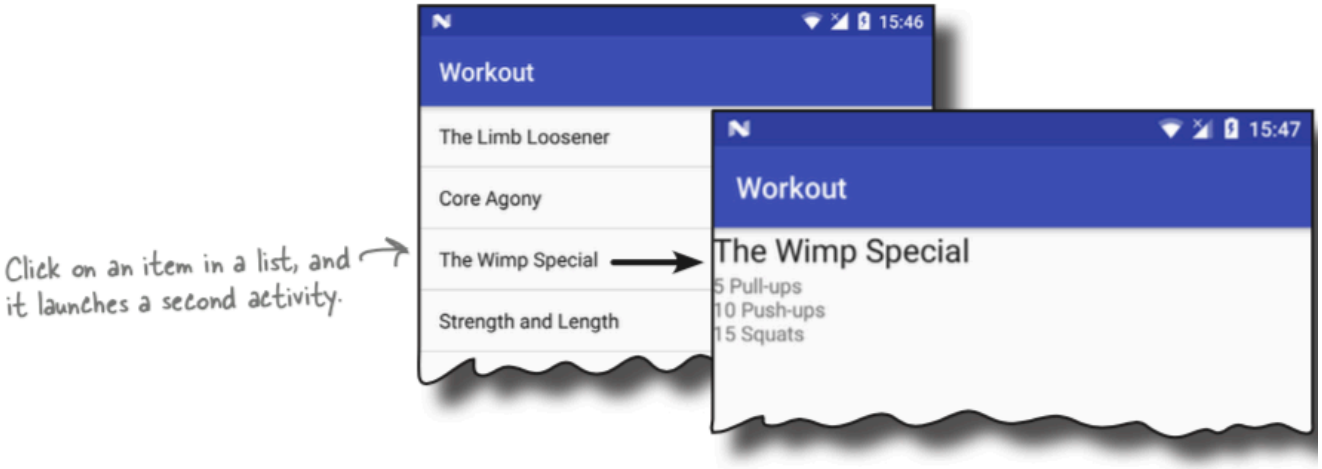
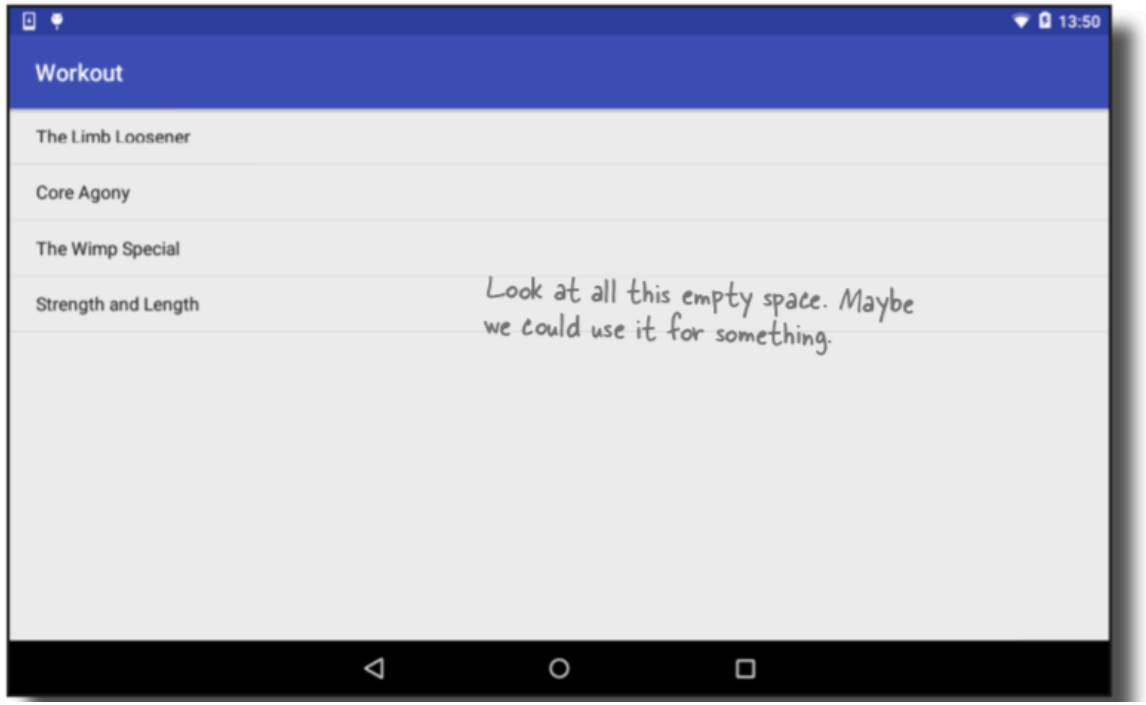
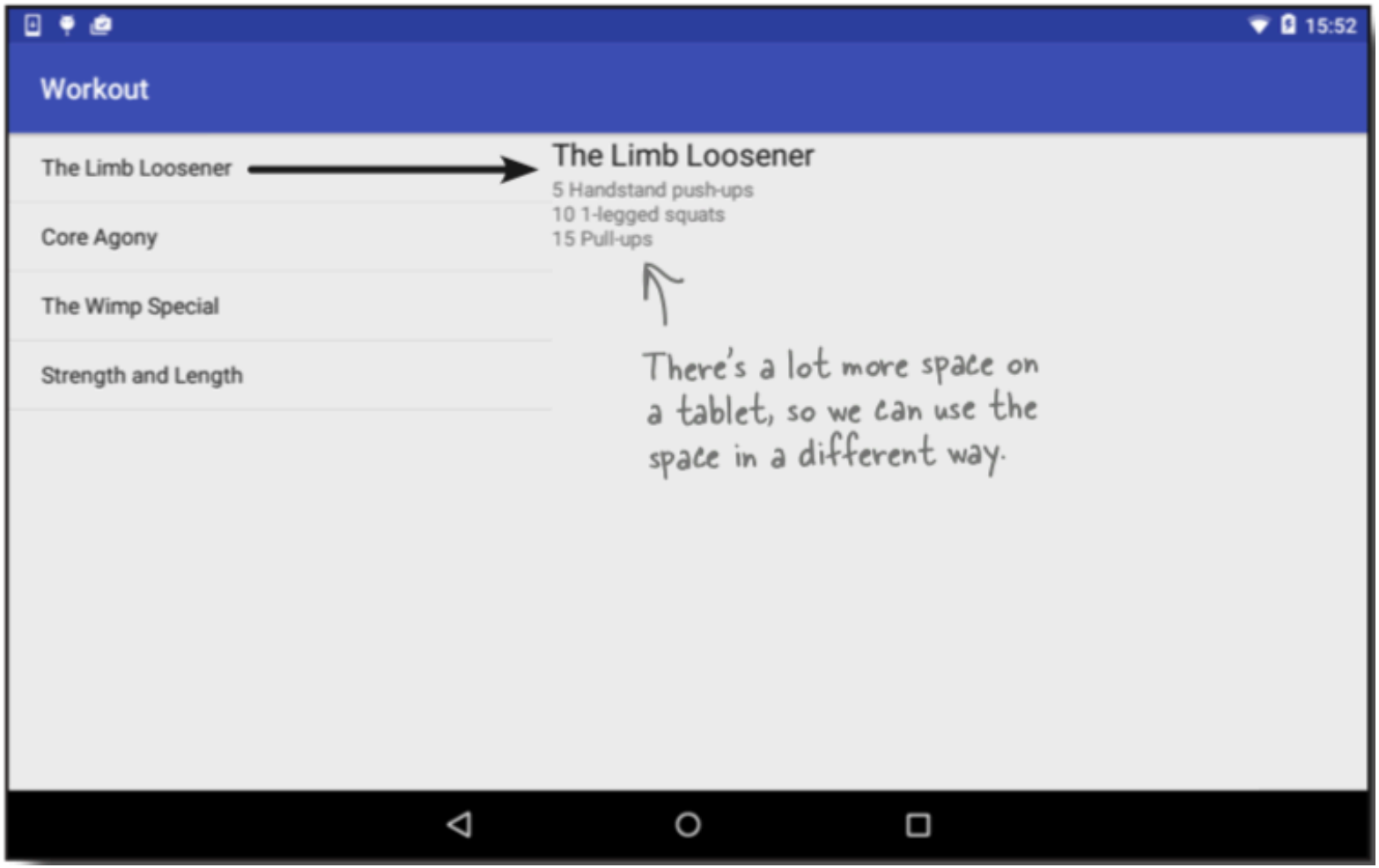
12
Fragmentos para interfaces más grandes

