2
Construcción de aplicaciones interactivas

Una aplicación consejera sobre cerveza

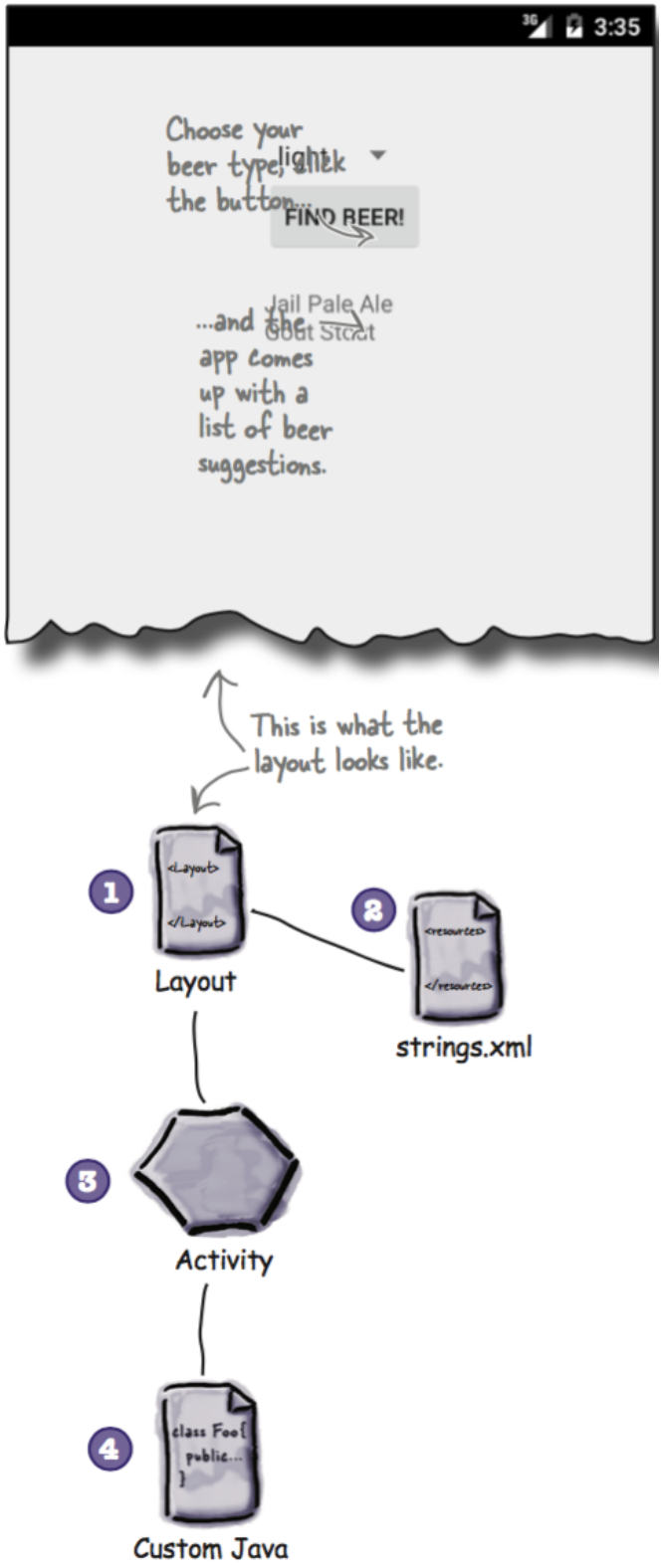
1. **The layout specifies what the app will look like.**
It includes three GUI components:
   - A drop-down list of values called a spinner, which allows the user to choose which type of beer they want.
   - A button that when pressed will return a selection of beer types.
   - A text field that displays the types of beer.

2. **The file strings.xml includes any string resources needed by the layout—for example, the label of the button specified in the layout.**

3. **The activity specifies how the app should interact with the user.**
It takes the type of beer the user chooses, and uses this to display a list of beers the user might be interested in. It achieves this with the help of a custom Java class.

4. **The custom Java class contains the application logic for the app.**
It includes a method that takes a type of beer as a parameter, and returns a list of beers of this type. The activity calls the method, passes it the type of beer, and uses the response.

Una aplicación consejera sobre cerveza



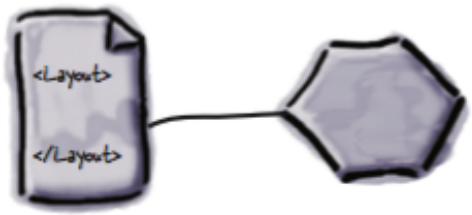
Pasos para la construcción de la aplicación

1 **Create a project.**
You're creating a brand-new app, so you'll need to create a new project. Just like before, you'll need to create a basic layout and activity.

2 **Update the layout.**
Once you have a basic app set up, you need to amend the layout so that it includes all the GUI components your app needs.

Pasos para la construcción de la aplicación

3 **Wire the layout to the activity.**
The layout only creates the visuals. To add smarts to your app, you need to wire the layout to the Java code in your activity.

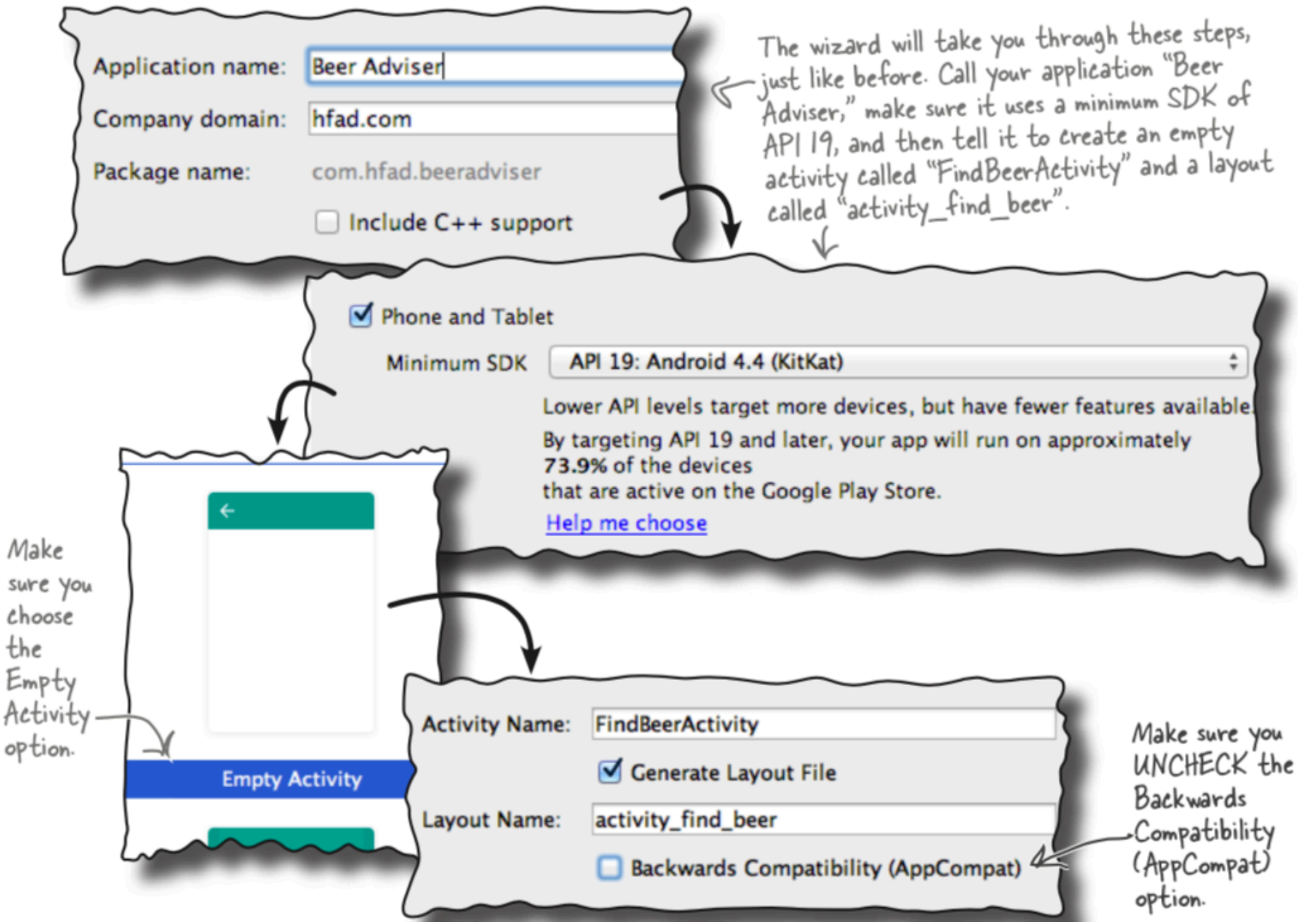


4 **Write the application logic.**
You'll add a Java custom class to the app, and use it to make sure users get the right beer based on their selection.

