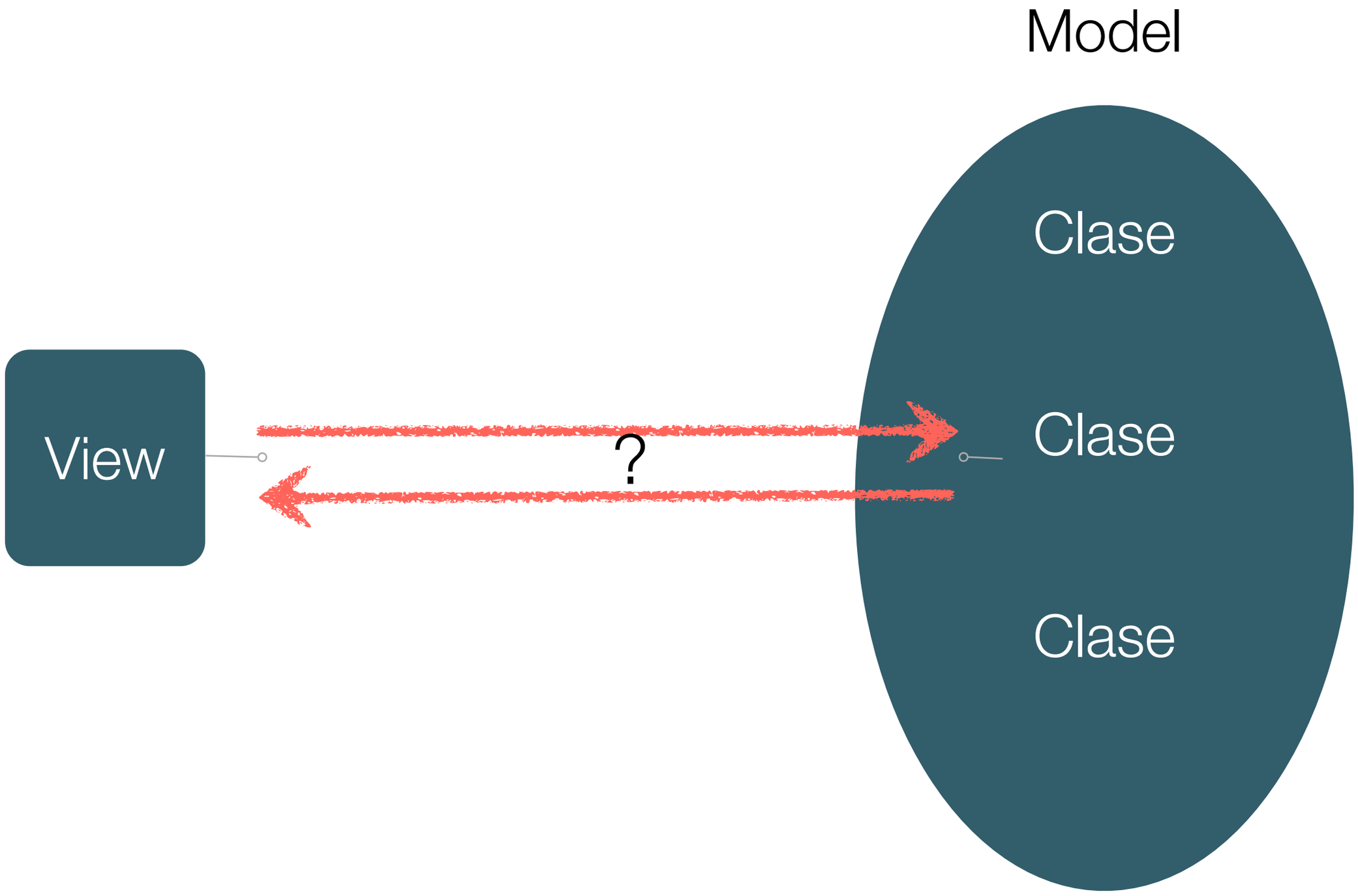
Introducción a Android

Sesión 4

