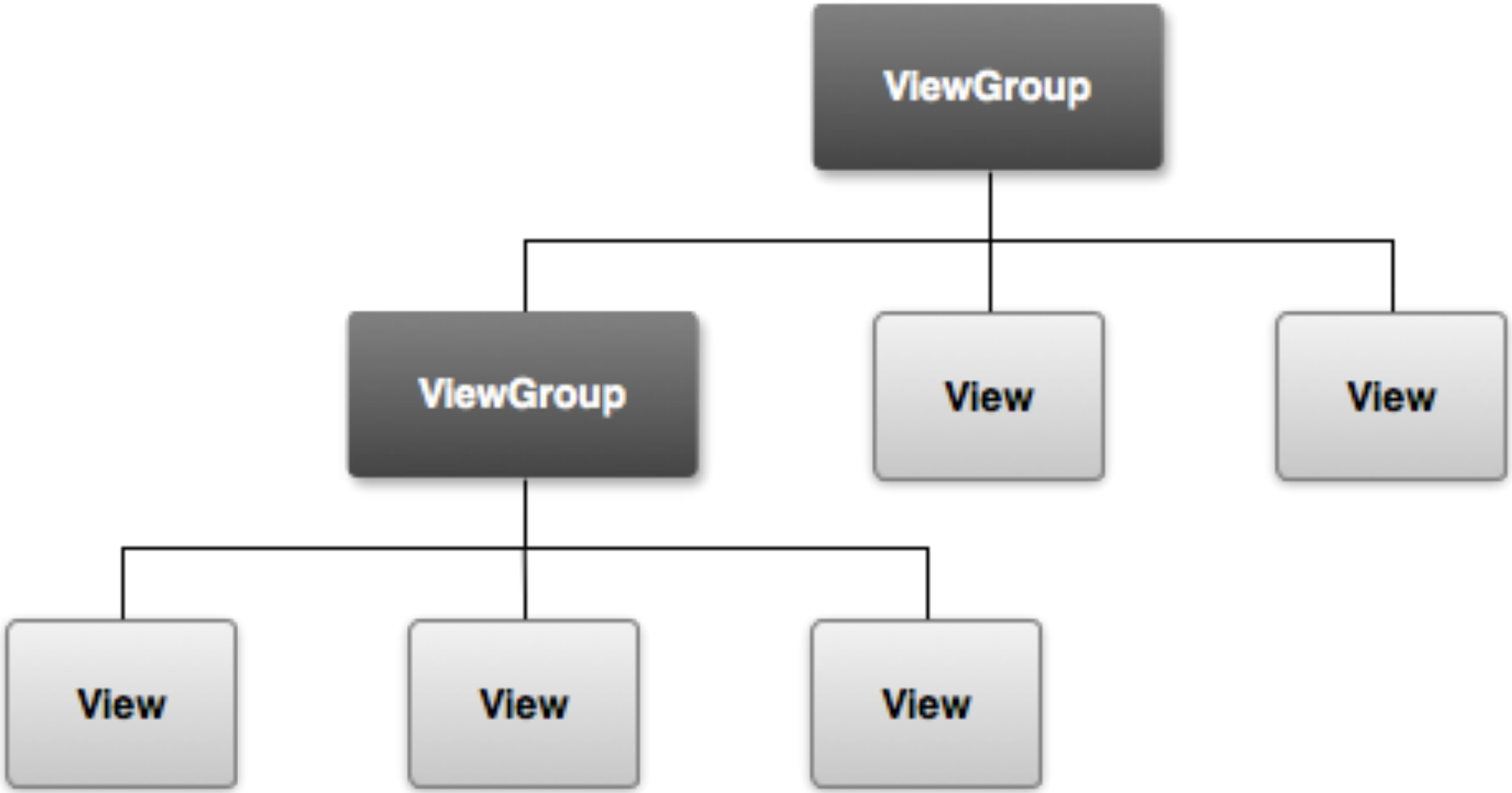
LAYOUT

Benoît Paques

View vs ViewGroup

- Toutes les interfaces utilisent des View et des ViewGroup
- Une View permet de dessiner un élément avec lequel l'utilisateur peut intéragir
- Un ViewGroup est un objet qui détient d'autres View ou ViewGroup pour définir leur layout
- Ils sont définis dans des fichiers xml dans le dossier layout
- Le nom de la balise xml est le nom de la classe qu'il représente
  - Ex : un élément <TextView> crée un widget TextView dans notre UI

