# 1 Drag & Drop

## 1.1 Sumario



## 1.2 Introdución

- **NOTA**: imos usar como SDK mínimo o 11.
- Se necesitamos controlar a versión o podemos facer coa clase VERSION da seguinte forma:

String androidOS = Build.VERSION.RELEASE; if (androidOS.startsWith("2")) ==> versión 2.x.x

- Con **Drag & Drop (Arrastrar e soltar)** permíteselle ao usuario que arrastre datos dende unha Vista á outra.
- O proceso divídese en:
    - O proceso comeza cando o usuario realiza algún xesto nun obxecto que reconozamos como inicio do arrastre. Por exemplo: un LongClick
    - A aplicación informa ao SO que o proceso de arrastre comezou e indícalle que elemento (View) é o que ten que debuxar mentres se arrastra, que é un elemento distinto do que iniciamos o proceso (pode ser a mesma imaxe para que pareza que movemos algo)
    - Hai que definir os obxectos Views sobre os que se vai recibir o Drop (Soltar). Cando se comeza a arrastrar un obxecto o SO manda unha mensaxe a todos os elementos gráficos da nosa pantalla que poidan recibir un drop, esperando unha contestación de se aceptarán ou non os diferentes eventos que se poidan dar durante o arrastre (que o obxecto entre dentro do View, saia, solte,....)

- **Referencias**:
    - http://developer.android.com/guide/topics/ui/drag-drop.html

## 1.3 Caso práctico sinxelo

- Comezamos creando un proxecto: **U3_20_DragDrop**
- Usaremos as seguintes imaxes no recurso /**res**/**drawable**

- O funcionamento da Activity é o seguinte:


- Ao facer un LongClick na pregunta pódese comezar a arrastrar.
- En función do botón ao que se arrastre obtemos unha mensaxe que indica a que botón se arrastrou: OK ou NO.

### 1.3.1 XML do layout

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/LinearLayoutPreguntas"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Drag &amp; Drop"
        android:layout_gravity="center"
        android:textSize="22sp" />

    <ImageView
        android:id="@+id/imgvwPreguntaArrastrar"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_gravity="center_horizontal"
        android:layout_weight="1"
        android:contentDescription="Imaxe pregunta arrastrar"
        android:src="@drawable/question" />

    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="5" >

        <ImageView
            android:id="@+id/imgVwPreguntasNo"
            android:layout_width="wrap_content"
```

```
            android:layout_height="wrap_content"
            android:layout_alignParentRight="true"
            android:layout_alignTop="@+id/imgvwPreguntasOk"
            android:contentDescription="Imaxe preguntas NO"
            android:src="@drawable/no" />

        <ImageView
            android:id="@+id/imgvwPreguntasOk"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentLeft="true"
            android:layout_centerVertical="true"
            android:contentDescription="Imaxe preguntas OK"
            android:src="@drawable/ok" />
    </RelativeLayout>

</LinearLayout>
```

### 1.3.2 Cógigo Java da Aplicación

- A continuación debullamos o código:

```
package com.example.u3_20_dragdrop;

import android.app.Activity;
import android.os.Build;
import android.os.Bundle;
import android.view.DragEvent;
import android.view.Menu;
import android.view.View;
import android.view.View.DragShadowBuilder;
import android.view.View.OnLongClickListener;
import android.view.ViewGroup;
import android.widget.ImageView;
import android.widget.Toast;

public class U3_20_DragDrop extends Activity {

@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_u3_20__drag_drop);

String androidOS = Build.VERSION.RELEASE;
if (!(androidOS.startsWith("2")) && !(androidOS.startsWith("1")))
xestionarArrastrar();

}

private void xestionarArrastrar() {
ImageView pregunta = (ImageView) findViewById(R.id.imgvwPreguntaArrastrar);

// Xestionamos cando prememos durante un tempo maior a 1 seg.
pregunta.setOnLongClickListener(new OnLongClickListener() {

@Override
public boolean onLongClick(View v) {
// TODO Auto-generated method stub
DragShadowBuilder sombra = new DragShadowBuilder(v);
v.startDrag(null, sombra, null, 0);
v.setVisibility(View.INVISIBLE);
return true;
}
});


