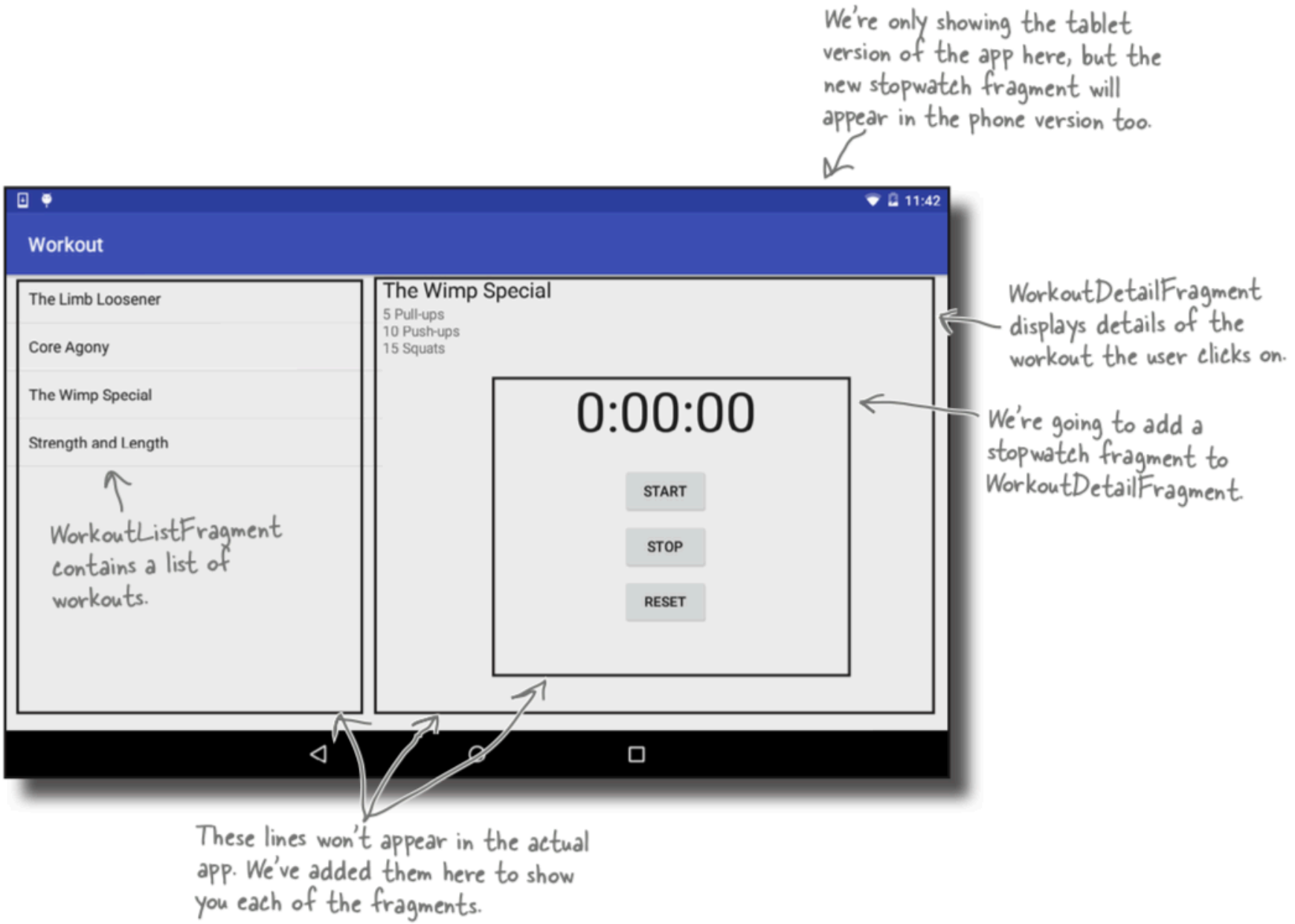
# 13
# Fragmentos anidados

# Creando fragmentos anidados



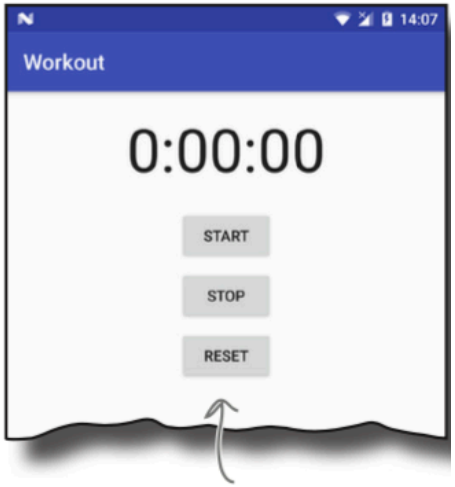
# Qué vamos a hacer ?

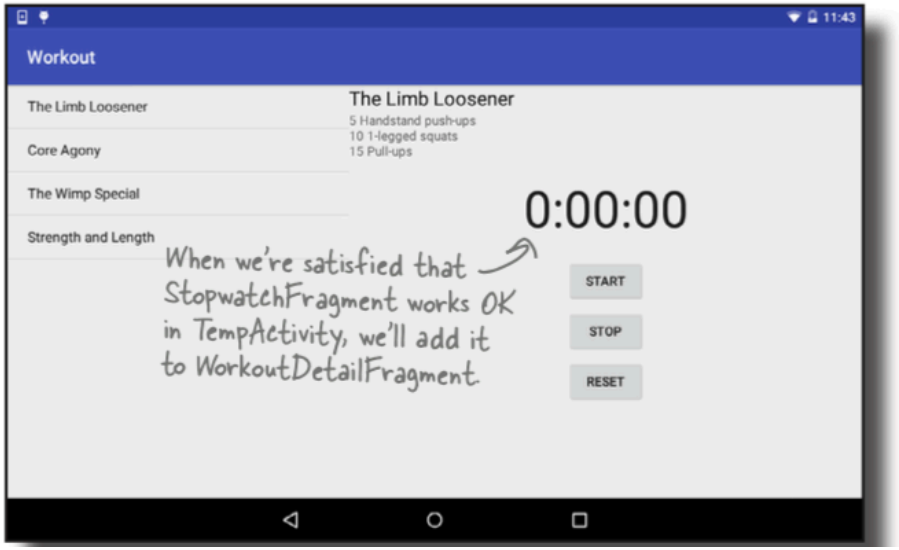
1. **Convert StopwatchActivity into StopwatchFragment.**
We'll take the `StopwatchActivity` code we created in Chapter 4, and change it into fragment code. We'll also display it in a new temporary activity called `TempActivity` so that we can check that it works. We'll temporarily change the app so that `TempActivity` starts when the app gets launched.

2. **Test StopwatchFragment.**
The `StopwatchActivity` included Start, Stop, and Reset buttons.We need to check that these still work when the stopwatch code is in a fragment.

   We also need to test what happens to `StopwatchFragment` when the user rotates the device.

3. **Add StopwatchFragment to WorkoutDetailFragment.**
Once we're satisfied that `StopwatchFragment` works, we'll add it to `WorkoutDetailFragment`.

