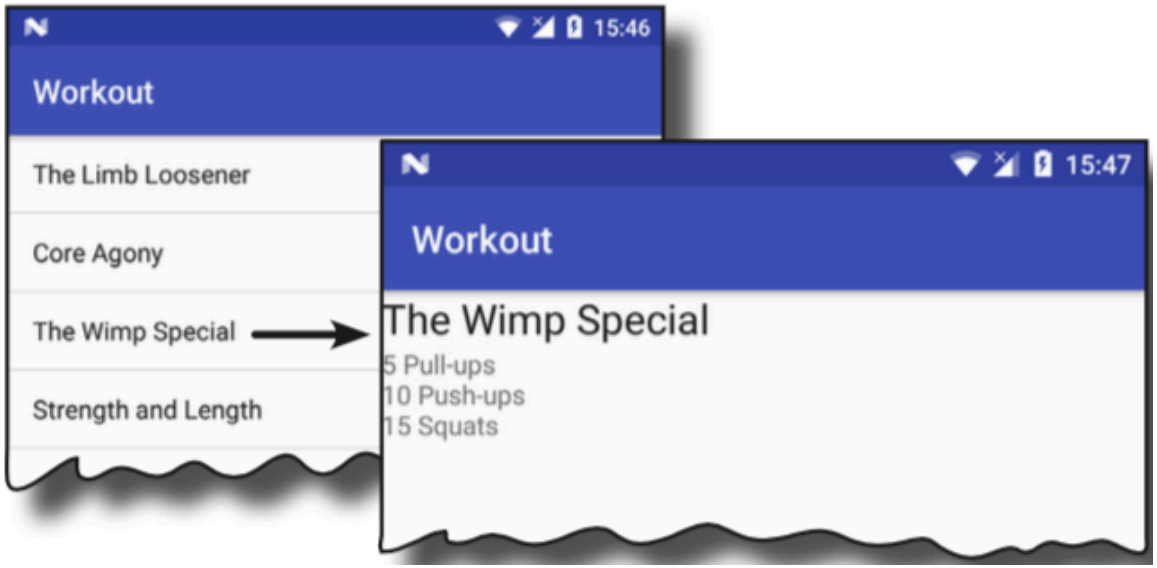
11
Fragmentos

Una aplicación debe verse bien en todos los dispositivos

On a phone:

Take a look at this image of an app on a phone. It displays a list of workouts, and when you click on one, you are shown the details of that workout.

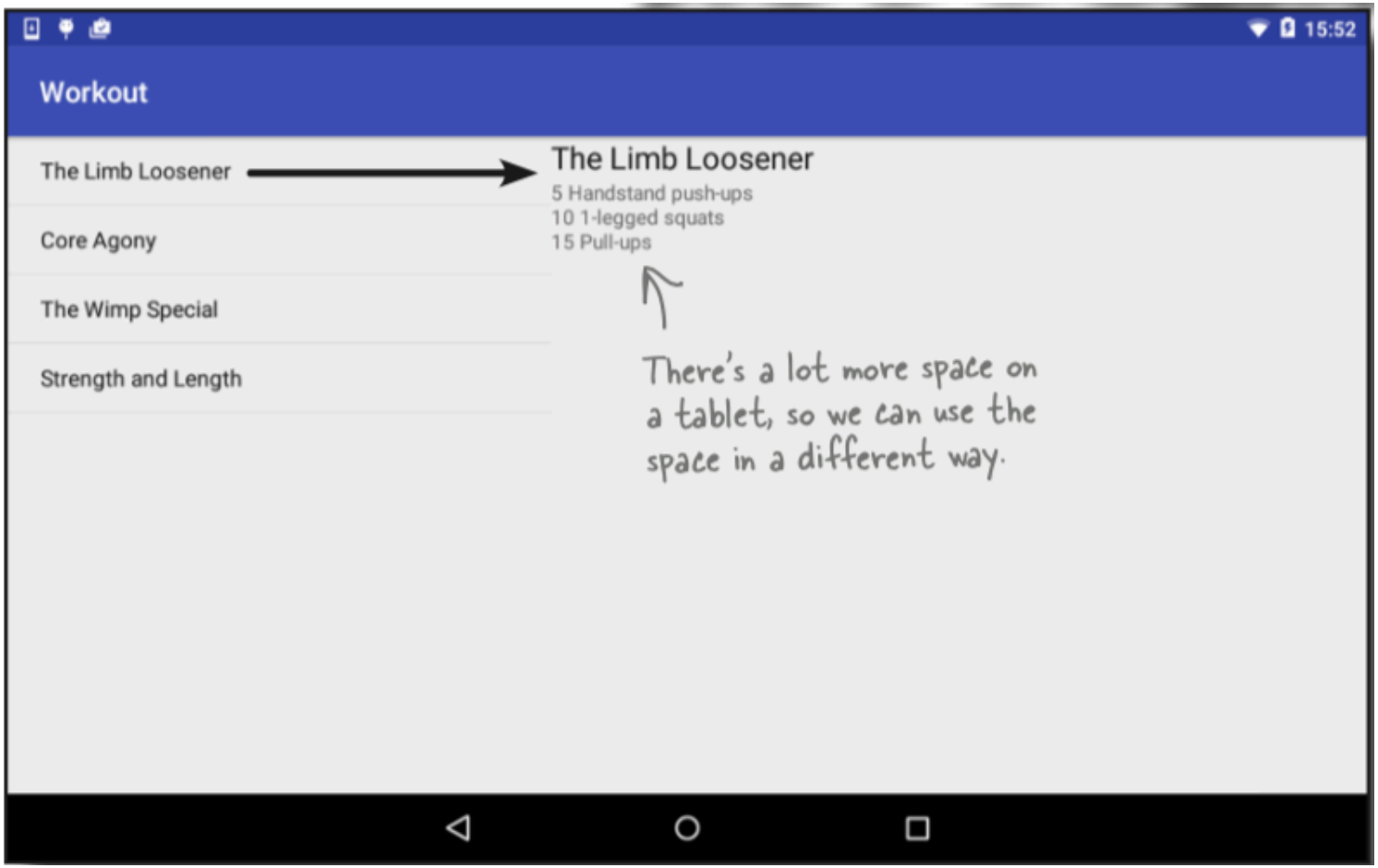
Click on an item in a list, and it launches a second activity.



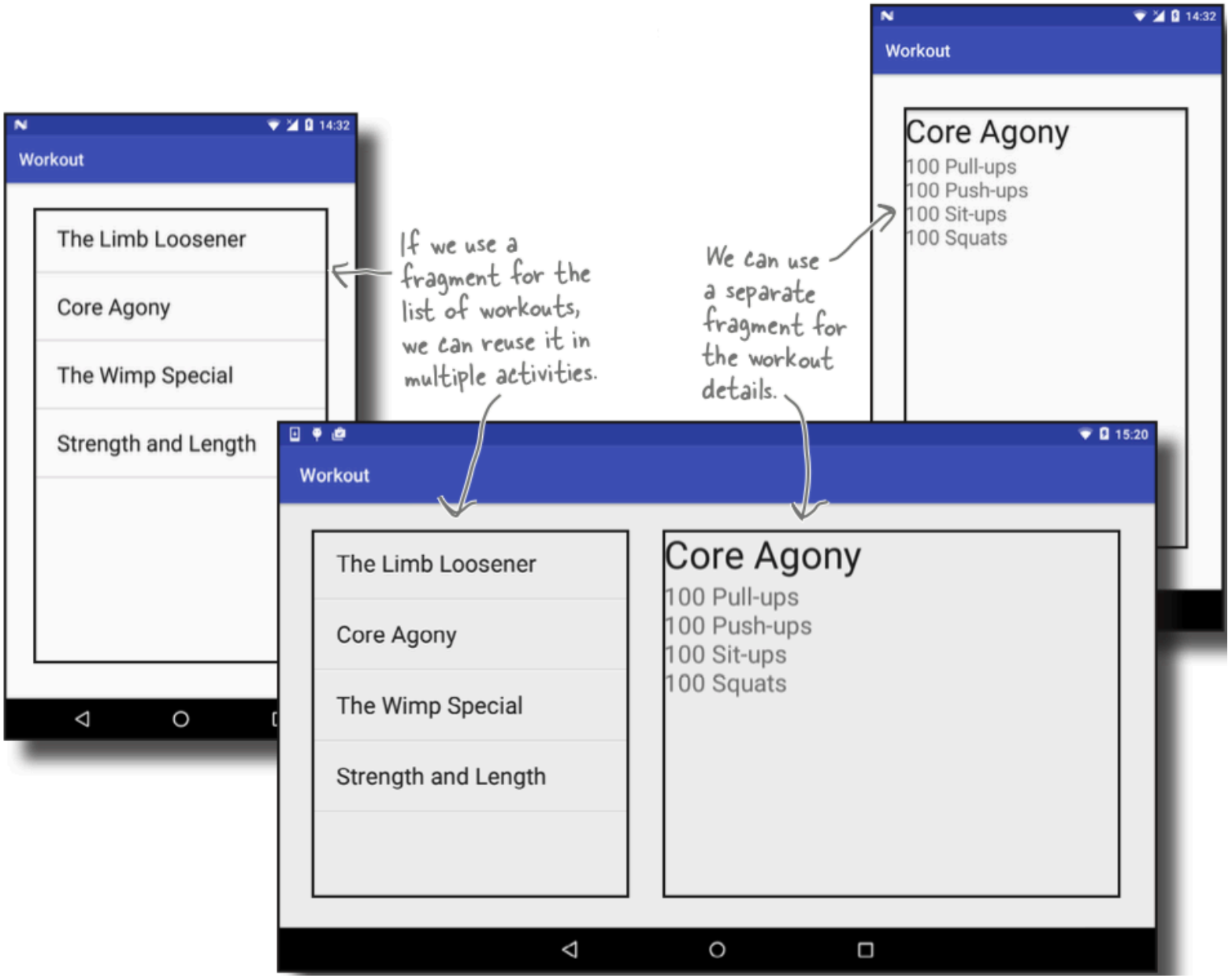
Una aplicación debe verse bien en todos los dispositivos

On a tablet:

On a larger device, like a tablet, you have a lot more screen space available, so it would be good if all the information appeared on the same screen. On the tablet, the list of workouts only goes partway across the screen, and when you click on an item, the details appear on the right.



Los fragmentos permiten reutilizar código



Estructura de la aplicación Workout

1. **When the app gets launched, it starts MainActivity.**
   `MainActivity` uses *activity_main.xml* for its layout, and contains a fragment called `WorkoutListFragment`.

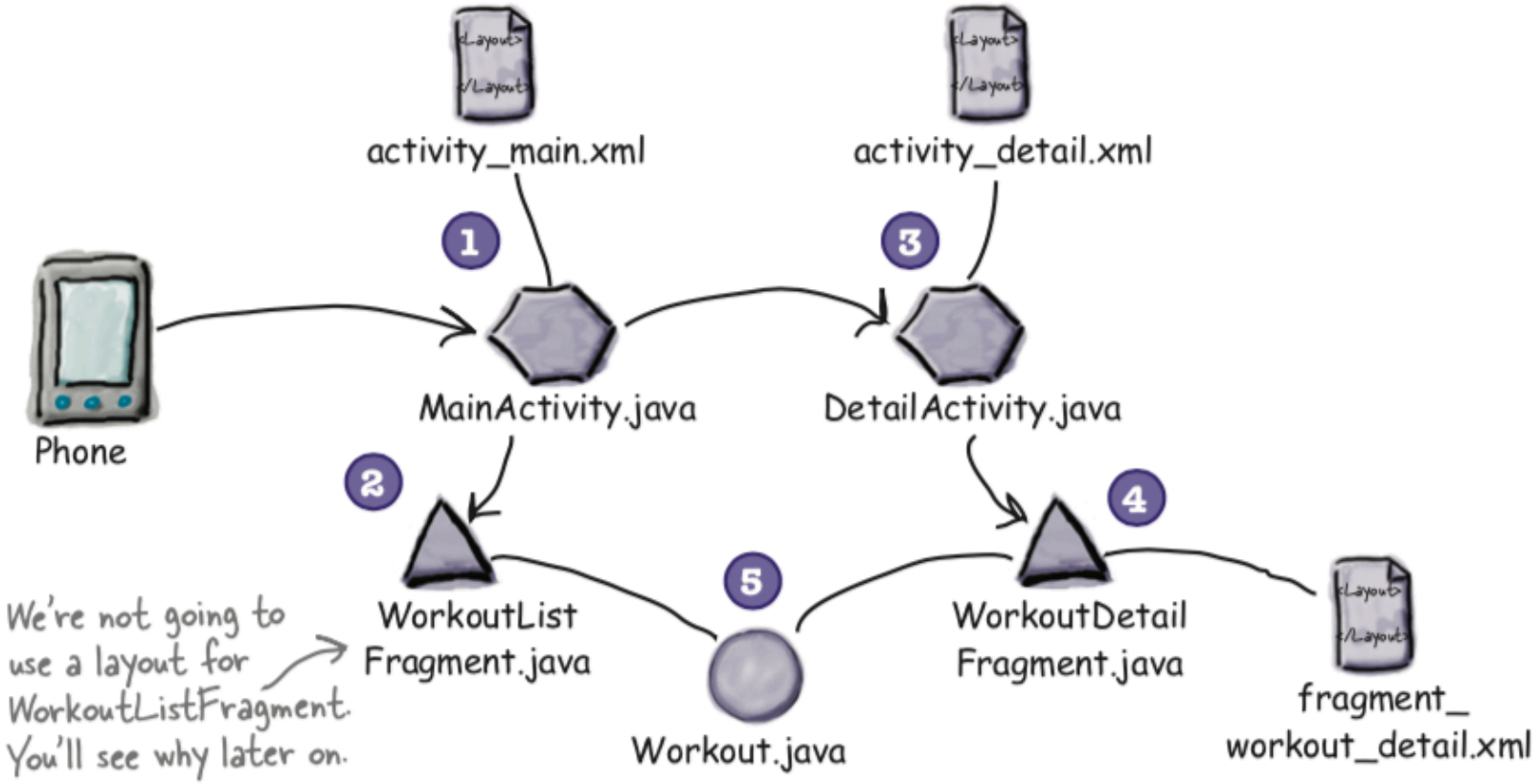
2. **WorkoutListFragment displays a list of workouts.**

3. **When the user clicks on one of the workouts, DetailActivity starts.**
   `DetailActivity` uses *activity_detail.xml* for its layout, and contains a fragment called `WorkoutDetailFragment`.