La aplicación Workout en teléfono y tableta





Diseñando para interfaces más grandes



La versión para teléfono

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

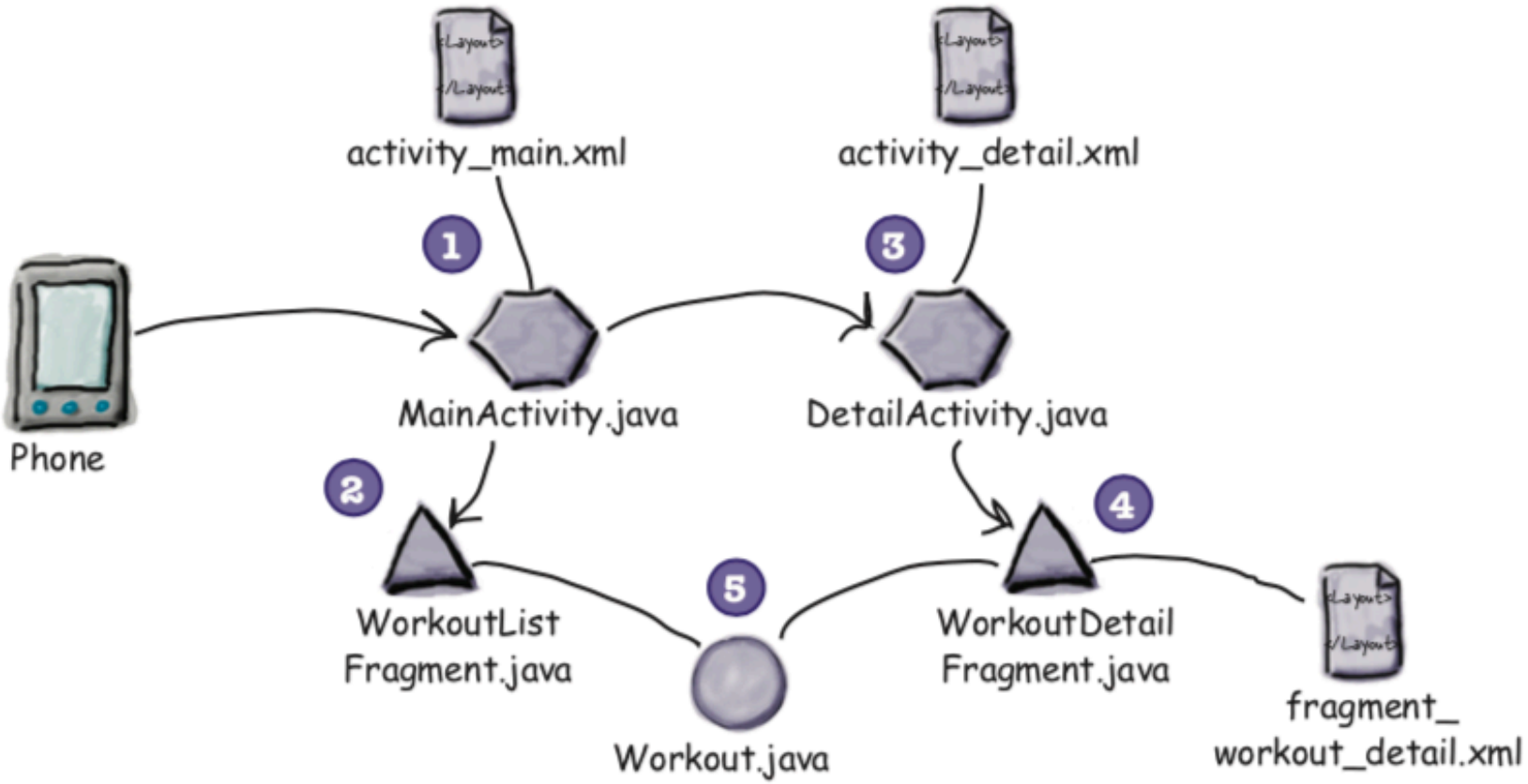
2. **WorkoutListFragment displays a list of workouts.**

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`.

La versión para teléfono



La versión para tableta

1. **When the app gets launched, it starts MainActivity as before.**
   `MainActivity` uses *activity_main.xml* for its layout.

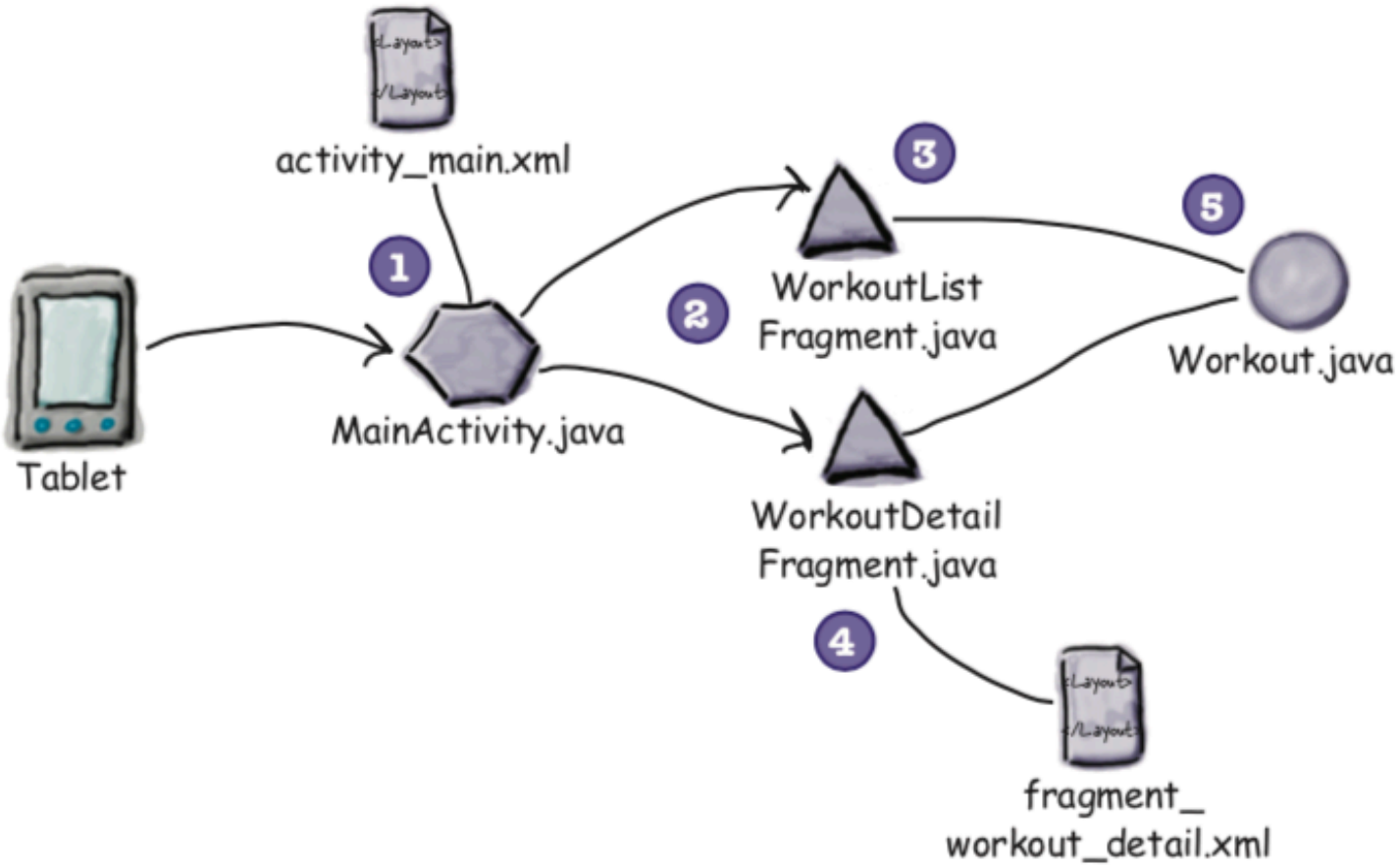
2. **MainActivity's layout displays two fragments, WorkoutListFragment and WorkoutDetailFragment.**

3. **WorkoutListFragment displays a list of workouts.**
   It's a list fragment, so it has no extra layout file.

4. **When the user clicks on one of the workouts, its details are displayed in WorkoutDetailFragment.**
   `WorkoutDetailFragment` uses *fragment_workout_detail.xml* for its layout.

