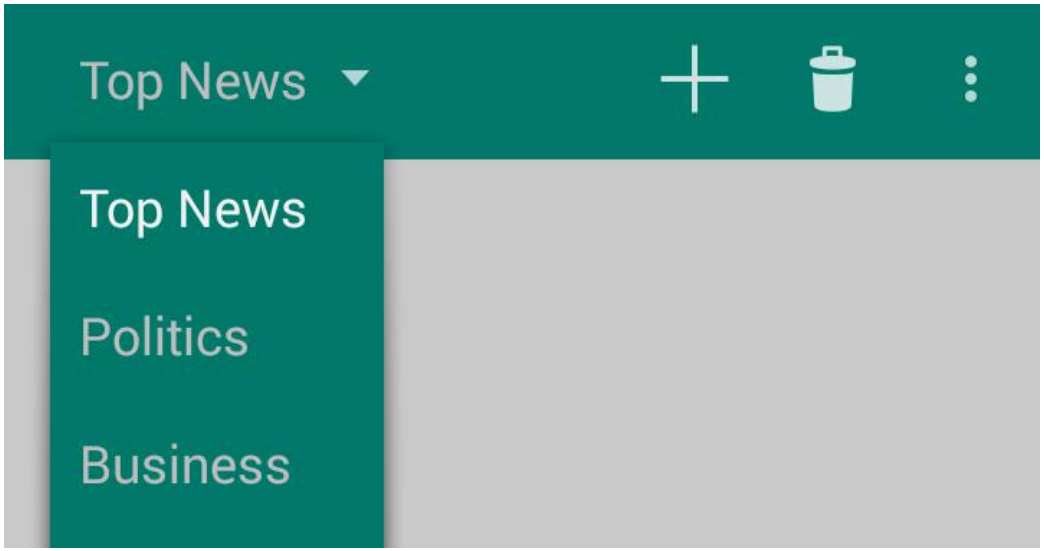
# Grille


GridView Example
Alyssum
Daisy
Jasmine
