17. Escuchadores de eventos

En la unidad anterior estudiamos una forma sencilla de ligar un método a la pulsación de un botón mediante el atributo `android:onClick` del botón. En esta unidad veremos una forma más general de añadir funcionalidad a las vistas mediante la implementación de interfaces.

La clase `View` posee una colección de interfaces denominadas escuchadores de eventos. Como sabes, las interfaces en Java son clases cuyos métodos no tienen cuerpo. Las clases que implementen la interfaz deben suministrar necesariamente una versión de cada método de la interfaz.

Los escuchadores de eventos contienen un único método, denominado método callback. Por ejemplo, la interfaz `View.onClickListener` declara el método `onClick()` que se llama automáticamente cuando el usuario toca la vista, por ejemplo.

17.1 Escuchando mediante clases con nombre

Para ilustrar la utilización de los escuchadores, vamos a programar una versión de nuestro juego funcionalmente equivalente a la de la unidad anterior pero con un escuchador en lugar de utilizar el atributo `android:onClick`. El fichero Java es el siguiente:

/src/MainActivity.java

```
package es.uam.eps.android.ccc17_1;

import android.app.Activity;
...
import android.widget.RadioButton;

public class MainActivity extends Activity {

    private int SIZE = 7;
    private RadioButton button5;
    private RadioButton button10;
    private RadioButton button17;

    private final int ids [][] = {
        {0, 0, R.id.f1, R.id.f2, R.id.f3, 0, 0},
        {0, 0, R.id.f4, R.id.f5, R.id.f6, 0, 0},
        {R.id.f7, R.id.f8, R.id.f9, R.id.f10, R.id.f11, R.id.f12, R.id.f13},
        {R.id.f14, R.id.f15, R.id.f16, R.id.f17, R.id.f18, R.id.f19, R.id.f20},
        {R.id.f21, R.id.f22, R.id.f23, R.id.f24, R.id.f25, R.id.f26, R.id.f27},
        {0, 0, R.id.f28, R.id.f29, R.id.f30, 0, 0},
        {0, 0, R.id.f31, R.id.f32, R.id.f33, 0, 0}};

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        setFigure();

        button5 = (RadioButton) findViewById(R.id.f5);
        button10 = (RadioButton) findViewById(R.id.f10);
        button17 = (RadioButton) findViewById(R.id.f17);
        button5.setOnClickListener(listener);
```

```
    }

    private void setFigure (){
        RadioButton button;

        for (int i=0; i<SIZE; i++)
          for (int j=0; j<SIZE; j++)
            if (ids[i][j]!=0){
              button = (RadioButton) findViewById(ids[i][j]);
              if (ids[i][j] != R.id.f17)
                button.setChecked(true);
            }
    }
    private OnClickListener listener = new OnClickListener() {
        public void onClick(View v) {
            RadioButton button = (RadioButton) v;
            button.setChecked(false);
            button10.setChecked(false);
            button17.setChecked(true);
        }
    };
}
```

La diferencia entre la versión del método `onCreate()` de esta unidad y la versión de la unidad anterior es la asignación de un escuchador al botón `button5`. En general, para que el método callback del escuchador de eventos se ejecute al hacer click en un elemento de la interfaz, el escuchador debe registrarse para dicho elemento:

```
button5.setOnClickListener(listener);
```

El escuchador `listener` es una interfaz de tipo `View.OnClickListener` y, por lo tanto, a la vez que se instancia es necesario implementar el método callback `onClick()`, que lleva a cabo la misma tarea que el método `onRadioButtonClick()` de la unidad anterior. Básicamente, como ya vimos en la unidad anterior, el método `setChecked()` se utiliza para ajustar el estado de los botones y simular el salto de la ficha 5 sobre la ficha 10:

```
private OnClickListener listener = new OnClickListener() {
    public void onClick(View v) {
        RadioButton button = (RadioButton) v;
        button.setChecked(false);
        button10.setChecked(false);
        button17.setChecked(true);
    }
};
```

El fichero de diseño del proyecto `CCC17_1` es el mismo que el de la unidad anterior (`activity_main.xml` del proyecto `CCC16`).

17.2 Escuchando mediante clases anónimas

Podemos ahorrarnos la declaración de la interfaz `listener` instanciándola directamente en la lista de argumentos de `setOnClickListener()`:

```
button5.setOnClickListener(new OnClickListener() {

    @Override
    public void onClick(View v) {
        // TODO Auto-generated method stub
       RadioButton button = (RadioButton) v;
       button.setChecked(false);
       button10.setChecked(false);
       button17.setChecked(true);
    }
});
```

Quizá recuerdes que en Java a este tipo de clases se les denomina internas anónimas. Aunque al principio esta sintaxis resulta un poco intrincada, es una práctica bastante habitual en Android que tiene la ventaja de situar el código callback justo donde se necesita. Es el estilo recomendado cuando el escuchador solo se registra en una vista.