5. **Both fragments get their workout data from *Workout.java* as before.**

La versión para tableta

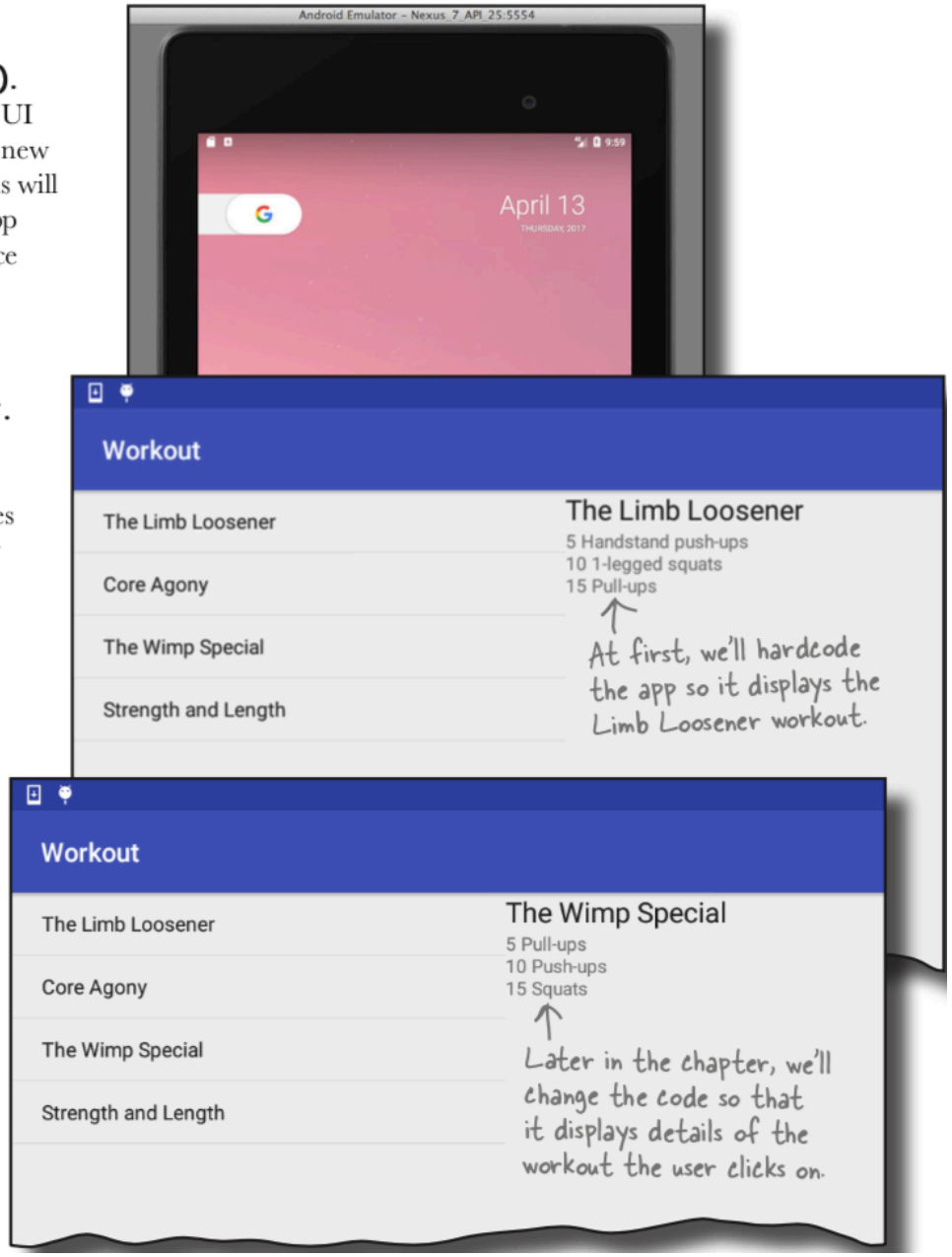


Qué vamos a hacer ?

1. **Create a tablet AVD (Android Virtual Device).**
We're going to create a new UI for a tablet, so we'll create a new tablet AVD to run it on. This will allow us to check how the app looks and behaves on a device with a larger screen.

2. **Create a new tablet layout.**
We'll reuse the fragments we've already created in a new layout that's designed to work on devices with larger screens. We'll display details of the first workout in the first instance so that we can see the fragments side by side.

3. **Display details of the workout the user selects.**
We'll update the app so that when the user clicks on one of the workouts, we'll display the details of the workout the user selected.



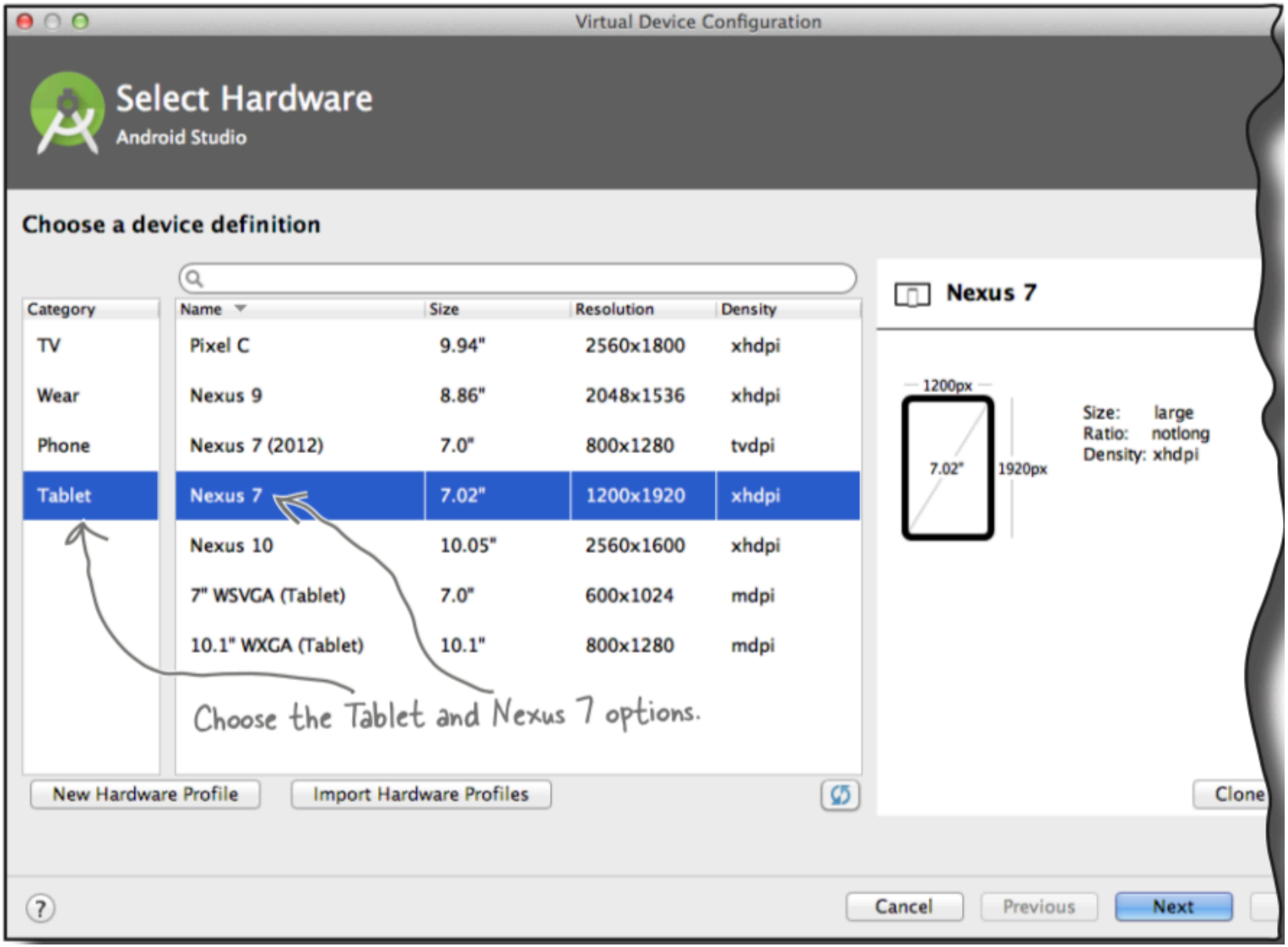
Creación de un AVD para tableta

Open the Android Virtual Device Manager

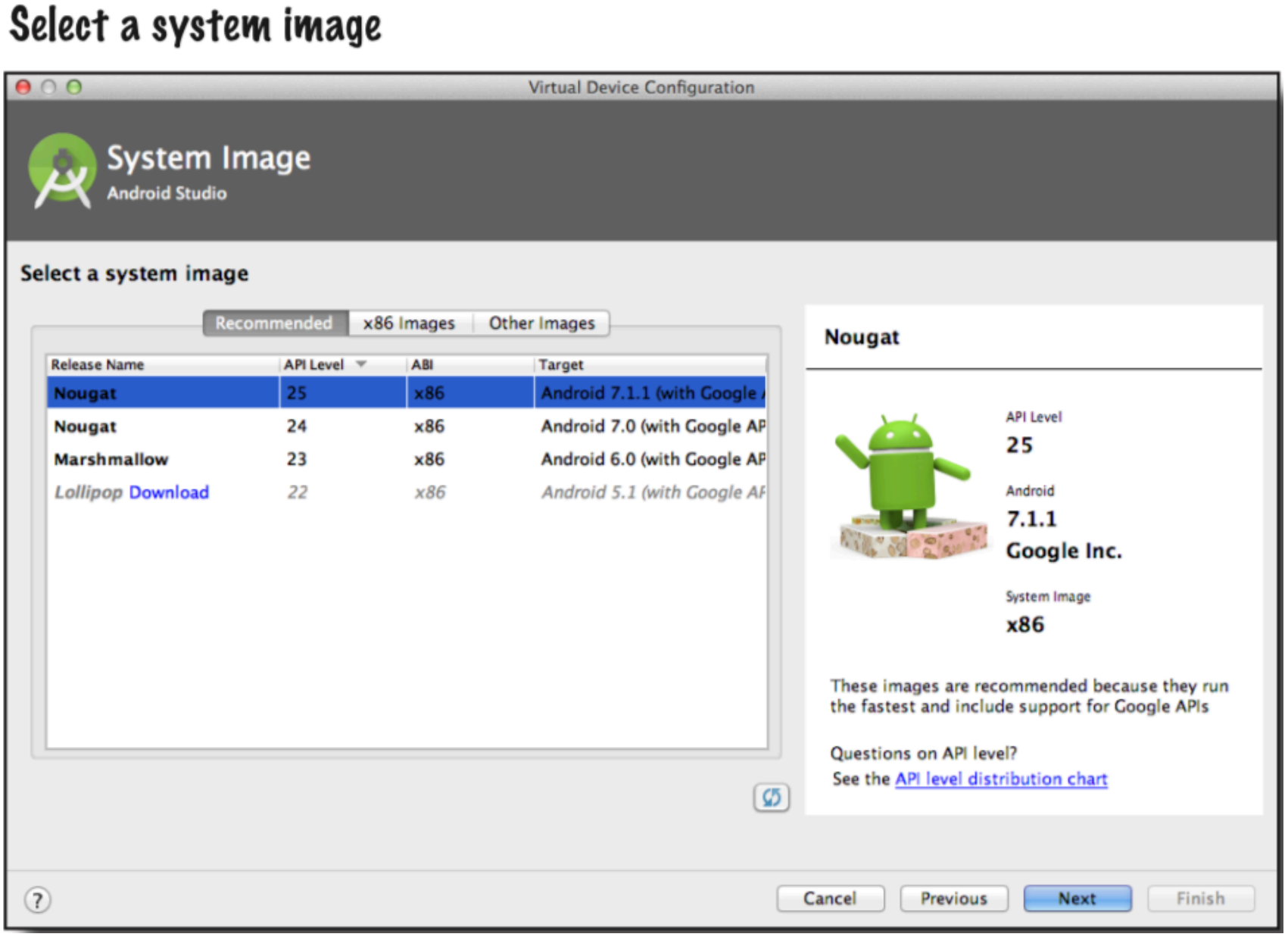
Click on the Create Virtual Device button to create an AVD.