Lily

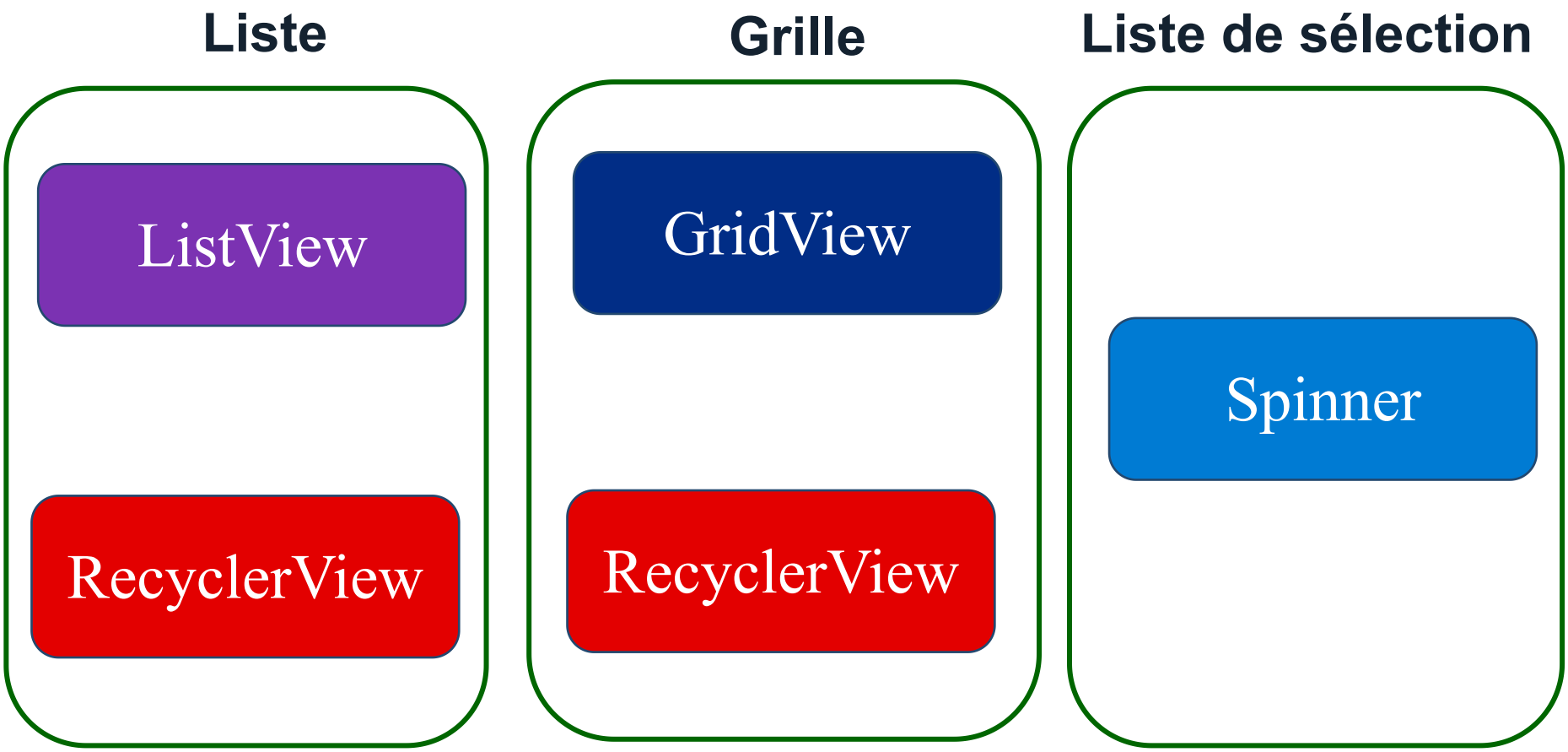

6:56
Recently Added

# Liste de selection

