Programación de eventos de componentes de

interface de usuario

* En general la programación de eventos de componentes de interface de usuario es la siguiente:
* Declarar en la vista layout el componente de usuario y sus atributos:
* Edit Text:

<**EditText android:id="@+id/*TxtInput*"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:inputType="text"  
 android:layout\_below="@id/LblEtiqueta"** />

* Botones

<**Button android:id="@+id/*BtnInputLayout*"  
android:layout\_width="wrap\_content"  
android:layout\_height="wrap\_content"  
android:layout\_below="@id/TxtImagenHint"  
android:layout\_alignParentRight="true"  
android:text="@string/negrita"** />

* RadioGroup – RadioButton

<**RadioGroup android:id="@+id/GrbGrupo1"  
 android:orientation="vertical"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"** >  
  
 <**RadioButton android:id="@+id/RbOpcion1"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="@string/opcion\_1"** />  
  
 <**RadioButton android:id="@+id/RbOpcion2"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="@string/opcion\_2"** />  
  
</**RadioGroup**>

* CheckBox

<**CheckBox android:id="@+id/ChkMarcame"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="@string/marcame"  
 android:checked="false"** />

* Declarar en el programa Java (MainActivity) una variable para cada componente:
* Edit Text

**private** EditText **txtInput**;

* Botones

**private** Button **btnNegrita**;  
**private** Button **btnComprobar**;

* RadioGroup – RadioButton

En un **RadioGroup** sólo se permite elegir sólo una opción.

**private** RadioGroup **rgOpciones**;  
**private** RadioButton **rbOpcion1**;  
**private** RadioButton **rbOpcion2**;

* CheckBox

En un **CheckBox** se permite elegir varias opciones.

**private** CheckBox **cbMarcame**;

* Asociar a la variable anterior, el componente de usuario de la vista a través del id y la clase R:
* Edit Text

**txtInput** = (EditText)findViewById(R.id.***TxtInput***);

* Botones

**btnComprobar** = (Button)findViewById(R.id.***BtnInputLayout***);

* RadioGroup – RadioButton

**rgOpciones** = (RadioGroup)findViewById(R.id.***GrbGrupo1***);

**rbOpcion1** = (RadioButton)findViewById(R.id.***RbOpcion1***);  
**rbOpcion2** = (RadioButton)findViewById(R.id.***RbOpcion2***);

* CheckBox

**cbMarcame** = (CheckBox)findViewById(R.id.***ChkMarcame***);

* Crear el método que se ejecutará cuando ocurra el evento del componente
* Edit Text

En este caso, no tiene eventos pero para leer el valor tecleado por el usuario hacer lo siguiente:

**txtInput**.getText().toString();

Y para asignarle un valor:

**txtInput***.setText(nuevoTexto);*

* Botones
* Crear una clase OnClickListener y sobreponer el método onClick que es la acción correspondiente del componente:

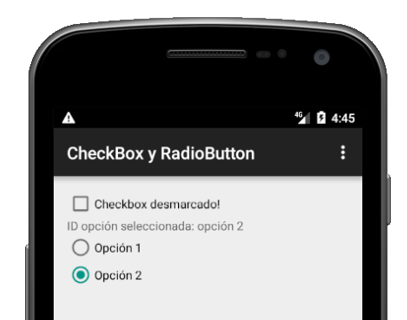
**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View view) {  
 String num = **txtInput**.getText().toString();  
  
 **if**(num.isEmpty() || Integer.*parseInt*(num)%2 != 0)  
 **txtInputLayout**.setError(**"Error: No es un**

**número par!"**);  
 **else  
 txtInputLayout**.setError(**null**);  
 }  
}

* RadioGroup – RadioButton
* Crear una clase OnClickListener y sobreponer el método onClick que es la acción correspondiente del componente:

View.OnClickListener list = **new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View view) {  
 String opcion = **""**;  
 **switch**(view.getId()) {  
 **case** R.id.***RbOpcion1***:  
 opcion = **"opción 1"**;  
 **break**;  
 **case** R.id.***RbOpcion2***:  
 opcion = **"opción 2"**;  
 **break**;  
 }  
  
 **lblMensaje**.setText(**"ID opción seleccionada: "** + opcion);  
 }  
};

Es decir, dentro de la definición del método onClick se verifica cual **RadioButton** del **RadioGroup** fue elegido mediante un switch a través de la clase view con view.getId()y case R.id.***RbOpcion1***, R.id.***RbOpcion2***, etc.