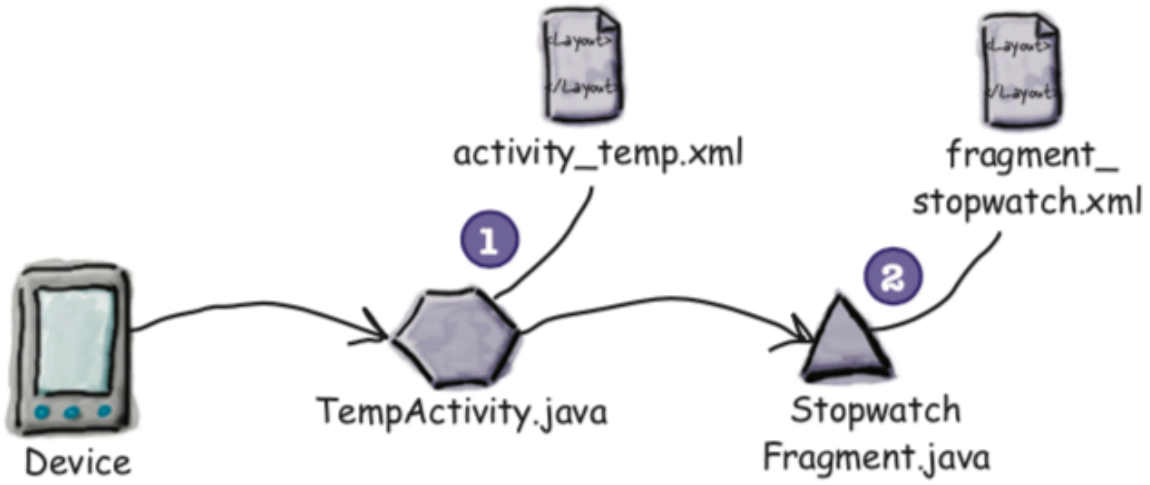


# La nueva versión de la aplicación

1. **When the app gets launched, it starts TempActivity.**
   `TempActivity` uses *activity_temp.xml* for its layout, and contains a fragment, `StopwatchFragment`.

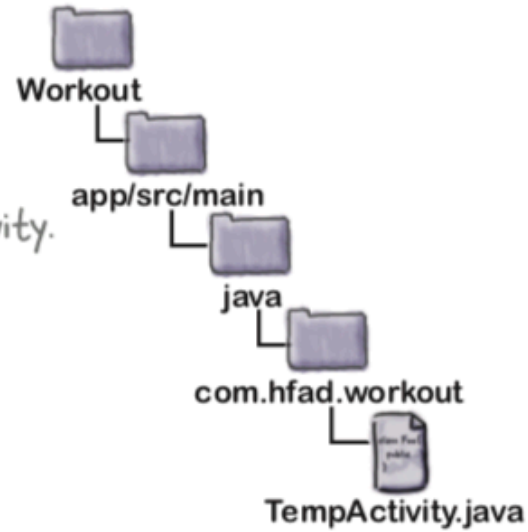
2. **StopwatchFragment displays a stopwatch with Start, Stop, and Reset buttons.**



# Agregando una clase temporal

```
package com.hfad.workout;

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

public class TempActivity extends AppCompatActivity {

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

The activity extends AppCompatActivity.



# Agregando una clase temporal

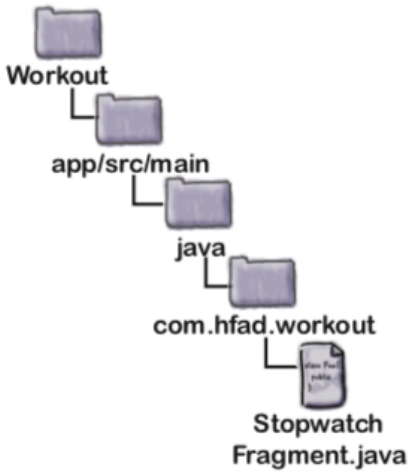
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.hfad.workout">

    <application
        ...
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name=".DetailActivity" />
        <activity android:name=".TempActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```


This bit specifies that it's the main activity of the app.

This says the activity can be used to launch the app.

# Agregando un fragmento basado en StopWatch

```
package com.hfad.workout;

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

public class StopwatchFragment extends Fragment {
    //Number of seconds displayed on the stopwatch.
    private int seconds = 0;
    //Is the stopwatch running?
    private boolean running;
    private boolean wasRunning;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        if (savedInstanceState != null) {
            seconds = savedInstanceState.getInt("seconds");
            running = savedInstanceState.getBoolean("running");
            wasRunning = savedInstanceState.getBoolean("wasRunning");
        }
    }
```




The number of seconds that have passed

running says whether the stopwatch is running.
wasRunning says whether the stopwatch was running before the stopwatch was paused.

Restore the state of the variables from the savedInstanceState Bundle.

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

Set the fragment's layout and start the runTimer() method, passing in the layout.

```
@Override
public void onPause() {
    super.onPause();
    wasRunning = running;
    running = false;
}
```

If the fragment's paused, record whether the stopwatch was running and stop it.


```
@Override
public void onResume() {
    super.onResume();
    if (wasRunning) {
        running = true;
    }
}
```

If the stopwatch was running before it was paused, set it running again.

```
@Override
public void onSaveInstanceState(Bundle savedInstanceState) {
    savedInstanceState.putInt("seconds", seconds);
    savedInstanceState.putBoolean("running", running);
    savedInstanceState.putBoolean("wasRunning", wasRunning);
}
```

Put the values of the variables in the Bundle before the activity is destroyed. These are used when the user turns the device.

```
public void onClickStart(View view) {
    running = true;
}
```

This code needs to run when the user clicks on the Start button.

# Agregando un fragmento basado en StopWatch

```
    public void onClickStop(View view) {
        running = false;
    }

    public void onClickReset(View view) {
        running = false;
        seconds = 0;
    }

    private void runTimer(View view) {
        final TextView timeView = (TextView) view.findViewById(R.id.time_view);
        final Handler handler = new Handler();
        handler.post(new Runnable() {
            @Override
            public void run() {
                int hours = seconds/3600;
                int minutes = (seconds%3600)/60;
                int secs = seconds%60;
                String time = String.format(Locale.getDefault(),
                        "%d:%02d:%02d", hours, minutes, secs);
                timeView.setText(time);
                if (running) {
                    seconds++;
                }
                handler.postDelayed(this, 1000);
            }
        });
    }
}
```

This code needs to run when the user clicks on the Stop button.

This code needs to run when the user clicks on the Reset button.

Putting the code in a Handler means it can run in the background thread.

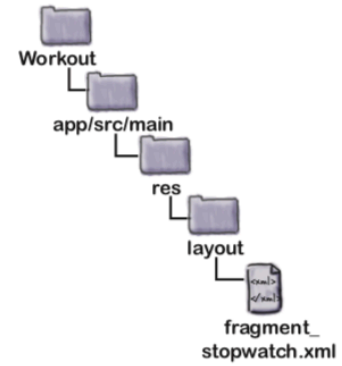
Display the number of seconds that have passed in the stopwatch.

If the stopwatch is running, increment the number of seconds.

Run the Handler code every second.


# Agregando un fragmento basado en StopWatch

```
<?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:orientation="vertical"
    android:padding="16dp">

    <TextView
        android:id="@+id/time_view"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:textAppearance="@android:style/TextAppearance.Large"
        android:textSize="56sp" />