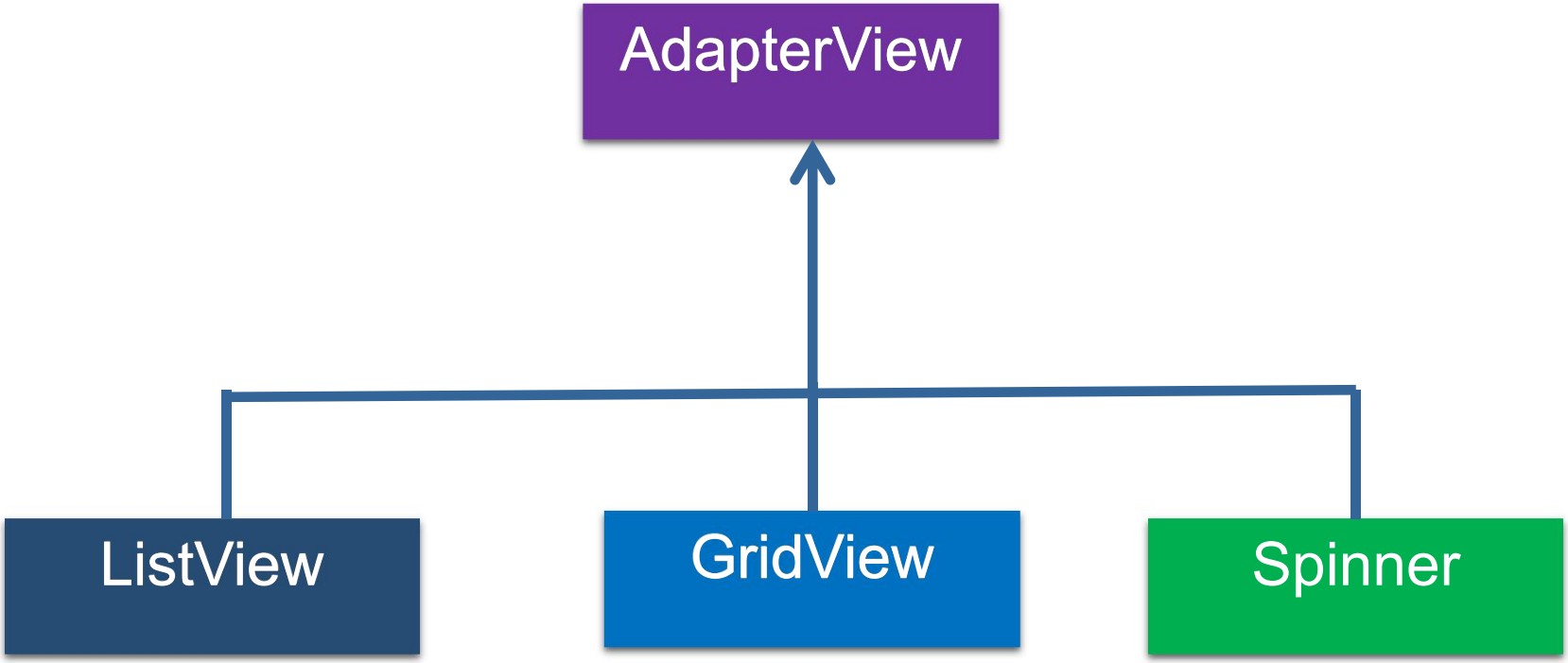


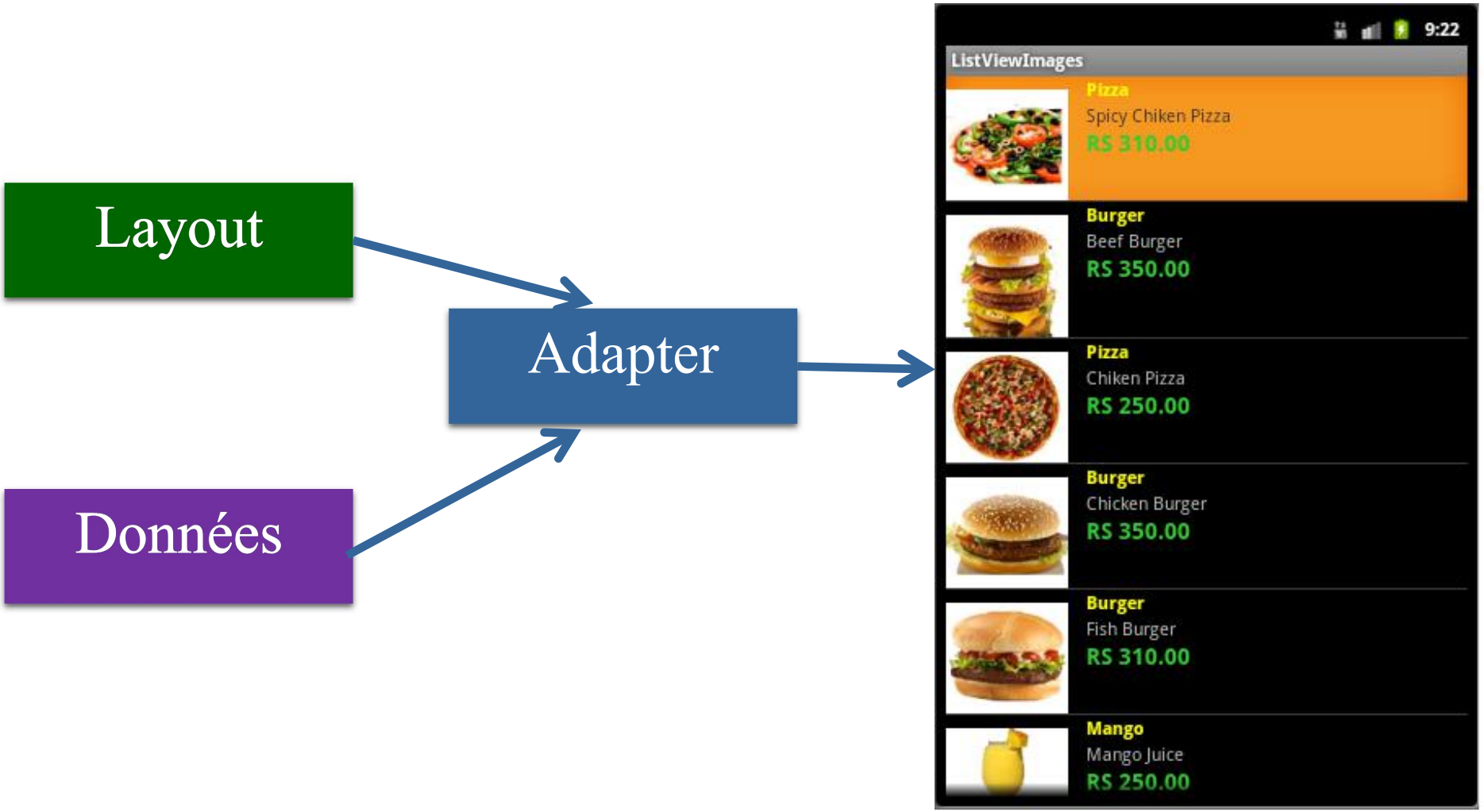
