Temari

1) Introducció
2) Framework Android
3) Projectes Android i Android SDK
4) Activity
5) Fragments, Views i ListViews
6) Intents
7) Layouts i Custom Views
8) Resources i Themes
9) Dialogs, Menus i WebView
10) Persistència de dades
11) Tasques en Background i internet
12) SQLite i content providers
13) Notificacions

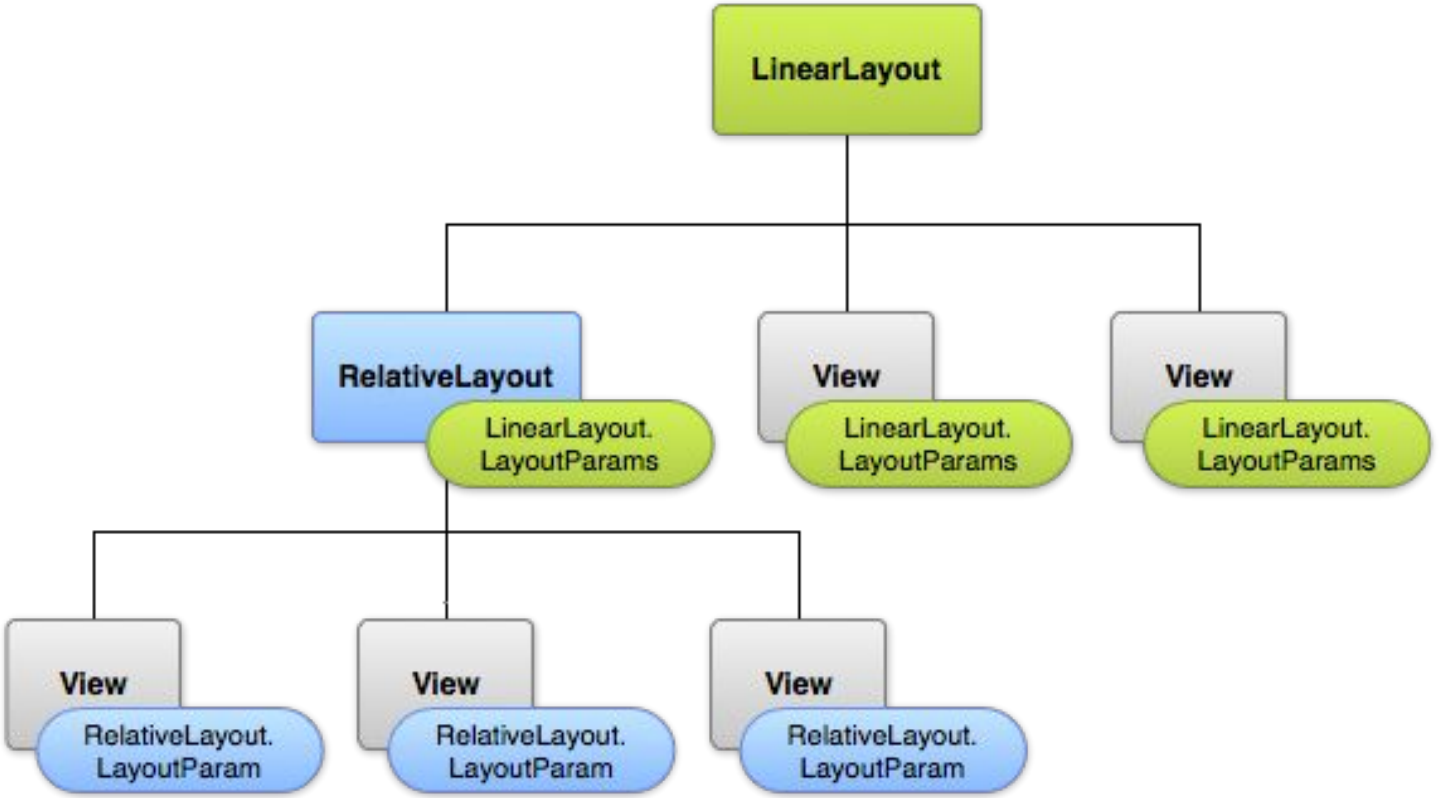
5 – Layouts i Custom Views

- Layouts disponibles
- Propietats dels Layouts
- Views pròpies
- Modificar Views existents

Layouts

Estructuren un grup de Views en una porció de la UI.