Arborescence



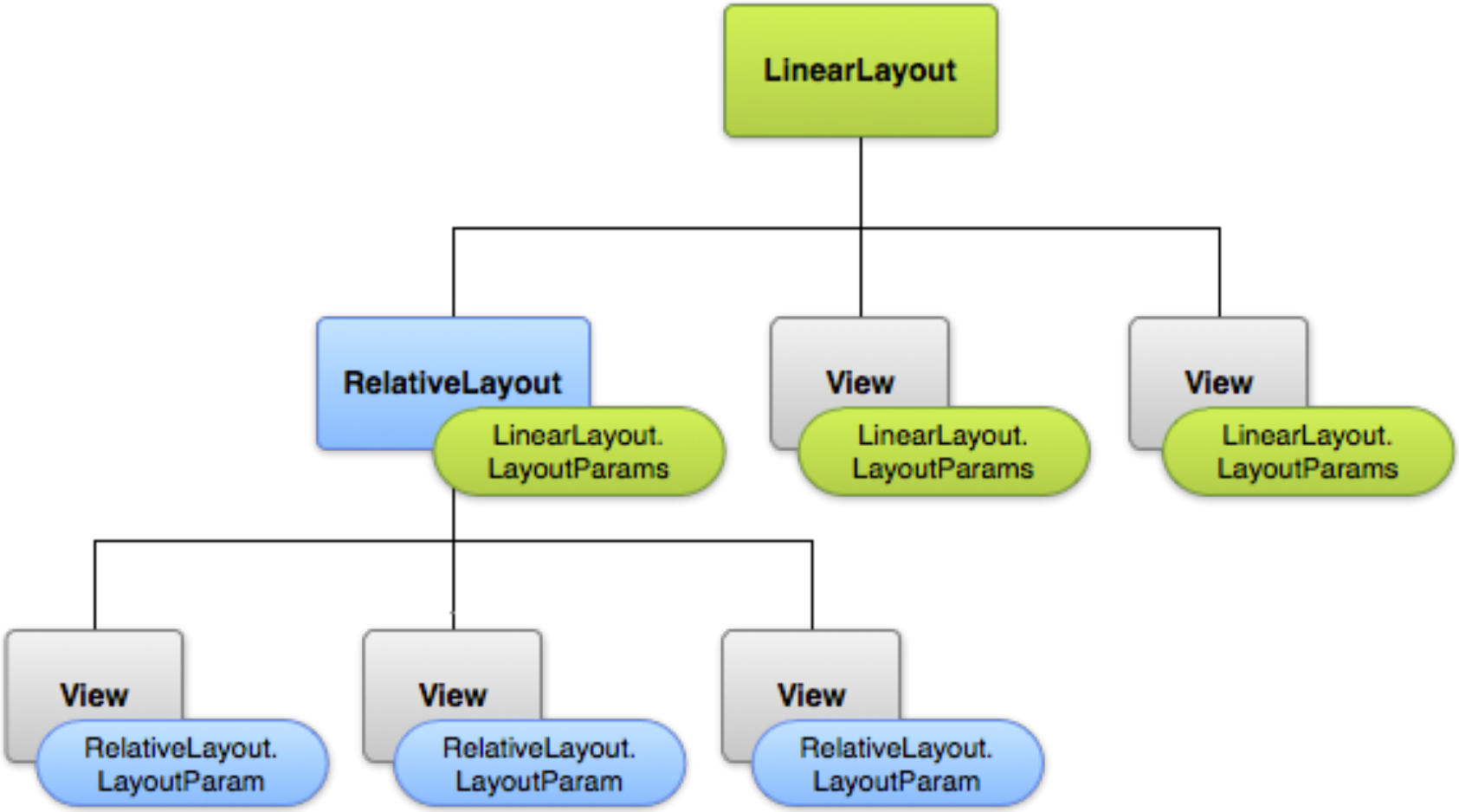
Création de l'interface

- Les View peuvent être créer de deux manières :
  - Déclaration dans le xml
  - Procéduralement
- Avantage de la déclaration en xml :
  - Séparation entre la définition des écrans et le code