    <Button
        android:id="@+id/start_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:layout_marginTop="20dp"
        android:onClick="onClickStart"
        android:text="@string/start" />

    <Button
        android:id="@+id/stop_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:layout_marginTop="8dp"
        android:onClick="onClickStop"
        android:text="@string/stop" />

    <Button
        android:id="@+id/reset_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:layout_marginTop="8dp"
        android:onClick="onClickReset"
        android:text="@string/reset" />
</LinearLayout>
```

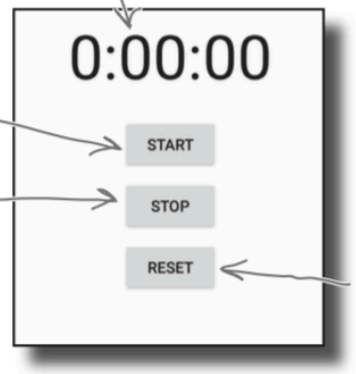



The number of hours, minutes, and seconds that have passed.

The Start button

The Stop button

The Reset button



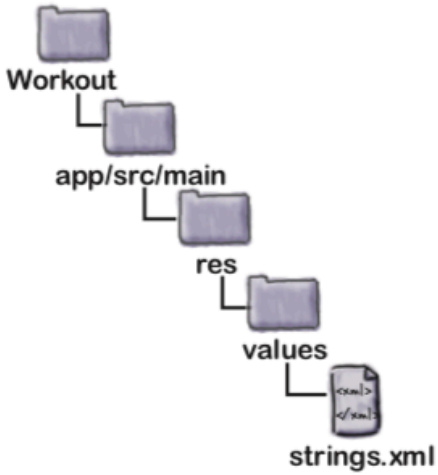

The Reset button code is on the next page.

# El layout del nuevo fragmento

# Cadenas en strings.xml

```
...
    <string name="start">Start</string>
    <string name="stop">Stop</string>
    <string name="reset">Reset</string>
...
```

These are the button labels.




The stopwatch looks the same as it did when it was an activity. But because it's now a fragment, we can reuse it in different places.

# Ciclo de vida de actividades y fragmentos

| Lifecycle Method | Activity? | Fragment? |
|---|---|---|
| onAttach() | | ✓ |
| onCreate() | ✓ | ✓ |
| onCreateView() | | ✓ |
| onActivityCreated() | | ✓ |
| onStart() | ✓ | ✓ |
| onPause() | ✓ | ✓ |
| onResume() | ✓ | ✓ |
| onStop() | ✓ | ✓ |
| onDestroyView() | | ✓ |
| onRestart() | ✓ | |
| onDestroy() | ✓ | ✓ |
| onDetach() | | ✓ |

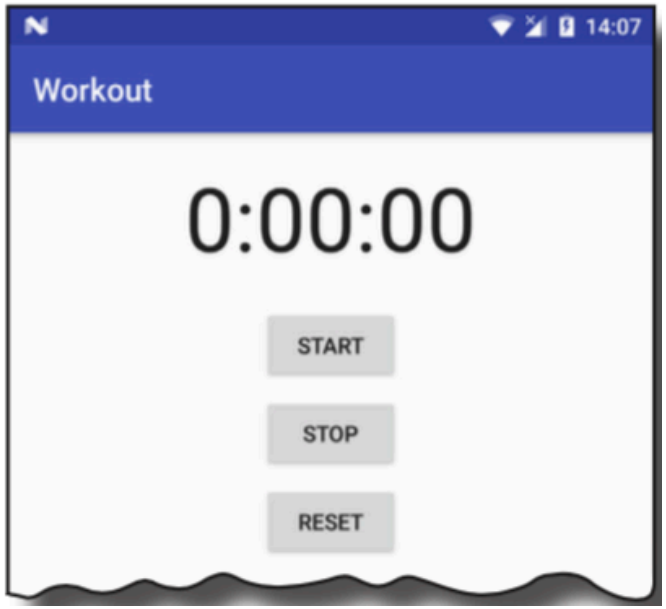
# Agregando StopwatchFragment en el layout de TempActivity

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

This adds the fragment to the activity.

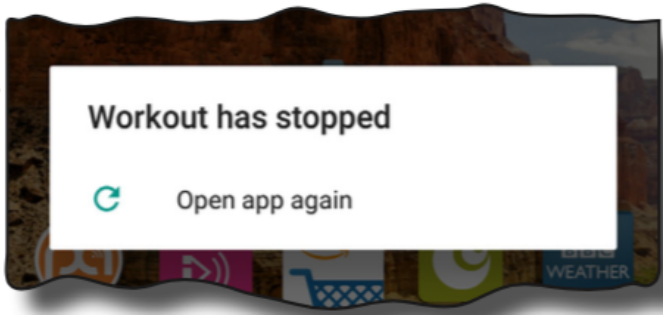
# Prueba

When we run the app, TempActivity starts, not MainActivity. TempActivity displays StopwatchFragment as expected.



# La aplicación falla al presionar los botones

This is what happened when we clicked on the Start button in StopwatchFragment.

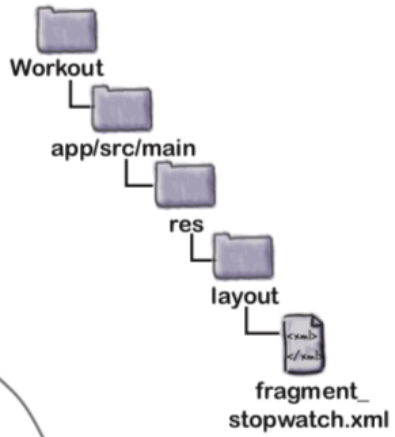



Yikes.

```
04-13 11:56:43.623 10583-10583/com.hfad.workout E/AndroidRuntime: FATAL EXCEPTION: main
    Process: com.hfad.workout, PID: 10583
    java.lang.IllegalStateException: Could not find method onClickStart(View) in a
    parent or ancestor Context for android:onClick attribute defined on view class
    android.support.v7.widget.AppCompatButton with id 'start_button'
        at android.support.v7.app.AppCompatViewInflater$DeclaredOnClickListener.
            resolveMethod(AppCompatViewInflater.java:327)
        at android.support.v7.app.AppCompatViewInflater$DeclaredOnClickListener.
            onClick(AppCompatViewInflater.java:284)
        at android.view.View.performClick(View.java:5609)
        at android.view.View$PerformClick.run(View.java:22262)
        at android.os.Handler.handleCallback(Handler.java:751)
        at android.os.Handler.dispatchMessage(Handler.java:95)
        at android.os.Looper.loop(Looper.java:154)
        at android.app.ActivityThread.main(ActivityThread.java:6077)
        at java.lang.reflect.Method.invoke(Native Method)
        at com.android.internal.os.ZygoteInit$MethodAndArgsCaller.
            run(ZygoteInit.java:865)
        at com.android.internal.os.ZygoteInit.main(ZygoteInit.java:755)
```

# Actualizando fragment_stopwatch.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    ...
    <Button
        android:id="@+id/start_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:layout_marginTop="20dp"
        android:onClick="onClickStart"
        android:text="@string/start" />

    <Button
        android:id="@+id/stop_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:layout_marginTop="8dp"
        android:onClick="onClickStop"
        android:text="@string/stop" />

    <Button
        android:id="@+id/reset_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:layout_marginTop="8dp"
        android:onClick="onClickReset"
        android:text="@string/reset" />
</LinearLayout>
```




Remove the onClick attributes for each of the buttons in the stopwatch.