# Les vues
# Implémentation



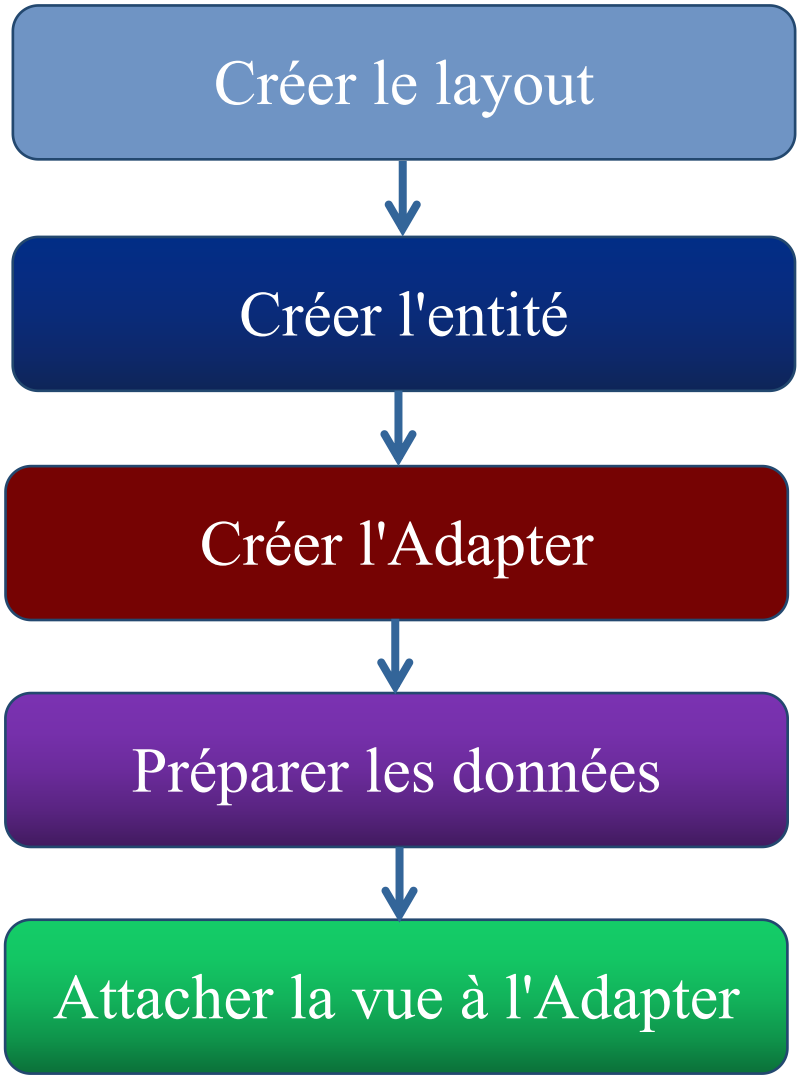
# Les vues
# AdapterView