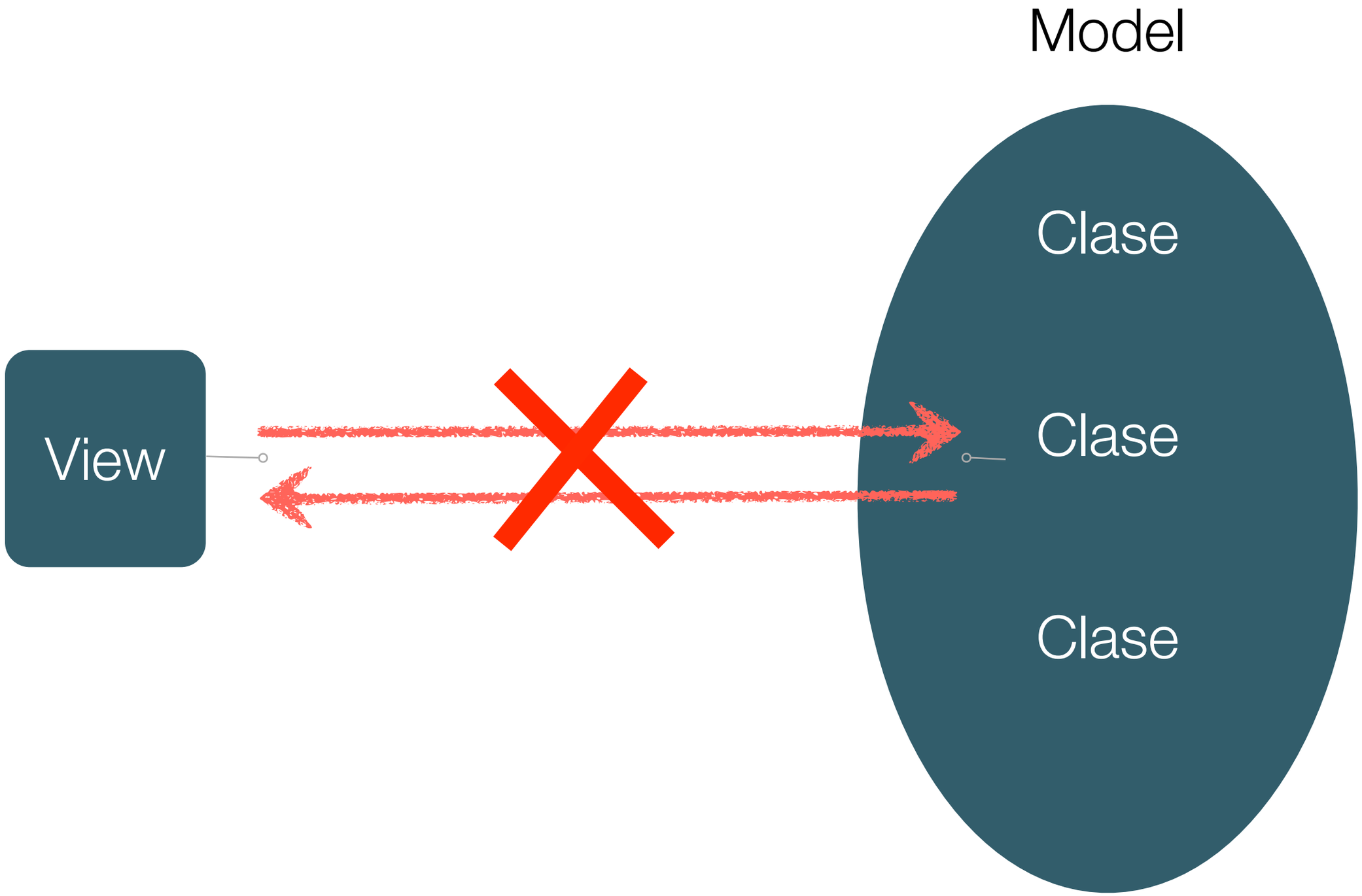
Qué vamos a ver el día de hoy?