4. **WorkoutDetailFragment uses fragment_workout_detail.xml for its layout.**
   It displays the details of the workout the user has selected.

5. **WorkoutListFragment and WorkoutDetailFragment get their workout data from *Workout.java*.**
   Workout.java contains an array of `Workouts`.

Estructura de la aplicación Workout

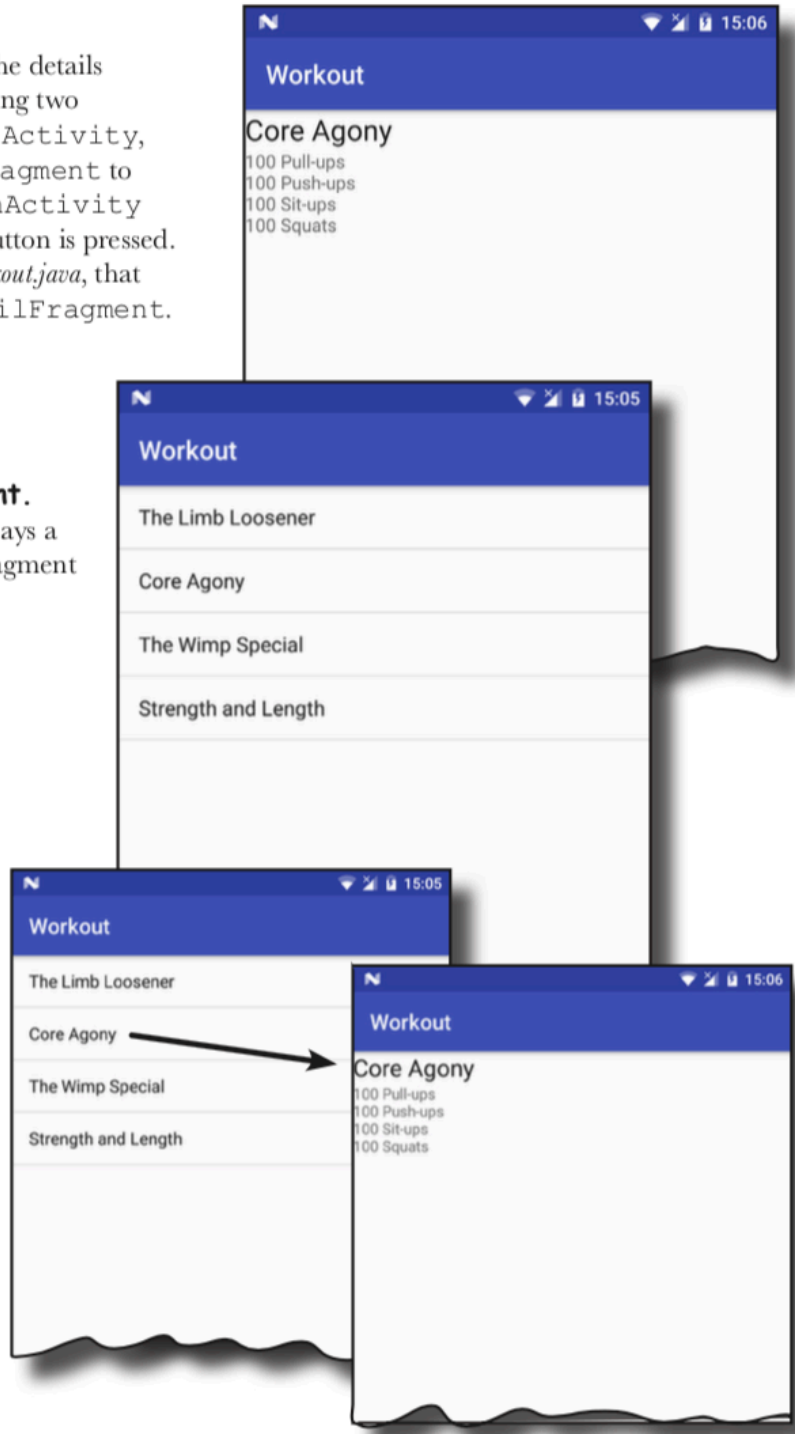


Pasos para construir la aplicación

1 **Create WorkoutDetailFragment.**
`WorkoutDetailFragment` displays the details of a specific workout. We'll start by creating two activities, `MainActivity` and `DetailActivity`, and then we'll add `WorkoutDetailFragment` to `DetailActivity`. We'll also get `MainActivity` to launch `DetailActivity` when a button is pressed. We'll also add a plain old Java class, *Workout.java*, that will provide the data for `WorkoutDetailFragment`.

2 **Create WorkoutListFragment.**
`WorkoutListFragment` displays a list of workouts. We'll add this fragment to `MainActivity`.

3 **Coordinate the fragments to display the correct workout.**
When the user clicks on an item in `WorkoutListFragment`, we'll start `DetailActivity` and get `WorkoutDetailFragment` to display details of the workout the user selected.

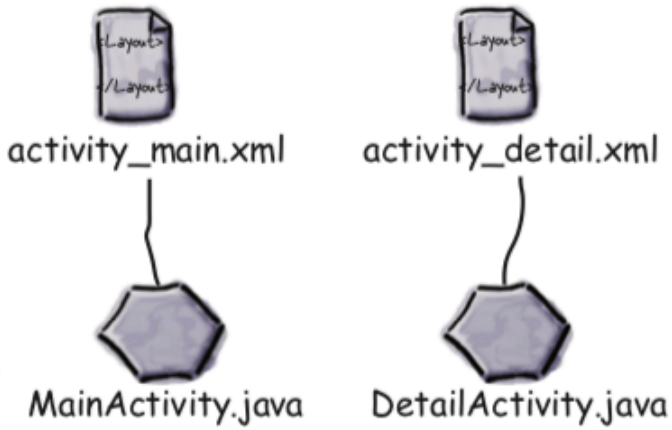


Creando el proyecto y actividades

We're going to start by creating a project that contains two activities, `MainActivity` and `DetailActivity`. `MainActivity` will be used for the fragment that displays a list of workouts, and `DetailActivity` will be used for the fragment that displays details of one particular workout.

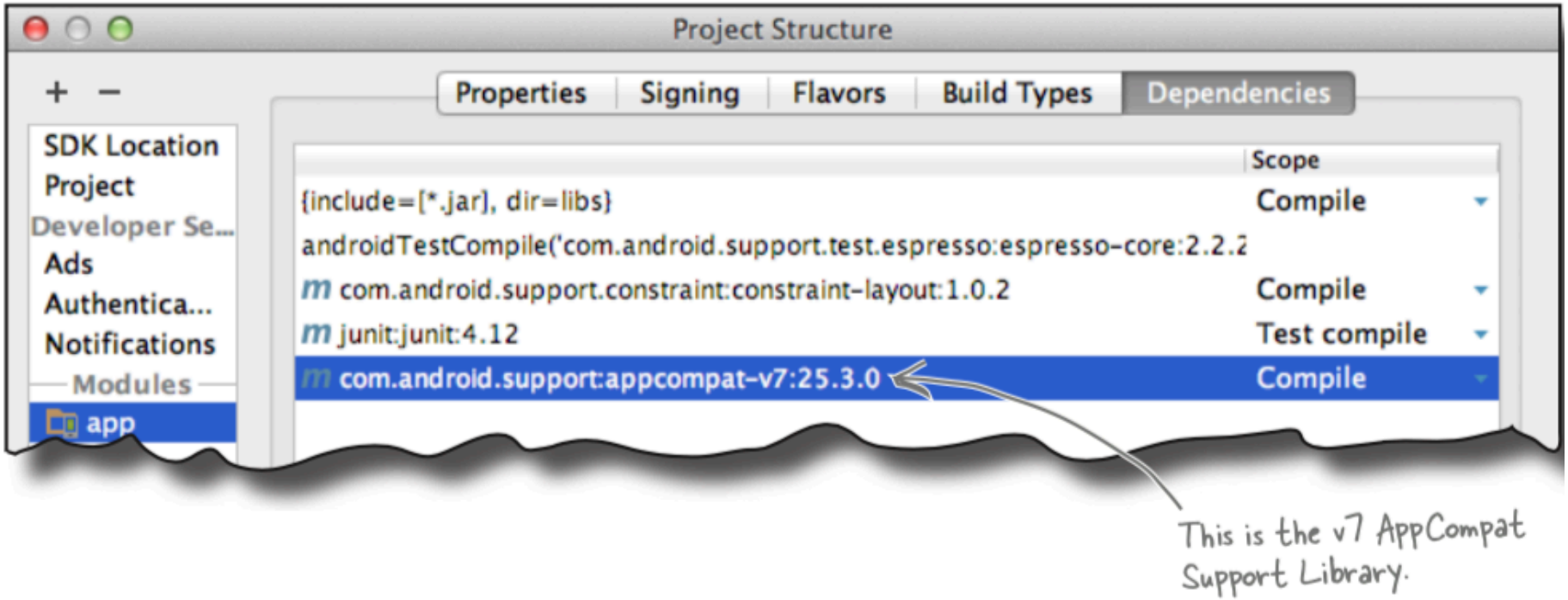
To do this, first create a new Android project with an empty activity for an application named "Workout" with a company domain of "hfad.com", making the package name `com.hfad.workout`. The minimum SDK should be API 19 so that it works on most devices. Name the activity "MainActivity" and name the layout "activity_main". **Make sure you check the Backwards Compatibility (AppCompat) checkbox.**

Next, create a second empty activity by highlighting the *com.hfad.workout* package in the *app/src/main/java* folder, and going to File→New...→Activity→Empty Activity. Name the activity "DetailActivity", name the layout "activity_detail", make sure the package name is `com.hfad.workout`, and **check the Backwards Compatibility (AppCompat) checkbox.**



If prompted for the activity's source language, select the option for Java.

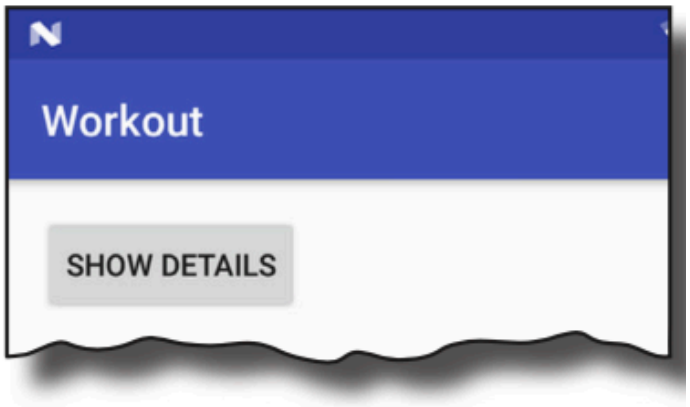
Agregando la biblioteca de apoyo AppCompat



Agregando un botón

```
<resources>
    ...
    <string name="details_button">Show details</string>
</resources>
```

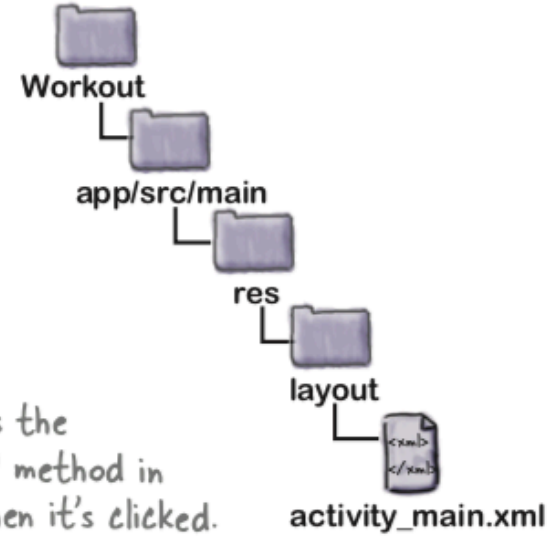
This text will be displayed on the button.




```
<?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.workout.MainActivity">

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:onClick="onShowDetails"
        android:text="@string/details_button" />
</LinearLayout>
```

This is the button we're adding.

The button calls the onShowDetails() method in MainActivity when it's clicked. We need to write this method.