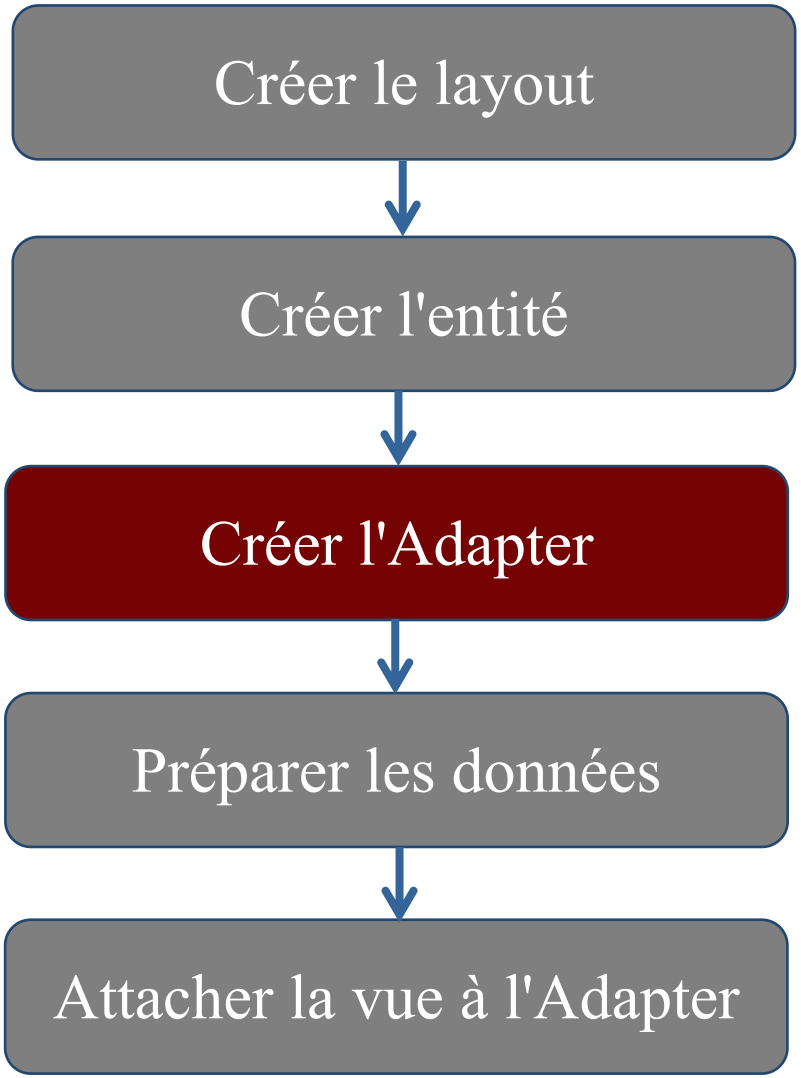
# AdapterView
# Architecture d'implémentation