Creación del proyecto

1. Open Android Studio and choose "Start a new Android Studio project" from the welcome screen. This starts the wizard you saw in Chapter 1.
2. When prompted, enter an application name of "Beer Adviser" and a company domain of "hfad.com", making your package name `com.hfad.beeradviser`. Make sure you uncheck the option to include C++ support.
3. We want the app to work on most phones and tablets, so choose a minimum SDK of API 19, and make sure the option for "Phone and Tablet" is selected. This means that any phone or tablet that runs the app must have API 19 installed on it as a minimum. Most Android devices meet this criterion.
4. Choose an empty activity for your default activity. Call the activity "FindBeerActivity" and the accompanying layout "activity_find_beer". Make sure the option to generate the layout is selected and you uncheck the Backwards Compatibility (AppCompat) option.

Creación del proyecto



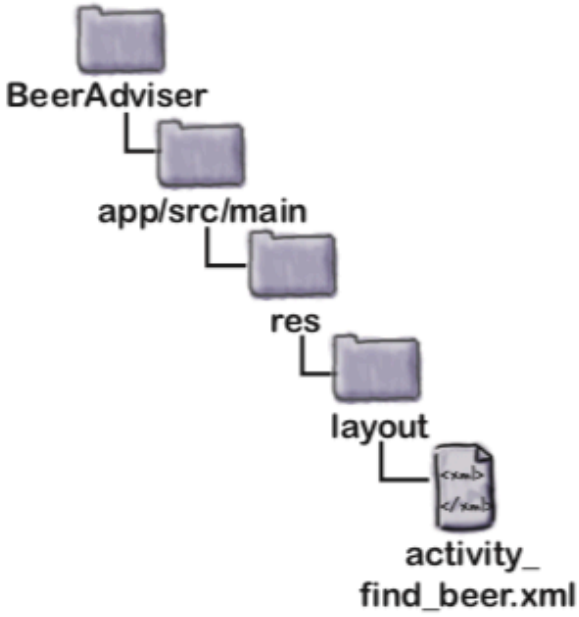
Cambiando el diseño

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="16dp"
    android:orientation="vertical"
    tools:context="com.hfad.beeradviser.FindBeerActivity">

    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="This is a text view" />
</LinearLayout>
```

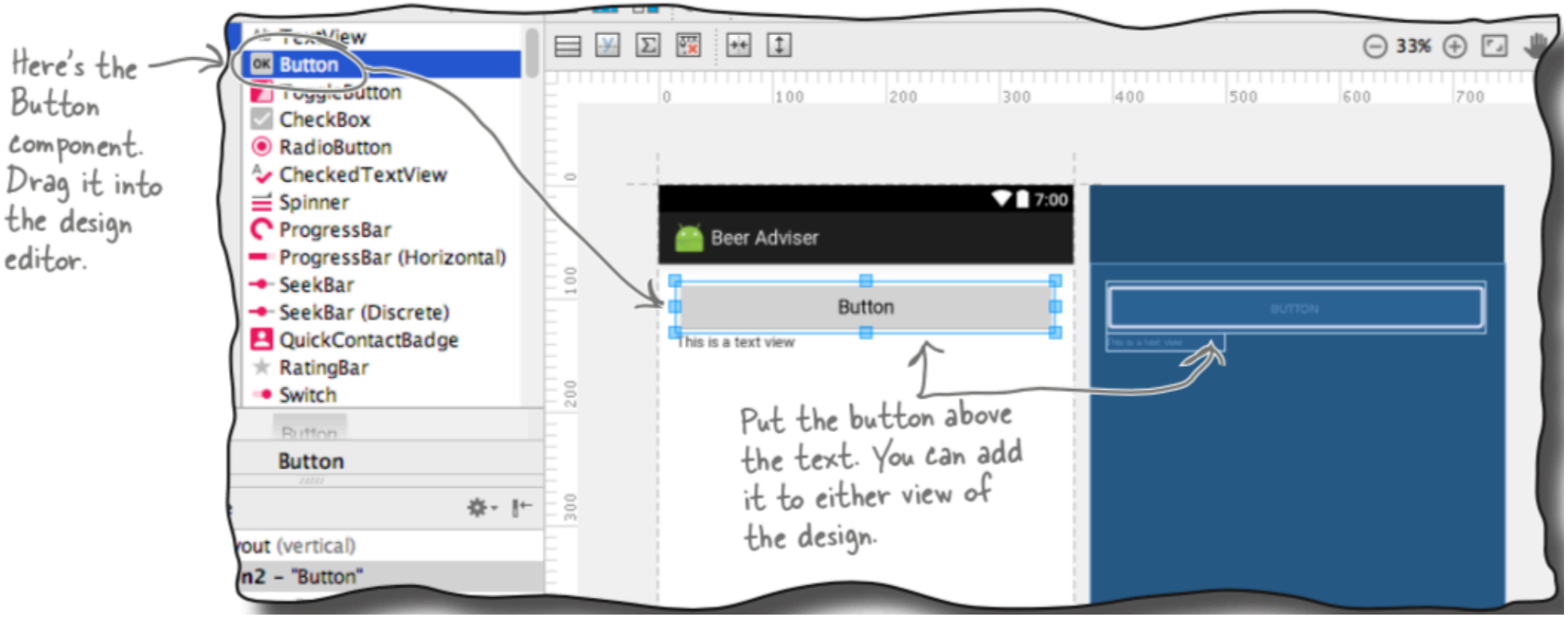
We're replacing the code Android Studio generated for us.

These elements relate to the layout as a whole. They determine the layout width and height, any padding in the layout margins, and whether components should be laid out vertically or horizontally.

This is used to display text.



Agregando un botón usando el editor de diseño

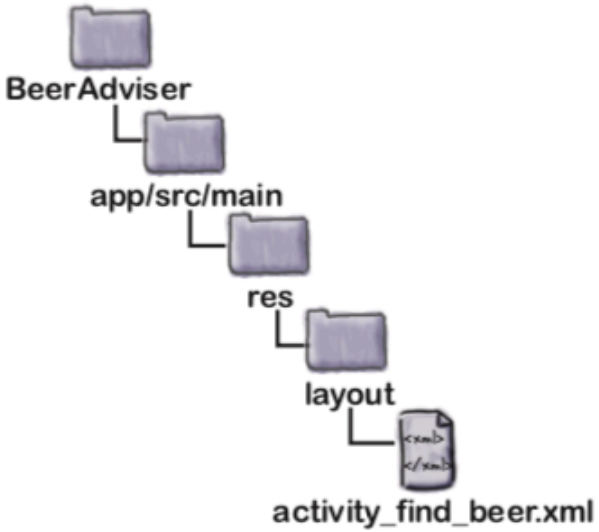


Cambios en el diseño

```
...
<Button
    android:id="@+id/button"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Button" />

<TextView
    android:id="@+id/textView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="This is a text view" />
...
```

There's a new <Button> element that describes the new button you've dragged to the layout. We'll look at this in more detail over the next few pages.

The code the design editor adds depends on where you place the button, so don't worry if your code looks different from ours.



Button y TextView son subclases de View

android:id

This gives the component an identifying name. The `ID` property enables you to control what components do via activity code, and also allows you to control where components are placed in the layout:

```
android:id="@+id/button"
```

android:text

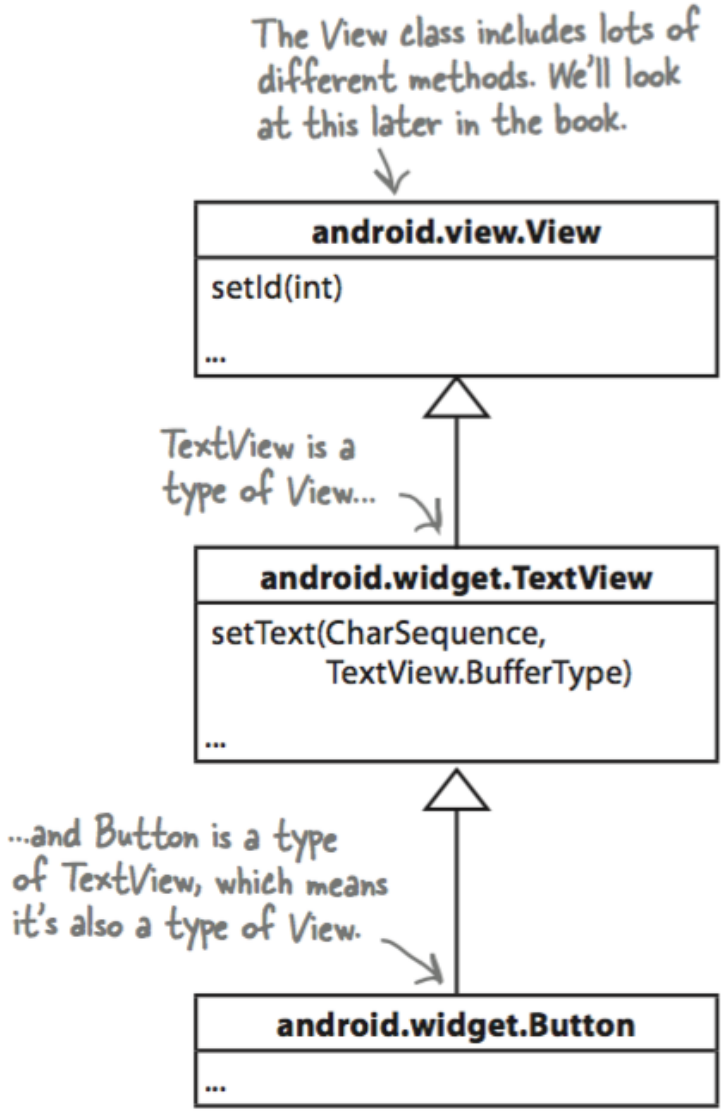
This tells Android what text the component should display. In the case of `<Button>`, it's the text that appears on the button:

```
android:text="New Button"
```

android:layout_width, android:layout_height

These properties specify the basic width and height of the component. `"wrap_content"` means it should be just big enough for the content:

```
android:layout_width="wrap_content"
android:layout_height="wrap_content"
```



El código del layout

The <LinearLayout> element

```
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="16dp"
    android:orientation="vertical"
    tools:context="com.hfad.beeradviser.FindBeerActivity">
```

This is the button.

```
    <Button
        android:id="@+id/button"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Button" />
```

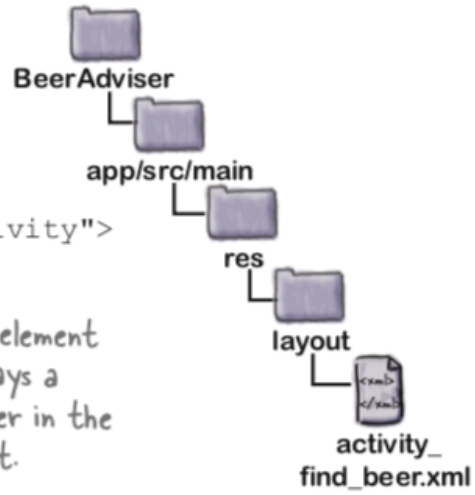
This is the text view.

```
    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="This is a text view" />
```

```
</LinearLayout>
```

← This closes the <LinearLayout> element.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="16dp"
    android:orientation="vertical"
    tools:context="com.hfad.beeradviser.FindBeerActivity">

    <Spinner
        android:id="@+id/color"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="40dp"
        android:layout_gravity="center"
        android:layout_margin="16dp" />

    <Button
        android:id="@+id/button find_beer"
        android:layout_width="match_parent wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:layout_margin="16dp"
        android:text="Button" />

    <TextView
        android:id="@+id/textView brands"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:layout_margin="16dp"
        android:text="This is a text view" />
</LinearLayout>
```



A spinner is the Android name for a drop-down list of values. It allows you to choose a single value from a selection.

This element displays a spinner in the layout.

Change the button's ID to "find_beer". We'll use this later.

Center the button horizontally and give it a margin.

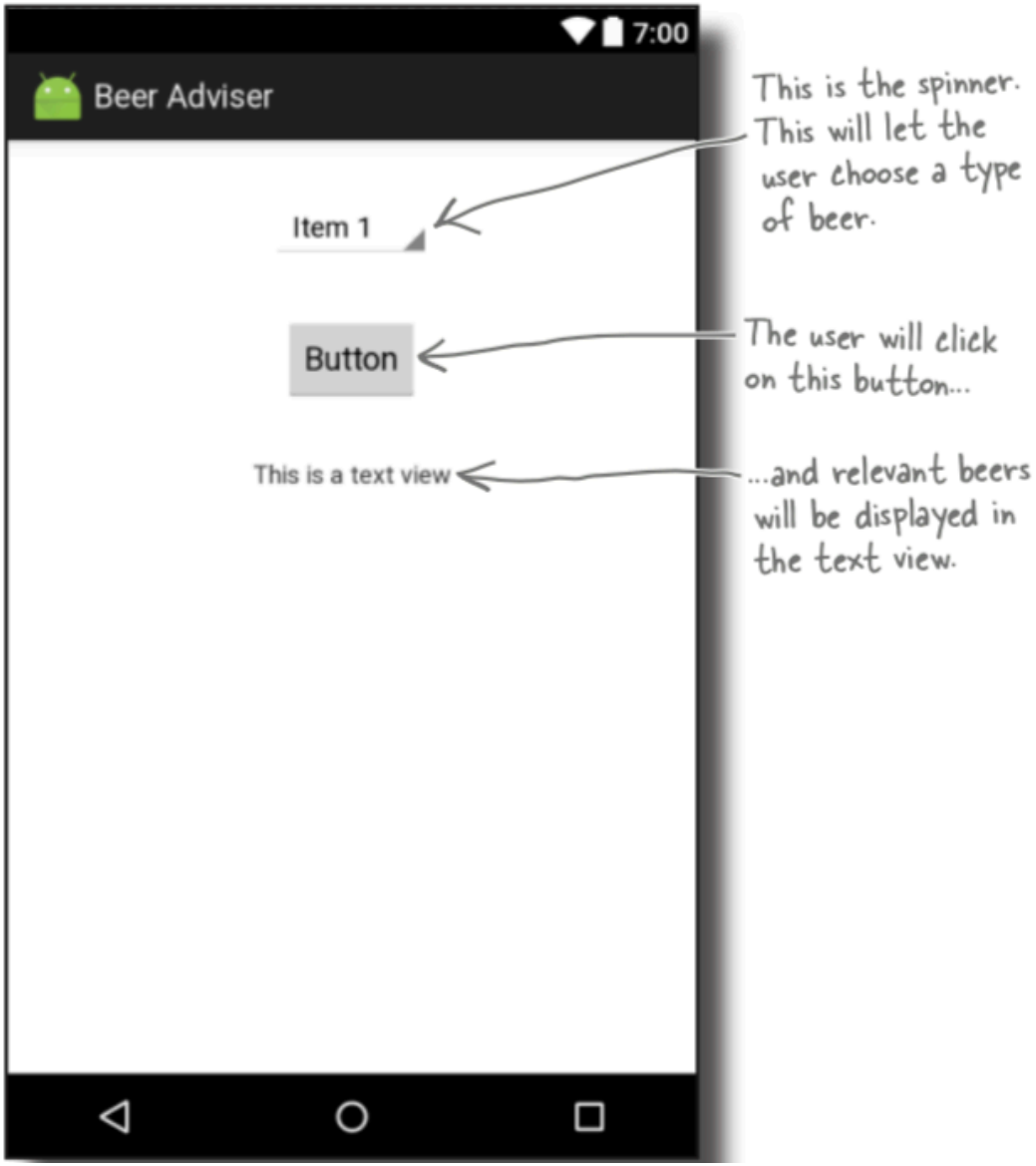
Change the button's width so it's as wide as its content.

Change the text view's ID to "brands".

Center the text view and apply a margin.