Otro evento, que existe para el componente RadioGroup – RadioButton es cuando sufre un cambio onCheckedChanged:

**rgOpciones**.setOnCheckedChangeListener(**new** RadioGroup.OnCheckedChangeListener() {  
 **public void** onCheckedChanged(RadioGroup group, **int** checkedId) {  
  
 String opcion = **""**;  
 **switch**(checkedId) {  
 **case** R.id.***RbOpcion1***:  
 opcion = **"opción 1"**;  
 **break**;  
 **case** R.id.***RbOpcion2***:  
 opcion = **"opción 2"**;  
 **break**;  
 }  
  
 **lblMensaje**.setText(**"ID opción seleccionada: "** + opcion);  
 }  
});

* CheckBox

Cuando es un **CheckBox** se hace de manera similar a la de un botón:

**cbMarcame**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View view) {  
 **boolean** isChecked = ((CheckBox)view).isChecked();  
  
 **if** (isChecked) {  
 **cbMarcame**.setText(**"Checkbox marcado!"**);  
 }  
 **else** {  
 **cbMarcame**.setText(**"Checkbox desmarcado!"**);  
 }  
 }  
});

verificando ahora que el **CheckBox** fue elegido mediante el método booleano:

((CheckBox)view).isChecked()

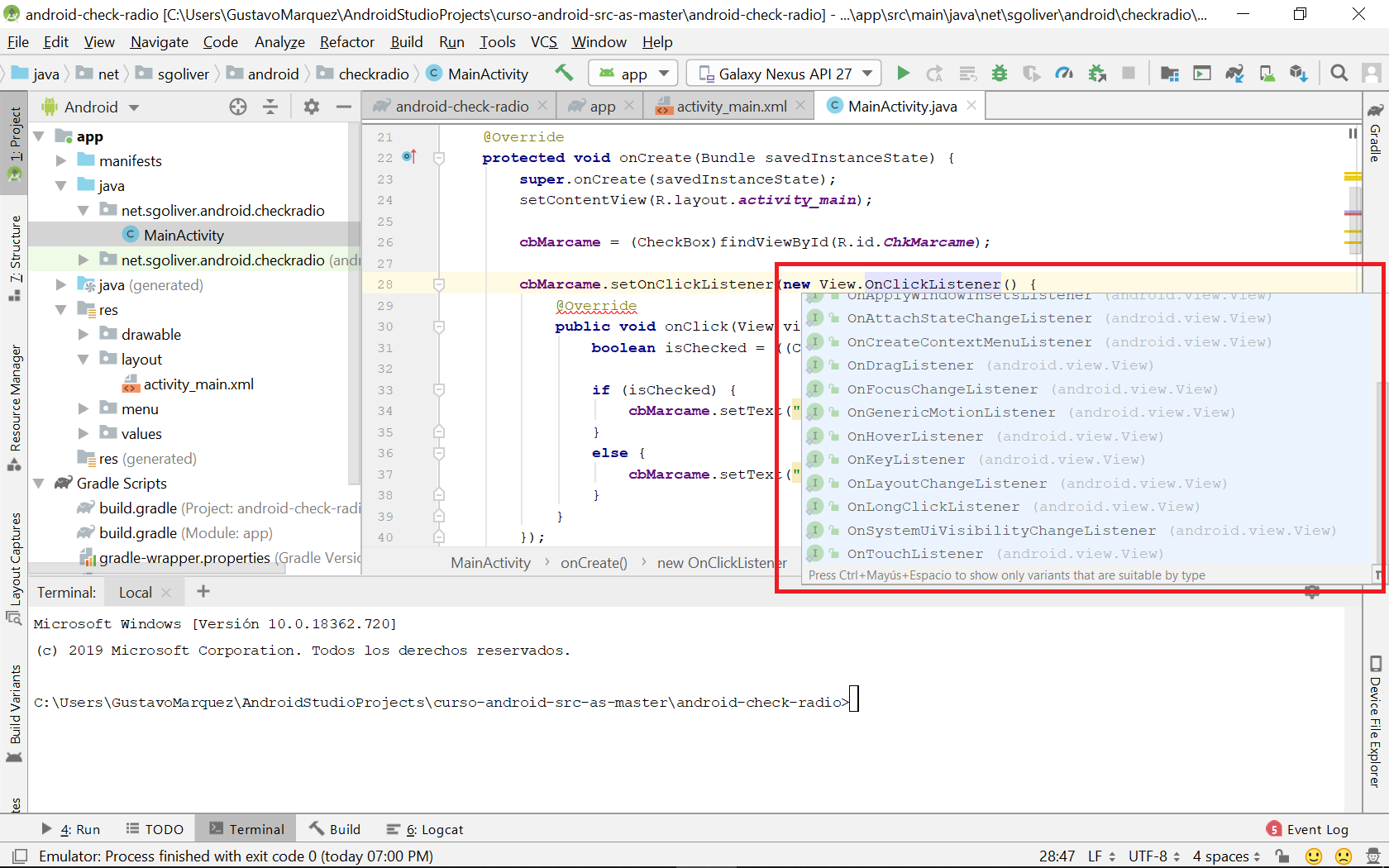
De manera similar al caso anterior, otro evento que existe para el componente checkBox es cuando sufre un cambio onCheckedChanged:

**cbMarcame**.setOnCheckedChangeListener(**new**

CheckBox.OnCheckedChangeListener() {  
 **public void** onCheckedChanged(CompoundButton

buttonView, **boolean** isChecked) {  
 **if** (isChecked) {  
 **cbMarcame**.setText(**"Checkbox marcado!"**);  
 }  
 **else** {  
 **cbMarcame**.setText(**"Checkbox desmarcado!"**);  
 }  
 }  
});

En el IDE se pueden ver otro tipo de métodos asociados a otro tipo de eventos:



En estos últimos casos, ver el programa android-check-radio

* Como último paso, asignar una instancia de la clase anterior a la variable declarada del componente:
* Botones

**btnComprobar**.setOnClickListener( <*instancia de la clase*

*anterior, ver el ejemplo de* RadioGroup – RadioButton > );

* RadioGroup – RadioButton

**rbOpcion1**.setOnClickListener(list);  
**rbOpcion2**.setOnClickListener(list);

* CheckBox

En este caso se puede hacer al mismo tiempo que se declara el método que se ejecutará cuando ocurra el evento del componente:

**cbMarcame**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View view) {  
 **boolean** isChecked = ((CheckBox)view).isChecked();  
  
 **if** (isChecked) {  
 **cbMarcame**.setText(**"Checkbox marcado!"**);  
 }  
 **else** {  
 **cbMarcame**.setText(**"Checkbox desmarcado!"**);  
 }  
 }  
});

Esta última técnica también se puede hacer emplear para los eventos de los botones y RadioGroup – RadioButton

La definición de la clase OnClickListener, OnCheckedChangeListener, se puede hacer:

* En línea, como en los ejemplos anteriores.
* En forma separada en la misma clase
* En un archivo aparte
* Algunos atributos adicionales de los componentes gráficos de usuario:
* <**TextView xmlns:android="http://schemas.android.com/apk/res/android"  
   android:layout\_width="match\_parent"  
   android:textColor="#07F" // Azul cielo  
   android:layout\_height="48dp"  
   android:fontFamily="sans-serif"  
   android:gravity="center\_vertical"  
   android:textSize="25sp"** />

* <**ListView android:id="@+id/left\_drawer"  
   android:layout\_width="240dp"  
   android:layout\_height="match\_parent"  
   android:layout\_gravity="start"  
   android:choiceMode="singleChoice"  
   android:divider="@android:color/transparent" // Separador**

**// android:divider="@android:color/color/holo\_red\_dark"**

**android**:**orientation**="**vertical**" // **Puede ser horizontal**

**(default)**

**android:showDividers="middle"   
android:dividerHeight="1dp" // Ancho del separador  
android:background="#FFF"**/>

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:layout\_gravity="top"

android:background="#33b5e5" // Azul Cielo

android:paddingBottom="4dp"

android:paddingTop="4dp"

android:textColor="#fff" /> // Blanco

**android:layout\_marginRight="10dp"**