Creación de un AVD para tableta

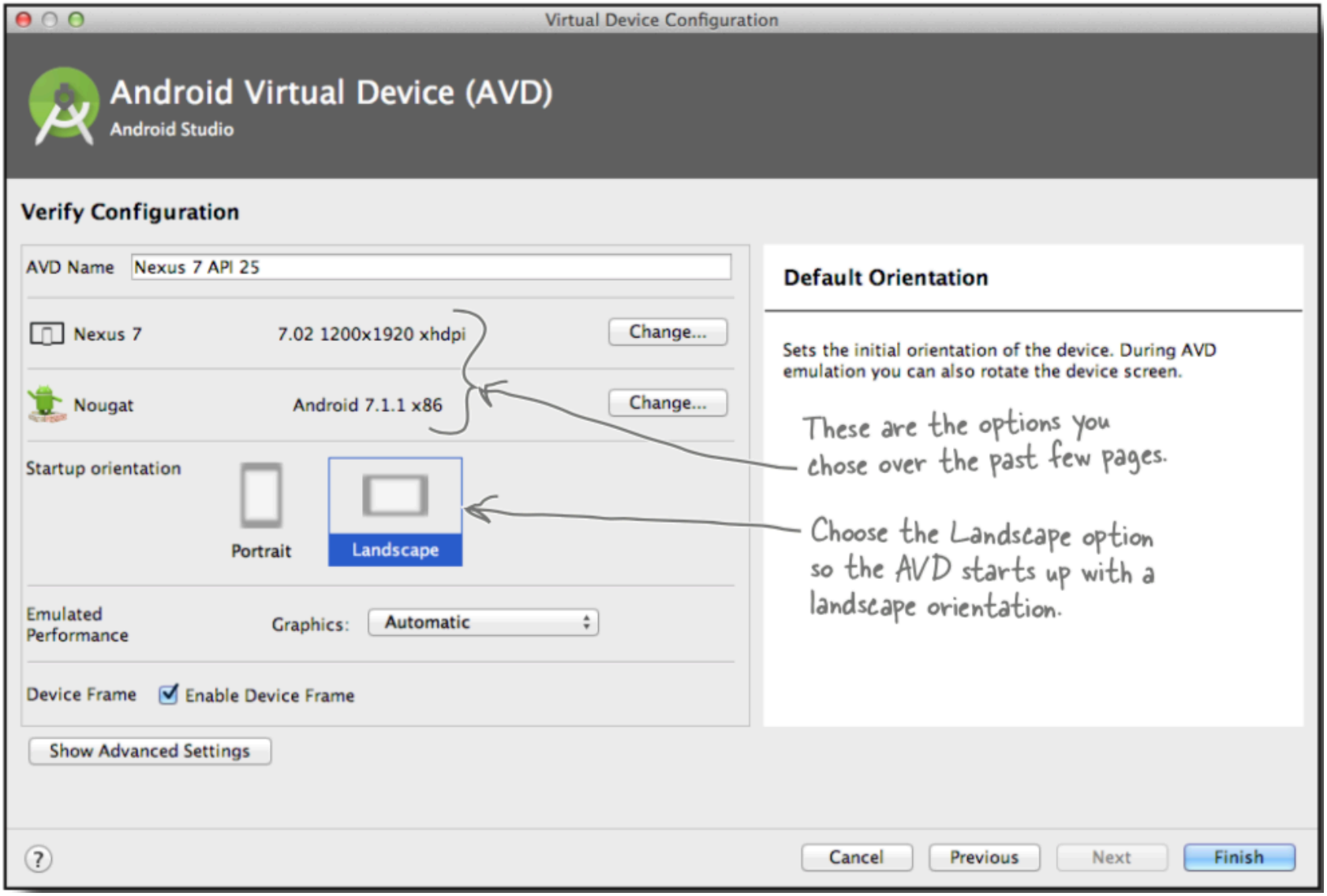
Select the hardware



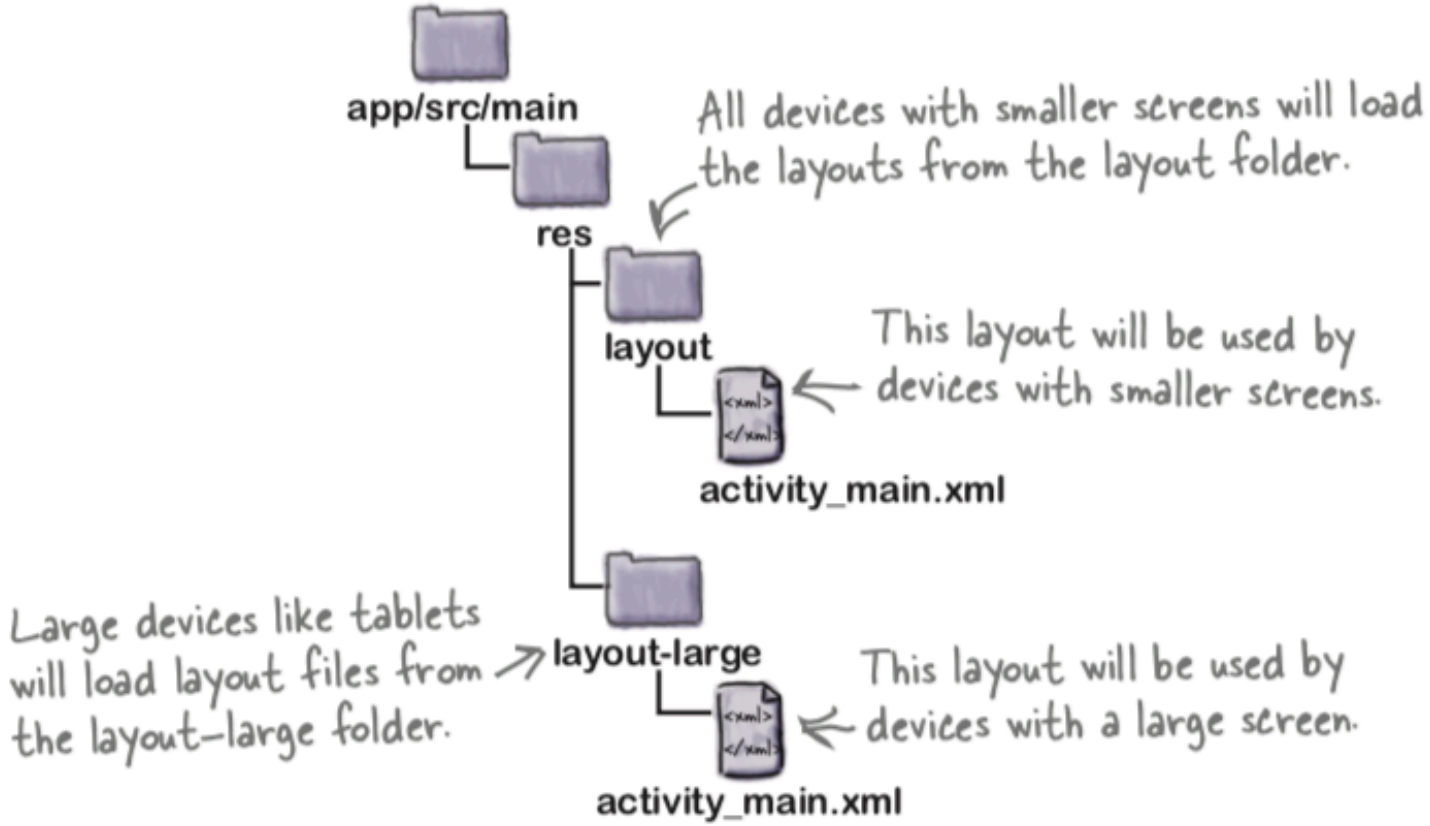
Creación de un AVD para tableta



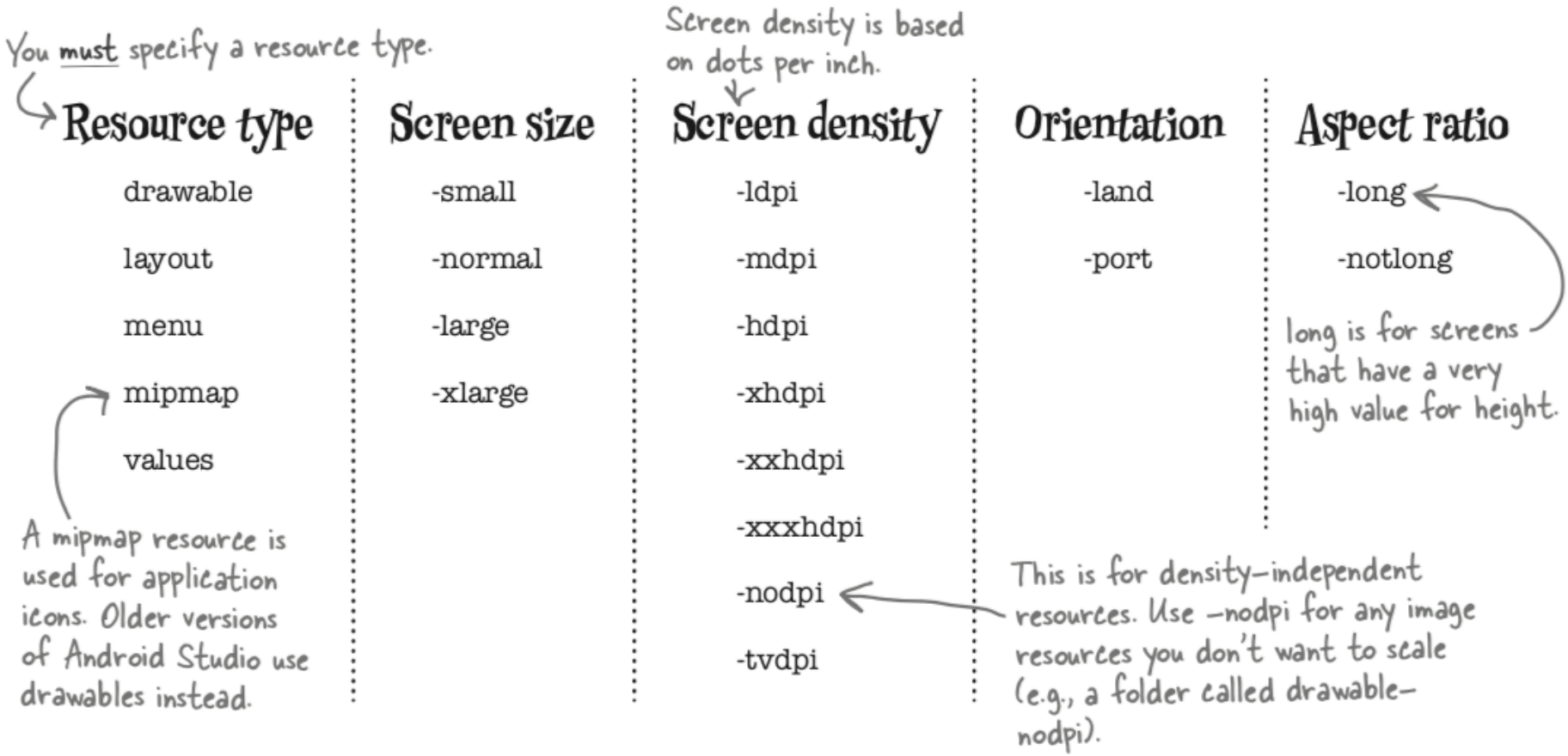
Creación de un AVD para tableta



Carpetas diferentes para cada tipo de dispositivo



Opciones para las carpetas



Las tabletas usan la carpeta layout-large

activity_main para tableta

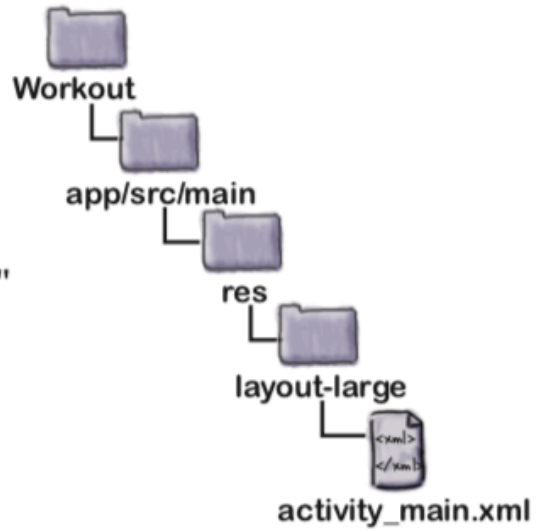
We're putting the fragments in a LinearLayout with a horizontal orientation so the two fragments will be displayed alongside each other.

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

    <fragment
        android:name="com.hfad.workout.WorkoutListFragment"
        android:id="@+id/list_frag"
        android:layout_width="0dp"
        android:layout_weight="2"
        android:layout_height="match_parent"/>