Cambios en el layout (XML)

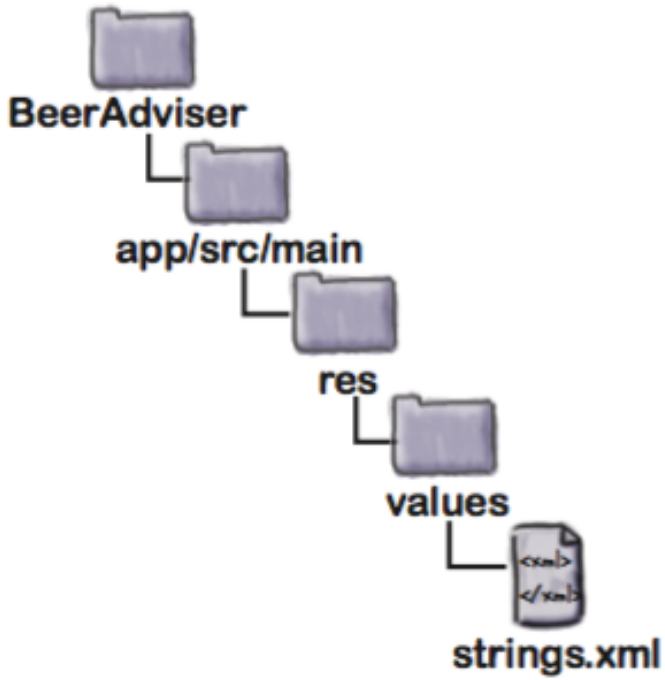
Cambios en el editor de diseño



A spinner provides a drop-down list of values. It allows you to choose a single value from a set of values.

GUI components such as buttons, spinners, and text views have very similar attributes, as they are all types of View. Behind the scenes, they all inherit from the same Android View class.

Cadenas como recursos en vez de codificación dura

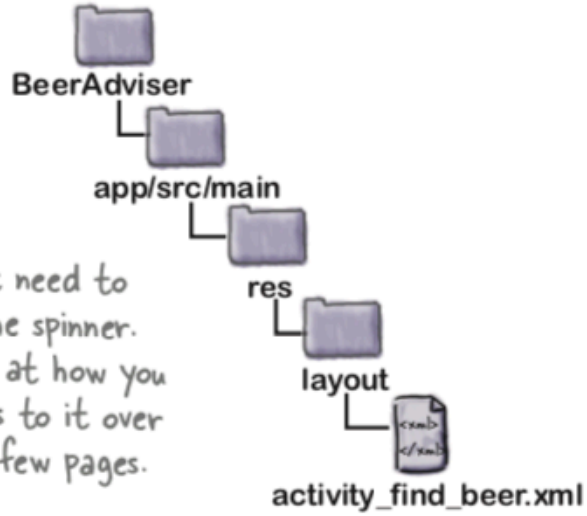



```
<resources>
    <string name="app_name">Beer Adviser</string>
    <string name="find_beer">Find Beer!</string>
    <string name="brands">No beers selected</string>
</resources>
```

This will be the default text in the text view.

```
...
    <Spinner
        android:id="@+id/color"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="40dp"
        android:layout_gravity="center"
        android:layout_margin="16dp" />

    <Button
        android:id="@+id/find_beer"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:layout_margin="16dp"
        android:text="Button@string/find_beer" />

    <TextView
        android:id="@+id/brands"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:layout_margin="16dp"
        android:text="This is a text view@string/brands" />

</LinearLayout>
```

We didn't need to change the spinner. We'll look at how you add values to it over the next few pages.



Cambios para utilizar los recursos cadenas

Delete the hardcoded text.

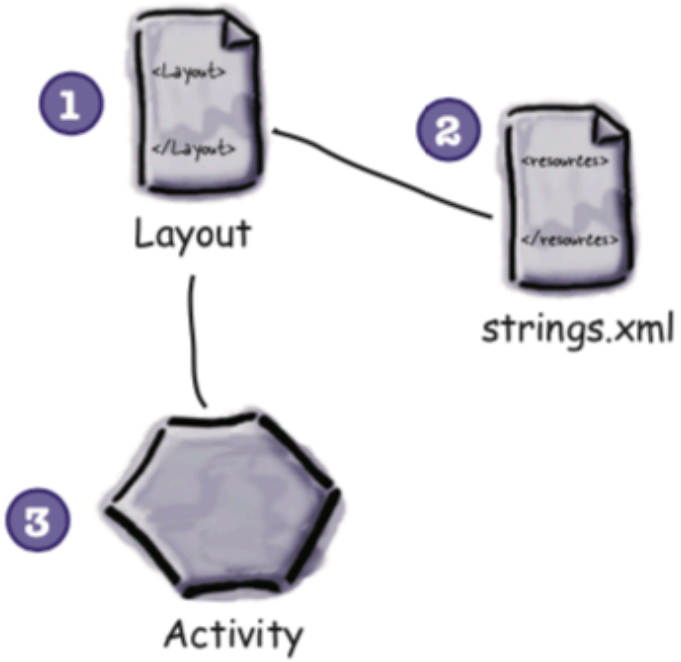
This will display the value of the find_beer String resource on the button.

Delete this hardcoded text too.

This will display the value of the brands String resource in the text view.

Lo hecho hasta ahora …

1. **We've created a layout that specifies what the app looks like.**
   It includes a spinner, a button, and a text view.
2. **The file strings.xml includes the String resources we need.**
   We've added a label for the button, and default text for the list of suggested beer brands to try.
3. **The activity specifies how the app should interact with the user.**
   Android Studio has created an activity for us, but we haven't done anything with it yet.



Agregando valores al Spinner

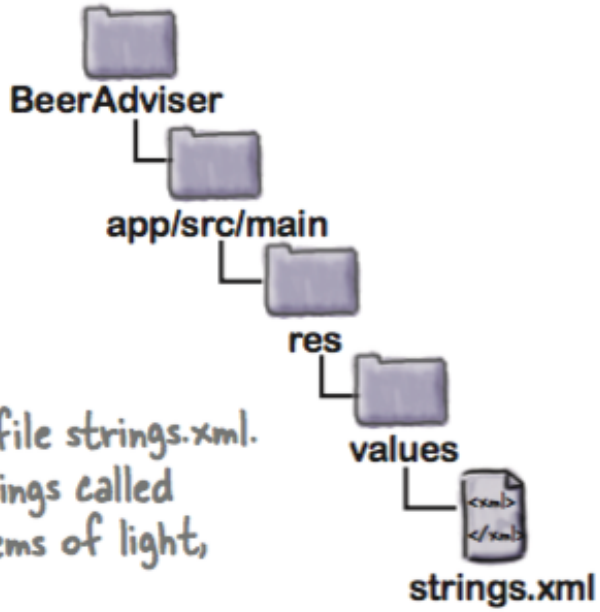
```
<string-array name="string_array_name">
    <item>string_value1</item>
    <item>string_value2</item>
    <item>string_value3</item>
    ...
</string-array>
```

This is the name of the array.

These are the values in the array. You can add as many as you need.

```
...
    <string name="brands"></string>
    <string-array name="beer_colors">
        <item>light</item>
        <item>amber</item>
        <item>brown</item>
        <item>dark</item>
    </string-array>
</resources>
```

Add this string-array to file strings.xml. It defines an array of strings called beer_colors with array items of light, amber, brown, and dark.



Asignación de un string-array al Spinner

```
...
    <Spinner
        android:id="@+id/color"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="40dp"
        android:layout_gravity="center"
        android:layout_margin="16dp"
        android:entries="@array/beer_colors" />
...
```

This means "the entries for the spinner come from array beer_colors".

Prueba


11:51
Beer Adviser
By default the top item in the spinner is selected.
light
FIND BEER!
11:52
Beer Adviser
Click on the spinner to see its entries.
light
amber
brown
dark
11:53
Beer Adviser
When you click on a value, it gets selected.
amber
FIND BEER!

Acciones para el botón

1. **The user chooses a type of beer from the spinner.**
   The user clicks on the button to find matching beers.