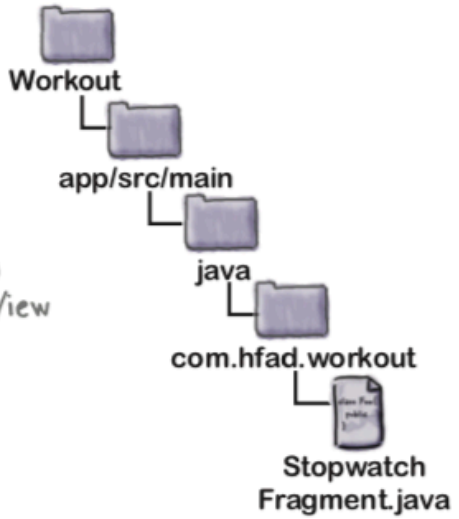
# Actualizando StopwatchFragment

`onClickStart()`, `onClickStop()`, and `onClickReset()` methods in *StopwatchFragment.java* to match ours:

```
...
    public private void onClickStart(View view) {
        running = true;
    }

    public private void onClickStop(View view) {
        running = false;
    }

    public private void onClickReset(View view) {
        running = false;
        seconds = 0;
    }
...
```

Change the methods to private.

Remove the View parameters.



# Actualizando StopwatchFragment

This turns the fragment into an OnClickListener.

```
public class StopwatchFragment extends Fragment implements View.OnClickListener {
    ...
}
```

```
@Override
public void onClick(View v) {
    switch (v.getId()) {
        case R.id.start_button:
            onClickStart();
            break;
        case R.id.stop_button:
            onClickStop();
            break;
        case R.id.reset_button:
            onClickReset();
            break;
    }
}
```

This is the View the user clicked on.

Check which View was clicked.

If the Start button was clicked, call the onClickStart() method.

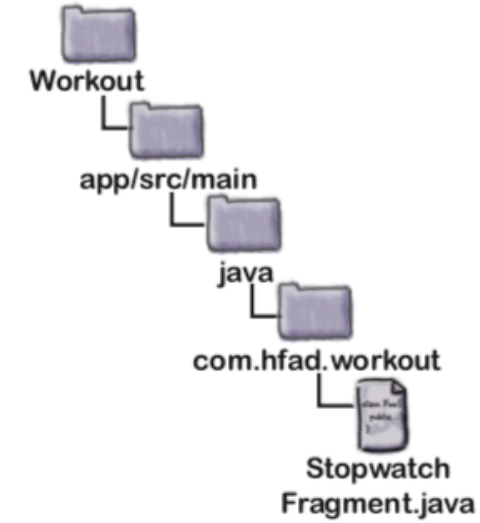
If the Stop button was clicked, call the onClickStop() method.

If the Reset button was clicked, call the onClickReset() method.

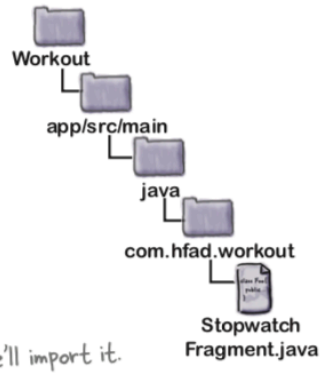
# Actualizando StopwatchFragment

```
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
                         Bundle savedInstanceState) {
    View layout = inflater.inflate(R.layout.fragment_stopwatch, container, false);
    runTimer(layout);
    Button startButton = (Button)layout.findViewById(R.id.start_button);
    startButton.setOnClickListener(this);
    Button stopButton = (Button)layout.findViewById(R.id.stop_button);
    stopButton.setOnClickListener(this);
    Button resetButton = (Button)layout.findViewById(R.id.reset_button);
    resetButton.setOnClickListener(this);
    return layout;
}
```

This attaches the listener to each of the buttons.



# El código completo de StopwatchFragment




```
package com.hfad.workout;

import java.util.Locale;
import android.os.Bundle;
import android.os.Handler;
import android.support.v4.app.Fragment;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;
import android.widget.Button;
```

We're using the Button class, so we'll import it.

```
public class StopwatchFragment extends Fragment implements View.OnClickListener {
    //Number of seconds displayed on the stopwatch.
    private int seconds = 0;
    //Is the stopwatch running?
    private boolean running;
    private boolean wasRunning;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        if (savedInstanceState != null) {
            seconds = savedInstanceState.getInt("seconds");
            running = savedInstanceState.getBoolean("running");
            wasRunning = savedInstanceState.getBoolean("wasRunning");
        }
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        View layout = inflater.inflate(R.layout.fragment_stopwatch, container, false);
        runTimer(layout);
        Button startButton = (Button)layout.findViewById(R.id.start_button);
        startButton.setOnClickListener(this);
        Button stopButton = (Button)layout.findViewById(R.id.stop_button);
        stopButton.setOnClickListener(this);
        Button resetButton = (Button)layout.findViewById(R.id.reset_button);
        resetButton.setOnClickListener(this);
        return layout;
    }
```

The fragment needs to implement the View.OnClickListener interface.

We're not changing the onCreate() method.

Update the onCreateView() method to attach the listener to the buttons.

# El código completo de StopwatchFragment

```
@Override
public void onClick(View v) {
    switch (v.getId()) {
        case R.id.start_button:
            onClickStart();
            break;
        case R.id.stop_button:
            onClickStop();
            break;
        case R.id.reset_button:
            onClickReset();
            break;
    }
}
```

As we're implementing the OnClickListener interface, we need to override the onClick() method.

Call the appropriate method in the fragment for the button that was clicked.

```
@Override
public void onPause() {
    super.onPause();
    wasRunning = running;
    running = false;
}

@Override
public void onResume() {
    super.onResume();
    if (wasRunning) {
        running = true;
    }
}

@Override
public void onSaveInstanceState(Bundle savedInstanceState) {
    savedInstanceState.putInt("seconds", seconds);
    savedInstanceState.putBoolean("running", running);
    savedInstanceState.putBoolean("wasRunning", wasRunning);
}
```

We've not changed these methods.