    <fragment
        android:name="com.hfad.workout.WorkoutDetailFragment"
        android:id="@+id/detail_frag"
        android:layout_width="0dp"
        android:layout_weight="3"
        android:layout_height="match_parent"/>
</LinearLayout>
```

Our layout already includes WorkoutListFragment.

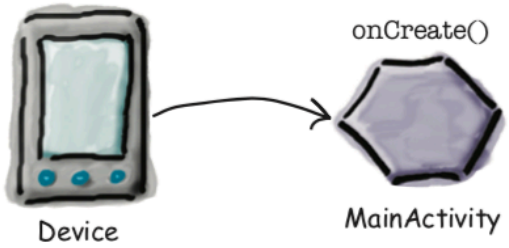
The fragments need IDs so that Android doesn't lose track of where to put each fragment.

We're adding WorkoutDetailFragment to MainActivity's layout.

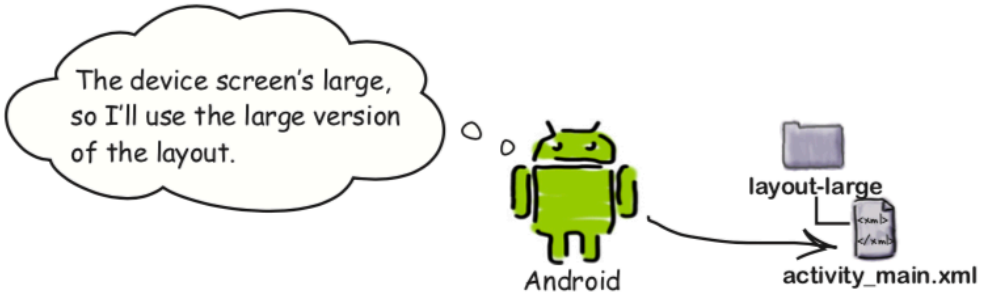


Qué hace el código ?

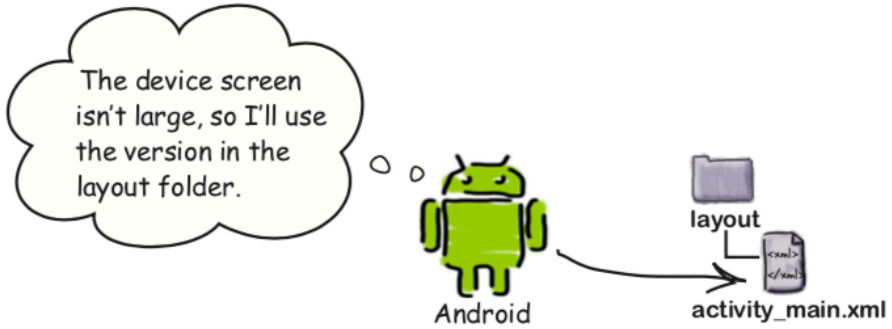
1 **When the app is launched, MainActivity gets created.**
`MainActivity`'s `onCreate()` method runs. This specifies that *activity_main.xml* should be used for `MainActivity`'s layout.



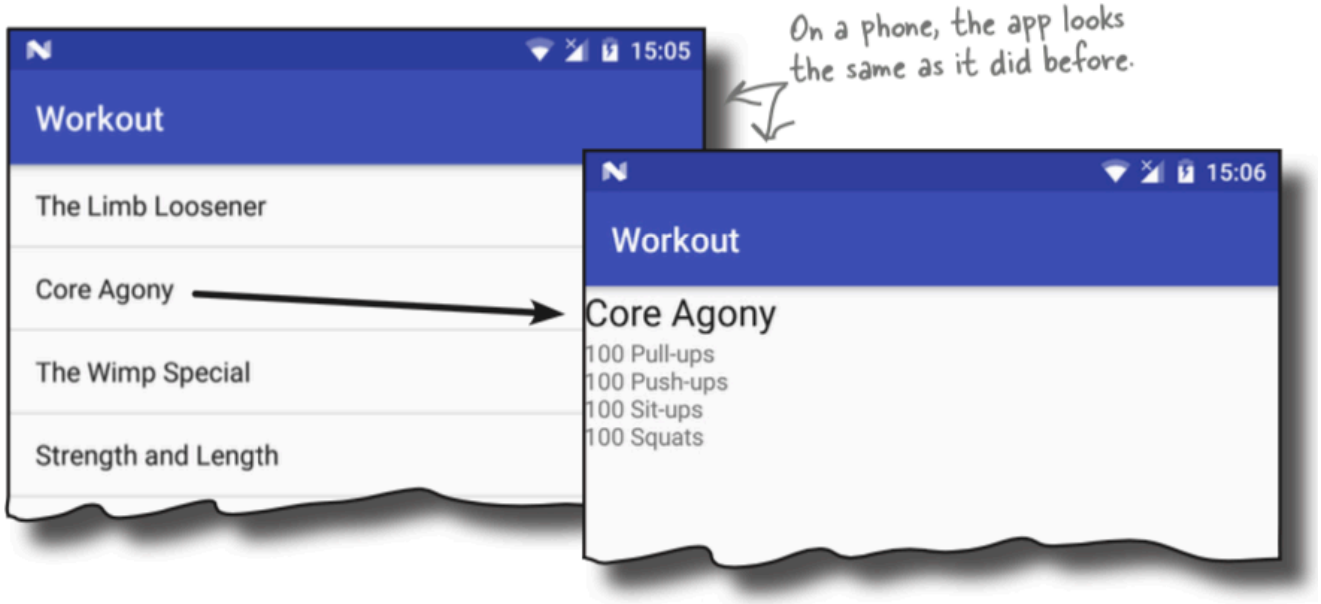
2a **If the app's running on a tablet, it uses the version of activity_main.xml that's in the layout-large folder.**
The layout displays `WorkoutListFragment` and `WorkoutDetailFragment` side by side.

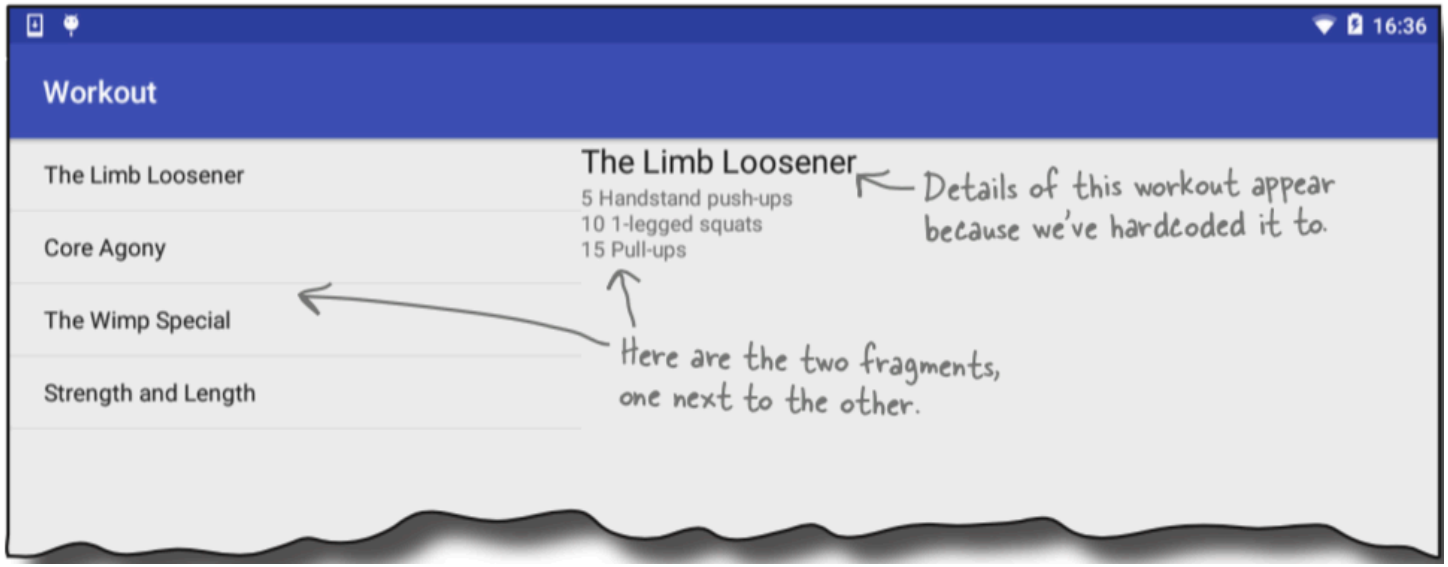


2b **If the app's running on a device with a smaller screen, it uses the version of activity_main.xml that's in the layout folder.**
The layout displays `WorkoutListFragment` on its own.



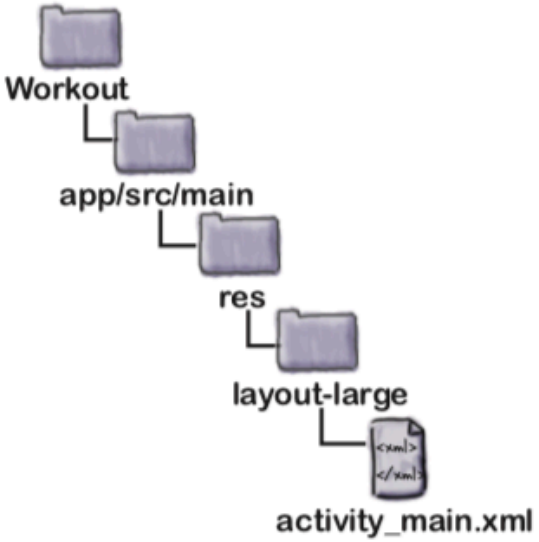
activity_main para tableta





Modificando el layout para tableta

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

    <fragment
        android:name="com.hfad.workout.WorkoutListFragment"
        android:id="@+id/list_frag"
        android:layout_width="0dp"
        android:layout_weight="2"
        android:layout_height="match_parent"/>

    <FrameLayout
        android:id="@+id/fragment_container"
        android:layout_width="0dp"
        android:layout_weight="3"
        android:layout_height="match_parent"/>
</LinearLayout>
```

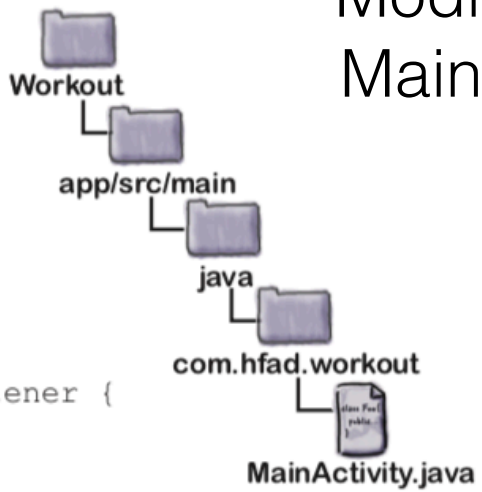


We're going to display the fragment inside a FrameLayout.

We'll add the fragment to the frame layout programmatically.

We'll give the FrameLayout an ID of fragment_container so we can refer to it in our activity code.

Modificando MainActivity




```
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 {

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

    @Override
    public void itemClicked(long id) {
        View fragmentContainer = findViewById(R.id.fragment_container);
        if (fragmentContainer != null) {
            //Add the fragment to the FrameLayout
        } else {
            Intent intent = new Intent(this, DetailActivity.class);
            intent.putExtra(DetailActivity.EXTRA_WORKOUT_ID, (int) id);
            startActivity(intent);
        }
    }
}
```

We've not changed this method.

Get a reference to the frame layout that will contain WorkoutDetailFragment. This will only exist if the app is being run on a device with a large screen.

We need to write code that will run if the frame layout exists.

If the frame layout doesn't exist, the app must be running on a device with a smaller screen. In that case, start DetailActivity and pass it the ID of the workout as before.

Utilizando transacciones en los fragmentos

1. **Begin the transaction.**
   This tells Android that you're starting a series of changes that you want to record in the transaction.

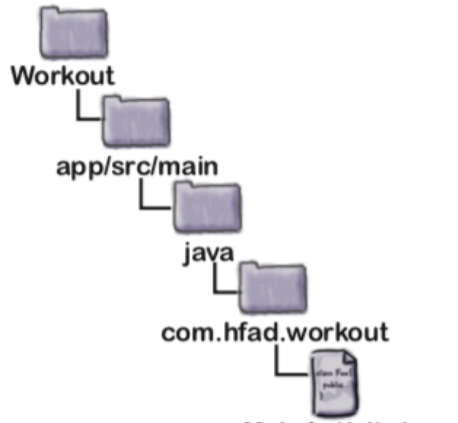
2. **Specify the changes.**
   These are all the actions you want to group together in the transaction. This can include adding, replacing, or removing a fragment, updating its data, and adding it to the back stack.