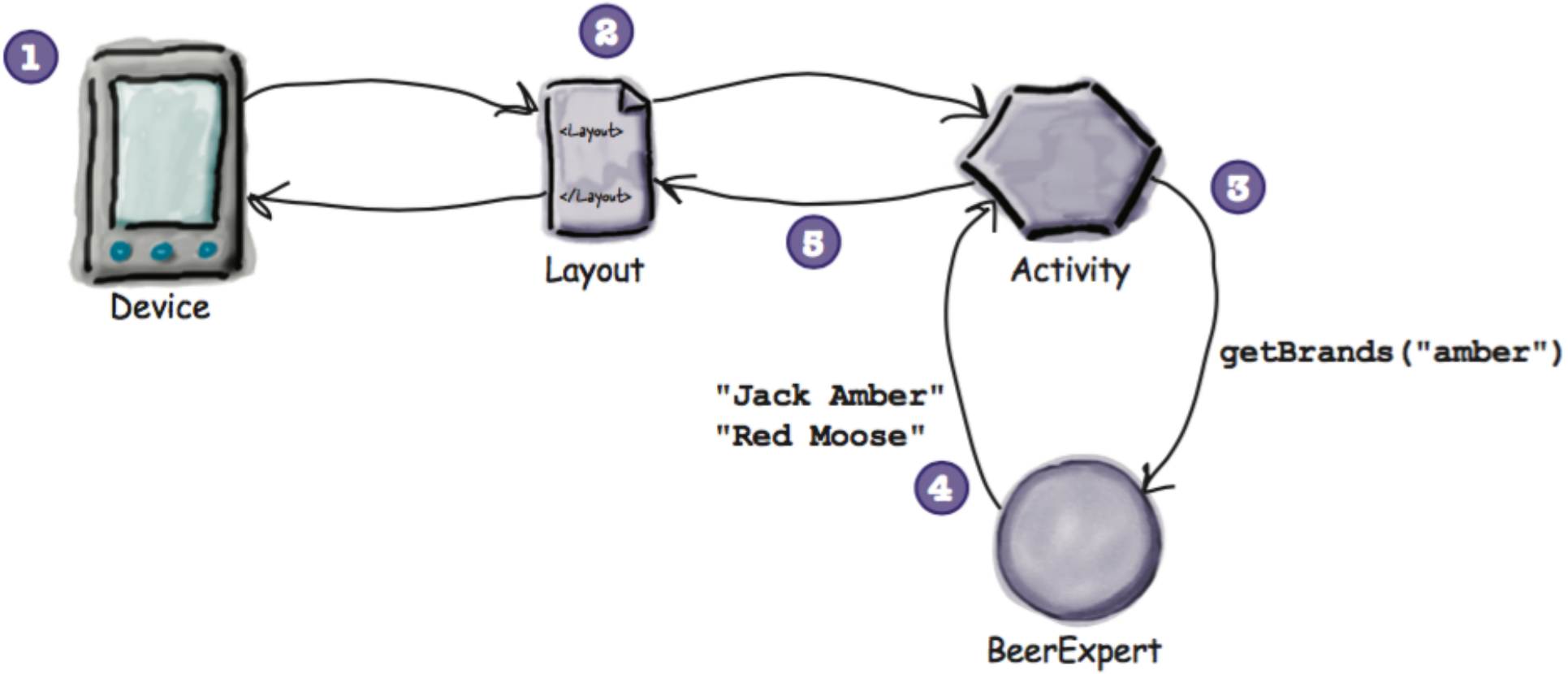
2. **The layout specifies which method to call in the activity when the button is clicked.**

3. **The method in the activity retrieves the value of the selected beer in the spinner and passes it to the getBrands() method in a Java custom class called BeerExpert.**

4. **BeerExpert's getBrands() method finds matching brands for the type of beer and returns them to the activity as an ArrayList of Strings.**

5. **The activity gets a reference to the layout text view and sets its text value to the list of matching beers.**
   This is displayed on the device.

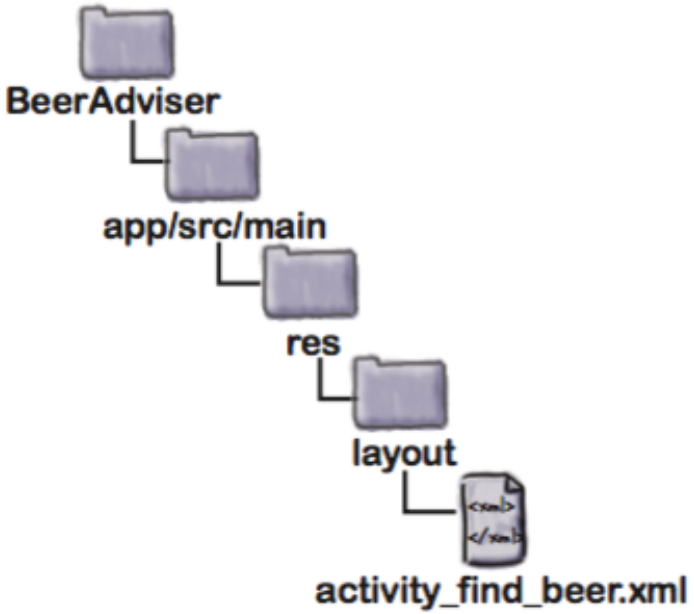
Acciones para el botón



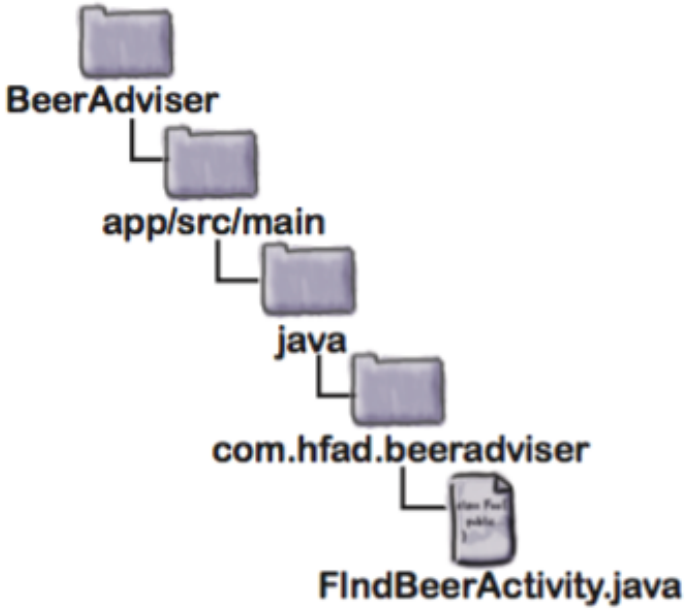
Acciones para el botón

```
...
    <Button
        android:id="@+id/find_beer"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignLeft="@+id/color"
        android:layout_below="@+id/color"
        android:text="@string/find_beer"
        android:onClick="onClickFindBeer" />
...
```

When the button is clicked, call method onClickFindBeer() in the activity. We'll create the method in the activity over the next few pages.



Acciones para el botón




```
package com.hfad.beeradviser;

import android.os.Bundle;
import android.app.Activity;

public class FindBeerActivity extends Activity {

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

The class extends the Android Activity class.

This is the onCreate() method. It's called when the activity is first created.

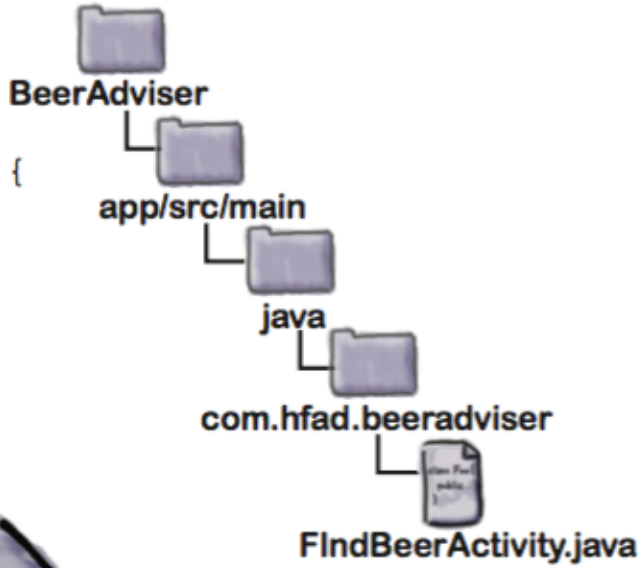
setContentView tells Android which layout the activity uses. In this case, it's activity_find_beer.

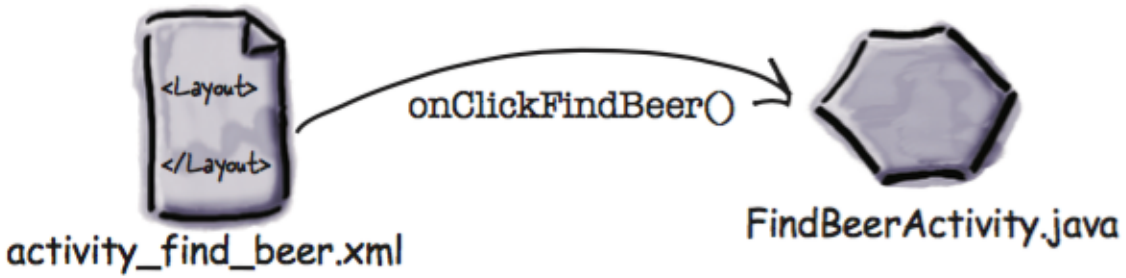
Acciones para el botón

We're using this class, so we need to import it.