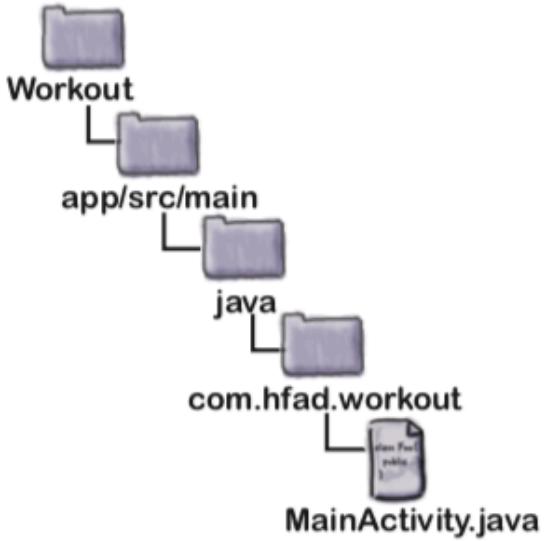


Respondiendo a clicks

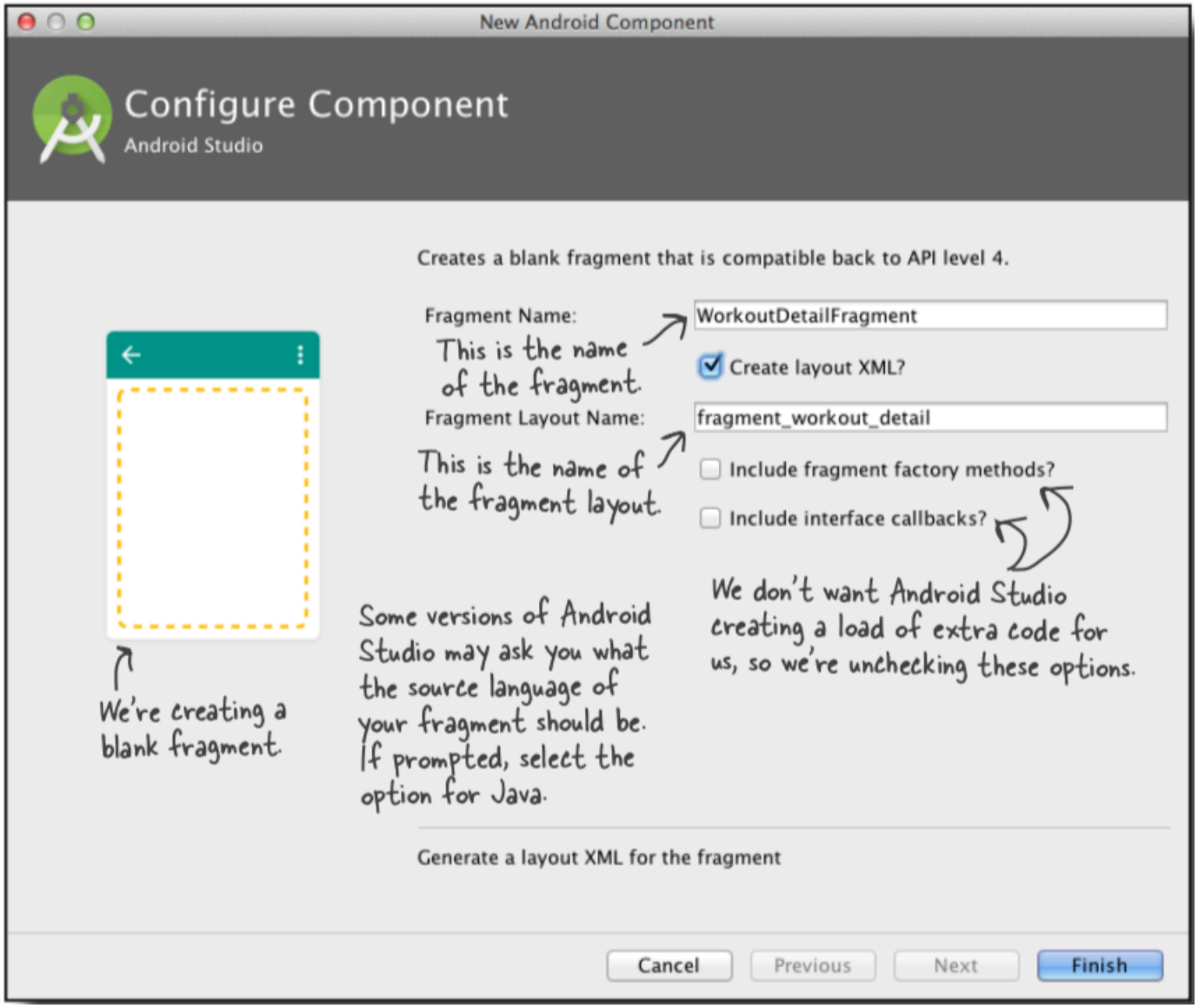
```
package com.hfad.workout;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.content.Intent;

public class MainActivity extends AppCompatActivity {

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

    public void onShowDetails(View view) {
        Intent intent = new Intent(this, DetailActivity.class);
        startActivity(intent);
    }
}
```

The activity extends AppCompatActivity.

This method is called when the button is clicked. It starts DetailActivity.



Agregando fragmentos al proyecto



El código de un fragmento

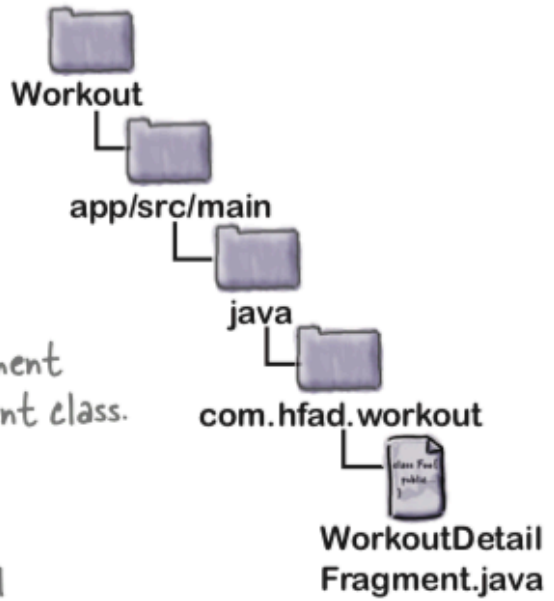
```
package com.hfad.workout;

import android.support.v4.app.Fragment;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

public class WorkoutDetailFragment extends Fragment {

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        return inflater.inflate(R.layout.fragment_workout_detail, container, false);
    }
}
```

We're using the Fragment class from the Android Support Library.

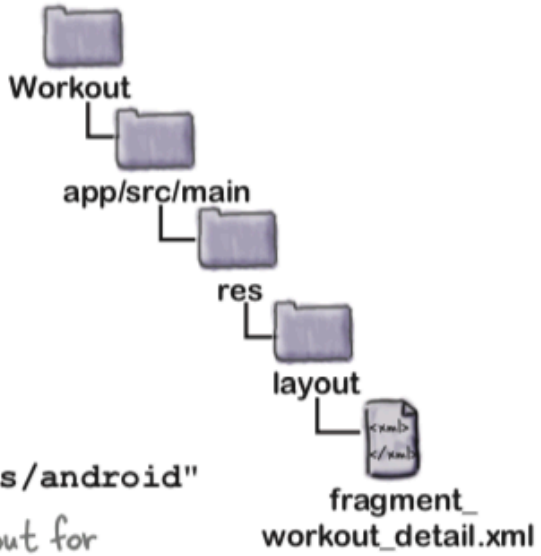
WorkoutDetailFragment extends the Fragment class.

This is the onCreateView() method. It's called when Android needs the fragment's layout.

This tells Android which layout the fragment uses (in this case, it's fragment_workout_detail).



El código del layout de un fragmento




```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_height="match_parent"
    android:layout_width="match_parent"
    android:orientation="vertical">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:text="@string/workout_title"
        android:id="@+id/textTitle" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/workout_description"
        android:id="@+id/textDescription" />

</LinearLayout>
```

We're using a LinearLayout for our fragment, but we could have used any of the other layout types we've looked at instead.

We'll display the workout title and description in two separate TextViews.

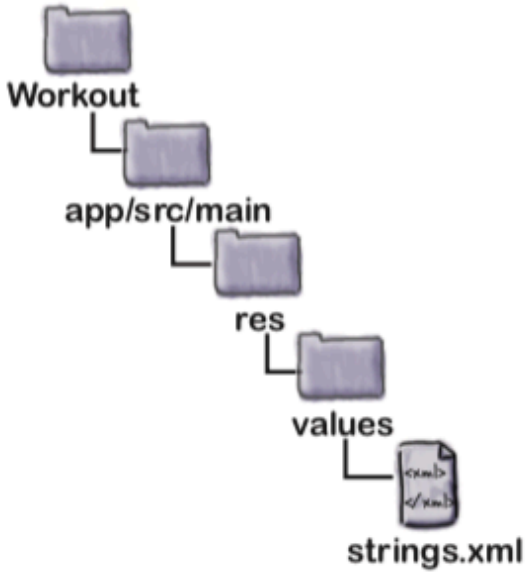
This makes the text large.

Static String resources.

El código del layout de un fragmento

```
<resources>
    ...
    <string name="workout_title">Title</string>
    <string name="workout_description">Description</string>
</resources>
```

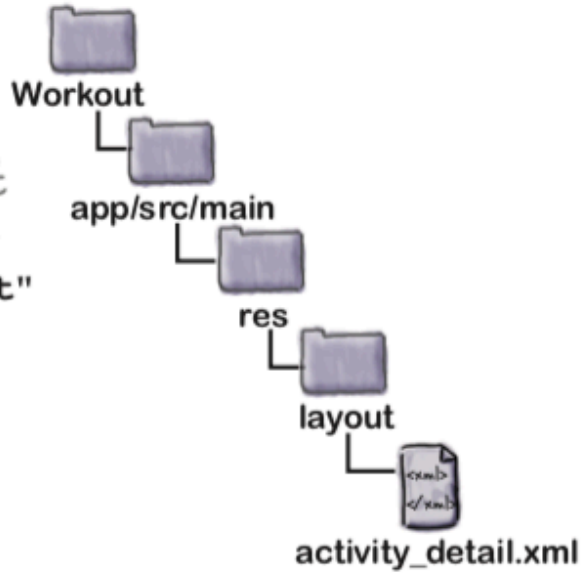
We'll use these to display default text in our fragment.



Agregando un fragmento al layout de una actividad

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <fragment
        android:name="com.hfad.workout.WorkoutDetailFragment"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />
</LinearLayout>
```

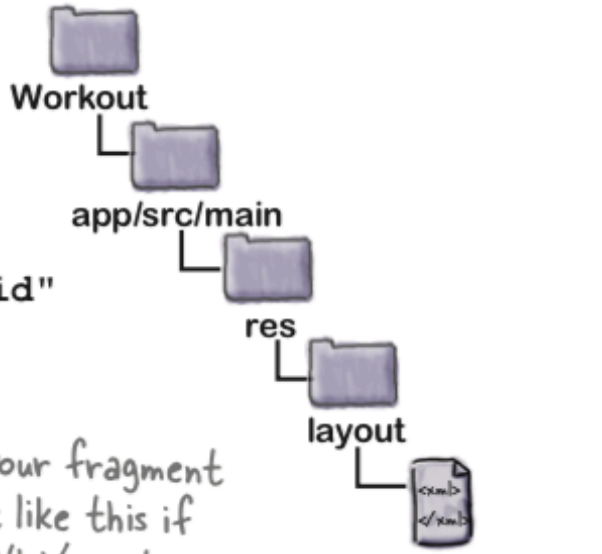
This adds the fragment to the activity's layout.



Simplificando el layout

```
<?xml version="1.0" encoding="utf-8"?>
<fragment
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:name="com.hfad.workout.WorkoutDetailFragment"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />
```

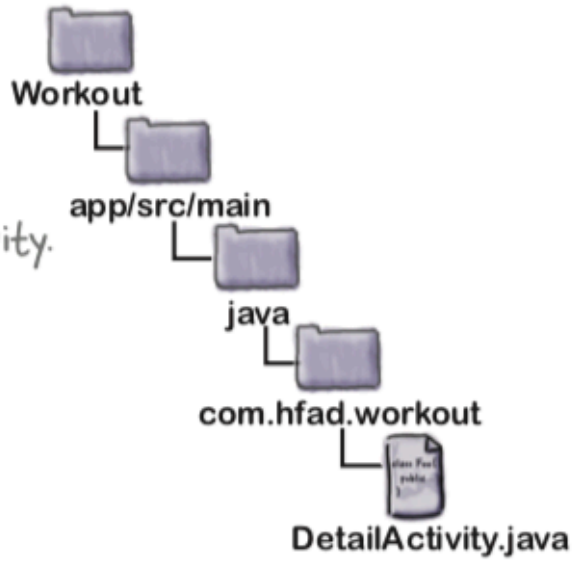
You can add your fragment to your layout like this if your layout ONLY contains a single fragment.



El código de DetailActivity

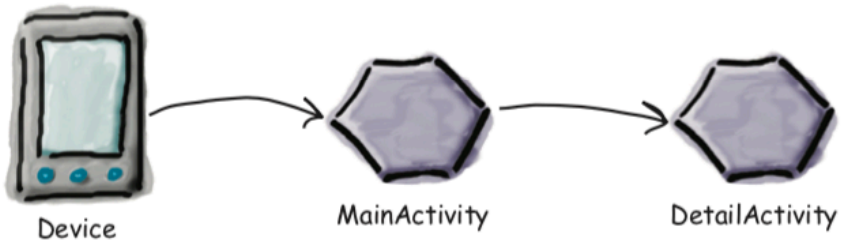
```
package com.hfad.workout;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

public class DetailActivity extends AppCompatActivity {

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

DetailActivity extends AppCompatActivity.

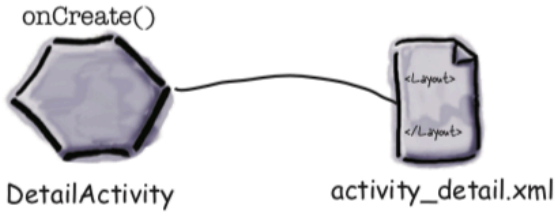


Que hace el código ?

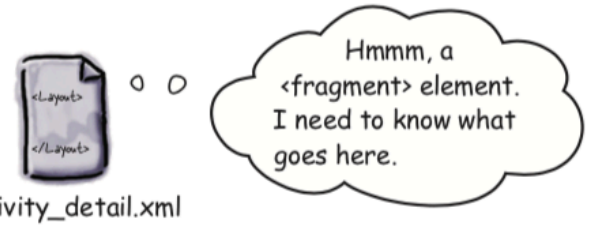
1. **When the app is launched, activity MainActivity gets created.**
The user clicks on the button in `MainActivity` to start `DetailActivity`.



