Temari

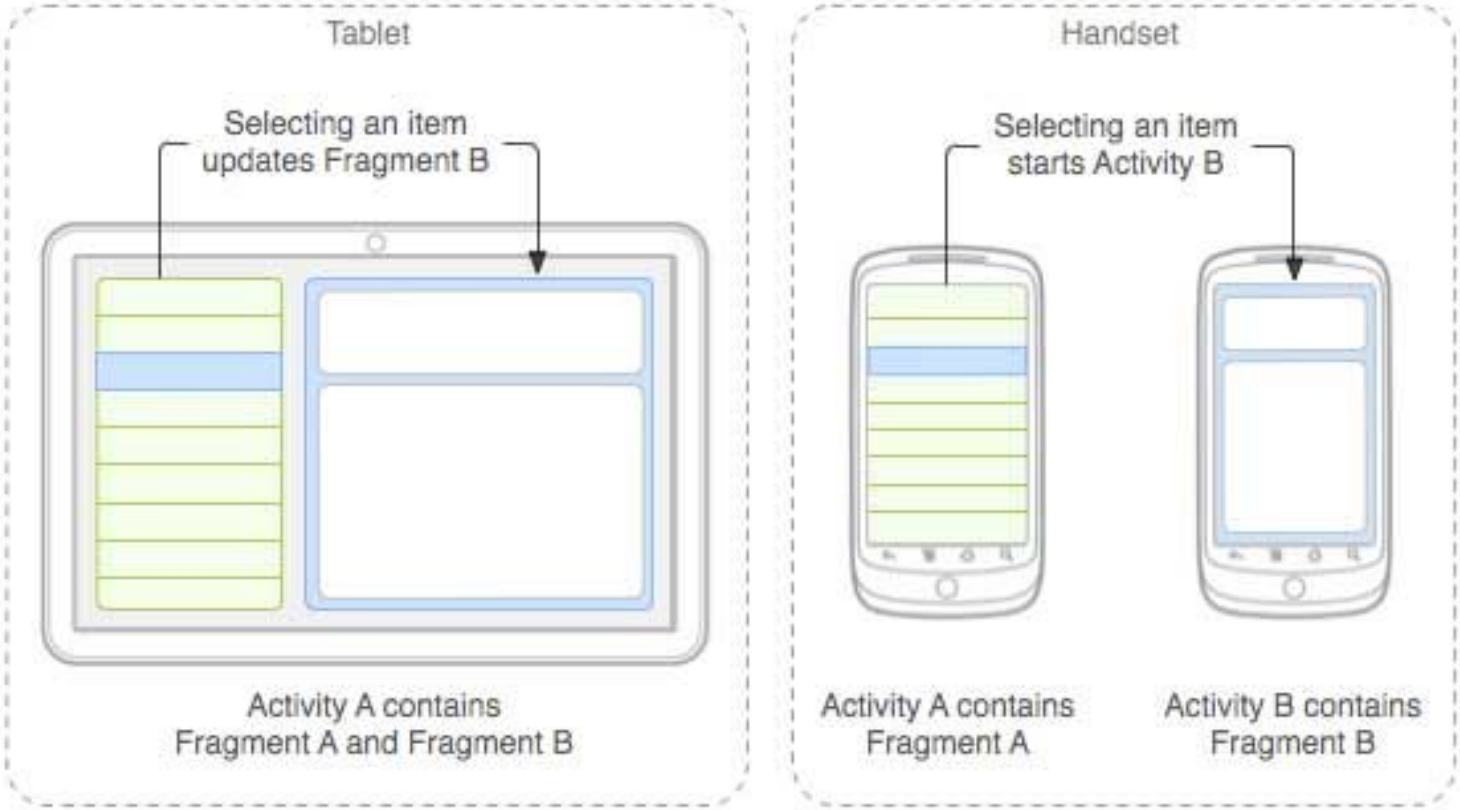
1) Introducción
2) Framework Android
3) Proyectos Android y Android SDK
4) Activity
5) Fragments, Views y ListViews
6) Intents
7) Layouts y Custom Views
8) Resources y Themes
9) Dialogs, Menus y WebView
10) Persisténcia de datos
11) Tareas en Background e internet
12) SQLite y content providers
13) Notificaciones

5 – Fragments, Views i Listviews

- Fragments
- Views
- ListView i ListActivity

Fragments, qué son?