```
...
import android.view.View;

public class FindBeerActivity extends Activity {
...
    //Call when the user clicks the button
    public void onClickFindBeer(View view){
    }
}
```

Add the onClickFindBeer() method to FindBeerActivity.java.

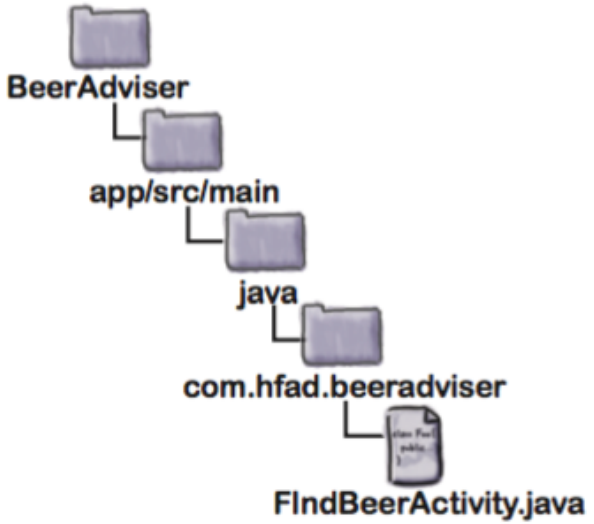




Acciones para el botón

```
//Call when the button gets clicked
public void onClickFindBeer( View view) {

    //Get a reference to the TextView
    TextView brands = (TextView) findViewById( R.id.brands );

    //Get a reference to the Spinner
    Spinner color = (Spinner) findViewById( R.id.color );

    //Get the selected item in the Spinner
    String beerType = String.valueOf(color. getSelectedItem() );

    //Display the selected item
    brands. setText (beerType);
}
```

Acciones para el botón




```
package com.hfad.beeradviser;

import android.os.Bundle;
import android.app.Activity;
import android.view.View;
import android.widget.Spinner;
import android.widget.TextView;