              View.OnDragListener _OnDragListener = new View.OnDragListener() {

                      @Override
                      public boolean onDrag(View arg0, DragEvent arg1) {
```

```
                                ImageView pregunta = (ImageView) findViewById(R.id.imgvwPreguntaArrastrar);

                                // Este método vaise chamar 3 veces: ao soltar a imaxe pregunta
                                // ou na imaxe OK
                                // ou na imaxe NO
                                // ou no Layout
                                if (arg1.getAction() == DragEvent.ACTION_DRAG_STARTED) {

                                        return true; // Aceptamos xestionar os eventos

                                }
                                if (arg1.getAction() == DragEvent.ACTION_DRAG_ENTERED) {
                                   // Para cambiar de imaxe ó entrar
                                   //((ImageView)arg0).setImageResource(R.drawable.imaxeoentrar);
                                }
                                if (arg1.getAction() == DragEvent.ACTION_DRAG_EXITED) {
                                  // Para cambiar de imaxe o ó saír
                                  //((ImageView)arg0).setImageResource(R.drawable.imaxeosair);
                                }
                                if (arg1.getAction() == DragEvent.ACTION_DROP) {
                                    // Soltamos a imaxe pregunta

                                        // COMPROBAMOS EN QUE IMAXE SE SOLTOU: OK, NO ou Layout
                                        switch (arg0.getId()) {
                                        case R.id.imgvwPreguntasOk:
                                                Toast.makeText(getApplicationContext(), "Soltaches en OK", Toast.LENGTH_SHORT).show()

                                                break;
                                        case R.id.imgVwPreguntasNo:
                                                Toast.makeText(getApplicationContext(), "Soltaches en NO", Toast.LENGTH_SHORT).show()
                                                break;
                                        default:
                                               // FACEMOS VISIBLE A IMAXE DA PREGUNTA
                                               // Soltouse a imaxe no layout
                                                pregunta.setVisibility(View.VISIBLE);
                                                return false;
                                        }

                                        // FACEMOS VISIBLE A IMAXE PREGUNTA
                                        pregunta.setVisibility(View.VISIBLE);
                                        return true;
                                }
                                return false;
                        }
                };

        // Asociamos os obxectos ImageView e Layout á interface definida antes.
        ImageView imgOk = (ImageView) findViewById(R.id.imgvwPreguntasOk);
        ImageView imgNo = (ImageView) findViewById(R.id.imgVwPreguntasNo);
        ViewGroup pantalla = (ViewGroup) findViewById(R.id.LinearLayoutPreguntas);

        imgNo.setOnDragListener(_OnDragListener);
        imgOk.setOnDragListener(_OnDragListener);
        pantalla.setOnDragListener(_OnDragListener);

        }

        @Override
        public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.u3_20__drag_drop, menu);
        return true;
        }

        }
```

#### 1.3.2.1 Comprobar versión de Android

- Comprobamos se a versión de Android e 3.x.x ou superior, nese caso chamamos ao método que vai xestionar o Drag&Drop.

```
String androidOS = Build.VERSION.RELEASE;
                if (!(androidOS.startsWith("2")) && !(androidOS.startsWith("1")))
                        xestionarArrastrar();
```

#### 1.3.2.2 Comezar a arrastrar

```
ImageView pregunta = (ImageView) findViewById(R.id.imgvwPreguntaArrastrar);

                // Xestionamos cando prememos durante un tempo maior a 1 seg.
                pregunta.setOnLongClickListener(new OnLongClickListener() {