- Combos (spinners), listas, gridviews -> patrón Adapter
- Distribución de aplicaciones
  - Ofuscación de aplicaciones
  - Firmado de aplicaciones
  - Android Market

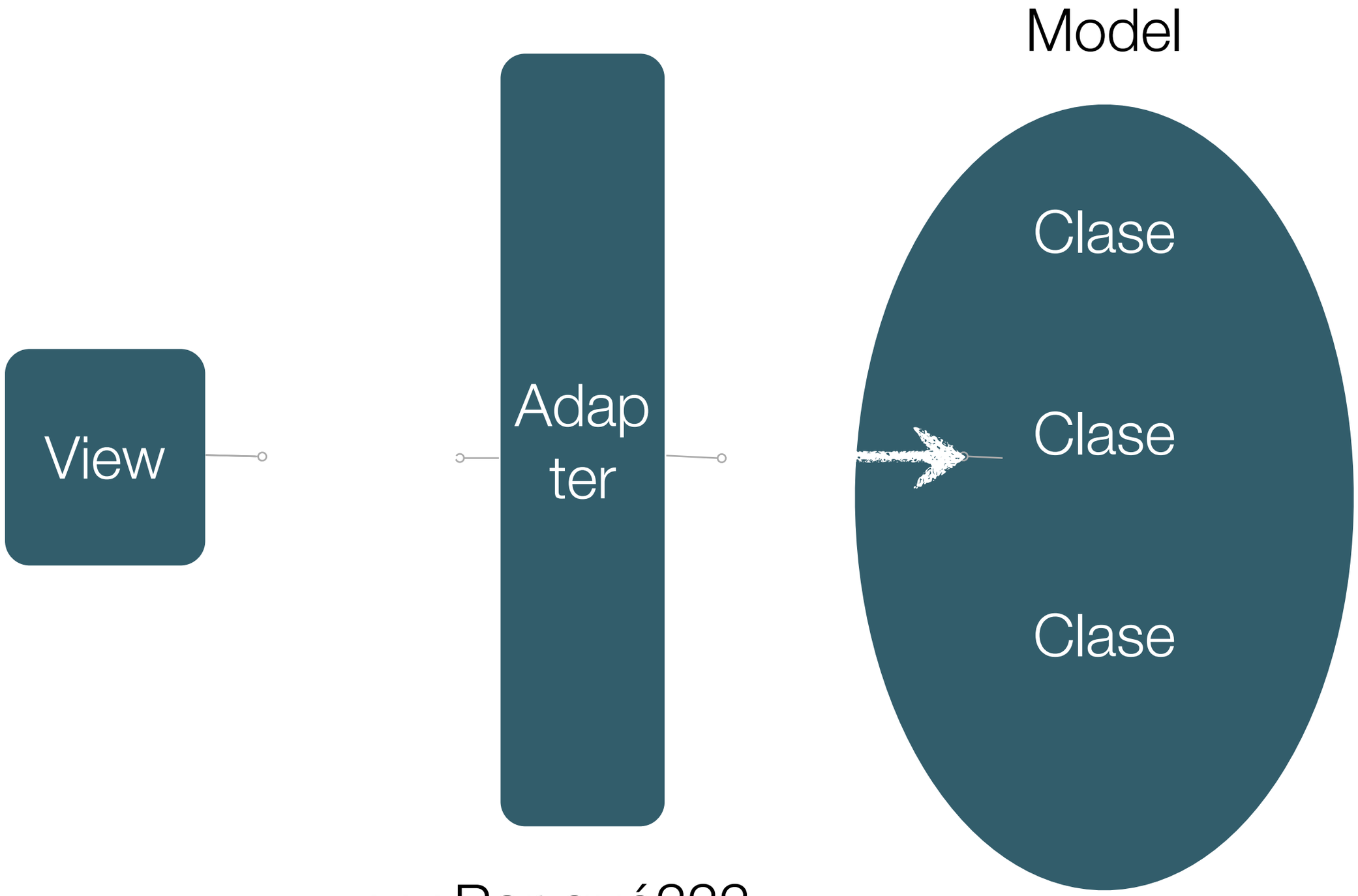
Patrón Adapter



Patrón Adapter



Patrón Adapter



¿¿¿Por qué???