El proyecto `CCC17_2` utiliza este enfoque. El código completo de la actividad principal es el siguiente:

/src/MainActivity.java

```
package es.uam.eps.android.ccc17_2;

import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.RadioButton;

public class MainActivity extends Activity {

     private int SIZE = 7;
     private RadioButton button5;
     private RadioButton button10;
     private RadioButton button17;

      private final int ids [][] = {
        {0, 0, R.id.f1, R.id.f2, R.id.f3, 0, 0},
        {0, 0, R.id.f4, R.id.f5, R.id.f6, 0, 0},
        {R.id.f7, R.id.f8, R.id.f9, R.id.f10, R.id.f11, R.id.f12, R.id.f13},
        {R.id.f14, R.id.f15, R.id.f16, R.id.f17, R.id.f18, R.id.f19, R.id.f20},
        {R.id.f21, R.id.f22, R.id.f23, R.id.f24, R.id.f25, R.id.f26, R.id.f27},
        {0, 0, R.id.f28, R.id.f29, R.id.f30, 0, 0},
        {0, 0, R.id.f31, R.id.f32, R.id.f33, 0, 0}};

     /** Called when the activity is first created. */
     @Override
     public void onCreate(Bundle savedInstanceState) {
         super.onCreate(savedInstanceState);
         setContentView(R.layout.activity_main);
         setFigure();

         button5 = (RadioButton) findViewById(R.id.f5);
         button10 = (RadioButton) findViewById(R.id.f10);
         button17 = (RadioButton) findViewById(R.id.f17);
```

```
        button5.setOnClickListener(new OnClickListener() {

            @Override
            public void onClick(View v) {
                // TODO Auto-generated method stub
                RadioButton button = (RadioButton) v;
                button.setChecked(false);
                button10.setChecked(false);
                button17.setChecked(true);
            }
        });
    }

    private void setFigure() {
        RadioButton button;

        for (int i = 0; i < SIZE; i++)
            for (int j = 0; j < SIZE; j++)
                if (ids[i][j] != 0) {
                    button = (RadioButton) findViewById(ids[i][j]);
                    if (ids[i][j] != R.id.f17)
                        button.setChecked(true);
                }
    }
}
```

17.3 La actividad también escucha

Otra forma de esquivar la declaración del objeto `listener` consiste en hacer que la actividad implemente la interfaz `OnClickListener`. Veamos cómo hacer esto con el proyecto `CCC17_3`:

/src/MainActivity.java

```
package es.uam.eps.android.ccc17_3;

import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.RadioButton;

public class MainActivity extends Activity implements OnClickListener {

    private int SIZE = 7;
    private RadioButton button5;
    private RadioButton button10;
    private RadioButton button17;

    private final int ids [][] = {
                {0, 0, R.id.f1, R.id.f2, R.id.f3, 0, 0},
                {0, 0, R.id.f4, R.id.f5, R.id.f6, 0, 0},
                {R.id.f7, R.id.f8, R.id.f9, R.id.f10, R.id.f11, R.id.f12, R.id.f13},
                {R.id.f14, R.id.f15, R.id.f16, R.id.f17, R.id.f18, R.id.f19, R.id.f20},
                {R.id.f21, R.id.f22, R.id.f23, R.id.f24, R.id.f25, R.id.f26, R.id.f27},
                {0, 0, R.id.f28, R.id.f29, R.id.f30, 0, 0},
                {0, 0, R.id.f31, R.id.f32, R.id.f33, 0, 0}};
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        setFigure();

        button5 = (RadioButton) findViewById(R.id.f5);
        button10 = (RadioButton) findViewById(R.id.f10);
        button17 = (RadioButton) findViewById(R.id.f17);
        button5.setOnClickListener(this);
    }
```

```
    private void setFigure (){
        RadioButton button;

        for (int i=0; i<SIZE; i++)
          for (int j=0; j<SIZE; j++)
            if (ids[i][j]!=0){
              button = (RadioButton) findViewById(ids[i][j]);
                if (ids[i][j] != R.id.f17)
                  button.setChecked(true);
            }
    }

    @Override
    public void onClick(View v) {
        // TODO Auto-generated method stub
        RadioButton button = (RadioButton) v;
        button.setChecked(false);
        button10.setChecked(false);
        button17.setChecked(true);
    }

}
```

Ahora es la actividad la que juega el papel del escuchador `listener` del proyecto `CCC17_1`, es decir, la que implementa el método `onClick()` de la interfaz y, por lo tanto, la que se registra como escuchador del botón 5:

```
button5.setOnClickListener(this);
```