3. **Commit the transaction.**
   This finishes the transaction and applies the changes.

```
package com.hfad.workout;

import android.support.v4.app.FragmentTransaction;
...

public class MainActivity extends AppCompatActivity
                          implements WorkoutListFragment.Listener {

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

    @Override
    public void itemClicked(long id) {
        View fragmentContainer = findViewById(R.id.fragment_container);
        if (fragmentContainer != null) {
            WorkoutDetailFragment details = new WorkoutDetailFragment();
            FragmentTransaction ft = getSupportFragmentManager().beginTransaction();
            details.setWorkout(id);
            ft.replace(R.id.fragment_container, details);
            ft.setTransition(FragmentTransaction.TRANSIT_FRAGMENT_FADE);
            ft.addToBackStack(null);
            ft.commit();
        } else {
            Intent intent = new Intent(this, DetailActivity.class);
            intent.putExtra(DetailActivity.EXTRA_WORKOUT_ID, (int) id);
            startActivity(intent);
        }
    }
}
```

We're using a FragmentTransaction from the Support Library as we're using Support Library Fragments.

We haven't changed this method.



Start the fragment transaction.

Add the transaction to the back stack.

Replace the fragment.

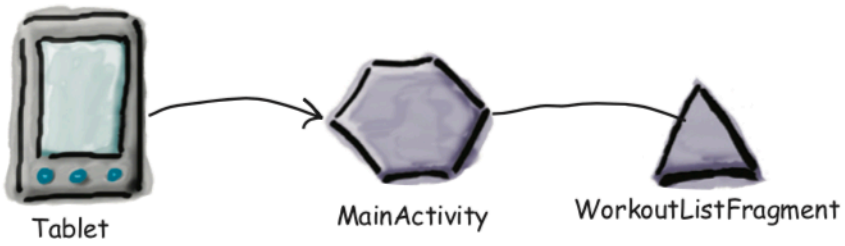
Get the new and old fragments to fade in and out.

Commit the transaction.

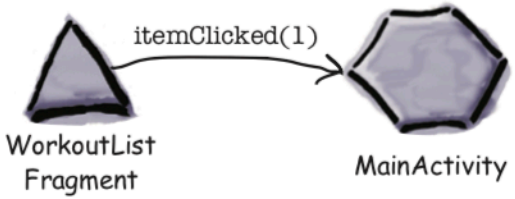
Modificando MainActivity

Qué pasa al ejecutar el código ?

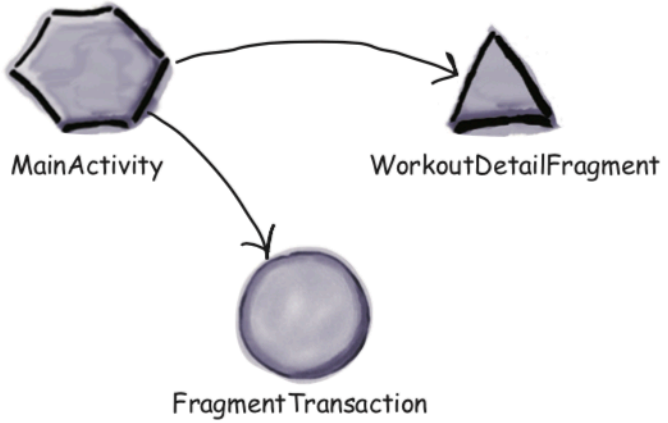
1 **The app is launched on a tablet and MainActivity starts.**
`WorkoutListFragment` is attached to `MainActivity`, and `MainActivity` is registered as a listener on `WorkoutListFragment`.



2 **When an item is clicked in WorkoutListFragment, the fragment's onListItemClick() method is called.**
This calls `MainActivity`'s `itemClicked()` method, passing it the ID of the workout that was clicked; in this example, the ID is `1`.

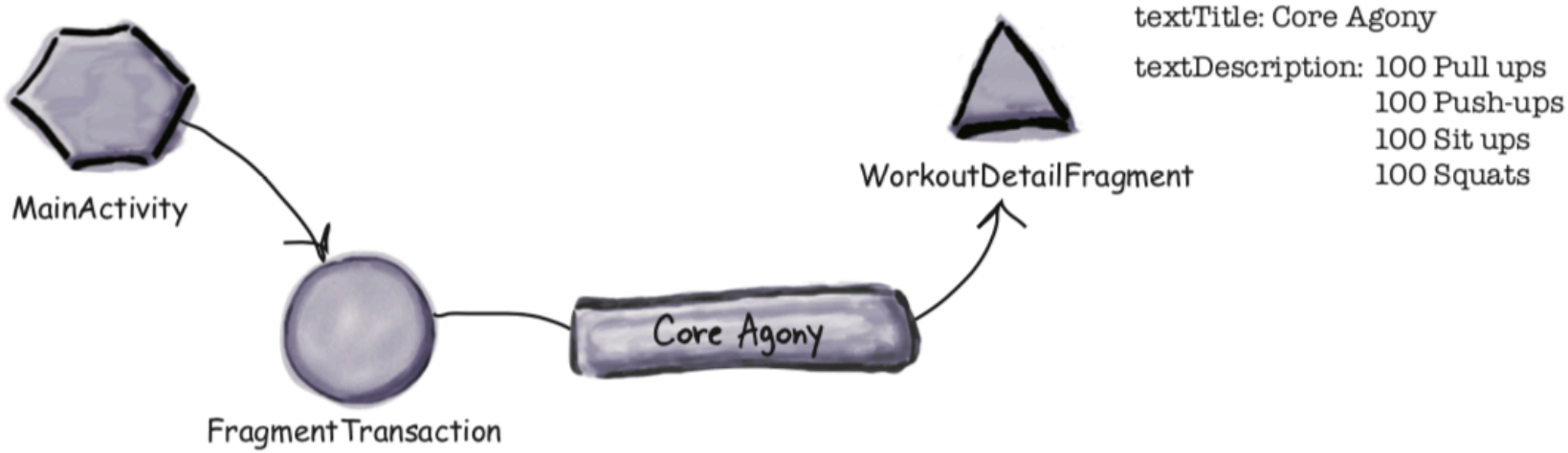


3 **MainActivity's itemClicked() method sees that the app is running on a tablet.**
It creates a new instance of `WorkoutDetailFragment`, and begins a new fragment transaction.



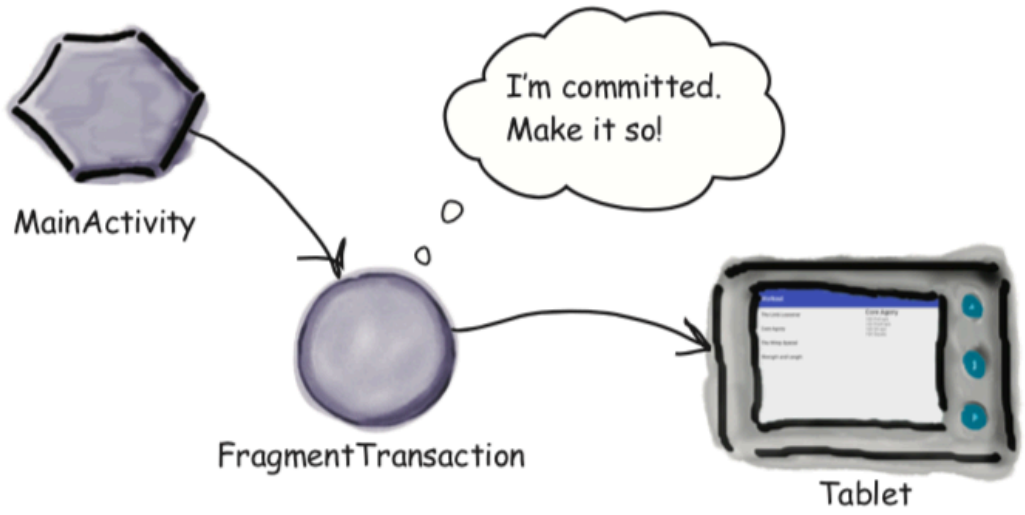
4 **As part of the transaction, WorkoutDetailFragment's views are updated with details of the workout that was selected, in this case the one with ID 1.**

The fragment is added to the `FrameLayout` `fragment_container` in `MainActivity`'s layout, and the whole transaction is added to the back stack.



5 **MainActivity commits the transaction.**

All of the changes specified in the transaction take effect, and the `WorkoutDetailFragment` is displayed next to `WorkoutListFragment`.

