

Interfaz de usuario: Controles de entrada (*inputs*):

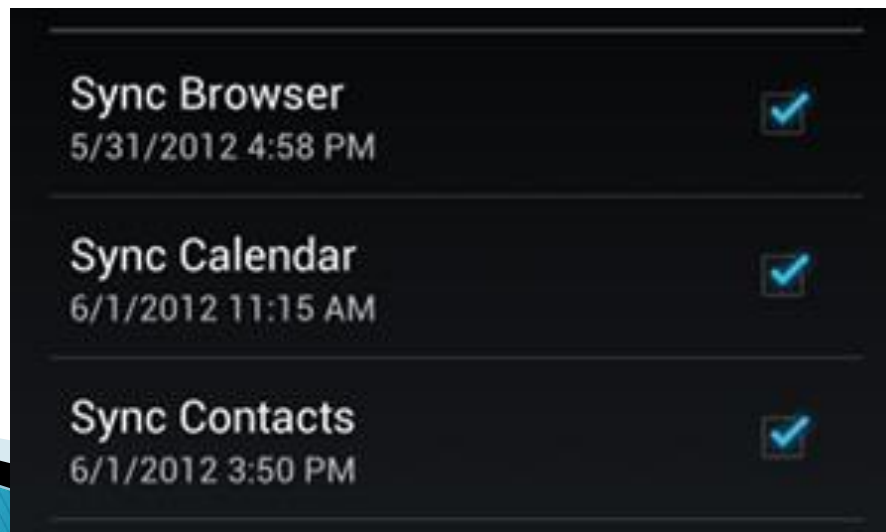
“CheckBox”

“RadioButton”

Control CheckBox (I)

(ver su API)

- ▶ Es un **control de entrada**.
- ▶ Permite que el usuario **pueda escoger una o más opciones** partiendo de un conjunto.
- ▶ **Se suele usar para** **marcar o desmarcar opciones en una aplicación**.
- ▶ Lo más **típico** es **presentarlos en una lista vertical**.



Control CheckBox (II)

(ver su API)

- ▶ Un **conjunto de checkboxes** permite al usuario **escoger múltiples cosas**, por lo que:
 - ✌ **cada checkbox es manejado por separado** y **debe registrarse un clickListener para cada uno.**
- ▶ La **forma de definir un CheckBox** y los **métodos disponibles** para manipularlo son:
 - ✌ **similares** a los del **control ToggleButton ya visto.**

Control CheckBox (III)

(ver su API)

► *Ejemplo de definición de un CheckBox en XML:*

```
<CheckBox android:id="@+id/ChkMarcame"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text="@string/marcame"  
    android:checked="false" />
```

► En cuanto a la personalización del control :

👍 **CheckBox** extiende [indirectamente] del control **TextView**.

👍 Por tanto, **todas las opciones de formato ya comentadas** anteriormente son **válidas** también para este control.

Control CheckBox (IV)

¿Cómo saber cuándo está pulsado o no?

- ▶ Atributo “**android:checked**”:
 - ☺ Se usa para **inicializar el estado del control** a **marcado** (*true*) o **desmarcado** (*false*).
 - ☺ Si no establecemos este atributo, **el control aparecerá por defecto** en estado **desmarcado**.
- ▶ **En el código de la aplicación** podremos hacer uso de los métodos:
 - ☺ **isChecked()** → para **conocer el estado del control**
 - ☺ **setChecked(*estado*)** → **para establecer un estado concreto para el control**

```
if (checkBox.isChecked()) {  
    checkBox.setChecked(false);  
}
```

Control CheckBox (V)

Control de Eventos: atributo "onClick"

- ▶ Cuando **el usuario selecciona un checkbox**, el **objeto "CheckBox"** recibe un evento **"on-click"**.
- ▶ Para **definir el comportamiento**, se le añade el **atributo onClick** al **<CheckBox>** en el **XML**.
- ▶ El **valor** debe ser el **nombre del método** que será llamado.
- ▶ **La activity que usa el layout** es la que **debe implementar el método**.
- ▶ El **método** debe ser:

 **public,**

 **devolver void y**

 **recibir un único argumento de tipo View**

Control de Eventos: evento “onCheckedChanged”

- ▶ El evento “onCheckedChanged” informa de que ha cambiado el estado del CheckBox.
- ▶ Para implementar las acciones de este evento se suele usar la siguiente lógica:
 - ➡ se asigna el escuchador de eventos con el método “setOnCheckedChangeListener”
 - ➡ se captura el evento con el método “onCheckedChanged”
 - ➡ y, dependiendo del nuevo estado del control (variable isChecked recibida como parámetro), haremos una acción u otra
- ▶ Veámoslo con este código:

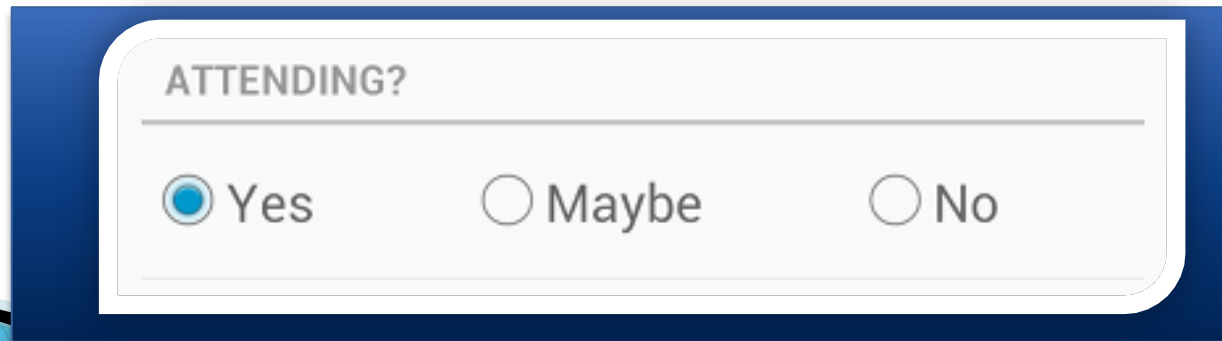
Control de Eventos: evento "onCheckedChangeListener"

```
private CheckBox cbMarcame;  
  
//...  
  
cbMarcame = (CheckBox)findViewById(R.id.chkMarcame);  
  
cbMarcame.setOnCheckedChangeListener(  
    new CheckBox.OnCheckedChangeListener() {  
        public void onCheckedChanged(CompoundButton buttonView,  
                                     boolean isChecked) {  
            if (isChecked) {  
                cbMarcame.setText("Checkbox marcado!");  
            }  
            else {  
                cbMarcame.setText("Checkbox desmarcado!");  
            }  
        }  
    });
```


Control RadioButton (I)

(ver su API)

- ▶ Es un **control de entrada**.
- ▶ Permite que el usuario **pueda escoger una opción** entre un conjunto de ellas.
- ▶ Son **opciones excluyentes**.
- ▶ **Deben usarse** cuando **las opciones deben verse todas al mismo tiempo**. En **caso contrario es mejor usar un Spinner**.



ATTENDING?

☒ Yes ☐ Maybe ☐ No

Control RadioButton (II)

(ver su API)

- ▶ Los **radiobuttons** vienen en **grupos** de al menos dos:
 - ☞ para asociarlos hay que crearlos en un elemento **<RadioGroup>** y crear varios **<RadioButton>** dentro.
- ▶ El **RadioGroup** es una **subclase** de **LinearLayout** que tiene **orientación vertical** **por defecto**.

Control **RadioButton** (III)

(ver su API)

► *Ejemplo:*

```
<RadioGroup
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical" >

    <RadioButton
        android:id="@+id/radio_pirates"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:onClick="onRadioButtonClicked"
        android:text="@string/pirates" />

    <RadioButton
        android:id="@+id/radio_ninjas"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:onClick="onRadioButtonClicked"
        android:text="@string/ninjas" />
</RadioGroup>
```

¿Cómo saber cuál está pulsado y manejarlos desde JAVA?

► Método “**check(id)**”:

☺ **Marca la opción** determinada por el **ID**.

☺ *Ejemplo:*

```
RadioGroup rg = (RadioGroup)findViewById(R.id.gruporb);  
rg.check(R.id.radio1);
```

► Método “**clearCheck()**”:

☺ **Desmarca todas las opciones.**

☺ *Ejemplo:*

```
RadioGroup rg = (RadioGroup)findViewById(R.id.gruporb);  
rg.clearCheck();
```

¿Cómo saber cuál está pulsado y manejarlos desde JAVA?

- ▶ Método “**getCheckedRadioButtonId()**”:

☺ Devuelve el ID de la opción marcada o “-1” si no hay ninguna marcada.

☺ *Ejemplo:*

```
RadioGroup rg = (RadioGroup)findViewById(R.id.gruporb);  
rg.clearCheck();  
rg.check(R.id.radio1);  
int idSeleccionado = rg.getCheckedRadioButtonId();
```

Control **RadioButton**(VI)

Control de Eventos: **atributo “onClick”**

- ▶ Cuando **el usuario selecciona un radioButton**, el objeto **“RadioButton”** recibe un evento **“on-click”**.
- ▶ Se hace de **forma similar a los checkboxes**.
- ▶ **Añadiendo el atributo “onClick” y especificando un método.**

Control de Eventos: evento “onCheckedChanged”

- ▶ El evento “onCheckedChanged” informa de que ha cambiado el estado del **RadioGroup**.
- ▶ Es el mismo evento que en los “**CheckBoxes**”.
- ▶ Para **implementar las acciones de este evento** se suele usar la siguiente lógica:
 - ➡ se **asigna el escuchador** de eventos con el método “**setOnCheckedChangeListener**”
 - ➡ se **captura el evento** con el método “**onCheckedChanged**”
 - ➡ y, **dependiendo del RadioButton seleccionado** (variable **checkedId** recibida como **parámetro**), **haremos una acción u otra**.

Control RadioButton (VIII)

Control de Eventos: evento "onCheckedChanged"

- ▶ Veámoslo con este código que cambia el texto de una etiqueta según qué RadioButton esté seleccionado:

```
private TextView lblMensaje;  
private RadioGroup rgOpciones;  
  
//...  
  
lblMensaje = (TextView)findViewById(R.id.LblSeleccion);  
rgOpciones = (RadioGroup)findViewById(R.id.gruporb);  
  
rgOpciones.setOnCheckedChangeListener(  
    new RadioGroup.OnCheckedChangeListener() {  
        public void onCheckedChanged(RadioGroup group, int checkedId) {  
            lblMensaje.setText("ID opción seleccionada: " + checkedId);  
        }  
    });
```