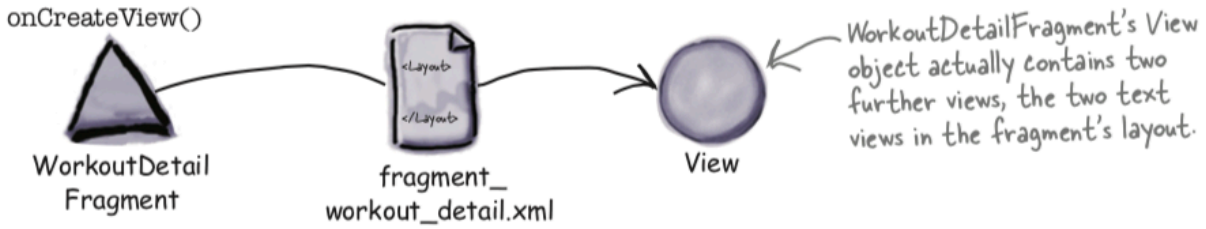
2. **DetailActivity's onCreate() method runs.**
The `onCreate()` method specifies that *activity_detail.xml* should be used for `DetailActivity`'s layout.



3. **activity_detail.xml sees that it includes a <fragment> element that refers to WorkoutDetailFragment.**

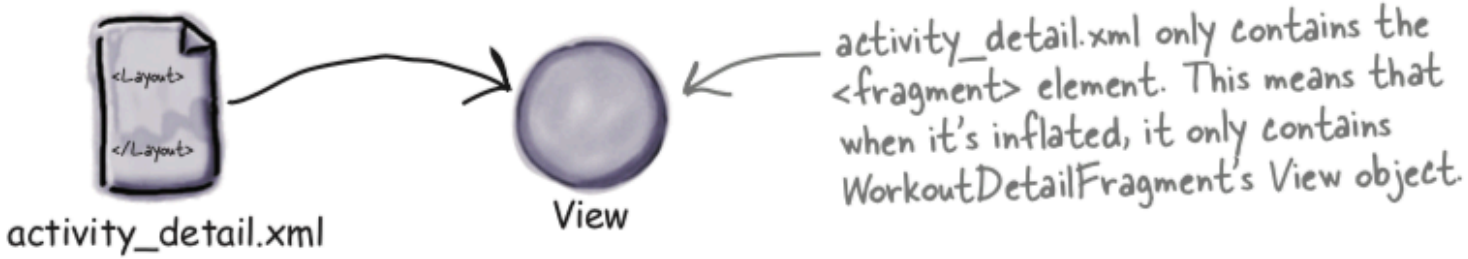


4. **WorkoutDetailFragment's onCreateView() method is called.**
The `onCreateView()` method specifies that *fragment_workout_detail.xml* should be used for `WorkoutDetailFragment`'s layout. It inflates the layout to a `View` object.

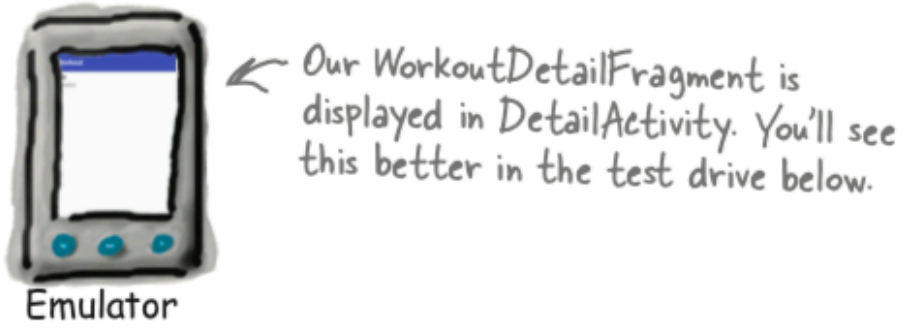


Que hace el código ?

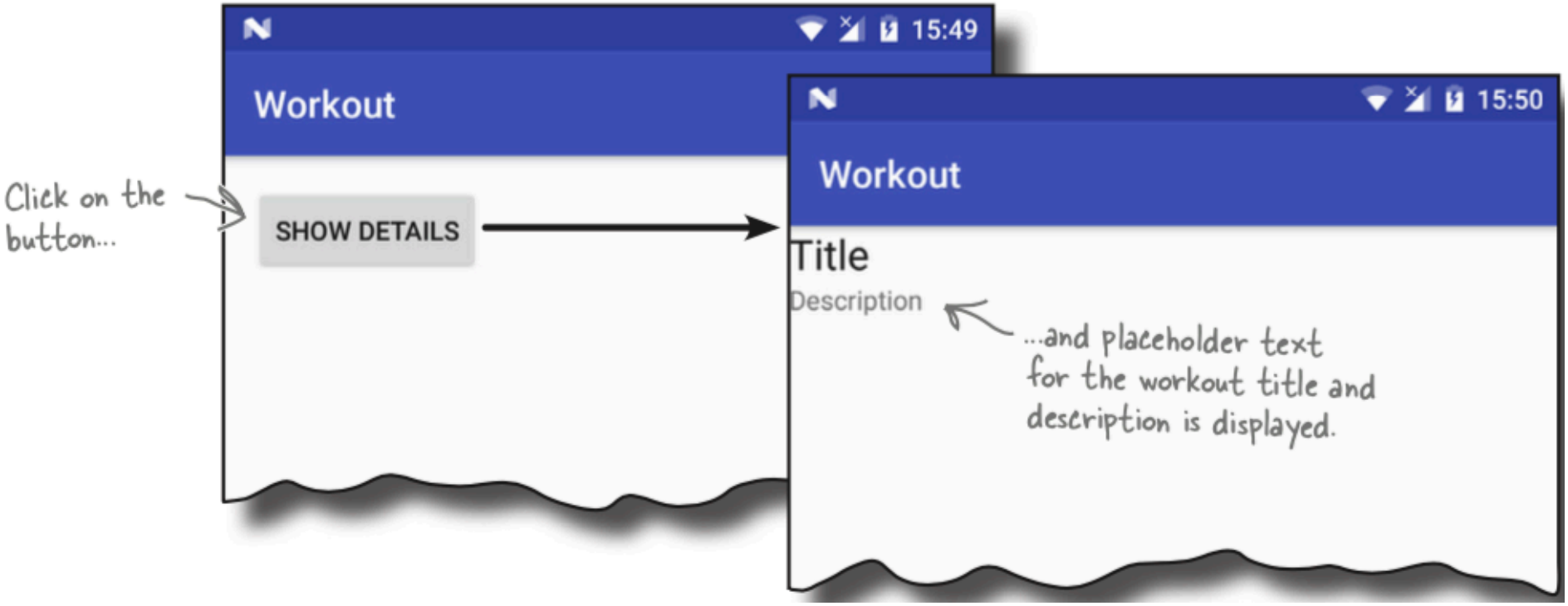
5 **activity_detail.xml's Views are inflated to View Java objects.**
`DetailActivity` layout uses `WorkoutDetailFragment`'s `View` object in place of the `<fragment>` element in its layout's XML.



6 **Finally, DetailActivity is displayed on the device.**
Its layout contains the fragment `WorkoutDetailFragment`.



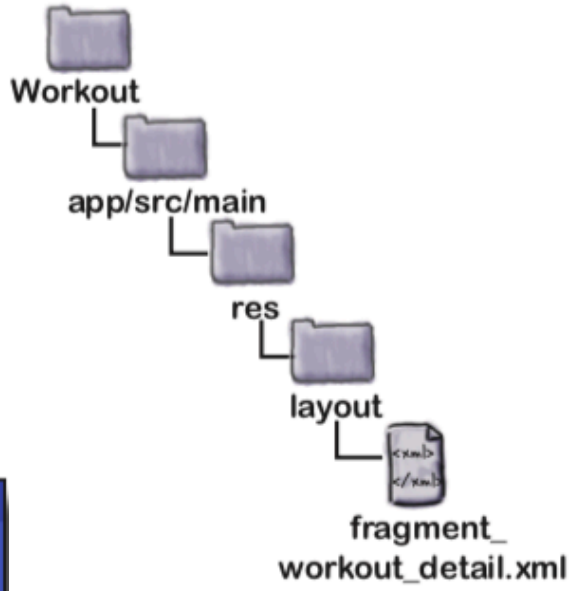
Prueba

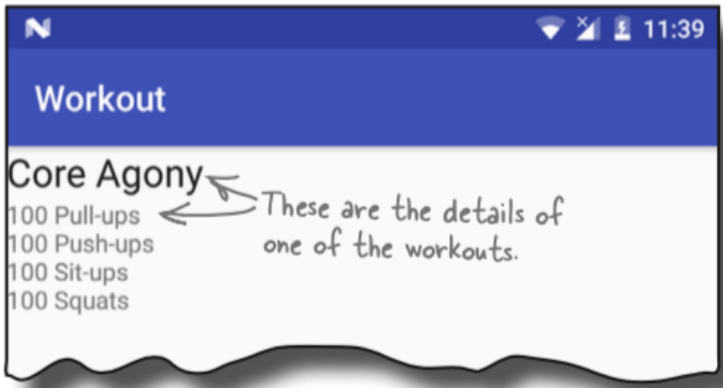

15:49
Workout
Click on the button...
SHOW DETAILS
15:50
Workout
Title
Description
...and placeholder text for the workout title and description is displayed.

Interacción entre el fragmento y la actividad

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    ...
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:text="@string/workout_title"
        android:id="@+id/textTitle" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/workout_description"
        android:id="@+id/textDescription" />
</LinearLayout>
```

Delete both these lines.





La clase Workout

```
package com.hfad.workout;

public class Workout {
    private String name;
    private String description;

    public static final Workout[] workouts = {
            new Workout("The Limb Loosener",
                    "5 Handstand push-ups\n10 1-legged squats\n15 Pull-ups"),
            new Workout("Core Agony",
                    "100 Pull-ups\n100 Push-ups\n100 Sit-ups\n100 Squats"),
            new Workout("The Wimp Special",
                    "5 Pull-ups\n10 Push-ups\n15 Squats"),
            new Workout("Strength and Length",
                    "500 meter run\n21 x 1.5 pood kettleball swing\n21 x pull-ups")
    };
```

Each Workout has a name and description.

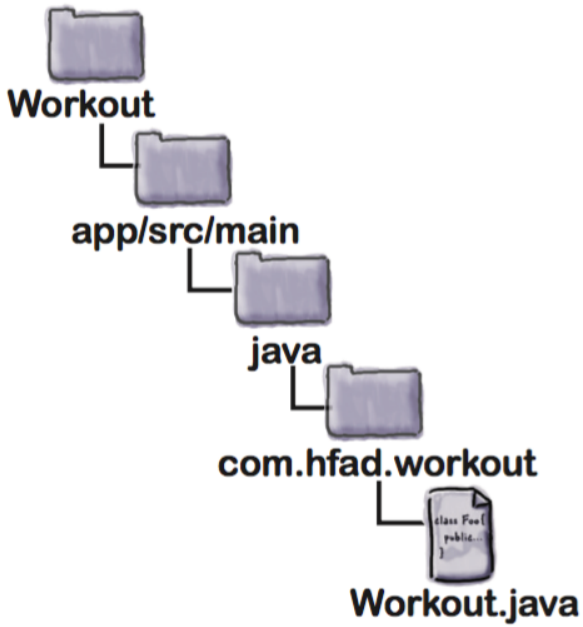
workouts is an array of four Workouts.

La clase Workout

```
    //Each Workout has a name and description
    private Workout(String name, String description) {
        this.name = name;
        this.description = description;
    }

    public String getDescription() {
        return description;
    }

    public String getName() {
        return name;
    }

    public String toString() {
        return this.name;
    }
}
```

These are getters for the private variables.

The String representation of a Workout is its name.

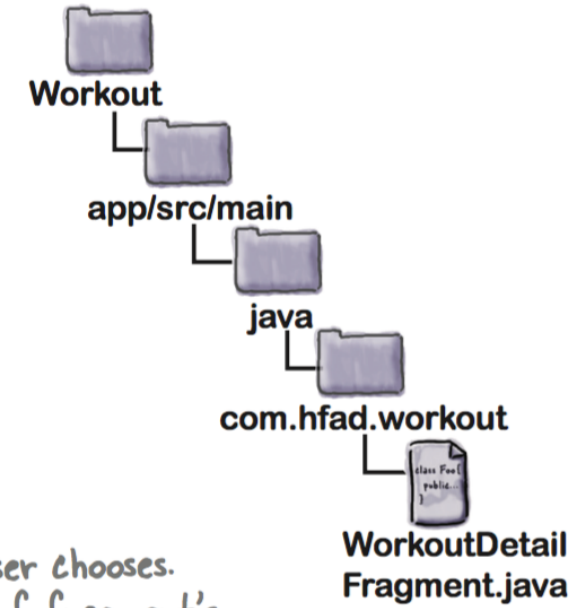


Pasando el ID de la clase al fragmento

```
package com.hfad.workout;


import android.app.Fragment;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;


public class WorkoutDetailFragment extends Fragment {
    private long workoutId;