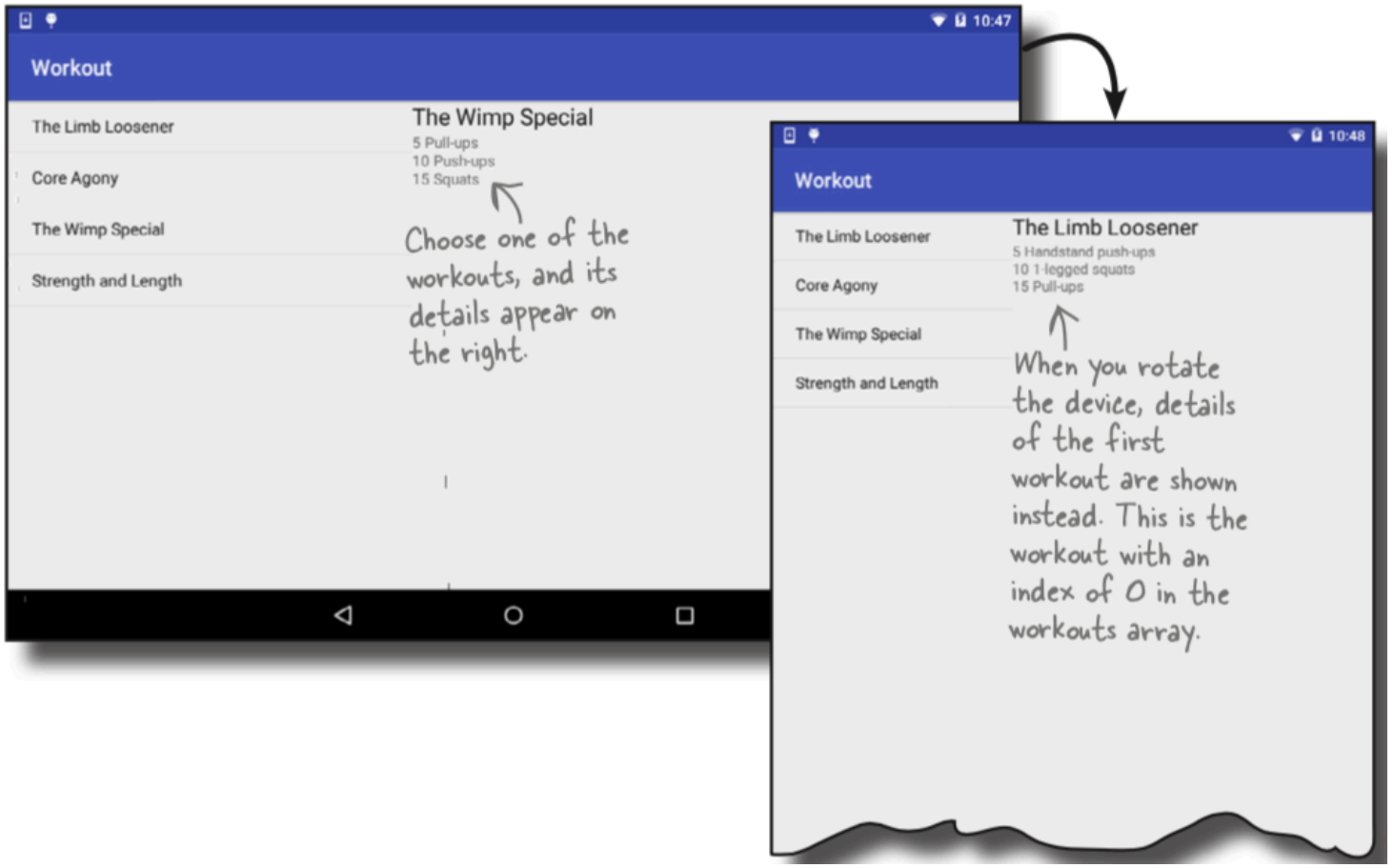


Qué pasa al ejecutar el código ?


15:43
Workout
The Limb Loosener
Core Agony
The Wimp Special
Strength and Length
The right side of the screen is empty when you start the app, as the user hasn't chosen a workout yet.
15:44
Workout
The Limb Loosener
Core Agony
The Wimp Special
Strength and Length
The Limb Loosener
5 Handstand push-ups
10 1-legged squats
15 Pull-ups
When the user clicks on the Limb Loosener workout, its details get displayed.
15:45
Workout
The Limb Loosener
Core Agony
The Wimp Special
Strength and Length
The Wimp Special
5 Pull-ups
10 Push-ups
15 Squats
The user then clicks on the Wimp Special workout and its details get displayed.
15:46
Workout
The Limb Loosener
Core Agony
The Wimp Special
Strength and Length
The Limb Loosener
5 Handstand push-ups
10 1-legged squats
15 Pull-ups
When the user clicks on the Back button, the app goes back to the Limb Loosener workout.

Prueba

Al girar el dispositivo la aplicación falla



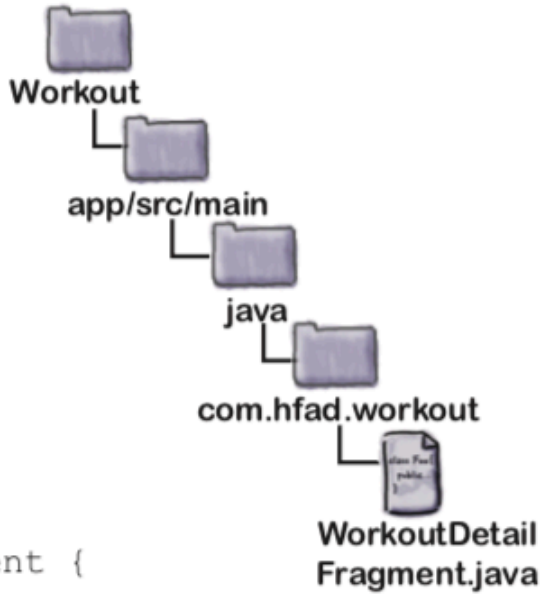
Almacenando el estado del fragmento

```
public void onCreate(Bundle savedInstanceState){
    super.onCreate(savedInstanceState);
    if (savedInstanceState != null) {
        workoutId = savedInstanceState.getLong("workoutId");
    }
}
```

We can use this Bundle to get the previous state of the workoutId variable.

```
public void onSaveInstanceState(Bundle savedInstanceState) {
    savedInstanceState.putLong("workoutId", workoutId);
}
```

The onSaveInstanceState() method gets called before the fragment is destroyed.




```
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 void onCreate(Bundle savedInstanceState){
        super.onCreate(savedInstanceState);
        if (savedInstanceState != null) {
            workoutId = savedInstanceState.getLong("workoutId");
        }
    }

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

Add the onCreate() method.

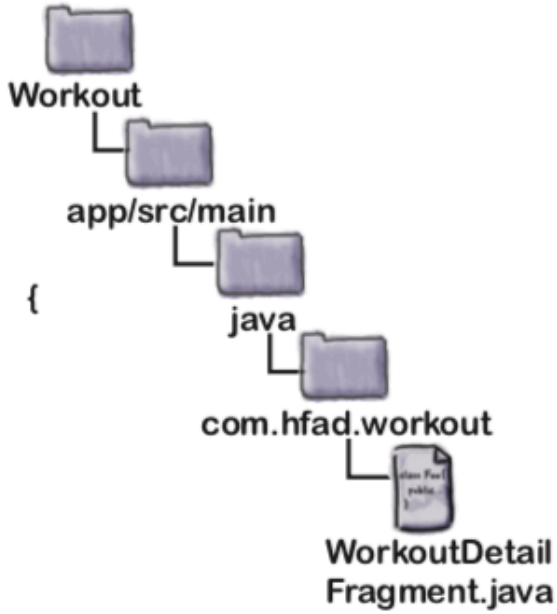
Set the value of the workoutId.

El código actualizado de WorkoutDetailFragment

El código actualizado de WorkoutDetailFragment

```
    @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());
        }
    }

    @Override
    public void onSaveInstanceState(Bundle savedInstanceState) {
        savedInstanceState.putLong("workoutId", workoutId);
    }

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

Save the value of the workoutId in the savedInstanceState Bundle before the fragment gets destroyed. We're retrieving it in the onCreate() method.



Prueba


10:47
Workout
The Limb Loosener
Core Agony
The Wimp Special
Strength and Length
The Wimp Special
5 Pull-ups
10 Push-ups
15 Squats
When you click on one of the workouts, its details continue to be displayed when you rotate the device.
10:47
Workout
The Limb Loosener
Core Agony
The Wimp Special
Strength and Length
The Wimp Special
5 Pull-ups
10 Push-ups
15 Squats

BULLET POINTS

- Make apps look different on different devices by putting separate layouts in device-appropriate folders.
- Android keeps track of places you've visited within an app by adding them to the back stack as separate transactions. Pressing the Back button pops the last transaction off the back stack.
- Use a frame layout to add, replace, or remove fragments programmatically using fragment transactions.
- Begin the transaction by calling the `FragmentManager beginTransaction()` method. This creates a `FragmentTransaction` object.
- Add, replace, and delete fragments using the `FragmentTransaction add()`, `replace()`, and `remove()` methods.
- Add a transaction to the back stack using the `FragmentTransaction addToBackStack()` method.
- Commit a transaction using the `FragmentTransaction commit()` method. This applies all the updates in the transaction.
- Save the state of a fragment's variables in the `Fragment onSaveInstanceState()` method.
- Restore the state of a fragment's variables in the `Fragment onCreate()` method.