                        @Override
                        public boolean onLongClick(View v) {
                                // TODO Auto-generated method stub
                                DragShadowBuilder sombra = new DragShadowBuilder(v);
                                v.startDrag(null, sombra, null, 0);
                                v.setVisibility(View.INVISIBLE);
                                return true;
                        }
                });
```

- Se na imaxe da pregunta se fai un LongClick capturamos ese evento cun Listener e
- Indicámoslle ao S.O. o que ten que debuxar cando faga Drag sobre a imaxe.
- Para debuxar a imaxe (ou o que queiramos) que vai a moverse pola pantalla, temos que utilizar un obxecto da clase **DragShadowBuilder**.
- Podemos crear unha subclase de dita clase e personalizar todo o que queiramos.
- Para o noso exemplo, imos a usar a mesma imaxe da pregunta que está en pantalla para facelo.
- Para iso só temos que crear un obxecto da clase DragShadowBuilder e pasarlle no constructor o View a debuxar pola pantalla. Neste caso o View asociado a imaxe da pregunta.
    - **DragShadowBuilder sombra = new DragShadowBuilder(v)**;

- Cando facemos un Drop, pode ser que veñan outros obxectos doutras aplicacións ou incluso da nosa, pero que non sexa o obxecto esperado.
- Podemos facer que coa imaxe que se arrastra, vaia asociado información dun tipo determinado, de tal forma que o obxecto que vai a recibir a imaxe (drop,) pode preguntar polo tipo de información que acompaña a imaxe (ou o que se estea arrastrando) e se non é do tipo que espera dicirlle o S.O. que non quere recibir máis eventos do obxecto que se arrastra.
    - Neste exemplo non imos asociar información á imaxe.

- Chamamos ao método **startDrag()**: v.startDrag(null, sombra, null, 0);
- Ten catro parámetros:
    - O primeiro: é onde iría a información que acompañaría á imaxe.
    - O segundo: é o obxecto da clase DragShadowBuilder (A vista que se vai mover pola pantalla)
    - O terceiro: serve para o mesmo que o primeiro pero cun tipo de información máis sinxela.
    - O cuarto: non se utiliza.

- **v.setVisibility(View.INVISIBLE)** fai que a vista que conte? a imaxe na súa posición inicial non sexa visible.
    - Lembrar que o que aparece visualmente como obxecto que se move pola pantalla non é o orixinal.

- Finalmente o método onLongClick devolve true se realmente se consumou o evento LongClick.

#### 1.3.2.3 Vistas nas que soltar

- Agora hai que informar o S.O. sobre que obxectos van poder recibir operacións de Drop.
- Cada unha das vistas receptoras (contenedoras) do Drop terá que ter asociado a interface OnDragListener, polo que terán que chamar o método setOnDragListener.
- Como queremos que todos elas (as contedoras) teñan o mesmo código, podemos usar un obxecto da clase onDragListener para facelo (Visto no apartado de Xestión de Eventos III)

```
ImageView imgOk = (ImageView) findViewById(R.id.imgvwPreguntasOk);
                ImageView imgNo = (ImageView) findViewById(R.id.imgVwPreguntasNo);
                ViewGroup pantalla = (ViewGroup) findViewById(R.id.LinearLayoutPreguntas);

                imgNo.setOnDragListener(_OnDragListener);
                imgOk.setOnDragListener(_OnDragListener);
                pantalla.setOnDragListener(_OnDragListener);
```

- Os 2 botóns e o layout poden ser recptores da imaxe que se anda arrastrando pola pantalla, como hai que facer case o mesmo nos tres casos, crease un obxecto que implemente o método **onDrag()** do interface OnDragListener.

#### 1.3.2.4 O método onDrag. Implementar o Interface

- Este evento é chamado cando se solta unha vista sobre outra, e é esta última quen o chama.

```
View.OnDragListener _OnDragListener = new View.OnDragListener() {

                        @Override
                        public boolean onDrag(View arg0, DragEvent arg1) {
                                ImageView pregunta = (ImageView) findViewById(R.id.imgvwPreguntaArrastrar);

                                // Este método vaise chamar 3 veces: ao soltar a imaxe pregunta
                                // ou na imaxe OK
                                // ou na imaxe NO
                                // ou no Layout
                                if (arg1.getAction() == DragEvent.ACTION_DRAG_STARTED) {