# El código completo de StopwatchFragment

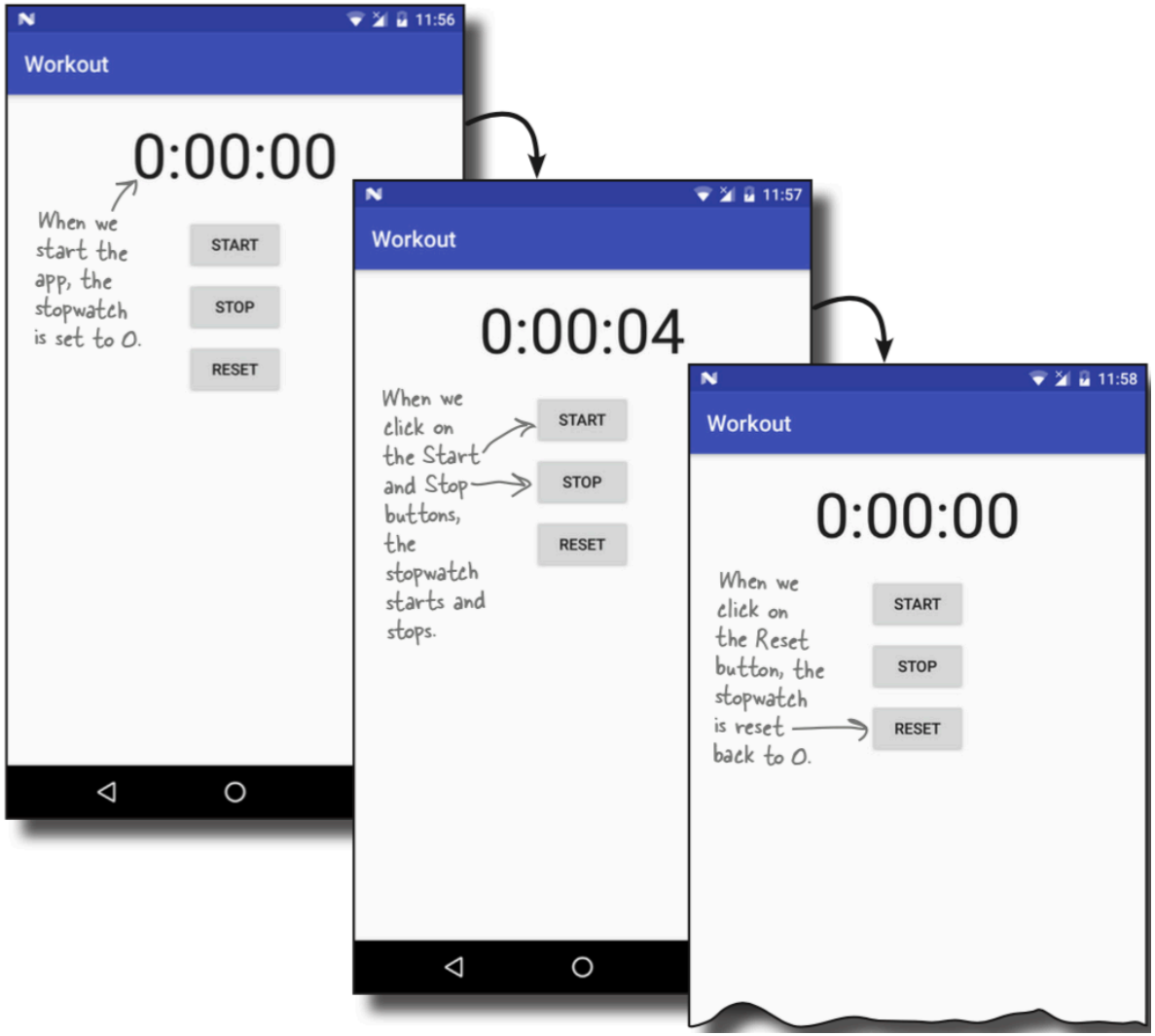
```
private void onClickStart() {
    running = true;
}

private void onClickStop() {
    running = false;
}

private void onClickReset() {
    running = false;
    seconds = 0;
}
```

We've changed these methods so they're private. We've also removed the View parameter, as we no longer needed it.

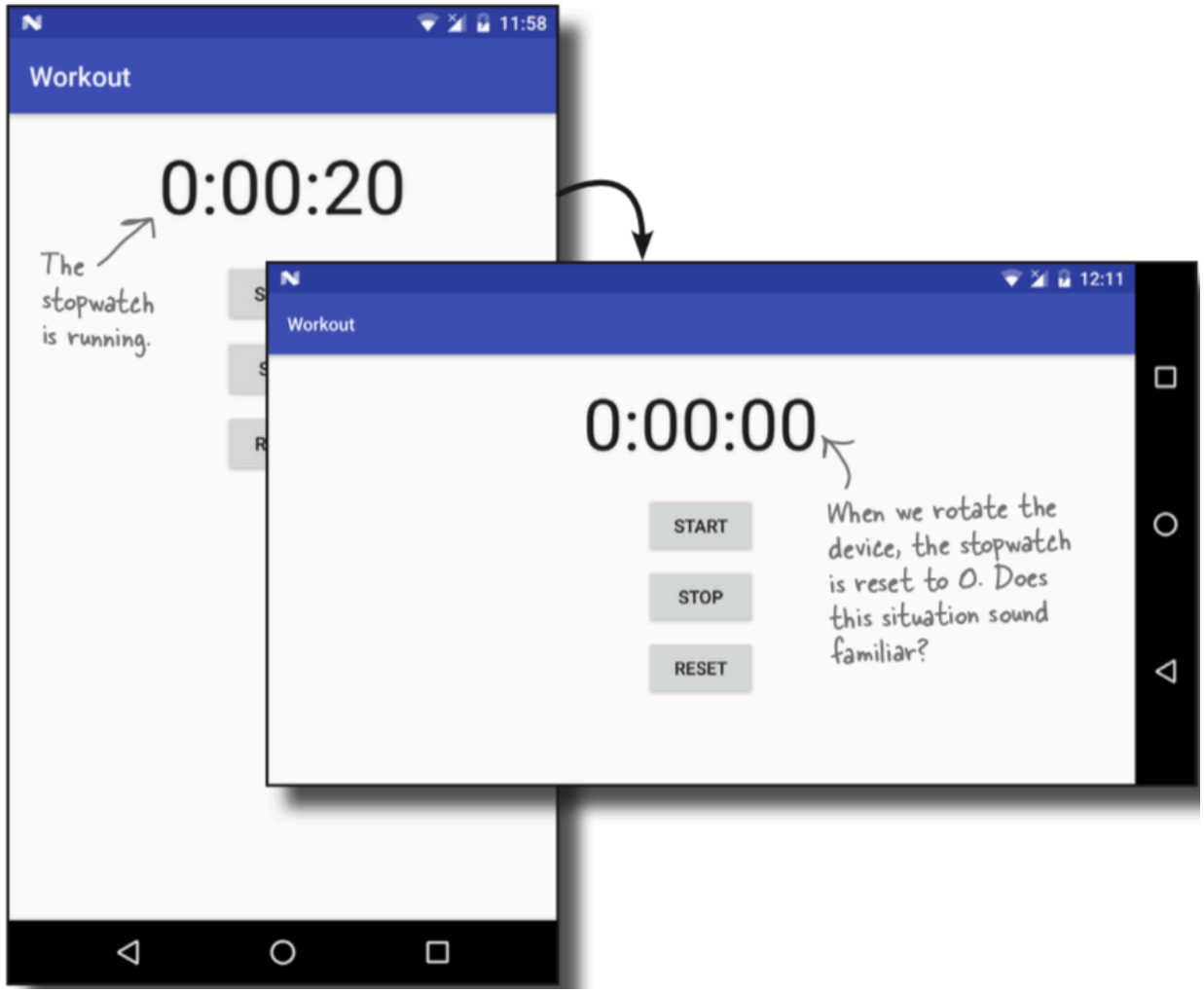

We've not changed this method.

```
private void runTimer(View view) {
    final TextView timeView = (TextView) view.findViewById(R.id.time_view);
    final Handler handler = new Handler();
    handler.post(new Runnable() {
        @Override
        public void run() {
            int hours = seconds/3600;
            int minutes = (seconds%3600)/60;
            int secs = seconds%60;
            String time = String.format(Locale.getDefault(),
                    "%d:%02d:%02d", hours, minutes, secs);
            timeView.setText(time);
            if (running) {
                seconds++;
            }
            handler.postDelayed(this, 1000);
        }
    });
}
}
```