public class FindBeerActivity extends Activity {

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

    //Call when the button gets clicked
    public void onClickFindBeer(View view) {
        //Get a reference to the TextView
        TextView brands = (TextView) findViewById(R.id.brands);
        //Get a reference to the Spinner
        Spinner color = (Spinner) findViewById(R.id.color);
        //Get the selected item in the Spinner
        String beerType = String.valueOf(color.getSelectedItem());
        //Display the selected item
        brands.setText(beerType);
    }
}
```

We're using these extra classes.

We've not changed this method.

findViewById returns a View. You need to cast it to the right type of View.

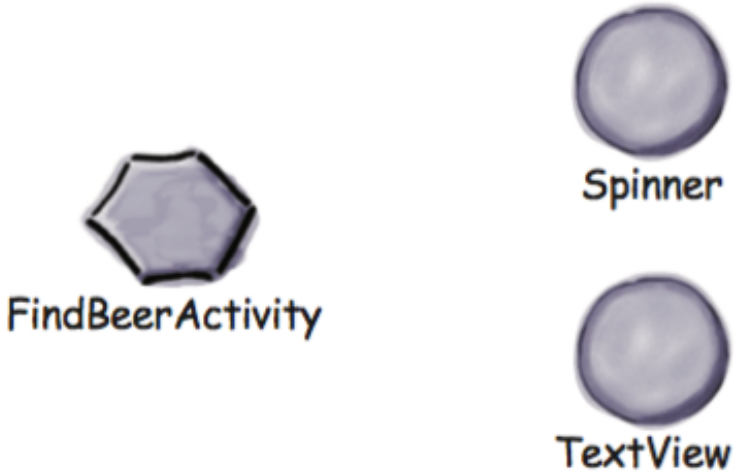
getSelectedItem returns an Object. You need to turn it into a String.

Que hace el código …

1. **The user chooses a type of beer from the spinner and clicks on the Find Beer button. This calls the public void onClickFindBeer(View) method in the activity.**

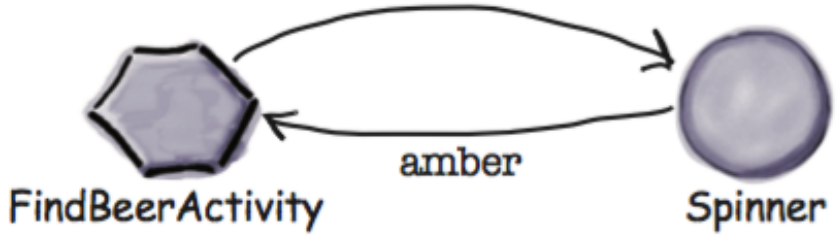
The layout specifies which method in the activity should be called when the button is clicked via the `android:onClick` property of the button.

2. **The activity gets references to the TextView and Spinner GUI components using calls to the findViewById() method.**

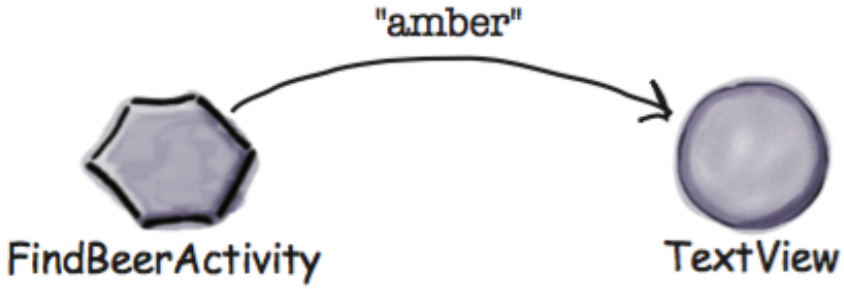


Que hace el código …

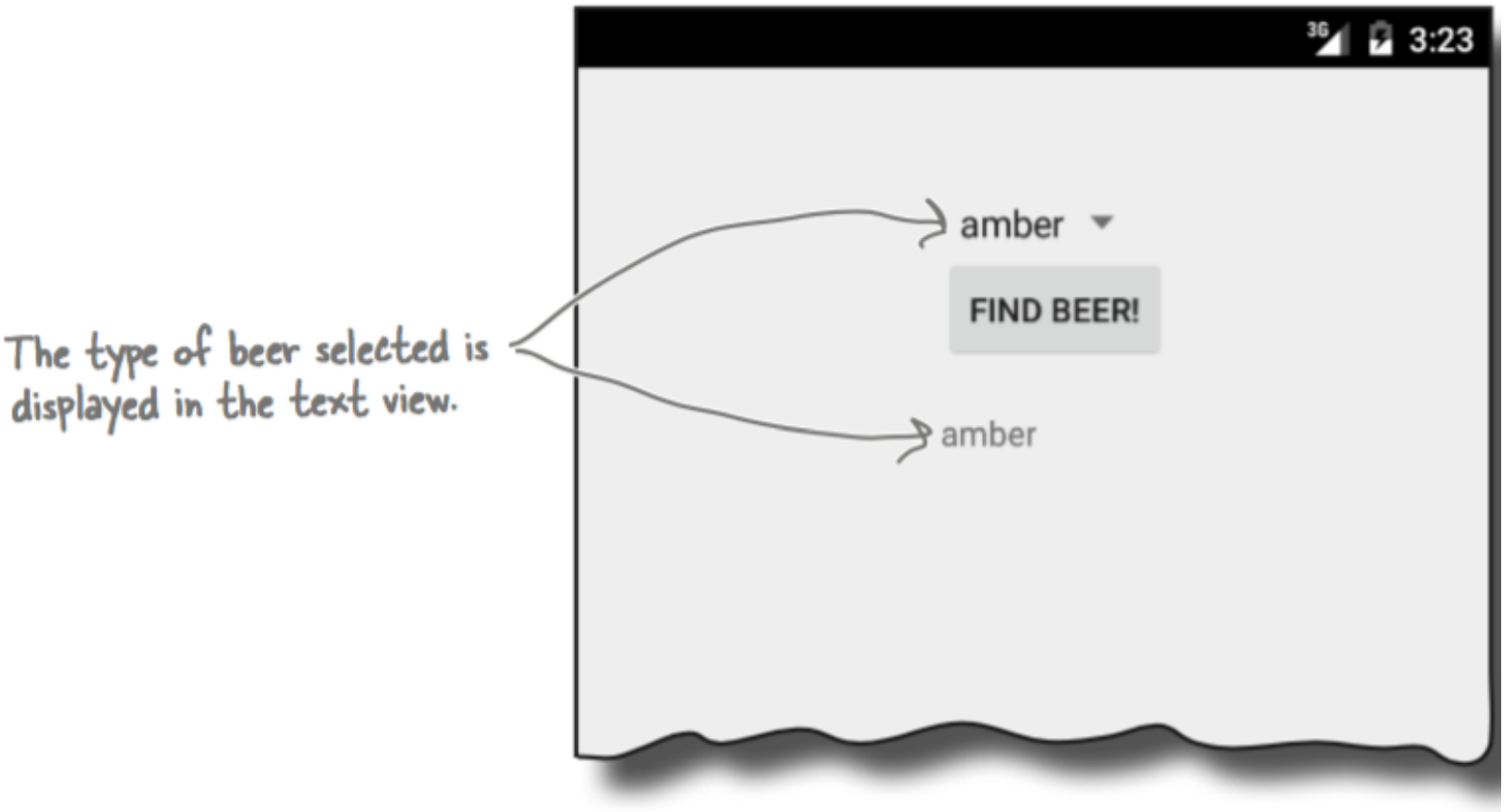
3 **The activity retrieves the currently selected value of the spinner, and converts it to a String.**



4 **The activity then sets the text property of the TextView to reflect the currently selected item in the spinner.**

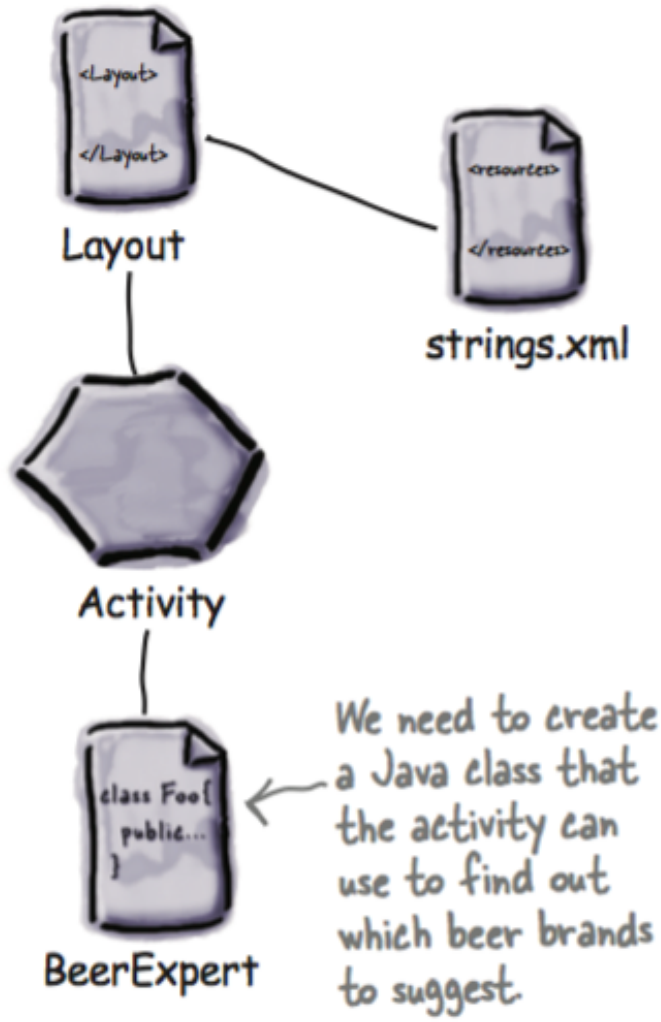


Prueba


3:23
amber
FIND BEER!
amber
The type of beer selected is displayed in the text view.

Clase Java personalizada

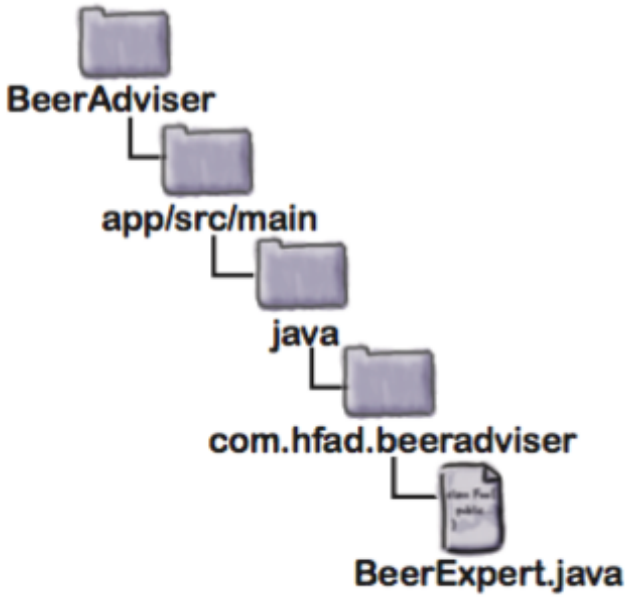
- The package name should be `com.hfad.beeradviser`.
- The class should be called `BeerExpert`.
- It should expose one method, `getBrands()`, that takes a preferred beer color (as a String), and return a `List<String>` of recommended beers.



Clase Java personalizada

```
package com.hfad.beeradviser;
import java.util.ArrayList;
import java.util.List;

public class BeerExpert {
    List<String> getBrands(String color) {
        List<String> brands = new ArrayList<String>();
        if (color.equals("amber")) {
            brands.add("Jack Amber");
            brands.add("Red Moose");
        } else {
            brands.add("Jail Pale Ale");
            brands.add("Gout Stout");
        }
        return brands;
    }
}
```

This is pure Java code, nothing Androidy about it.



Clase Java personalizada

```
package com.hfad.beeradviser;

import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;
import android.view.View;
import android.widget.Spinner;
import android.widget.TextView;
import java.util.List;
```

← We're using this extra class.

```
public class FindBeerActivity extends Activity {
    private BeerExpert expert = new BeerExpert();