                                        return true; // Aceptamos xestionar os eventos

                                }
                                if (arg1.getAction() == DragEvent.ACTION_DRAG_ENTERED) {
                                   // Para cambiar de imaxe ó entrar
                                   //((ImageView)arg0).setImageResource(R.drawable.imaxeoentrar);
                                }
                                if (arg1.getAction() == DragEvent.ACTION_DRAG_EXITED) {
                                  // Para cambiar de imaxe o ó saír
                                  //((ImageView)arg0).setImageResource(R.drawable.imaxeosair);
                                }
                                if (arg1.getAction() == DragEvent.ACTION_DROP) {
                                    // Soltamos a imaxe pregunta

                                        // COMPROBAMOS EN QUE IMAXE SE SOLTOU: OK, NO ou Layout
                                        switch (arg0.getId()) {
                                        case R.id.imgvwPreguntasOk:
                                                Toast.makeText(getApplicationContext(), "Soltaches en OK", Toast.LENGTH_SHORT).show()

                                                break;
                                        case R.id.imgVwPreguntasNo:
                                                Toast.makeText(getApplicationContext(), "Soltaches en NO", Toast.LENGTH_SHORT).show()
                                                break;
                                        default:
                                               // FACEMOS VISIBLE A IMAXE DA PREGUNTA
                                               // Soltouse a imaxe no layout
                                                pregunta.setVisibility(View.VISIBLE);
                                                return false;
                                        }

                                        // FACEMOS VISIBLE A IMAXE PREGUNTA
                                        pregunta.setVisibility(View.VISIBLE);
                                        return true;
                                }
                                return false;
                        }
                };
```

- Creamos o obxecto _OnDragListener.

- Implementamos o método **onDrag(View arg0, DragEvent arg1)**
- Ten dous parámetros:
    - Un view que é quen recibe o a vista a soltar
    - un obxecto da clase DragEvent, que entre outras cousas nos podería servir para recoller a información que puidese acompañar á vista que arrastramos. Tamén serve para ver que acción se está realizando co elemento que se está arrastrando sobre a vista:
        - Entrando no obxecto destino, saíndo, soltando, etc.
    - Para saber o tipo de acción drag&drop que se produce, temos que usar o método **getAction()** da clase DragEvent.
        - Seguindo os comentarios do código pódense ver posibles accións se están levando a cabo co obxecto que se está arrastrando.

- Se o evento que se produce é o Drop (Soltar):if (arg1.getAction() == DragEvent.ACTION_DROP)
- Entón debemos executar o código que desexamos.
- Como temos un obxecto _OnDragListener para tódalas posibles vistas destinarias temos que comprobar en que vista se soltou a imaxe e actuar en consecuencia.

- Observar como facemos visible a imaxe inicial da pregunta unha vez que imos saír do método().

- O **return** é moi importante, xa que cando prememos sobre a pregunta para arrastrala, o S.O. manda unha mensaxe a todas as Views que poden recoller un "Drag" (no noso caso OK, NO) preguntando quen quere aceptar o view que se vai arrastrar.
- Se o procedemento onDrag devolve false, o S.O. non manda máis eventos a dito View, se devolve True si.

## 1.4 Caso práctico 2: Preguntas

- Comezaremos creando un novo proxecto: **U3_21_Preguntas**
- Neste caso baseándonos na aplicación anterior imos crear unha nova na que o sistema terá un array de preguntas (Verdadeiro/Falso) que amosara ao usuario de forma aleatoria.
- O usuario debe responder arrastrando a imaxe da pregunta cara a imaxe OK ou NO.
- A aplicación informará se a resposta é correcta ou non.
- Unha vez que se acaben as preguntas informarase desa situación.


- No exemplo respondeuse mal á pregunta da primeira imaxe.
- Na segunda imaxe xa se amosa unha nova pregunta e o informe da aplicación á nosa anterior resposta.

### 1.4.1 O XML do Layout

- Nas liñas 15-29 están os Elementos que van amosar as preguntas.

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/LinearLayoutPreguntas"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:text="Drag &amp; Drop: Preguntas"
        android:textSize="22sp" />

    <ScrollView
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="3" >

        <TextView
            android:id="@+id/lblPregunta"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:gravity="center_horizontal"
            android:text="Aquí van as preguntas"
            android:textColor="#00F"
            android:textSize="26sp" />
    </ScrollView>
```

```
    <ImageView
        android:id="@+id/imgvwPreguntaArrastrar"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_gravity="center_horizontal"
        android:layout_weight="1"
        android:contentDescription="Imaxe pregunta arrastrar"
        android:src="@drawable/question" />

    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="2" >

        <ImageView
            android:id="@+id/imgVwPreguntasNo"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentRight="true"
            android:layout_alignTop="@+id/imgvwPreguntasOk"
            android:contentDescription="Imaxe preguntas NO"
            android:src="@drawable/no" />

        <ImageView
            android:id="@+id/imgvwPreguntasOk"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentLeft="true"
            android:layout_centerVertical="true"
            android:contentDescription="Imaxe preguntas OK"
            android:src="@drawable/ok" />
    </RelativeLayout>

</LinearLayout>
```