Paramètre de Layout

- Les attributs de xml commençant par layout_ définissent des paramètres pour la View qui seront interprétés par le ViewGroup dans lequel il est.
- Tous les ViewGroup implémentent une classe qui étend ViewGroup.LayoutParams
  - Cette sous classe définit la taille et la position de toutes les View enfants
- Tous les ViewGroup ont les attributs layout_width et layout_height qui peuvent prendre deux valeurs
  - wrap_content définit sa taille par rapport au contenu de la View
  - match_parent la View sera aussi grande que son ViewGroup parent

Paramètres de layout



Padding et Margin

Layout courant

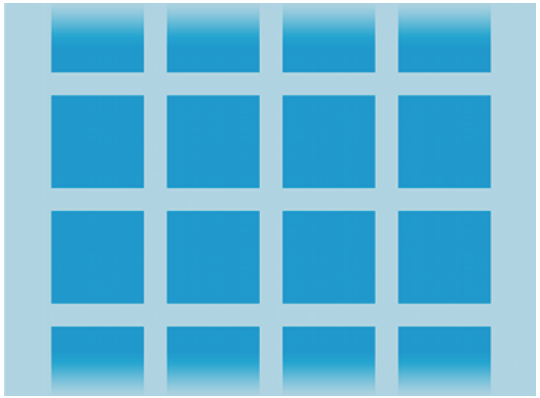
- LinearLayout
  - Organise ses enfants en une ligne horizontale ou vertical
  - android:orientation, android:layout_weight

- RelativeLayout
  - Permet de spécifier la position relative des enfants par rapport aux autres ou à la vue parente
  - android:layout_alignParentTop, android:layout_centerVertical, android:layout_below, android:layout_toRightOf

Layout courant

- ListView
  - Définit une liste scrollable d'objet
  - Remplit par un Adapter

- GridView
  - Listview à colonne
  - Remplit par un Adapter



Animations

- Définit dans le xml dans le dossier res/anim
- Le fichier doit avoir une racine <alpha>, <scale>, <translate>, <rotate> or <set>
- Pour avoir une séquence d'animation il faut spécifier un startOffset

Animation xml

```
<set android:shareInterpolator="false">
    <scale
        android:interpolator="@android:anim/accelerate_decelerate_interpolator"
        android:fromXScale="1.0"
        android:toXScale="1.4"
        android:fromYScale="1.0"
        android:toYScale="0.6"
        android:pivotX="50%"
        android:pivotY="50%"
        android:fillAfter="false"
        android:duration="700" />
    <set android:interpolator="@android:anim/decelerate_interpolator">
        <scale
           android:fromXScale="1.4"
           android:toXScale="0.0"
           android:fromYScale="0.6"
           android:toYScale="0.0"
           android:pivotX="50%"
           android:pivotY="50%"
           android:startOffset="700"
           android:duration="400"
           android:fillBefore="false" />
        <rotate
           android:fromDegrees="0"
           android:toDegrees="-45"
           android:toYScale="0.0"
           android:pivotX="50%"
           android:pivotY="50%"
           android:startOffset="700"
           android:duration="400" />
    </set>
</set>
```

Animations lancement

- Procédure
  - Récupération de la vue à animer
  - Chargement de l'animation depuis les ressources
  - Lancement de l'animation

```
ImageView spaceshipImage = (ImageView) findViewById(R.id.spaceshipImage);
Animation hyperspaceJumpAnimation = AnimationUtils.loadAnimation(this, R.anim.hyperspace_jump);
spaceshipImage.startAnimation(hyperspaceJumpAnimation);
```

TP #7

- Création d'une Activity affichant
  - Une TextView dont le contenu apparaît dans un fade-in
  - 3 boutons sur une ligne horizontale chacun lançant une animation différente sur la TextView ci dessus (scale, rotate, translate)
  - Positionnement d'un footer contenant une image

TP #7 - notation

- 1 point pour le fade-in de la TextView
- 3 points pour les animations lancées par les boutons
- 1 point pour le footer bien positionné