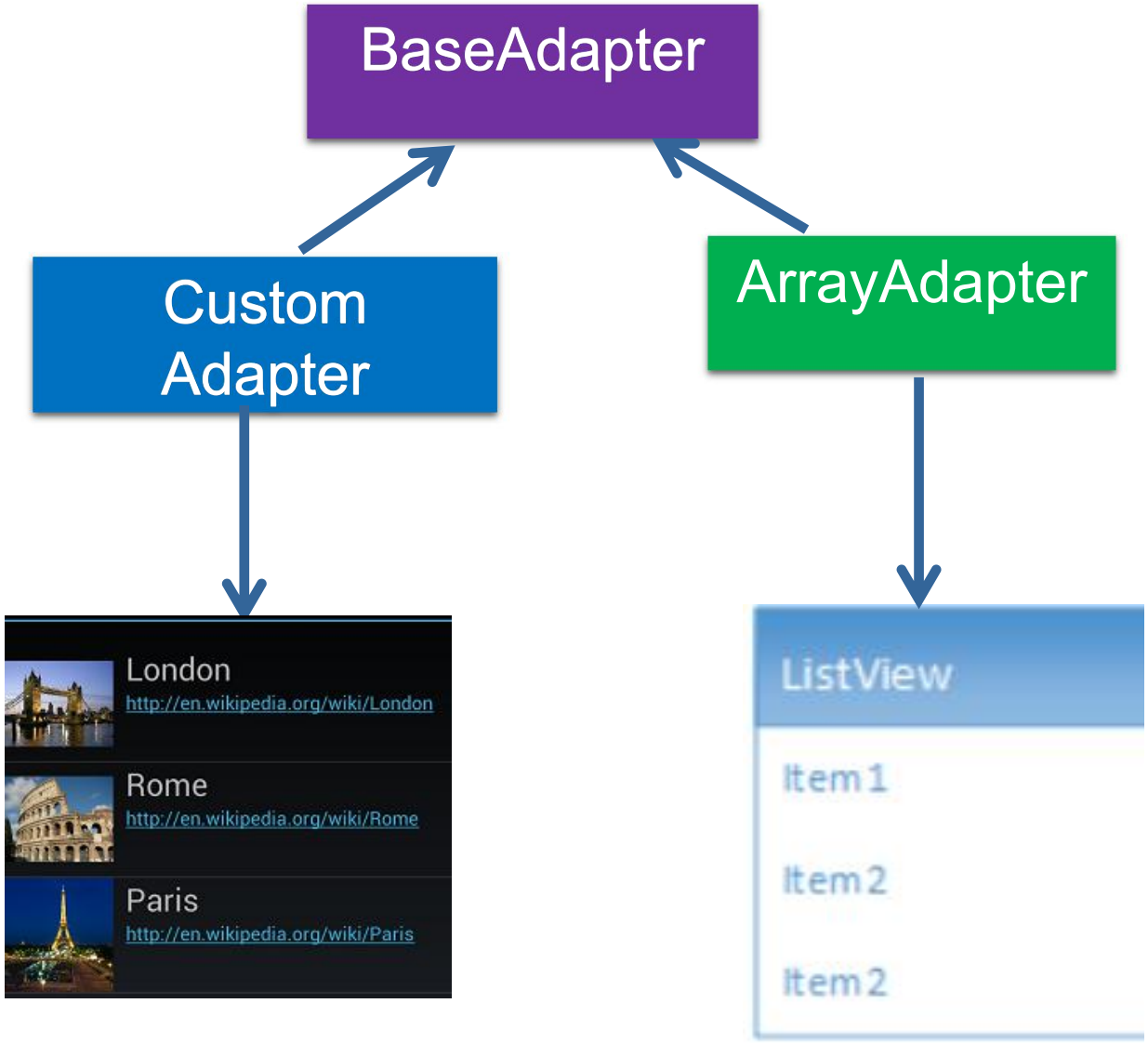
# AdapterView
# Porcessus d'implémentation



# AdapterView
# Porcessus d'implémentation



# AdapterView
# Création de l'adapter



# AdapterView
# Création de l'adapter

**Quatre méthodes à redéfinir**

- *getItem(i: Int)*. Retourne l'élément de la position i.
- *getItemId(i: Int)*. Retourne l'identiant de l'élément à la position i.
- *getCount()*. Retourne la taille de la liste.
- *getView(i: Int, p0: View?, parent: ViewGroup?)*.
  - Retourne la vue à afficher.
  - Affiche la vue sur la liste avec les données.

# Références


1. https://developer.android.com/guide/topics/resources/accessing-resources.html
2. https://material.io/guidelines/layout/units-measurements.html#units-measurements-density-independent-pixels-dp
3. http://tutos-android-france.com/images-vectorielles/
4. https://developer.android.com/training/constraint-layout/index.html
5. http://developer.android.com/guide/topics/ui/layout/listview.html