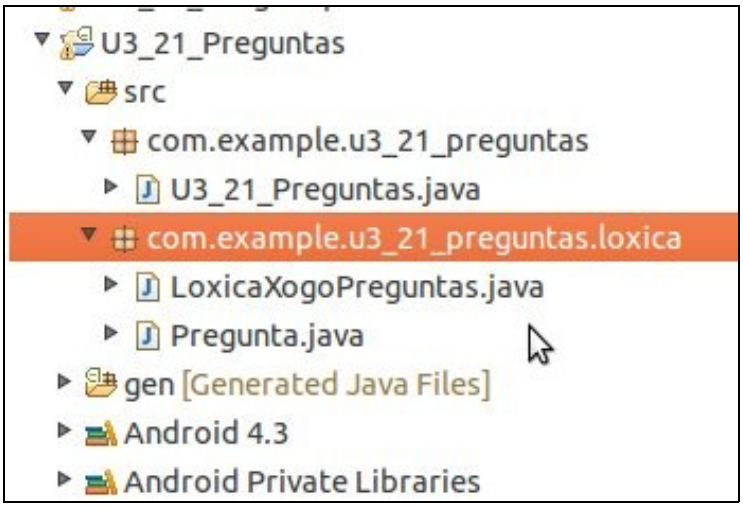
## 1.5 Cógigo Java

- Unha posible solución é a que se amosa a continuación.

- Imos separar o código:
    - ♦ Lóxica do programa por un lado
        - ◊ Cando facemos unha aplicación, normalmente imos traballar con datos que temos que almacenar nalgún sitio. Na aplicación que estamos a desenvolver atopámonos coas preguntas. Todas as preguntas virán dunha táboa e teremos que recuperalas para amosalas ao usuario no xogo das preguntas.
        - ◊ Para facer máis claro o proxecto, imos crear un paquete onde gardaremos estas clases 'especiais'. Dentro de dito paquete teremos:
            - · A clase Pregunta que terá todo o necesario para gardar a información dunha pregunta (id, enunciado, unha variable booleana que indique cal é a resposta correcta, outra variable booleana que indique se a pregunta foi lida).
            - · Unha clase que sirva para gardar aquela información que teña que ver coa configuración do xogo e tamén que teña métodos que nos sirvan para o desenvolvemento do mesmo.
                - Gardar nun array o conxunto de preguntas que imos usar.
                - Un método que devolva unha pregunta que non fose lida do array anterior.