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

    //Call when the button gets clicked
    public void onClickFindBeer(View view) {
        //Get a reference to the TextView
        TextView brands = (TextView) findViewById(R.id.brands);
        //Get a reference to the Spinner
        Spinner color = (Spinner) findViewById(R.id.color);
        //Get the selected item in the Spinner
        String beerType = String.valueOf(color.getSelectedItem());
        //Get recommendations from the BeerExpert class
        List<String> brandsList = expert.getBrands(beerType);
        StringBuilder brandsFormatted = new StringBuilder();
        for (String brand : brandsList) {
            brandsFormatted.append(brand).append('\n');
        }
        //Display the beers
        brands.setText(brandsFormatted);
    }
}
```

Add an instance of BeerExpert as a private variable.

Use the BeerExpert class to get a List of brands.

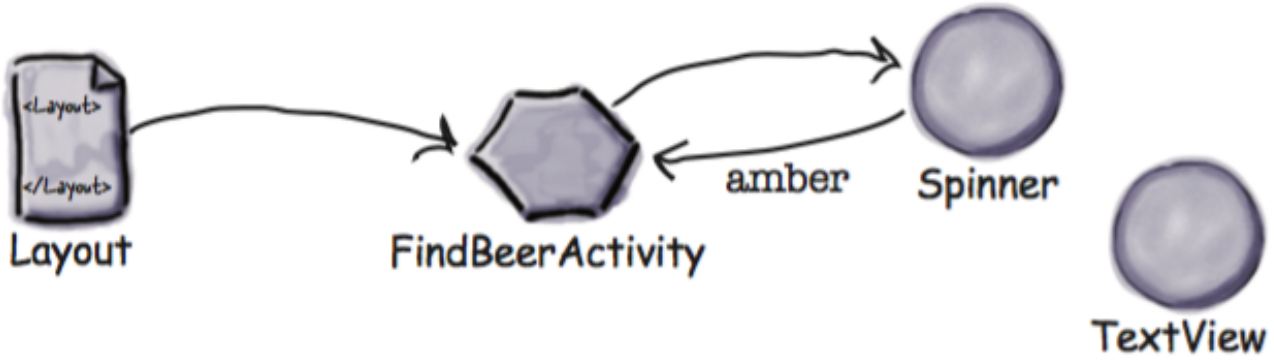
Build a String, displaying each brand on a new line

Display the String in the TextView.

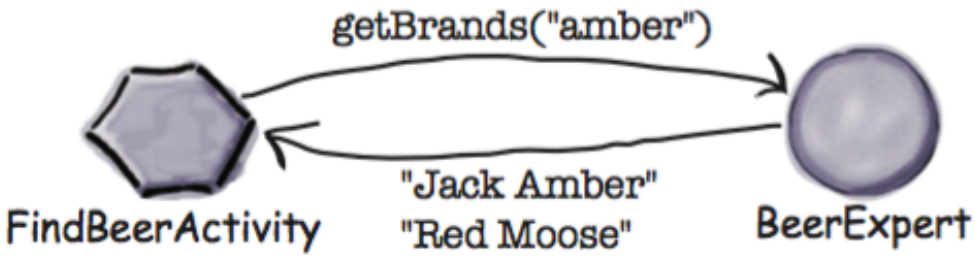
Clase Java personalizada

Que pasa al ejecutar el código …

1 **When the user clicks on the Find Beer button, the onClickFindBeer() method in the activity gets called.**
The method creates a reference to the spinner and text view, and gets the currently selected value from the spinner.

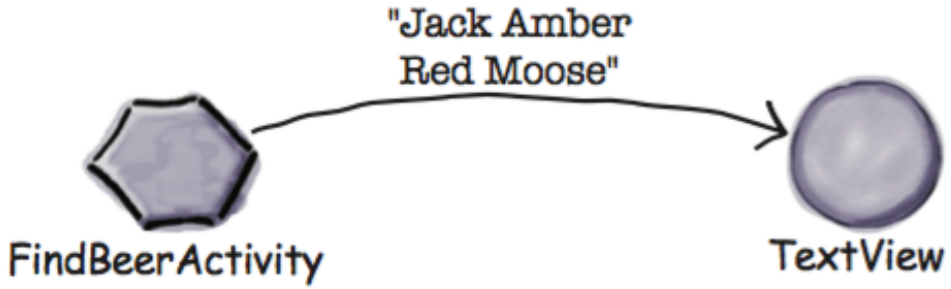


2 **The onClickFindBeer() calls the getBrands() method in the BeerExpert class, passing in the type of beer selected in the spinner.**
The `getBrands()` method returns a list of brands.

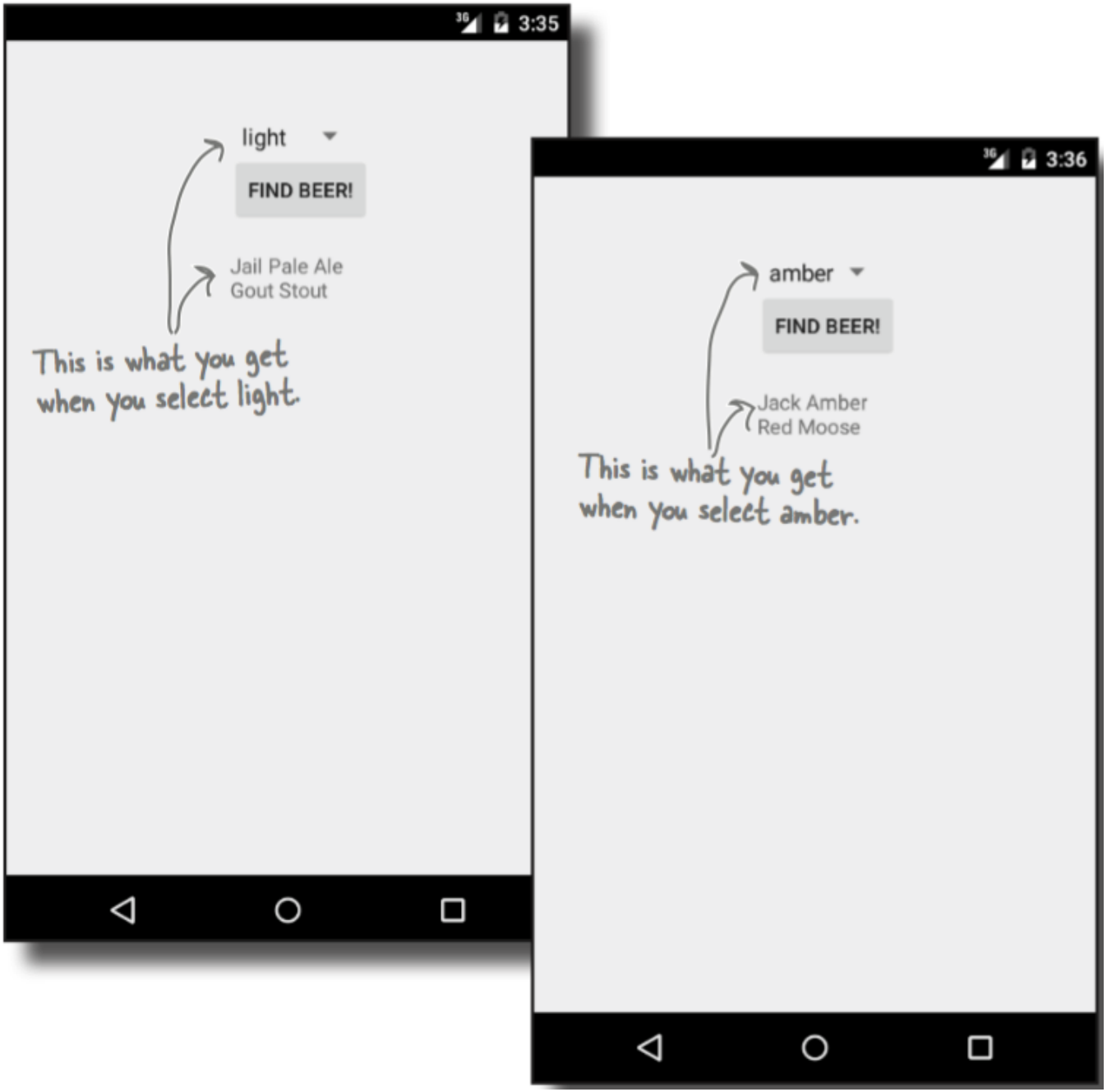


Que pasa al ejecutar el código …

3 **The onClickFindBeer() method formats the list of brands and uses it to set the text property in the text view.**



Prueba


3:35
light
FIND BEER!
Jail Pale Ale
Gout Stout
This is what you get when you select light.
3:36
amber
FIND BEER!
Jack Amber
Red Moose
This is what you get when you select amber.

BULLET POINTS

- The `Button` element is used to add a button.
- The `Spinner` element is used to add a spinner. A spinner is a drop-down list of values.
- All GUI components are types of view. They inherit from the Android `View` class.
- Add an array of string values using:

```
<string-array name="array">
    <item>string1</item>
    ...
</string-array>
```

- Reference a `string-array` in the layout using:

```
"@array/array_name"
```

- Make a button call a method when clicked by adding the following to the layout:

```
android:onClick="clickMethod"
```

  There needs to be a corresponding method in the activity:

```
public void clickMethod(View view){
}
```

- *R.java* is generated for you. It enables you to get references for layouts, GUI components, Strings, and other resources in your Java code.
- Use `findViewById()` to get a reference to a view.
- Use `setText()` to set the text in a view.
- Use `getSelectedItem()` to get the selected item in a spinner.
- Add a custom class to an Android project by going to File menu→New...→Java Class.