Componentes de listas

- Spinner
- ListView
- GridView

Spinner

- Selector de opciones (combo) que te permite elegir solamente una a la vez.
- Pertenece a la capa View por lo que NO se puede comunicar con el model (ni viceversa).
- Tres partes:
  - Definición de presentación (cómo se verá cada elemento de la lista)
  - Obtención de los datos
    - Datos constantes? -> Listado de strings
    - Datos dinámicos? -> Clase Adapter
  - Administración desde el código (Activity).

Formas de obtener datos

- Por medio de un arreglo de constantes
  - Opciones fijas, no cambian en tiempo de ejecución
- Por medio de un arreglo obtenido en tiempo de ejecución
  - Opciones dinámicas
  - Pueden provenir de un servicio de red, una base de datos, etc.

Caso 1: Arreglo de constantes

- Se obtiene como un recurso del archivo strings.xml

```
<resources>
    <string name="app_name">User Application</string>
    <string-array name="numero cuotas">
        <item>Una</item>
        <item>Dos</item>
        <item>Tres</item>
        <item>Cuatro</item>
    </string-array>
</resources>
```

Caso 2: Datos dinámicos

- Listas obtenidas en tiempo de ejecución (`List` o arreglos)

- `CursorAdapter`: para datos obtenidos de SQLite o Content Providers

Presentación

- Presentación simple / por defecto: `ArrayAdapter` y layouts predefinidos

- Presentación personalizada: dos partes
  - Definir nuestro propio layout para los ítems
  - Definir nuestra propia implementación de `Adapter`

`ArrayAdapter`

- Se encarga de proporcionar la data necesaria al View.
- `createFromResource` permite crear directamente desde un recurso `<string-array>`
- Permite configurar la apariencia del listado y del item seleccionado

```
Spinner spinner = (Spinner) findViewById(R.id.spinner);
ArrayAdapter<CharSequence> adapter =
    ArrayAdapter.createFromResource(this, R.array.planets_array,
            android.R.layout.simple_spinner_item);
adapter.setDropDownViewResource(
            android.R.layout.simple_spinner_dropdown_item);
spinner.setAdapter(adapter);
```

Presentación personalizada

- Definimos nuestro layout con la apariencia que tendrá cada ítem del `Spinner`

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent">
    <ImageView
      android:layout_height="wrap_content"
      android:layout_width="wrap_content"
      android:src="@drawable/icon"
      android:id="@+id/iviLogoTarjeta"/>
    <TextView
      android:text="TextView"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:id="@+id/tviNombreTarjeta"/>
</LinearLayout>
```

- Creamos nuestro propio Adapter

```
public class ListTarjetasAdapter extends BaseAdapter
        implements Filterable{

    @Override
    public int getCount() {
        // Calcular y devolver la cantidad de elementos del listado
    }

    @Override
    public Object getItem(int position) {
        // Buscar y devolver el ítem que se mostrará en la posición indicada
    }

    @Override
    public long getItemId(int position) {
        // Devolver un número identificador para el ítem en la posición
indicada
    }

    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        // Aquí creamos la vista que se mostrará por cada elemento
        // La creamos a partir de un layout y la llenamos con la data
correspondiente
        // Para hacerlo más eficiente, podemos reutilizar las vistas
    }
}
```

Administración desde el código (Activity)

- Creamos una nueva instancia de nuestro Adapter y lo asignamos al Spinner