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        return inflater.inflate(R.layout.fragment_workout_detail, container, false);
    }


    public void setWorkout(long id) {
        this.workoutId = id;
    }
}
```

This is the ID of the workout the user chooses. Later, we'll use it to set the values of fragment's views with the workout details.

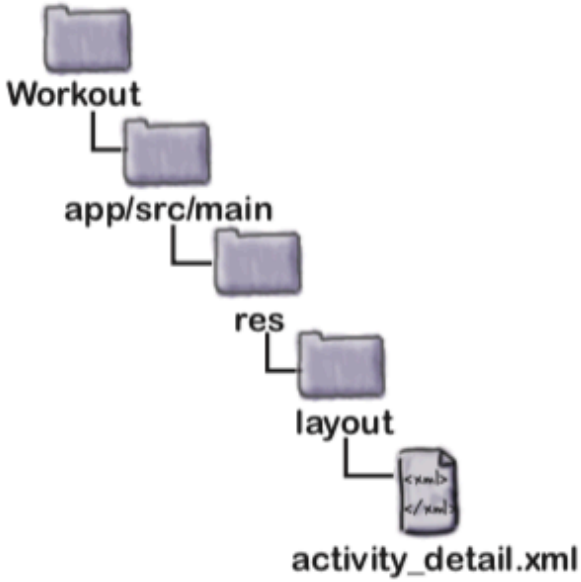
This is a setter method for the workout ID. The activity will use this method to set the value of the workout ID.



Actualizando activity_detail.xml

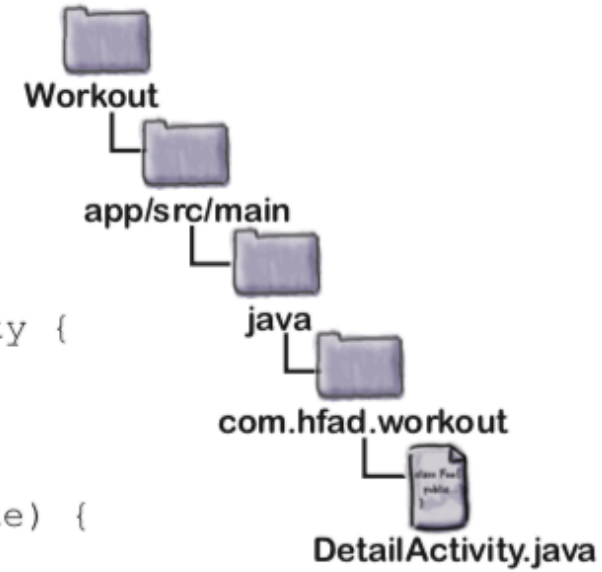
```
<?xml version="1.0" encoding="utf-8"?>
<fragment xmlns:android="http://schemas.android.com/apk/res/android"
    android:name="com.hfad.workout.WorkoutDetailFragment"
    android:id="@+id/detail_frag"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />
```

Add an ID to the fragment.



Actualizando DetailActivity

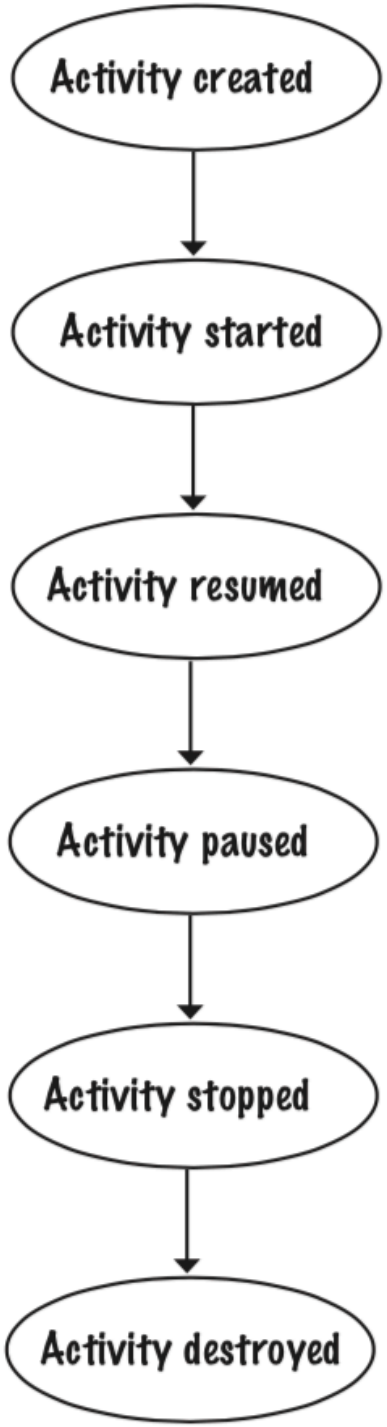
```
package com.hfad.workout;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

public class DetailActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_detail);
        WorkoutDetailFragment frag = (WorkoutDetailFragment)
                getSupportFragmentManager().findFragmentById(R.id.detail_frag);
        frag.setWorkout(1);
    }
}
```



We're going to get WorkoutDetailFragment to display details of a workout here to check that it's working.

This gets us a reference to WorkoutDetailFragment. Its id in the activity's layout is detail_frag.

Los estados de una actividad



The activity is created when its onCreate() method runs.
At this point, the activity is initialized, but isn't visible.

The activity is started when its onStart() method runs.
The activity is visible, but doesn't have the focus.

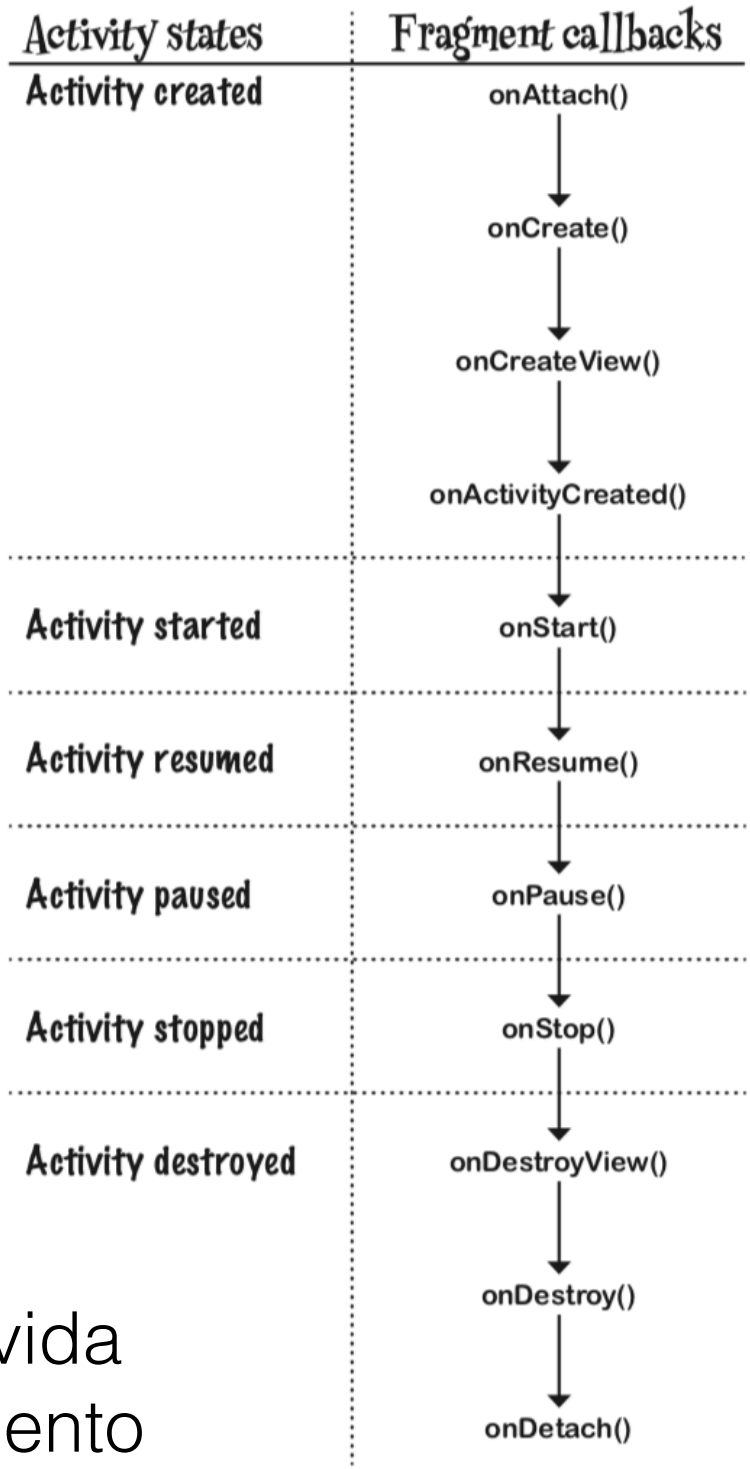
The activity is resumed when its onResume() method runs.
The activity is visible, and has the focus.

The activity is paused when its onPause() method runs.
The activity is still visible, but no longer has the focus.

The activity is stopped when its onStop() method runs.
The activity is no longer visible, but still exists.

The activity is destroyed when its onDestroy() method runs.
The activity no longer exists.

El ciclo de vida de un fragmento



onAttach(Context)
This happens when the fragment is associated with a context, in this case an activity.

onCreate(Bundle)
This is very similar to the activity's `onCreate()` method. It can be used to do the initial setup of the fragment.

onCreateView(LayoutInflater, ViewGroup, Bundle)
Fragments use a layout inflater to create their view at this stage.

onActivityCreated(Bundle)
Called when the `onCreate()` method of the activity has completed.

onStart()
Called when the fragment is about to become visible.

onResume()
Called when the fragment is visible and actively running.

onPause()
Called when the fragment is no longer interacting with the user.

onStop()
Called when the fragment is no longer visible to the user.

onDestroyView()
Gives the fragment the chance to clear away any resources that were associated with its view.

onDestroy()
In this method, the fragment can clear away any other resources it created.

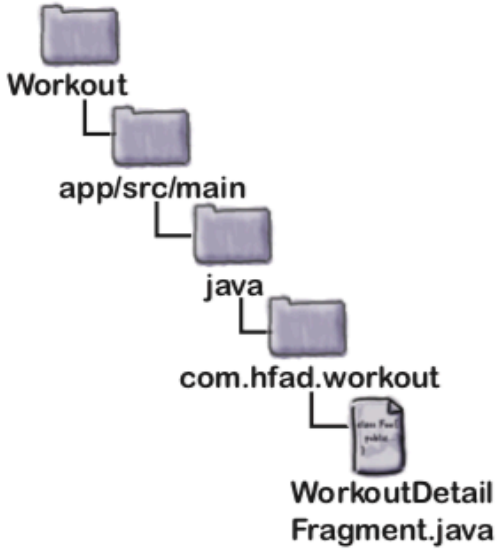
onDetach()
Called when the fragment finally loses contact with the activity.

El fragmento hereda los métodos del ciclo de vida

Object class
(java.lang.Object)

Fragment class
(android.support.v4.app.Fragment)
The `Fragment` class implements default versions of the lifecycle methods. It also defines other methods that fragments need, such as `getView()`.

OurFragment class
(com.hfad.foo)
Most of the behavior of our fragment is handled by superclass methods our fragment inherits. All you do is override the methods you need.




```
package com.hfad.workout;

import android.support.v4.app.Fragment;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;

public class WorkoutDetailFragment extends Fragment {
    private long workoutId;

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        return inflater.inflate(R.layout.fragment_workout_detail, container, false);
    }

    @Override
    public void onStart() {
        super.onStart();
        View view = getView();
        if (view != null) {
            TextView title = (TextView) view.findViewById(R.id.textTitle);
            Workout workout = Workout.workouts[(int) workoutId];
            title.setText(workout.getName());
            TextView description = (TextView) view.findViewById(R.id.textDescription);
            description.setText(workout.getDescription());
        }
    }