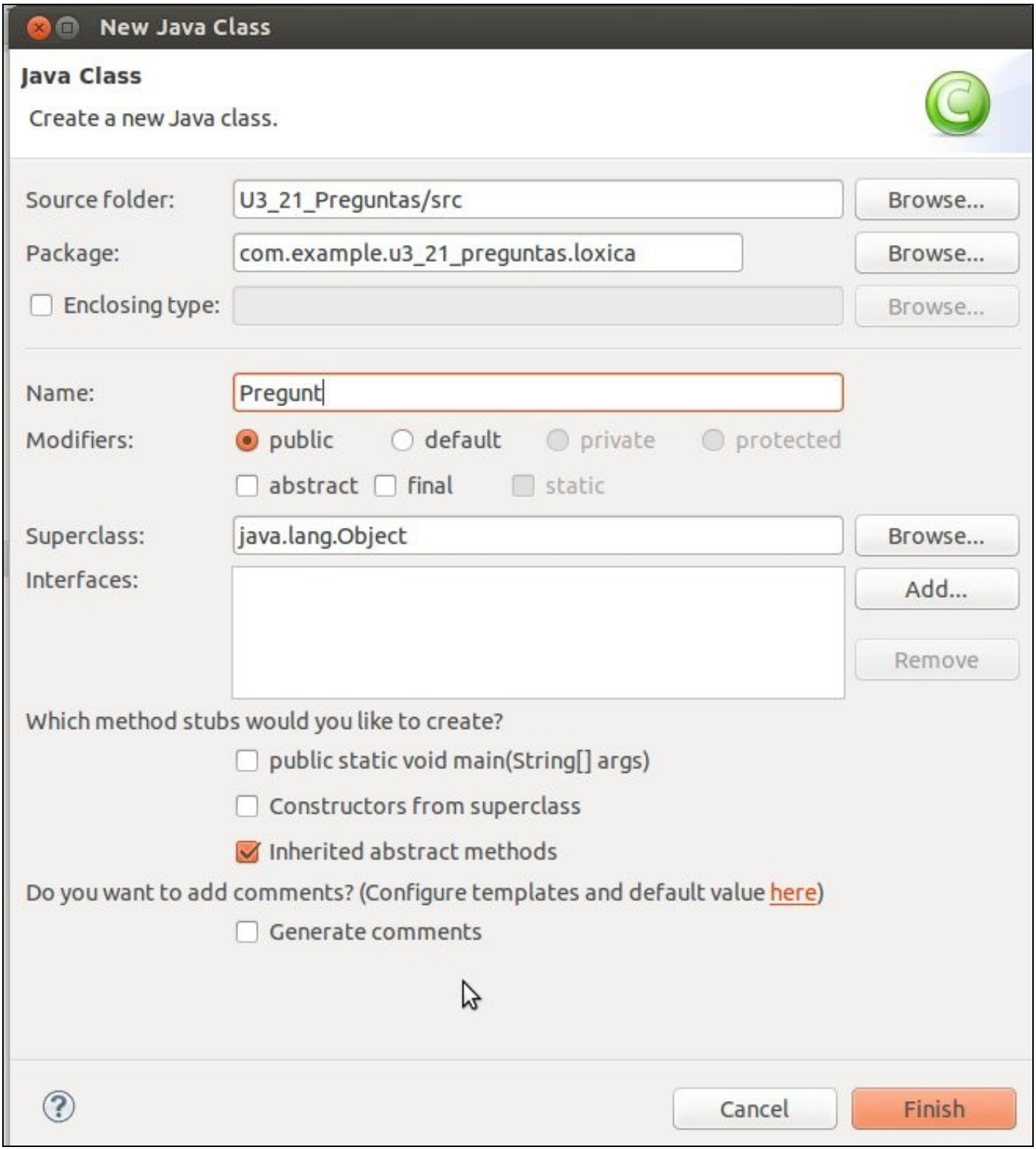
- ♦ Activities polo outro
    - ◊ Cando facemos unha aplicación, normalmente imos traballar con datos que temos que almacenar nalgún sitio. Na aplicación que estamos a desenvolver nos atopamos coas preguntas. Todas as preguntas virán dunha táboa e teremos que recuperalas para amosalas ao usuario no xogo das preguntas.

- Imos crear un **Paquete** para albergar a lóxica da aplicación:




- Na imaxe podemos ver que no paquete **...loxica** hai dúas clases Java,

- Para crear un paquete/clase: Co botón dereito sobre /**src** ou sobre o paquete indicar **New --> Class**




- Cubrir os campos do paquete e do nome da nova clase.