```
List arrTarjetas = cargaComboTarjetas();
spiTarjetas =
     (Spinner)findViewById(R.id.spiTarjetas);
ListTarjetasAdapter adapter =
     new ListTarjetasAdapter(this, arrTarjetas);
spiTarjetas.setAdapter(adapter);
```

ListView

- Otro widget de listado muy utilizado
- Preferido cuando se tiene que mostrar al usuario una lista de opciones de tamaño extenso, con opción para selección

GridView

- Widget que muestra una grilla / cuadrícula scroleable de dos dimensiones de items.

- Permite configurar número de columnas, ancho de columna y otros parámetros de presentación

- También emplea `Adapter`s para mostrar su contenido

Manejando la selección de un elemento

- Dos eventos principales:
  - toque sobre un elemento: `OnItemClickListener`
  - cambio de la selección: `OnItemSelectedListener`

```
ListView lv = (ListView) findViewById(r.id.lviCosteos)
lv.setOnItemClickListener(new OnItemClickListener() {
    public void onItemClick(AdapterView<?> parent, // ListView
            View view, // View del item seleccionado
            int position, // Posición del item seleccionado
            long id) { // Identificador (Adapter.getItemId)

    }
}
```

AutoCompleteTextView

- Para implementar autocompletado en un cuadro de texto, también se emplean `Adapters`

- Se emplea la clase `AutoCompleteTextView`:

```
AutoCompleteTextView textView =
    (AutoCompleteTextView) findViewById(R.id.autocomplete_country);
String[] countries =
    getResources().getStringArray(R.array.countries_array);
ArrayAdapter<String> adapter = new ArrayAdapter<String>(
        this, android.R.layout.simple_list_item_1, countries);
textView.setAdapter(adapter);
```

Publicando aplicaciones

Preparación

- Eliminar los logs no necesarios
- Data de ejemplo
  - Dejar data necesaria (por defecto)
  - Eliminar información delicada
- Datos finales en AndroidManifest.xml

Preparación

- Testing
  - Pruebas funcionales (regresión)
  - Probar en varios teléfonos
- Probar bajo la mayor cantidad de condiciones posibles.
  - Bajo 3G, Wifi, GPS, Network location, etc.
- Atención a cómo la aplicación responde a eventos (p. ej. cambio de landscape a portrait)

AndroidManifest.xml

- Especificar los valores de nombre de aplicación e icono.
- Especificar el android-versionCode (entero) y el android-versionName (string)
- Definir el SDK level mínimo (entero). Investigar alcance del API level utilizado.

AndroidManifest.xml

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.diplomadourp.busquedagoogle"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="19" />

    <!-- No debemos olvidar estos permisos para que la app pueda conectarse a Internet -->
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
    <uses-permission android:name="android.permission.INTERNET" />

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
```

Ofuscación

- Hacer complicado para los decompiladores obtener el código fuente
- Además sirve para comprimir código fuente
- En Android: Proguard

Firmado de aplicaciones

- Es obligatorio firmar digitalmente los .apk para subirlos al market
- Eclipse/ADT lo hace automáticamente con un debug key (para poder ejecutar aplicaciones sin problemas en los dispositivos)
- Para subir al Google Play Store, se debe firmar con un key diferente al de debug
    - Conservar el key con cuidado: no se puede actualizar la app con un .apk firmado con un key distinto

Firmado de aplicaciones

- Comandos a utilizar:
  - keytool: Creación de una llave

```
keytool -genkey -v -keystore ~/.android/releasekey.keystore -
alias releasekey -keyalg
RSA -validity 10000
```

  - jarsigner: Aplicación para la firma (utiliza la llave creada por keytool

```
jarsigner -verbose -keystore ~/.android/releasekey.keystore
MyApp-unaligned.apk releasekey
```

  - zipalign: Para alinear los bytes del .apk final y reducir la memoria utilizada por la aplicación

```
zipalign -v 4 MyApp-unaligned.apk MyApp.apk
```

Google Play Store (Ex-Android Market)

- Pago único de US$ 25
- Piden imágenes de la aplicación y un logo de un tamaño determinado
- Nos da estadísticas de la aplicación
- Ahora permite aplicaciones de hasta 50MB (y más con archivos de expansión)
- Para inscribirse y publicar: http://play.google.com/apps/publish