    public void setWorkout(long id) {
        this.workoutId = id;
    }
}
```

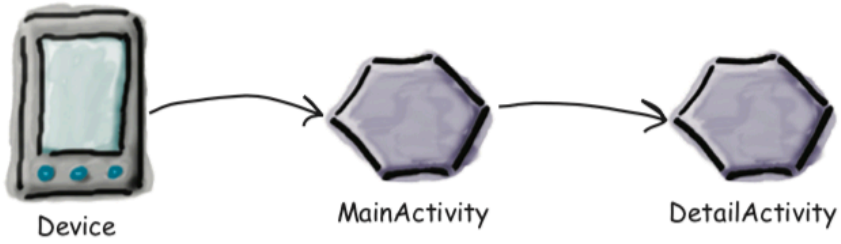
We're using this class in the onStart() method.

The getView() method gets the fragment's root View. We can then use this to get references to the workout title and description text views.

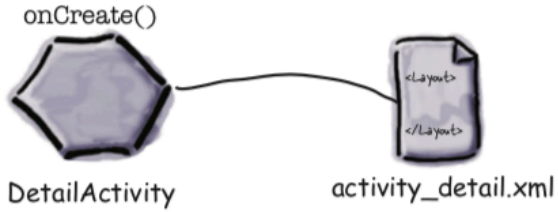
Actualizando el método onStart() en WorkoutDetailFragment

Que pasa al ejecutar el código

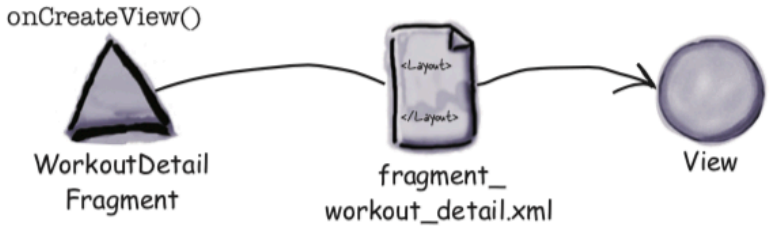
1 **When the app is launched, MainActivity gets created.**
The user clicks on the button in `MainActivity` to start `DetailActivity`.



2 **DetailActivity's onCreate() method runs.**
The `onCreate()` method specifies that *activity_detail.xml* should be used for `DetailActivity`'s layout. *activity_detail.xml* includes a `<fragment>` element with an ID of `detail_frag` that refers to the fragment `WorkoutDetailFragment`.

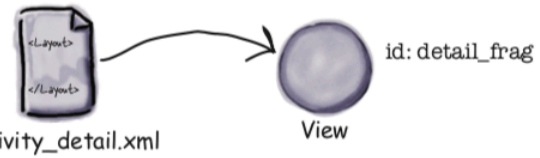


3 **WorkoutDetailFragment's onCreateView() method runs.**
The `onCreateView()` method specifies that *fragment_workout_detail.xml* should be used for `WorkoutDetailFragment`'s layout. It inflates the layout to a `View` object.

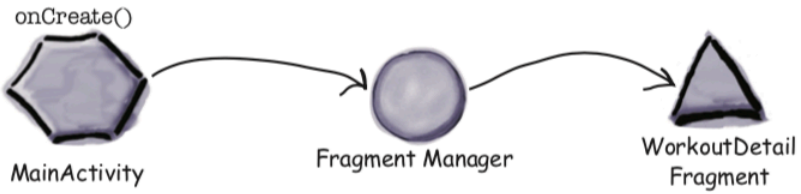


Que pasa al ejecutar el código

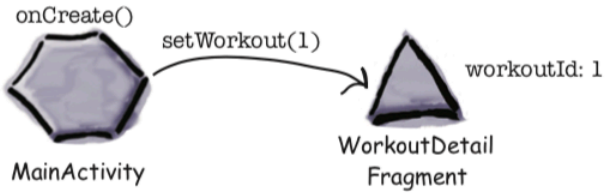
4 **activity_detail.xml's Views are inflated to View Java objects.**
`DetailActivity` uses `WorkoutDetailFragment`'s `View` object in place of the `<fragment>` element in its layout's XML, and gives it an ID of `detail_frag`.



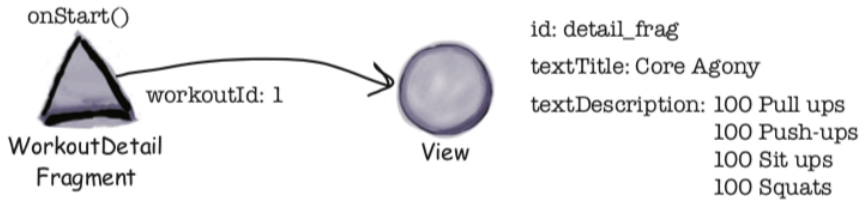
5 **DetailActivity's onCreate() method continues to run.**
`DetailActivity` gets a reference to `WorkoutDetailFragment` by asking the fragment manager for the fragment with an ID of `detail_frag`.



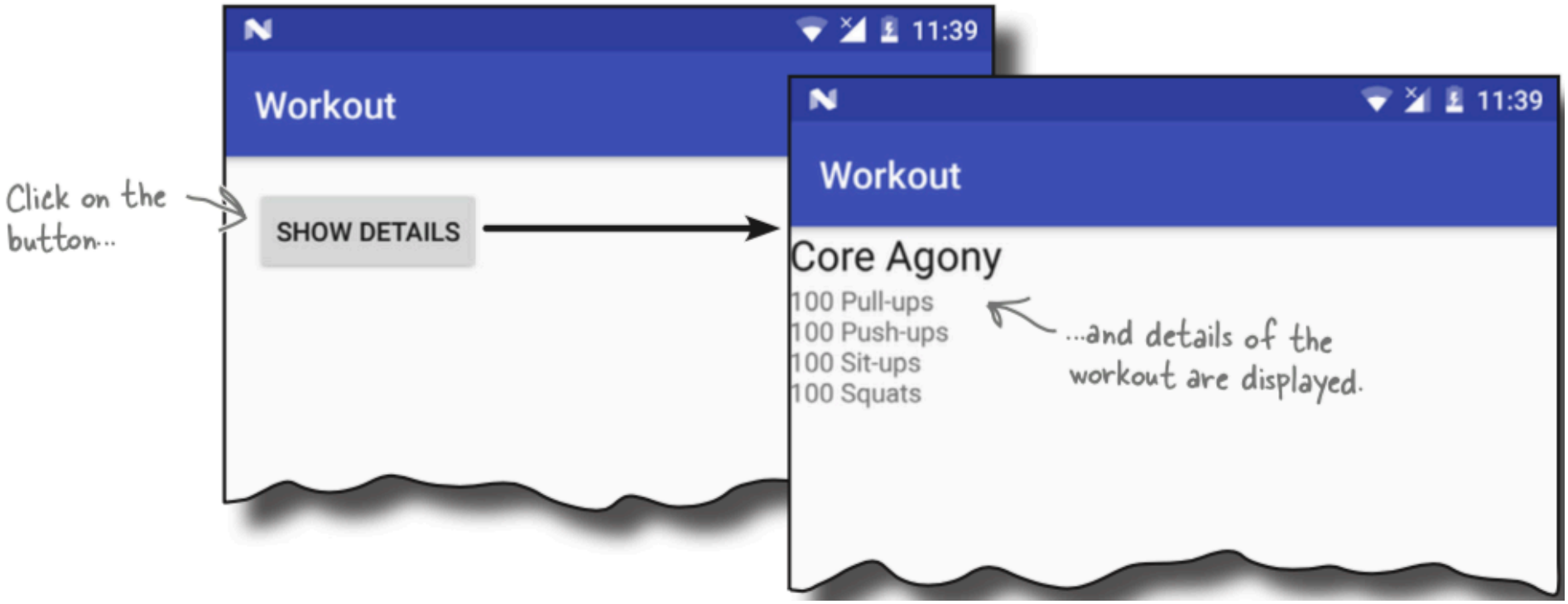
5 **DetailActivity calls WorkoutDetailFragment's setWorkout() method.**
`DetailActivity` passes `WorkoutDetailFragment` a workout ID of `1`. The fragment sets its `workoutId` variable to `1`.



6 **The fragment uses the value of the workout ID in its onStart() method to set the values of its views.**



Prueba


11:39
Workout
Click on the button...
SHOW DETAILS
11:39
Workout
Core Agony
100 Pull-ups
100 Push-ups
100 Sit-ups
100 Squats
...and details of the workout are displayed.

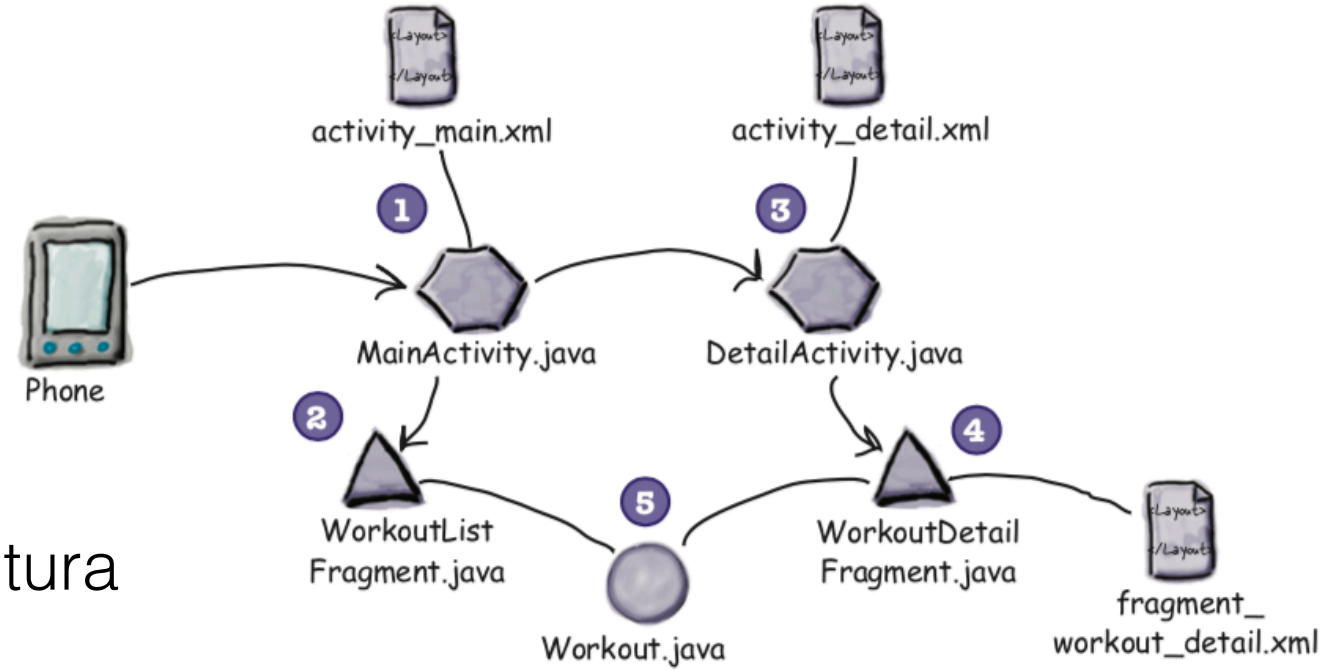
1. **When the app gets launched, it starts MainActivity.**
   `MainActivity` uses *activity_main.xml* for its layout, and contains a fragment called `WorkoutListFragment`.

2. **WorkoutListFragment displays a list of workouts.**

3. **When the user clicks on one of the workouts, DetailActivity starts.**
   `DetailActivity` uses *activity_detail.xml* for its layout, and contains a fragment called `WorkoutDetailFragment`.

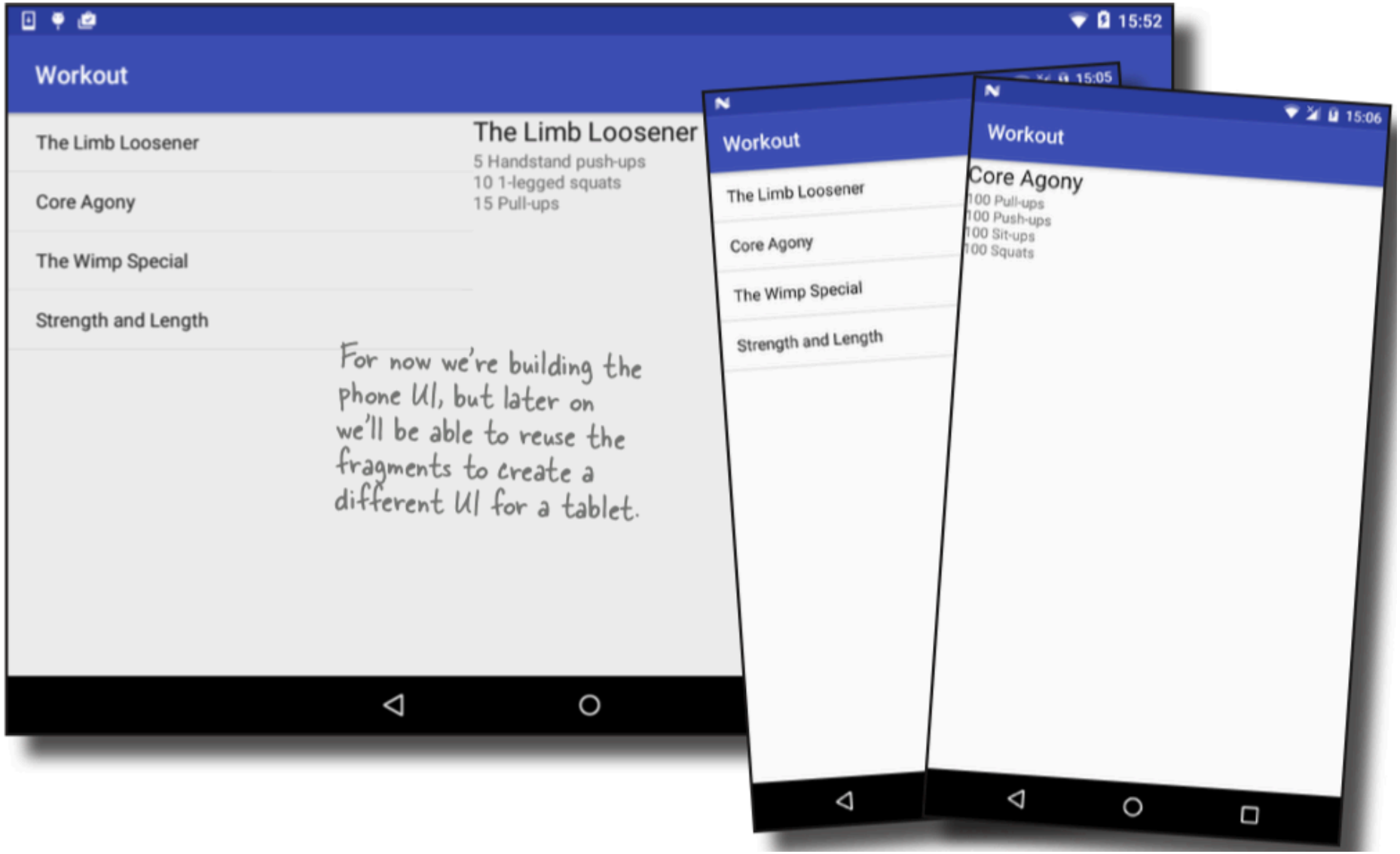
4. **WorkoutDetailFragment uses fragment_workout_detail.xml for its layout.**
   It displays the details of the workout the user has selected.

5. **WorkoutListFragment and WorkoutDetailFragment get their workout data from *Workout.java*.**
   Workout.java contains an array of `Workouts`.

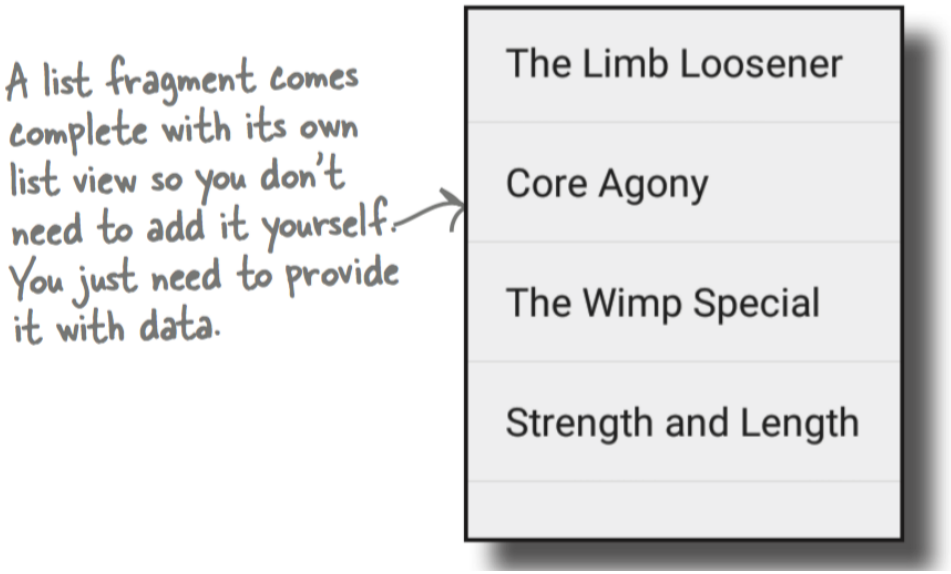


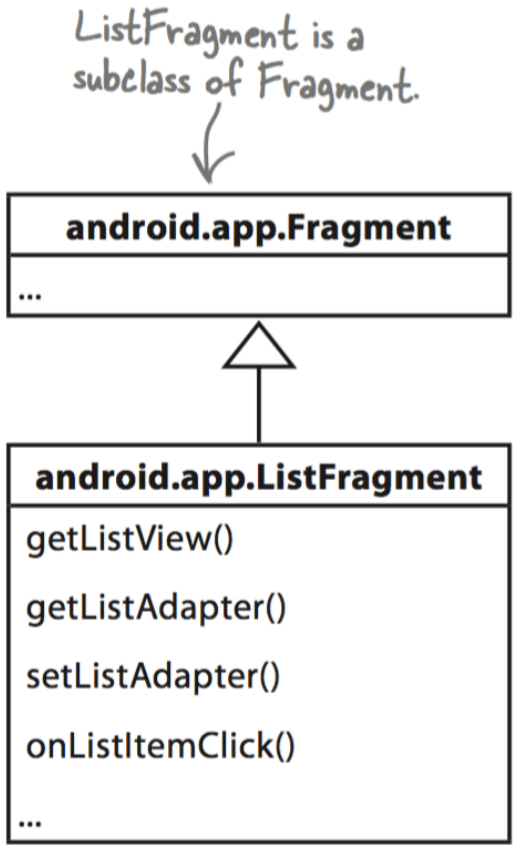
Recordando la estructura de la aplicación

Necesitamos un fragmento con una lista



Un ListFragment es un fragmento que contiene sólo una lista

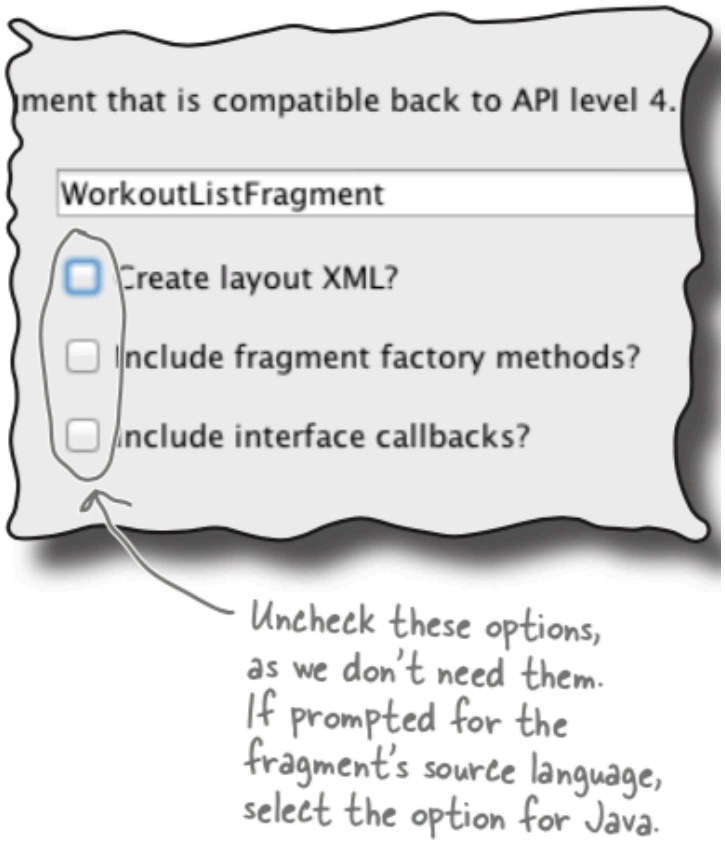




Como crear un fragmento lista

You add a list fragment to your project in the same way you add a normal fragment. Highlight the *com.hfad.workout* package in the *app/src/main/java* folder, then go to File→New...→Fragment→Fragment (Blank). Name the fragment "WorkoutListFragment", and then uncheck the options to create layout XML, and also the options to include fragment factory methods and interface callbacks (list fragments define their own layouts programmatically, so you don't need Android Studio to create one for you). When you click on the Finish button, Android Studio creates a new list fragment in a file called *WorkoutListFragment.java* in the *app/src/main/java* folder.

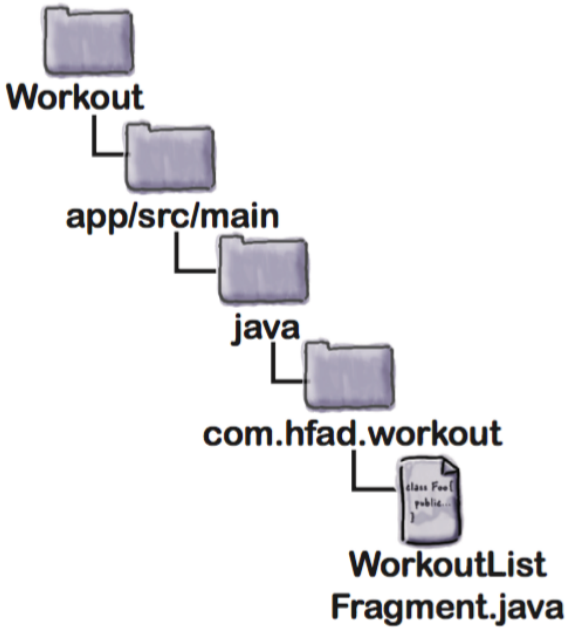
Here's what the basic code looks like to create a list fragment. As you can see, it's very similar to that of a normal fragment. Replace the code in *WorkoutListFragment.java* with the code below:



Creando un fragmento lista

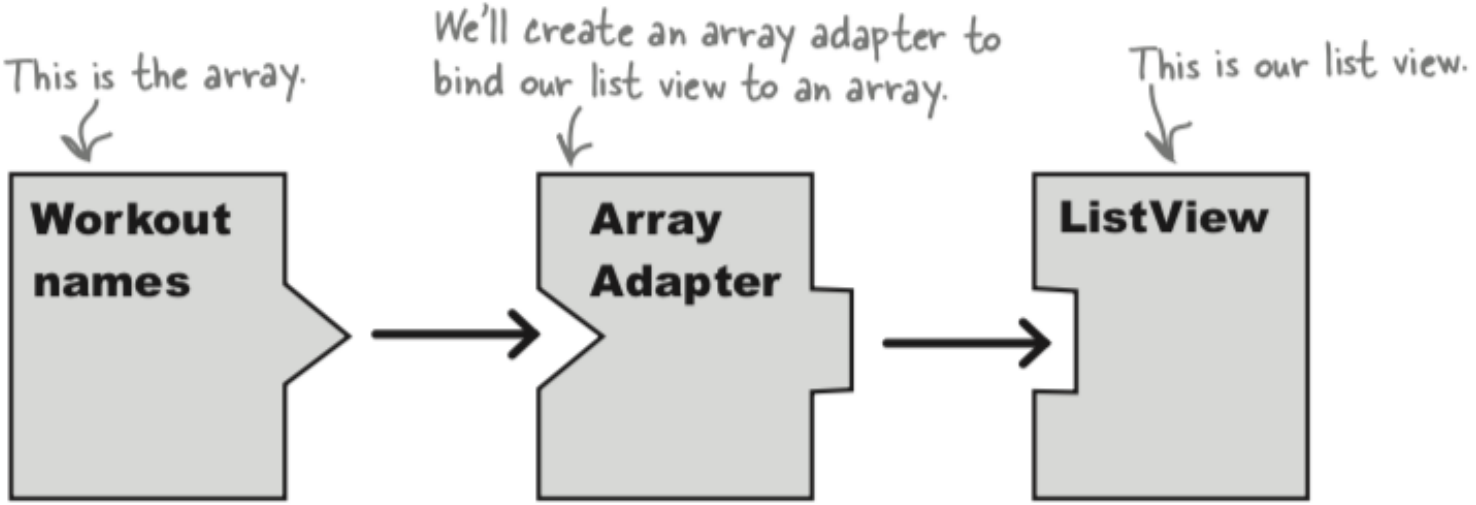
```
package com.hfad.workout;