Aparecen en Android 3.0.
Representan una **porción de la UI de una Activity**, se pueden **reusar** en más de una activity, y aparecer o no **en función de características**



Fragments

- Una Activity puede tener más de un Fragment
- Los Fragments se pueden añadir o sacar de la activity, bien en el Layout (XML) o dinámicamente en el codigo Java
- Tienen un ciclo de vida propio, con callbacks propias, però ligadas a las de la Activity.
- Un mismo Fragment se puede usar en varias Activities
- Si se quieren tener diferentes UI segun características de pantalla (medidas, landscape-portrait...) se necesitan fragments
- Hay que sobreescribir la classe **Fragment**

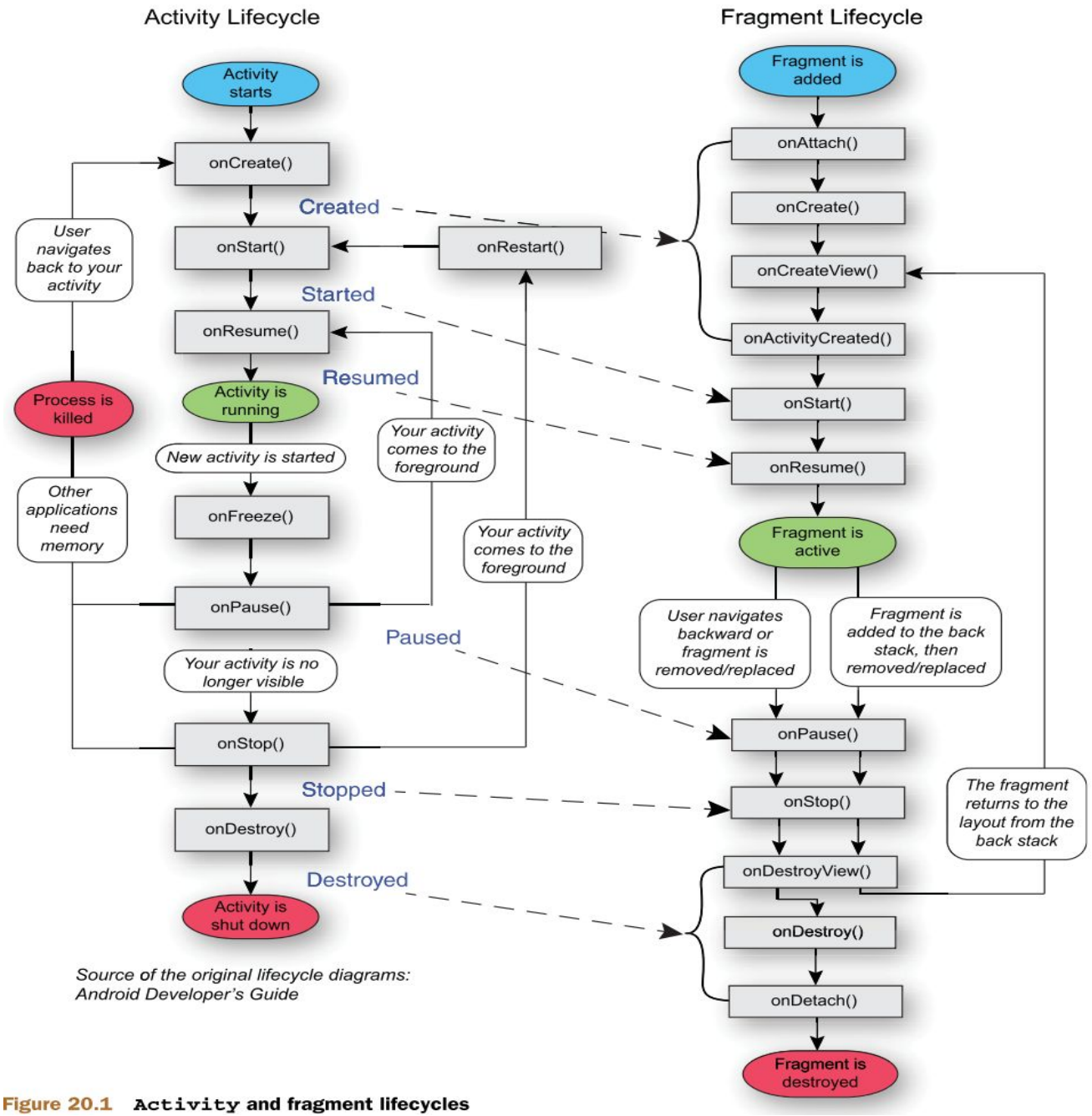


Figure 20.1 **`Activity` and fragment lifecycles**

Fragments, Ciclo de vida

- **OnCreateView()** → inicialitzar la UI → inflarla con un layout
- **OnCreate()** inicialitzar componentes y recuperar datos guardados cuando se pause el fragment (pasue/stop)
- **OnAttach()** se ejecuta cuando se añade el fragment a una activity. Mediante una **interface** java permite comunicacarse con la activity
- **OnPause()** Se ejecuta cuando se pausa el fragment (porque se ha pausado la activity, o porque se ha desvinculado el fragment de la activity). Persistir los datos aquí!