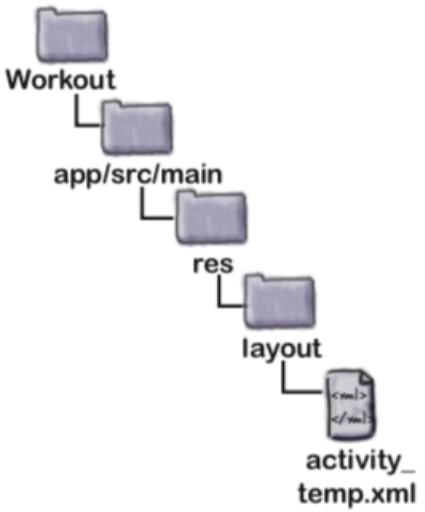
# Prueba


11:56
Workout
0:00:00
When we start the app, the stopwatch is set to 0.
START
STOP
RESET
11:57
Workout
0:00:04
When we click on the Start and Stop buttons, the stopwatch starts and stops.
START
STOP
RESET
11:58
Workout
0:00:00
When we click on the Reset button, the stopwatch is reset back to 0.
START
STOP
RESET

# Al girar el dispositivo se reinicia el cronómetro



# Cambiando Fragment por FrameLayout




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

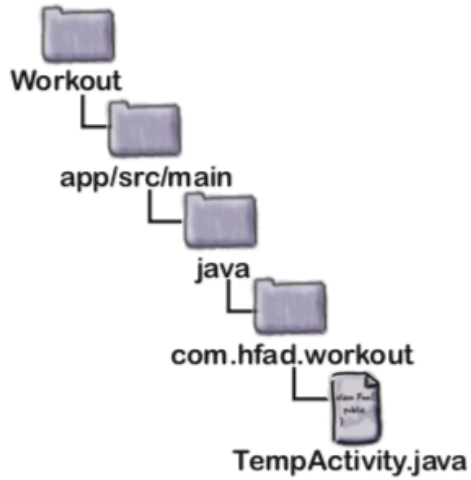
Replace the fragment with a FrameLayout.

Delete this line.

# Agregando transacciones a TempActivity

```
package com.hfad.workout;

import android.support.v4.app.FragmentTransaction;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

public class TempActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_temp);
        if (savedInstanceState == null) {
            StopwatchFragment stopwatch = new StopwatchFragment();
            FragmentTransaction ft = getSupportFragmentManager().beginTransaction();
            ft.add(R.id.stopwatch_container, stopwatch);
            ft.addToBackStack(null);
            ft.setTransition(FragmentTransaction.TRANSIT_FRAGMENT_FADE);
            ft.commit();
        }
    }
}
```

You need to import the FragmentTransaction class from the Support Library.




We only want to add the fragment if the activity isn't being recreated after having been destroyed.

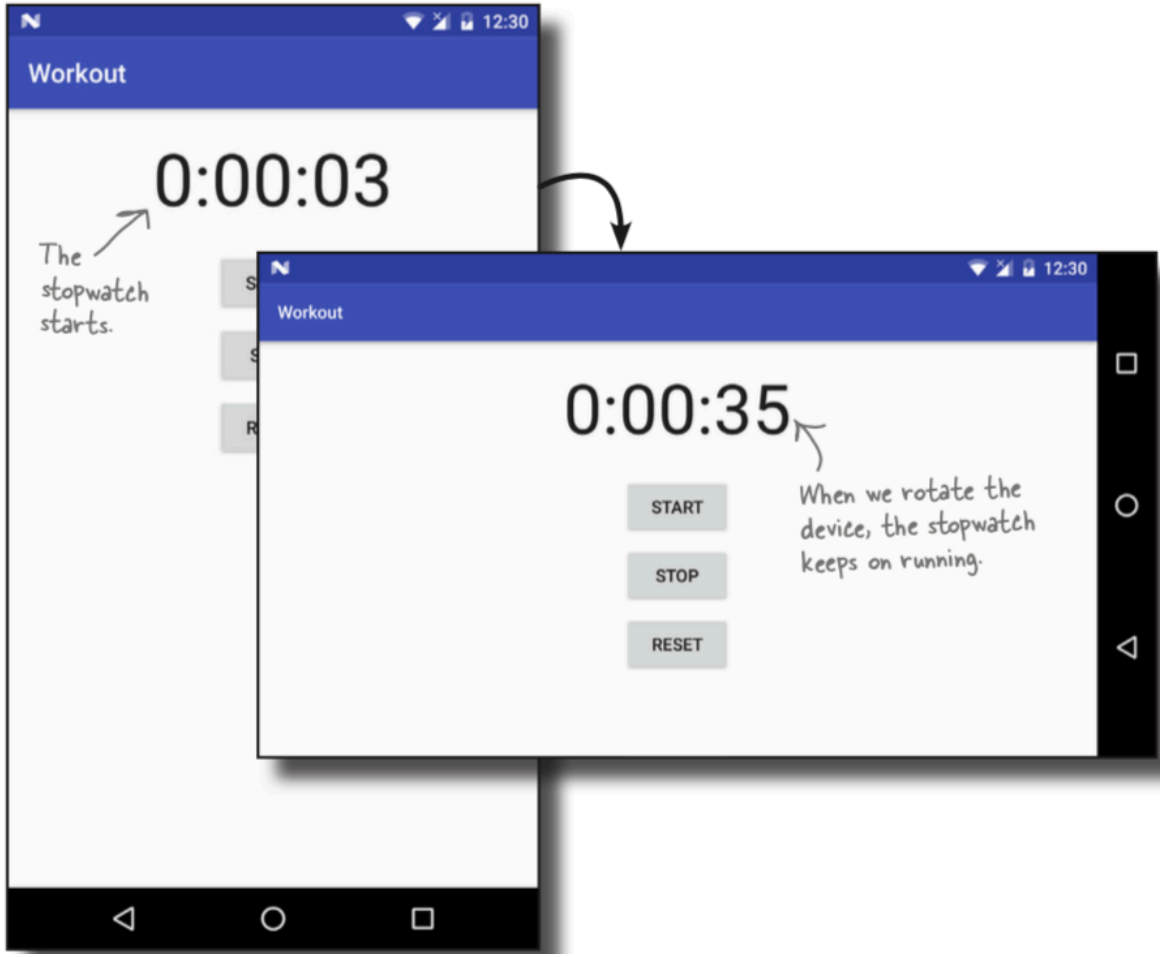
Begin the fragment transaction.

Add the stopwatch, and add the transaction to the back stack.

Set the fragment transition to fade in and out.

Commit the transaction. This applies the changes.

