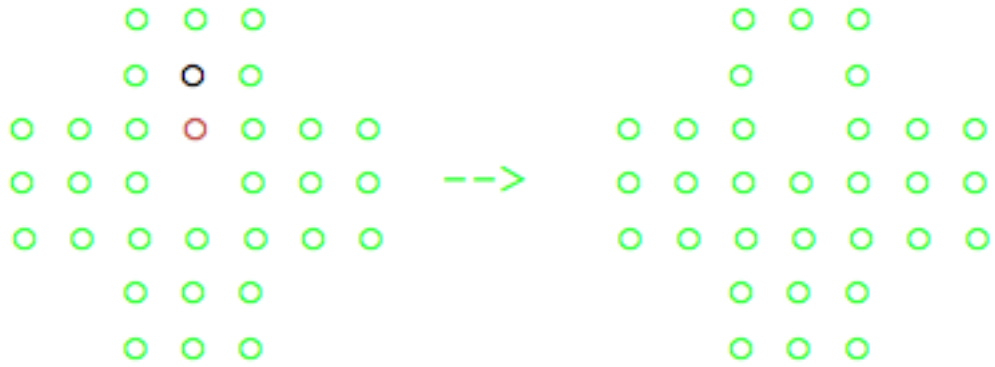
16. El atributo android:onclick

¿Cómo podemos hacer que nuestra aplicación ejecute una tarea en respuesta a eventos de la interfaz de usuario tal y como la pulsación de uno de los botones? Existen distintas formas de resolver esta cuestión:

- Asignando el atributo `android:onClick` del botón. El valor de este atributo es el nombre del método que se ejecuta al pulsar el botón.
- Implementando un escuchador de eventos (*event listener*) y registrándolo en el botón.
- Haciendo que la actividad implemente la interfaz escuchador de eventos.

La forma más sencilla es la primera y se ilustra en el proyecto de esta unidad, `CCC16`, en el que vamos a empezar a gestionar los clicks del jugador sobre el tablero programando un método y asignándoselo al botón número 5 (el de color negro en la figura inferior). Como resultado de la pulsación el estado de los botones 5 y 10 debe pasar a `false`, y a `true` el del botón 17, como se ve en la figura de la derecha:



El fichero de diseño del tablero es prácticamente igual al de la unidad 13 (proyecto `CCC13`) salvo por estas dos diferencias:

- Hemos eliminado las líneas correspondientes al atributo `android:checked`, que especifica el estado del botón, pues ahora fijaremos el estado de los botones con código Java.
- El botón con identificador `f5` tiene su atributo `android:onClick` asignado al método `onRadioButtonClick()`. Este es el método que se ejecutará al pulsar el botón:

```
<RadioButton
    android:id="@+id/f5"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@id/f1"
    android:layout_toRightOf="@id/f4"
    android:onClick="onRadioButtonClick"/>
```

Este es el aspecto de las dos primeras líneas de botontes del nuevo fichero de diseño:

/res/layout/activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >
    <RadioButton
        android:id="@+id/f2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"/>
    <RadioButton
        android:id="@+id/f1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_toLeftOf="@id/f2"/>
    <RadioButton
        android:id="@+id/f3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_toRightOf="@id/f2"/>

    <RadioButton
        android:id="@+id/f4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@id/f1"
        android:layout_alignLeft="@id/f1"/>
    <RadioButton
        android:id="@+id/f5"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@id/f1"
        android:layout_toRightOf="@id/f4"
        android:onClick="onRadioButtonClick"/>
    <RadioButton
        android:id="@+id/f6"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@id/f1"
        android:layout_toRightOf="@id/f5"/>
    ...

</RelativeLayout>
```

El código de `MainActivity.java` es el siguiente:

/src/MainActivity.java

```
package es.uam.eps.android.ccc3;

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

publicclass MainActivity extends Activity {
    int SIZE = 7;

    private RadioButton button5;
    private RadioButton button10;
    private RadioButton button17;

    privatefinalint 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
    publicvoid 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);
    }

    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);
                  }
    }

    public void onRadioButtonClick(View view) {
         button5.setChecked(false);
         button10.setChecked(false);
         button17.setChecked(true);
    }

}
```

El movimiento de las fichas se va a simular mediante el cambio de su estado que ya no se fija en el fichero de diseño. Para cambiar el estado dinámicamente se utiliza el método `setChecked()`. Este método hemos de invocarlo desde una referencia al botón correspondiente. Estas referencias se consiguen llamando al método `findViewById()` de la actividad, al que hay que pasar el identificador del botón correspondiente.

Los identificadores de los botones se definen en el fichero de diseño y también se almacenan en un array de enteros de nombre `ids`. Cada elemento de este array, de dimensión 7x7, se corresponde con una posición del tablero, lo cual nos permitirá en unaidades posteriores obtener fácilmente las coordenadas de un botón en el tablero a partir de su identificador.

En el método `onCreate()` de `MainActivity`, una vez inflada la interfaz especificada en el fichero de diseño, llamamos al método `setFigure()` y conseguimos referencias a los botones 5, 10 y 17:

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

La referencia devuelta por el método `findViewById()` es de tipo `View` y la convertimos mediante un cast al tipo `RadioButton`.

El método `setFigure()` es el encargado de poner el estado de todos los botones a `true` salvo el central (`R.id.f17`):

```
privatevoid 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);
          }
}
```

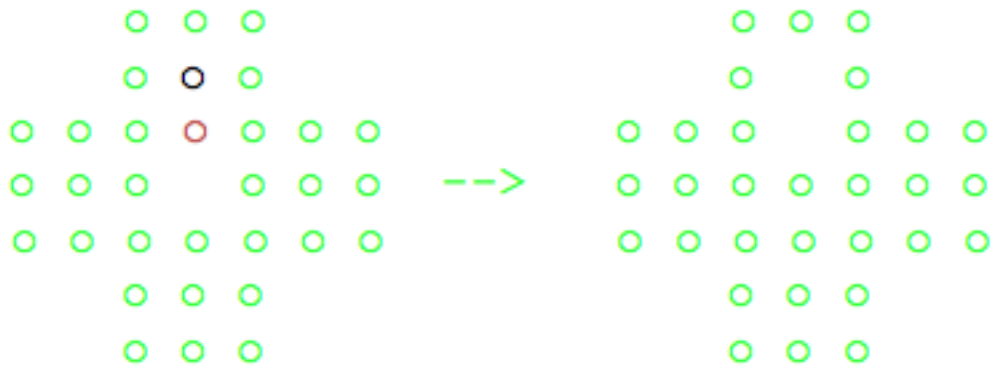
Una vez que disponemos de una referencia al objeto `RadioButton`, ajustamos su estado a `true` (marcado), siempre que no se trate del botón central:

```
if (ids[i][j] != R.id.f17)
    button.setChecked(true);
```

Solo nos falta por estudiar el método `onRadioButtonClicked()`, el método que se ejecuta al pulsar el botón con identificador `R.id.f5`. Este método ajusta a `false` los estados de los botones 5 y 10, y a `true` el estado del botón 17:

```
Public void onRadioButtonClick(View view) {
    button5.setChecked(false);
    button10.setChecked(false);
    button17.setChecked(true);
}
```

De esta manera tan sencilla simulamos el salto de la ficha 5 sobre la ficha 10, como se observa en la parte derecha de la siguiente figura:



Como ves, todavía estamos lejos de poder echar una partida al solitario. Para poder jugar de verdad hemos de dotar de funcionalidad al resto de los botones e implementar la lógica del juego, como veremos en unidades posteriores.