import android.os.Bundle;
import android.app.ListFragment;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

public class WorkoutListFragment extends ListFragment {

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        return super.onCreateView(inflater, container, savedInstanceState);
    }
}
```

The activity needs to extend ListFragment, not Fragment.



Calling the superclass onCreateView() method gives you the default layout for the ListFragment.

Utilizando un ArrayAdapter para asignar valores al ListView



Los fragmentos no son un tipo de Context

```
ArrayAdapter<DataType> listAdapter = new ArrayAdapter<DataType>(
                context, android.R.layout.simple_list_item_1, array);
```

```
ArrayAdapter<DataType> listAdapter = new ArrayAdapter<DataType>(
inflator.getContext(), android.R.layout.simple_list_item_1, array);
```

This gets you the current context

```
setListAdapter(listAdapter);
```

```
package com.hfad.workout;

import android.os.Bundle;
import android.support.v4.app.ListFragment;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;

public class WorkoutListFragment extends ListFragment {

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        String[] names = new String[Workout.workouts.length];
        for (int i = 0; i < names.length; i++) {
            names[i] = Workout.workouts[i].getName();
        }

        ArrayAdapter<String> adapter = new ArrayAdapter<>(
                inflater.getContext(), android.R.layout.simple_list_item_1,
                names);
        setListAdapter(adapter);

        return super.onCreateView(inflater, container, savedInstanceState);
    }
}
```

We're using this class in the onCreateView() method.

Create a String array of the workout names.

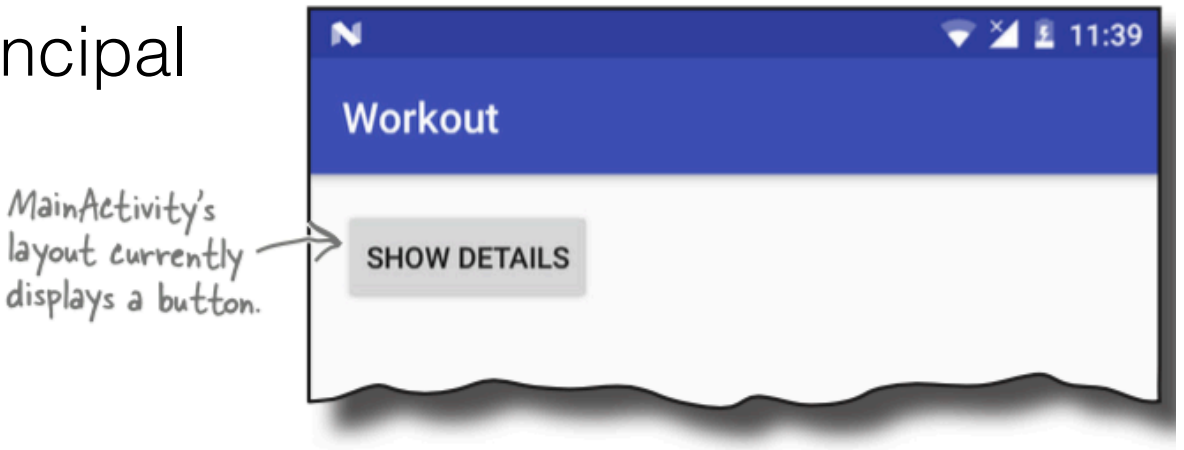
Create an array adapter.

Get the context from the layout inflater.

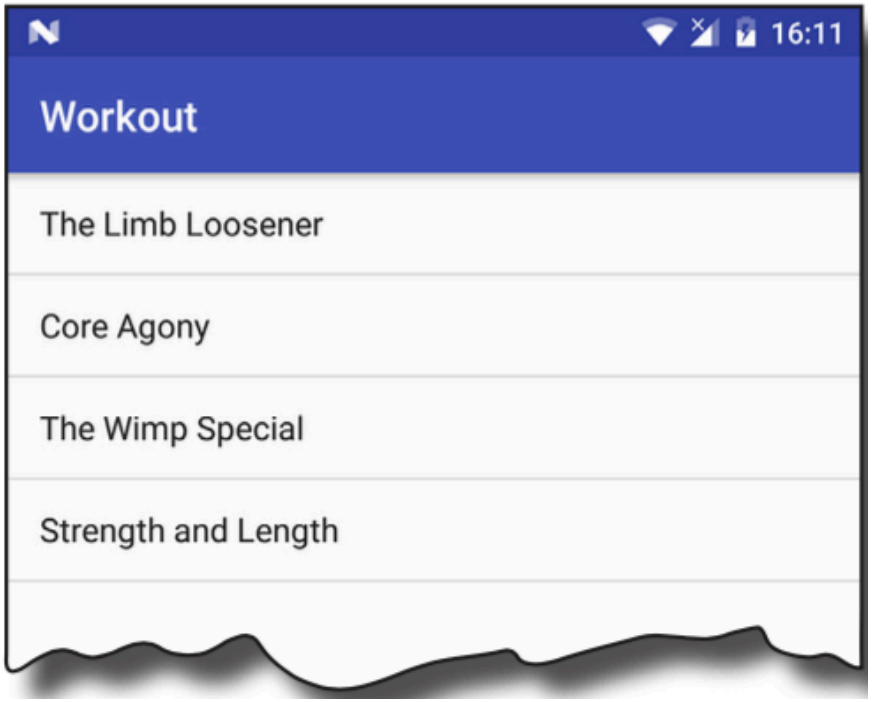
Bind the array adapter to the list view.

Actualización del código de WorkoutListFragment

Mostrando el fragmento en el layout de la actividad principal



MainActivity's layout currently displays a button.



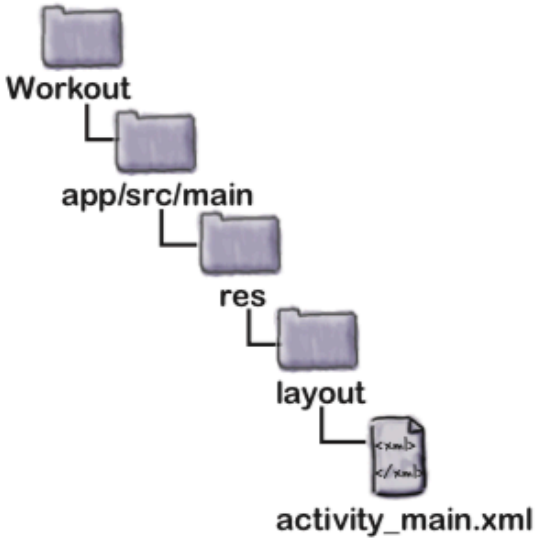
We're going to change the layout so that it displays WorkoutListFragment instead of the button.

Modificando el código de activity_main.xml

Our layout only contains a single fragment, so we can get rid of the LinearLayout.