# Prueba


12:30
Workout
0:00:03
The stopwatch starts.
12:30
Workout
0:00:35
When we rotate the device, the stopwatch keeps on running.
START
STOP
RESET

# Agregando cronómetro a WorkoutDetailFragment

# Agregando cronómetro a WorkoutDetailFragment

1. **When the app gets launched, it starts MainActivity.**
   `MainActivity` includes `WorkoutListFragment`, which displays a list of workouts.
2. **The user clicks on a workout and WorkoutDetailFragment is displayed.**
   `WorkoutDetailFragment` displays details of the workout, and contains `StopwatchFragment`.
3. **StopwatchFragment displays a stopwatch.**

We've simplified the app structure here, but these are the key points.

# Reiniciando MainActivity

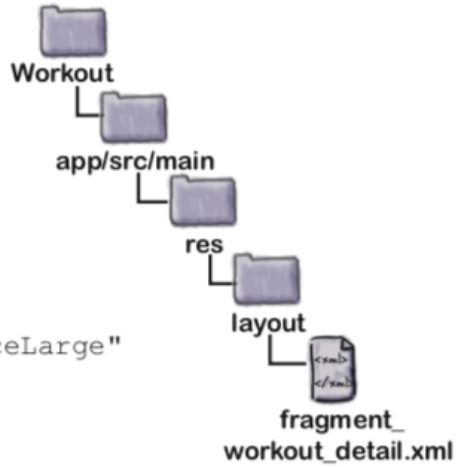

```
...
    <application
        ...
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name=".DetailActivity" />
        <activity android:name=".TempActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
...
```

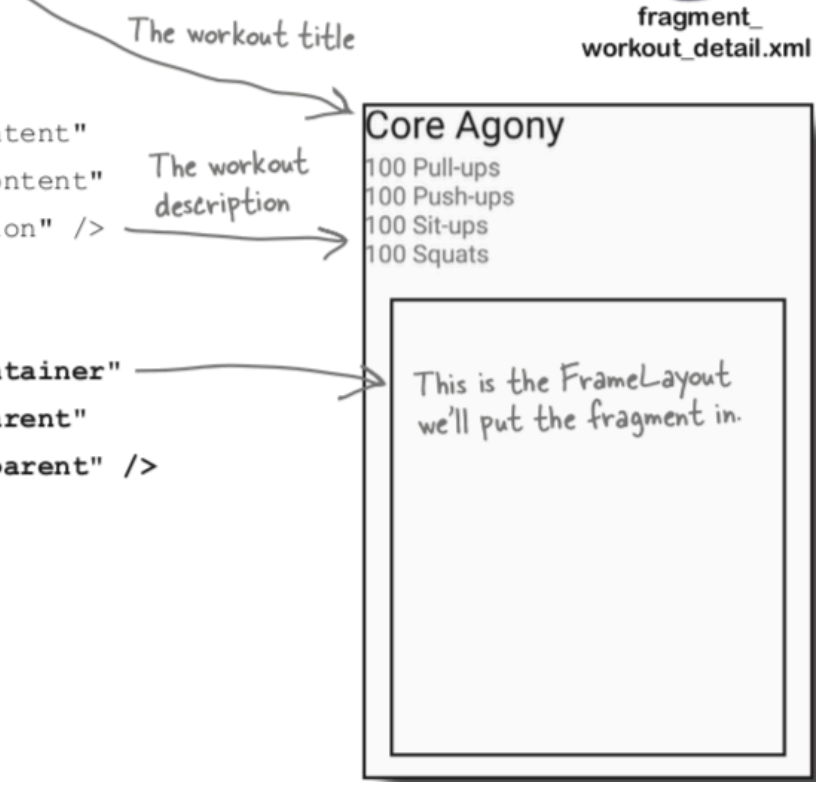
Add an intent filter to start MainActivity when the app is launched.

Remove the intent filter from TempActivity.

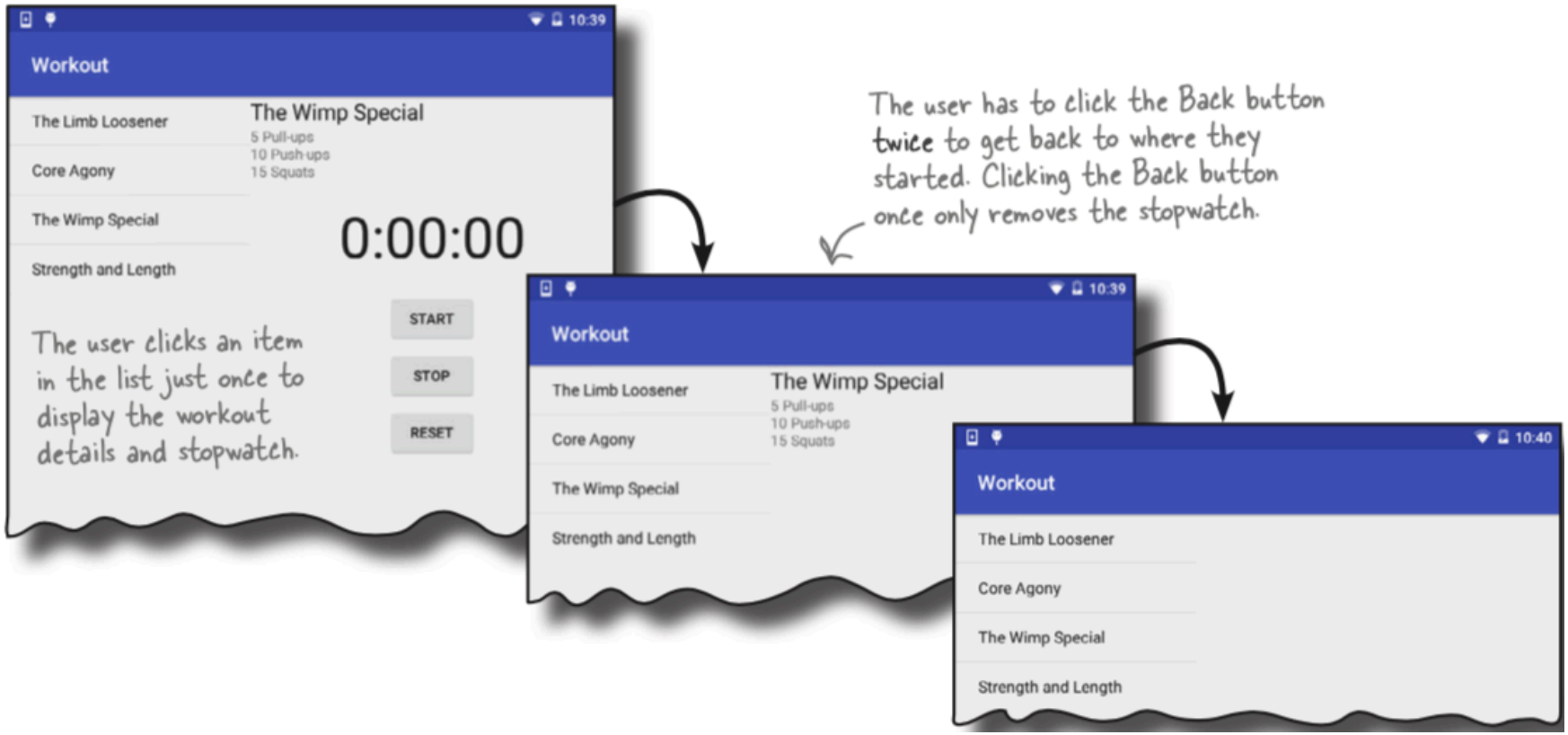
# Agregando un FrameLayout

```
<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:id="@+id/textTitle" />

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

    <FrameLayout
        android:id="@+id/stopwatch_container"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />
</LinearLayout>
```