### 1.5.1 Clase Pregunta

- Conten os atributos que pode ter unha pregunta e os métodos Setter e Getter dos atributos, así como unha sobre escritura do método toString();

```
package com.example.u3_21_preguntas.loxica;

public class Pregunta {
```

```
private int _id;
private String enunciado;
private boolean resposta;
private boolean vista; //Almacena se a pregunta xa foi amosada ao usuario

public Pregunta(int id, String enunciado, boolean resposta) {
this._id = id;
this.enunciado = enunciado;
this.resposta = resposta;
vista = false;
}

public long get_id() {
return _id;
}

public void set_id(int _id) {
this._id = _id;
}

public String getEnunciado() {
return enunciado;
}

public void setEnunciado(String enunciado) {
this.enunciado = enunciado;
}

public boolean isResposta() {
return resposta;
}

public void setResposta(boolean resposta) {
this.resposta = resposta;
}

public boolean isVista() {
return vista;
}

public void setVista(boolean vista) {
this.vista = vista;
}

@Override
public String toString() {
return enunciado;
}
}
```

### 1.5.2 Clase LoxicaXogoPreguntas

```
package com.example.u3_21_preguntas.loxica;

import java.util.ArrayList;
import java.util.Random;

public class LoxicaXogoPreguntas {

public static ArrayList<Pregunta> preguntas = new ArrayList<Pregunta>();

public static int xerarAleatorio(int min, int max) {
Random aleat = new Random();
return (aleat.nextInt(max - min) + min);
}

public static Pregunta obterNovaPregunta() {
```

```
boolean quedan = false;
for (Pregunta pregunta : preguntas) {
if (!pregunta.isVista()) {
quedan = true;
break;
}
}
if (!quedan)
return null;

int aleaotorio = xerarAleatorio(0, preguntas.size());
Pregunta pregunta;
do {
pregunta = preguntas.get(aleaotorio);
aleaotorio = xerarAleatorio(0, preguntas.size());
} while (pregunta.isVista());
return pregunta;

}

}
```

- **Liña 8**: preguntas como un array de obxectos de tipo Pregunta.

- **Método xerarAleatorio()**: devolve un número aleatorio entre un mínimo e un máximo

- **Método obterNovaPregunta()**: Revisa no array se quedan preguntas sen amosar e se é asi busca unha nova aleatoriamente e devolve esa Pregunta a que lle pedira unha pregunta nova