Diferents layouts per estructures diferents



5 – Layouts i Custom Views

Layouts

- **LinearLayout**
- **RelativeLayout**
- **TableLayout**
- **FrameLayout**
- **GridLayout**
- **DrawerLayout**
- **CoordinatorLayout**
- **...**

Totes subclass de **ViewGroup**

Layout Properties

Qualsevol ViewGroup conte una *nested class* del tipus **ViewGroup.LayoutParams**, que defineixen com es veuen les Views que conte

- Es **obligatori** definir un layout_**width** i un layout_**height,** que pot ser**:**
  - Un valor exacte, en **dp**
  - **match_parent**
  - **wrap_content**
- Altres propietats:
  - layout_margin[Botton,Top,Right,Left]
  - layout_gravity → com es posicionen les Views que conte
  - layout_weight → Com es distribueix l'espai sobrant del layout
  - layout_x i layout_y: les coordenades del layout

Inflar amb un layout

- **A l'Activity** → setContentView(int resourceId)
- **En un Fragment** → inflater.inflate(int resourceId, ViewGroup container, boolean)

Custom Views

A part de les Views que per defecte proporciona Android, si tenim una necessitat especial en podem fer de **Custom.** Hi ha 3 vies:

- **Modificar una View existent**: Les Views existents tenen paràmetres per configurar aspectes: el color, la mida, la direcció...
- **Combinar vàries Views en una nova**: es poden combinar vàries Views en una de nova que les agrupi totes, i llavors fer servir aquesta coma View
- **Fer una View completament nova**: Crear una View extenent la classe original: android.view.View

Modificar una View existent

Provem-ho amb un TextView

- Fem un **extends** de la View que volem modificar
- Creem els constructors (cridant al de la superclass!)
  - Hem de passar-li com a minim el **Context** i el **AttributeSet** → així apareix al layout editor
- Sobreescriure metodes:
  - **Init():** Inicialitzem els objectes Paint que usarem
  - **onDraw(Canvas canvas)**: Utilitzant els objectes Paint, pintem sobre el canvas formes, text, imatges

Canvas: **Quina forma es pinta**

Paint: **Com es pinta**

Compound Views

- **Definir atributs opcionals** en un fitxer de recursos res/values/attrs.xml → els atributs que es podran configurar de la nova View

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <declare-styleable name="Options">
        <attr name="titleText" format="string" localization="suggested" />
        <attr name="valueColor" format="color" />
    </declare-styleable>

</resources>
```


Compound Views (I)

- Crear un XML que combini les diferents Views en un element *<merge>*
- Crear la classe corresponent → extends un Layout
- Al codi, sobreescriure el constructor que se li passa **Context** i **AttributeSet**
    - Obtenir els valors de les configuracions en un **TypedArray** del Context

```
TypedArray a = context.obtainStyledAttributes(attrs,
    R.styleable.ColorOptionsView, 0, 0);
String titleText = a.getString(R.styleable.ColorOptionsView_titleText);
int valueColor = a.getColor(R.styleable.ColorOptionsView_valueColor,
    android.R.color.holo_blue_light);
a.recycle();
```

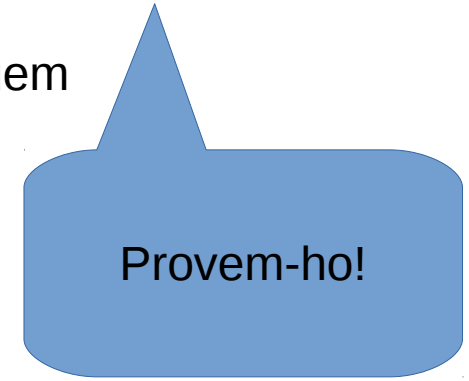

Compound Views (II)

- Inflar la Custom View amb el XML que combinava les altres Views

```
LayoutInflater inflater = (LayoutInflater) context
    .getSystemService(Context.LAYOUT_INFLATER_SERVICE);
inflater.inflate(R.layout.view_color_options, this, true);

TextView title = (TextView) getChildAt(0);
title.setText(titleText);
```

- Modificar les Views com vulguem



View completament nova (I)

- Crear una nova class que extengui **android.view.View**
- Crear nous recursos **declare-styleable** en el fitxer */res/values/attrs.xml*

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <declare-styleable name="Options">
        <attr name="titleText" format="string" localization="suggested" />
        <attr name="valueColor" format="color" />
    </declare-styleable>

</resources>
```

- Declarar variables de tipus Paint que necessitarem per pintar

View completament nova (II)

- Definir un constructor: **public MyView(Context ctx, AttributeSet attrs)**
    - Obtenir els atributs del XML del **TypedArray** que refereixen al *attrs.xml*

```
public PieChart(Context context, AttributeSet attrs) {
   super(context, attrs);
   TypedArray a = context.getTheme().obtainStyledAttributes(
        attrs,
        R.styleable.PieChart,
        0, 0);

   try {
       mShowText = a.getBoolean(R.styleable.PieChart_showText, false);
       mTextPos = a.getInteger(R.styleable.PieChart_labelPosition, 0);
   } finally {
       a.recycle();
   }
}
```


View completament nova (III)

Inicialitzar els Paint al mètode **init**()

Sobreescriure el **onDraw**()

```
protected void onDraw(Canvas canvas) {
    super.onDraw(canvas);

    // Draw the shadow
    canvas.drawOval(
            mShadowBounds,
            mShadowPaint
    );

    // Draw the label text
    canvas.drawText(mData.get(mCurrentItem).mLabel, mTextX, mTextY, mTextPaint);
```


View completament nova (IV)

- Implementar els getters i setters
    - Després dels setters cal cridar al **invalidate() i requestLayout**()