Fragments, añadidos en el layout

- Usar el tag <fragment> en el layout de la activity
- Especificar la *class* del Fragment en el elemento XML
- Ponerle un ID para referenciarlo desde Java

```
<fragment
        android:id="@+id/listFragment"
        android:layout_width="0dp"
        android:layout_weight="1"
        android:layout_height="match_parent"
        class="com.example.android.rssreader.MyListFragment"
></fragment>
```


Fragments, programáticamente

- Obtener un **FragmentManager:**

  FragmentManager fm = getFragmentManager();

- Iniciar una transacción:

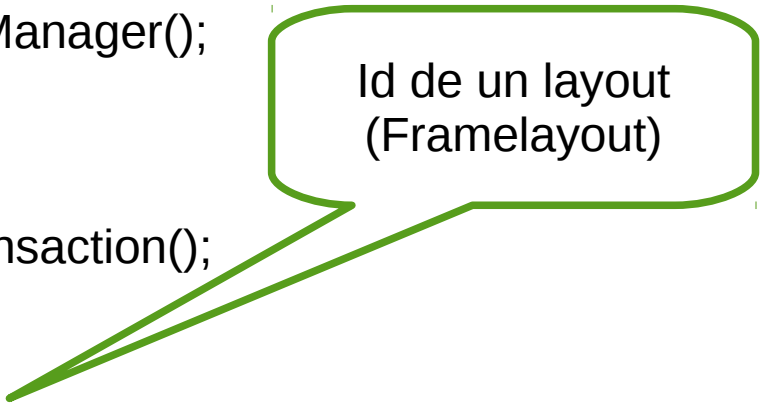
  FragmentTransaction ft = fm.beginTransaction();

- Realizar las modificaciones:

  ft.add(R.id.your_placehodler, new YourFragment());
  ft.replace(R.id.your_placehodler, new YourFragment());
  Fragment fragment =
  fm.findFragmentById(R.id.your_placehodler);
  ft.remove(fragment);

- Hacer commit:

  ft.commit();



Fragments, provémoslo

- Hacer una app que:
  - use 3 fragments, cada uno de ellos con un botón
  - Para disposición *portrait,* se muestra **sólo el primer fragment**, para *landscape,* los dos primeros
  - En portrait, al clicar el botón, abrir el siguiente fragment. Al ir atrás volver a la anterior. En landscape, el 3^{r} fragment se abre en lugar del segundo

Views

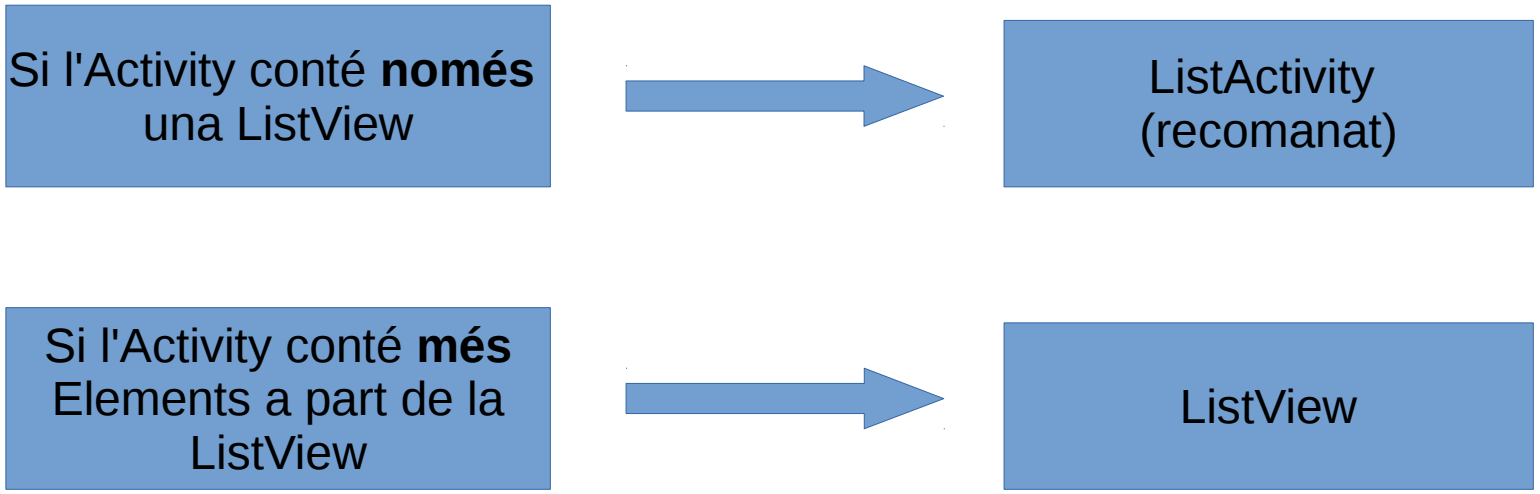
- Es la *parent class* de todos los elementos interactivos de la UI
- Es responsable de **visualizarse** i gestionar **los eventos**
- **ViewGroup** es una subclasse, y es un contenedor para otras Views
- Se definen a través de XML o directamente en código
- Configuraremos:
    - **Propiedades**: según la view, tendrá unas u otras
    - **Focus**: Si se quiere que inicie el *user input:* **setFocus()**
    - **Listeners:** objetos que contienen el método o métodos a ejecutar cuando pasa un evento
    - **Visibilidad**: controlar con ***setVisibility(int)***