### 1.5.3 Clase Activity Preguntas

- Importamos as 2 clases anteriores (Liñas 15,16)

```
package com.example.u3_21_preguntas;

import android.app.Activity;
import android.os.Bundle;
import android.view.DragEvent;
import android.view.Menu;
import android.view.View;
import android.view.View.DragShadowBuilder;
import android.view.View.OnLongClickListener;
import android.view.ViewGroup;
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.Toast;

import com.example.u3_21_preguntas.loxica.LoxicaXogoPreguntas;
import com.example.u3_21_preguntas.loxica.Pregunta;

public class U3_21_Preguntas extends Activity {
Pregunta preguntaActual = null;

@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_u3_21__preguntas);

crearPreguntas();
xestionarArrastrar();

}

private void xestionarArrastrar() {
```

```
ImageView pregunta = (ImageView) findViewById(R.id.imgvwPreguntaArrastrar);

// Xestionamos cando prememos durante un tempo maior a 1 seg.
pregunta.setOnLongClickListener(new OnLongClickListener() {

@Override
public boolean onLongClick(View v) {
// TODO Auto-generated method stub
DragShadowBuilder sombra = new DragShadowBuilder(v);
v.startDrag(null, sombra, null, 0);
v.setVisibility(View.INVISIBLE);
return true;
}
});

View.OnDragListener _OnDragListener = new View.OnDragListener() {

@Override
public boolean onDrag(View arg0, DragEvent arg1) {
ImageView pregunta = (ImageView) findViewById(R.id.imgvwPreguntaArrastrar);

// Este método vaise chamar 3 veces: ao soltar a imaxe pregunta
// ou na imaxe OK
// ou na imaxe NO
                                // ou Layout
if (arg1.getAction() == DragEvent.ACTION_DRAG_STARTED) {

return true; // Aceptamos xestionar os eventos
}
if (arg1.getAction() == DragEvent.ACTION_DRAG_ENTERED) {
}
if (arg1.getAction() == DragEvent.ACTION_DRAG_EXITED) {
}
if (arg1.getAction() == DragEvent.ACTION_DROP) {
                                        // Soltamos a imaxe pregunta

// COMPROBAMOS EN QUE IMAXE SE SOLTOU: ok, no ou layout
switch (arg0.getId()) {
case R.id.imgvwPreguntasOk:
comprobarPregunta(true);

break;
case R.id.imgVwPreguntasNo:
comprobarPregunta(false);
break;
default:
pregunta.setVisibility(View.VISIBLE);
return false;
}

// FACEMOS VISIBLE A IMAXE PREGUNTA
pregunta.setVisibility(View.VISIBLE);
return true;
}
return false;
}
};

// Asociamos os obxectos ImageView e Layout á interface definida antes.
ImageView imgOk = (ImageView) findViewById(R.id.imgvwPreguntasOk);
ImageView imgNo = (ImageView) findViewById(R.id.imgVwPreguntasNo);
ViewGroup pantalla = (ViewGroup) findViewById(R.id.LinearLayoutPreguntas);

imgNo.setOnDragListener(_OnDragListener);
imgOk.setOnDragListener(_OnDragListener);
pantalla.setOnDragListener(_OnDragListener);
}

private void cambiarPregunta() {
Pregunta preguntaSeguinte = LoxicaXogoPreguntas.obterNovaPregunta();
if (preguntaSeguinte == null) {
Toast.makeText(getApplicationContext(), "Non hai máis preguntas", Toast.LENGTH_LONG).show();
preguntaActual = null;
```

```
} else {

preguntaActual = preguntaSeguinte;
preguntaActual.setVista(true);

TextView texto = (TextView) findViewById(R.id.lblPregunta);

texto.setText(preguntaActual.getEnunciado());
}

}

private void crearPreguntas() {
// Creamos tres preguntas e metémolas no array de preguntas
Pregunta pregunta1 = new Pregunta(0, "Está o sol cerca ?", false);
Pregunta pregunta2 = new Pregunta(1, "É a lúa un satélite ?", true);
Pregunta pregunta3 = new Pregunta(2, "A velocidade da luz é de 300.000 km/seg ?", true);

LoxicaXogoPreguntas.preguntas.add(pregunta1);
LoxicaXogoPreguntas.preguntas.add(pregunta2);
LoxicaXogoPreguntas.preguntas.add(pregunta3);

cambiarPregunta();
}

private void comprobarPregunta(boolean resposta) {
if (preguntaActual.isResposta() == resposta)
Toast.makeText(getApplicationContext(), "Parabéns acertaches", Toast.LENGTH_SHORT).show();

else
Toast.makeText(getApplicationContext(), "Hmmm! Fallaches", Toast.LENGTH_SHORT).show();

cambiarPregunta();

}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
// Inflate the menu; this adds items to the action bar if it is present.
getMenuInflater().inflate(R.menu.u3_21__preguntas, menu);
return true;
}

}
```

- **Liña 19**: creamos unha variable preguntaActual para saber que pregunta estamos procesando.
- **Liña 26**: Chamamos ao método para crear as preguntas
- **Liñas 117...: Método crearPreguntas**: Creamos as preguntas e engadímolas ao array de Preguntas. Cando finalicemos de crear as preguntas chamamos ao método que vaia cambiando as preguntas que se van amosando.
    - ♦ O ideal sería traballar con ficheiros ou unha BD como se verá máis adiante.
- **Liñas 100...: Método cambiarPreguntas**: busca a pregunta seguinte, se non hai máis preguntas saca un aviso, en caso contrario amosa en pantalla a nova pregunta.
- **Liñas 130...: Método comprobarPregunta**: é chamado cando se solta a imaxe da pregunta nunha das imaxes (OK/NO).
    - ♦ Recibe como parámetro true/false en función do botón no que fose soltada a imaxe.
    - ♦ Compárase a resposta dada coa resposta gardada na pregunta e actúase en consecuencia ...