```
<?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.workout.MainActivity">

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:onClick="onShowDetails"
        android:text="@string/details_button" />
</LinearLayout>
```

We no longer need this button.

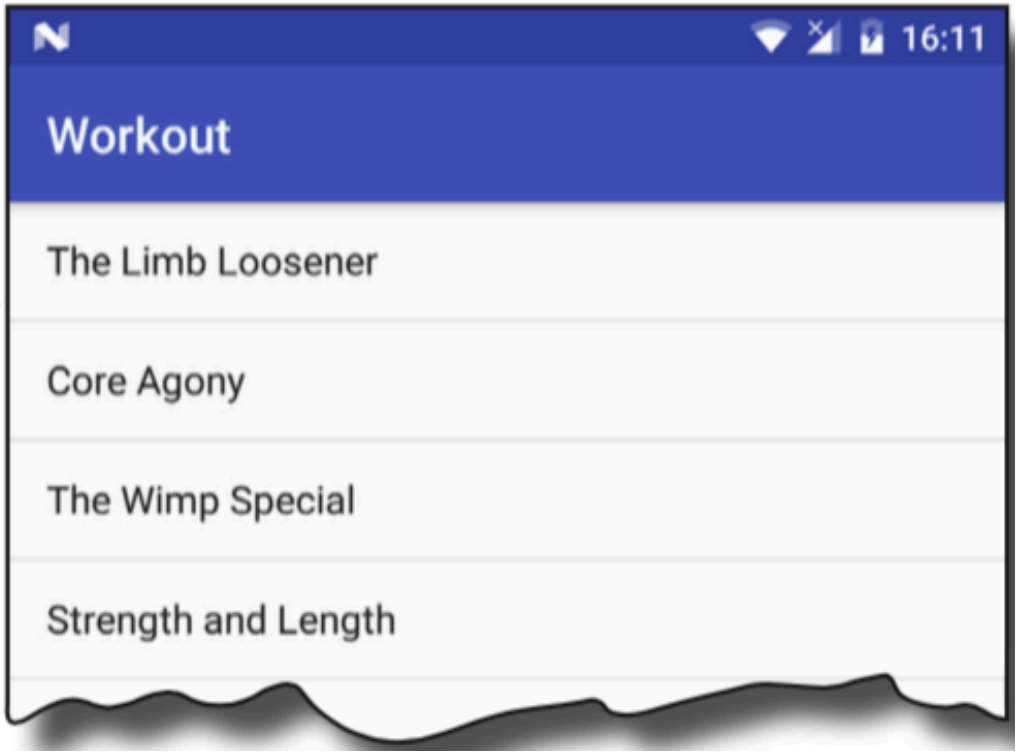


Here's the fragment.

```
<fragment xmlns:android="http://schemas.android.com/apk/res/android"
    android:name="com.hfad.workout.WorkoutListFragment"
    android:layout_width="match_parent"
    android:layout_height="match_parent"/>
```

Prueba

Here's a list of all the workout titles from the Workout class.



Conectando la lista con los detalles

1. Add code to `WorkoutListFragment` that waits for a workout to be clicked.
2. When that code runs, call some code in *MainActivity.java* that will start `DetailActivity`, passing it the ID of the workout.
3. Get `DetailActivity` to pass the ID to `WorkoutDetailFragment` so that the fragment can display details of the correct workout.

Desacoplando el fragmento con una interfaz

1. Define the listener interface

We'll call the interface Listener.

```
interface Listener {
    void itemClicked(long id);
};
```

Any activities that implement the Listener interface must include this method. We'll use it to get the activity to respond to items in the fragment being clicked.

Desacoplando el fragmento con una interfaz

2. Register the listener

```
public void onAttach(Context context) {
    super.onAttach(context);
    this.listener = (Listener)context;
}
```

This is the context (in this case, the activity) the fragment is attached to.

Desacoplando el fragmento con una interfaz

3. Respond to clicks

```
public void onListItemClick(ListView listView, View itemView, int position, long id) {
    if (listener != null) {
        listener.itemClicked(id);
    }
}
```

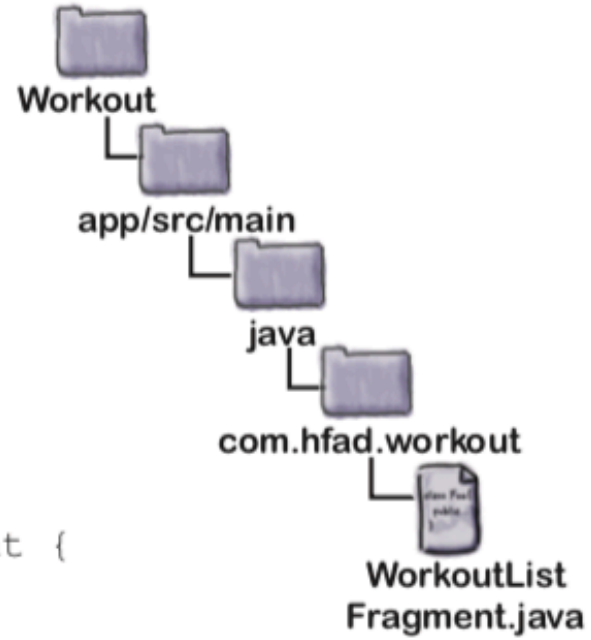
Call the itemClicked() method in the activity, passing it the ID of the workout the user selected.

Agregando la interfaz al fragmento de lista

```
package com.hfad.workout;

import android.os.Bundle;
import android.support.v4.app.ListFragment;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.content.Context;
import android.widget.ListView;

public class WorkoutListFragment extends ListFragment {

    static interface Listener {
        void itemClicked(long id);
    };

    private Listener listener;
```

Import these classes.

Add the listener to the fragment.



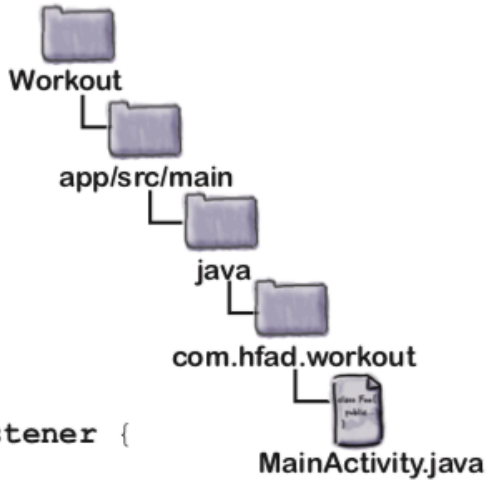
Agregando la interfaz al fragmento de lista

```
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        String[] names = new String[Workout.workouts.length];
        for (int i = 0; i < names.length; i++) {
            names[i] = Workout.workouts[i].getName();
        }
        ArrayAdapter<String> adapter = new ArrayAdapter<>(
                inflater.getContext(), android.R.layout.simple_list_item_1,
                names);
        setListAdapter(adapter);
        return super.onCreateView(inflater, container, savedInstanceState);
    }

    @Override
    public void onAttach(Context context) {
        super.onAttach(context);
        this.listener = (Listener)context;
    }

    @Override
    public void onListItemClick(ListView listView, View itemView, int position, long id) {
        if (listener != null) {
            listener.itemClicked(id);
        }
    }
}
```

This is called when the fragment gets attached to the activity. Remember, the Activity class is a subclass of Context.

Tell the listener when an item in the ListView is clicked.

Implementando la interfaz




```
package com.hfad.workout;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.content.Intent;

public class MainActivity extends AppCompatActivity
                          implements WorkoutListFragment.Listener {
```

Implement the listener interface defined in WorkoutListFragment.

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

This is the method called by MainActivity's button. We've removed the button, so we no longer need this method.

```
    public void onShowDetails(View view) {
        Intent intent = new Intent(this, DetailActivity.class);
        startActivity(intent);
    }
```

This method is defined by the interface, so we need to implement it.

```
    @Override
    public void itemClicked(long id) {
        Intent intent = new Intent(this, DetailActivity.class);
        intent.putExtra(DetailActivity.EXTRA_WORKOUT_ID, (int)id);
        startActivity(intent);
    }
}
```

Pass the ID of the workout to DetailActivity. EXTRA_WORKOUT_ID is the name of a constant we'll define in DetailActivity.

Pasando el ID a WorkoutDetailFragment

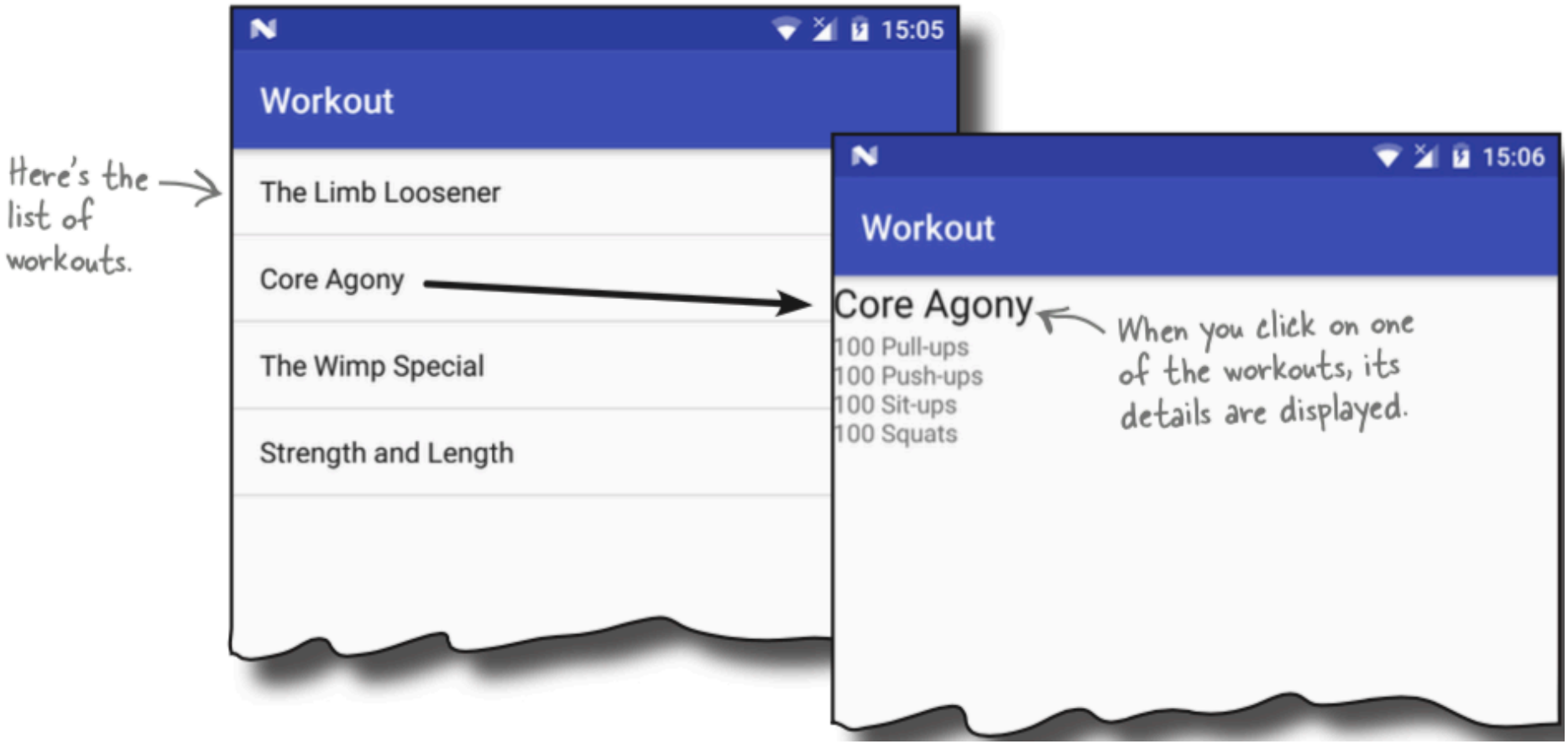
```
package com.hfad.workout;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

public class DetailActivity extends AppCompatActivity {

    public static final String EXTRA_WORKOUT_ID = "id";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_detail);
        WorkoutDetailFragment frag = (WorkoutDetailFragment)
                getSupportFragmentManager().findFragmentById(R.id.detail_frag);
        frag.setWorkout(1);
        int workoutId = (int) getIntent().getExtras().get(EXTRA_WORKOUT_ID);
        frag.setWorkout(workoutId);
    }

}
```

We're using a constant to pass the ID from MainActivity to DetailActivity to avoid hardcoding this value.

We're no longer hardcoding an ID of 1, so remove this line.

Get the ID from the intent, and pass it to the fragment via its setWorkout() method.

Prueba


Here's the list of workouts.
15:05
Workout
The Limb Loosener
Core Agony
The Wimp Special
Strength and Length
15:06
Workout
Core Agony
100 Pull-ups
100 Push-ups
100 Sit-ups
100 Squats
When you click on one of the workouts, its details are displayed.

BULLET POINTS

- A fragment is used to control part of a screen. It can be reused across multiple activities.
- A fragment has an associated layout.
- The `onCreateView()` method gets called each time Android needs the fragment's layout.
- Add a fragment to an activity's layout using the `<fragment>` element and adding a `name` attribute.
- The fragment lifecycle methods tie in with the states of the activity that contains the fragment.
- The `Fragment` class doesn't extend the `Activity` class or implement the `Context` class.
- Fragments don't have a `findViewById()` method. Instead, use the `getView()` method to get a reference to the root view, then call the view's `findViewById()` method.
- A list fragment is a fragment that comes complete with a `ListView`. You create one by subclassing `ListFragment`.