Views

- Hemos visto ya algunas
  - Button
  - ToggleButton
  - Checkbox
  - ImageView
  - ...
  - LinearLayout
  - RelativeLayout
  - ...

ListView i ListActivity

- ListView és una View pensada per a mostrar llistats verticals d'elements
- Dues opcions per a mostrar una ListView:



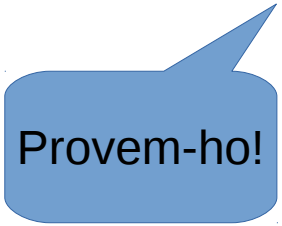
ListActivity

- Subclass de Activity, cuya UI es una ListView
- Para usar una ListActivity:
  - Crear una class que **extends** ListActivity

    public class MyListAct extends ListActivity{ ...}
  - Crear un array de Strings (o List<String>) con el contenido

    final String[] ANDORID_OS={''Cupcake'', ''Donut'',...};
  - Crear un ArrayAdapter en el onCreate()

    setListAdapter(new ArrayAdapter<String>(this, android.R.layout.simple_list_item1, ANDROID_OS));

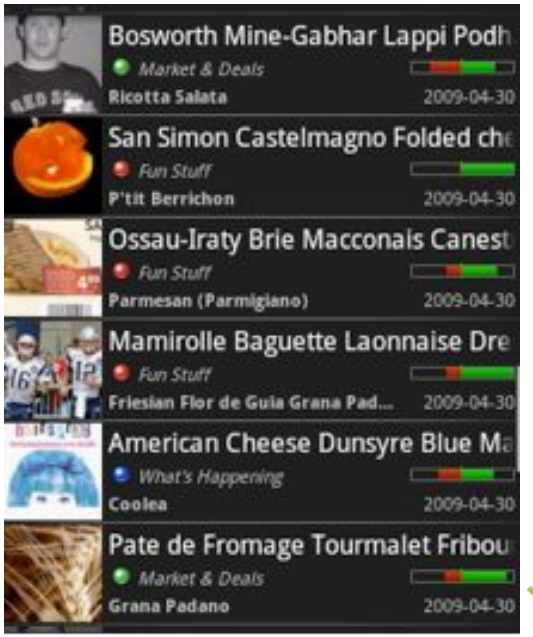


Adapters

- Classes del SDK que conectan datos con la presentación
- Existen varias por defecto:
  - ArrayAdapter: adapta un array o List
  - CursorAdapter: adapta un cursor (acceso a BDD o a ContentProvider)
  - Una creada por nosotros **extends** BaseAdapter
- Especifica cuántos elementos hay, el elemento en cada posición, qué layout se usa para cada elemento de la lista...
- Si los datos cambian (añadir o sacar elementos del array, o de la BDD...) → **notifyDataSetChanged()**

ListView

- Widget dentro de Layout complejo
- Crear un Layout que será el que se usará para cada uno de los items de la lista
- Hay que añadir un **adapter** (per exemple un ArrayAdapter), que liga:
  - Elementos a mostrar
  - El layout a aplicar a cada elemento
  - La activity



Eventos en una ListView

- Evento de click sobre un elemento

  listView.setOnItemClickListener(...)

- Event de click largo sobre un elemento:

  listView.setOnItemLongClickListener(...)