# La aplicación no funciona correctamente



# La aplicación no funciona correctamente

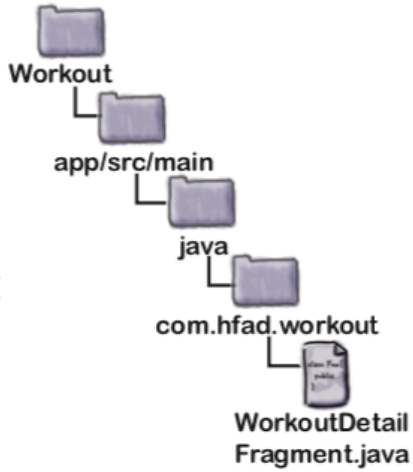

```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    if (savedInstanceState == null) {
        StopwatchFragment stopwatch = new StopwatchFragment();
        FragmentTransaction ft = getChildFragmentManager().beginTransaction();
        ft.add(R.id.stopwatch_container, stopwatch);
        ft.addToBackStack(null);
        ft.setTransition(FragmentTransaction.TRANSIT_FRAGMENT_FADE);
        ft.commit();
    } else {
        workoutId = savedInstanceState.getLong("workoutId");
    }
}
```

We're using getChildFragmentManager() instead of getSupportFragmentManager(). Apart from that, the code is the same as we had earlier.

# El código completo de WorkoutDetailFragment

```
package com.hfad.workout;

import android.support.v4.app.Fragment;
import android.support.v4.app.FragmentTransaction;
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) {
        if (savedInstanceState == null) {
            StopwatchFragment stopwatch = new StopwatchFragment();
            FragmentTransaction ft = getChildFragmentManager().beginTransaction();
            ft.add(R.id.stopwatch_container, stopwatch);
            ft.addToBackStack(null);
            ft.setTransition(FragmentTransaction.TRANSIT_FRAGMENT_FADE);
            ft.commit();
        } else {
            workoutId = savedInstanceState.getLong("workoutId");
        }
    }
```

You need to import the FragmentTransaction class from the Support Library.




Delete this line.

We only want to add the fragment if the activity isn't being recreated after having been destroyed.

Begin the fragment transaction.

Add the stopwatch, and add the transaction to the back stack.

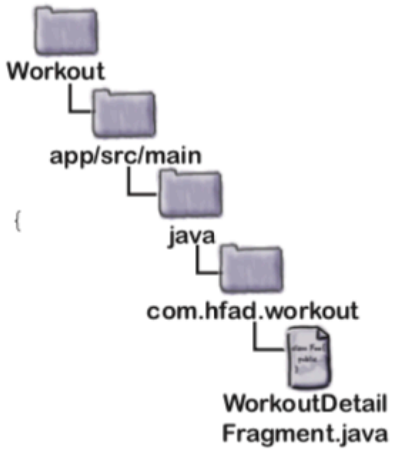
Commit the transaction.

Set the fragment transition to fade in and out.

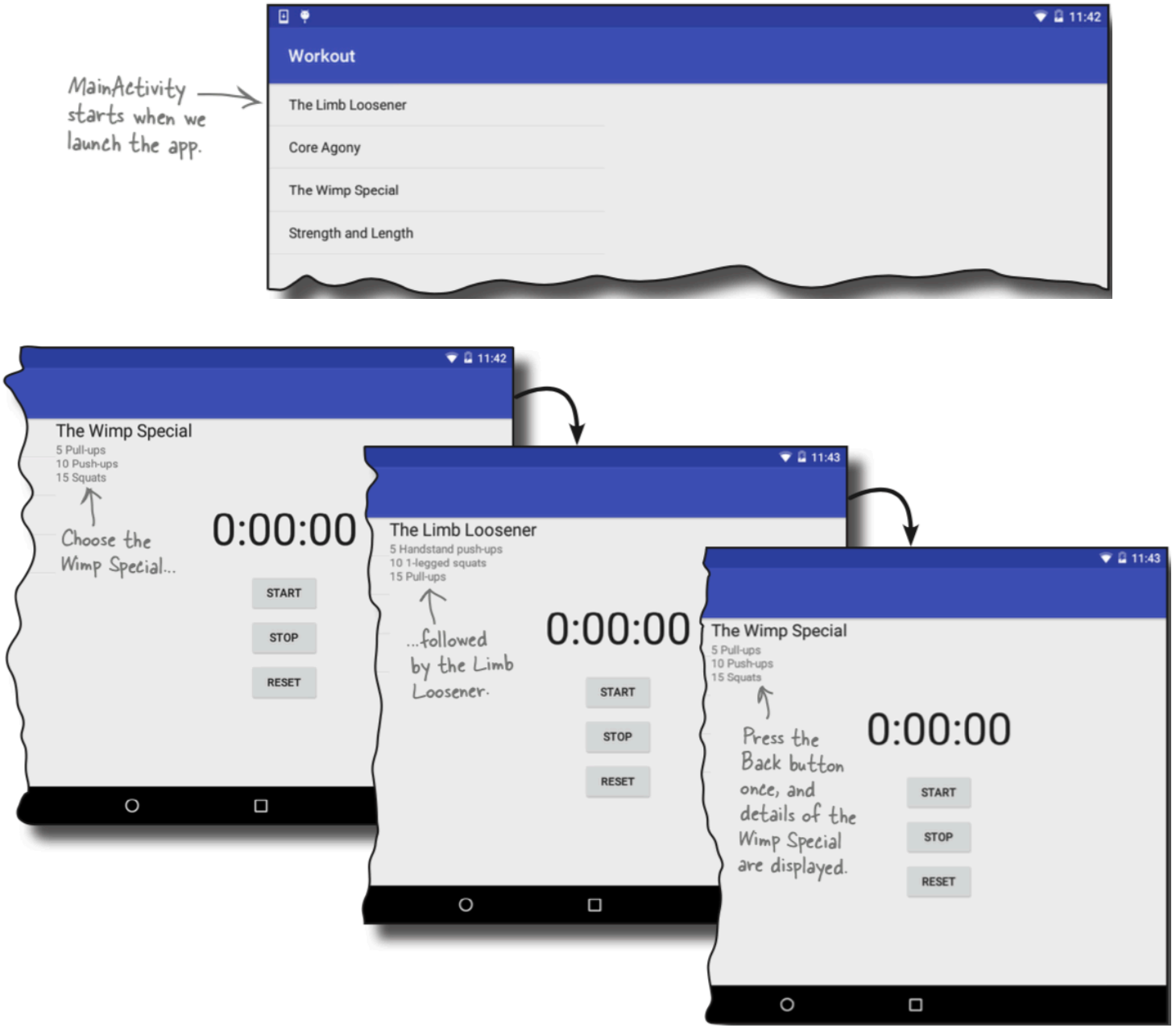
# El código completo de WorkoutDetailFragment

```
    @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());
        }
    }

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

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

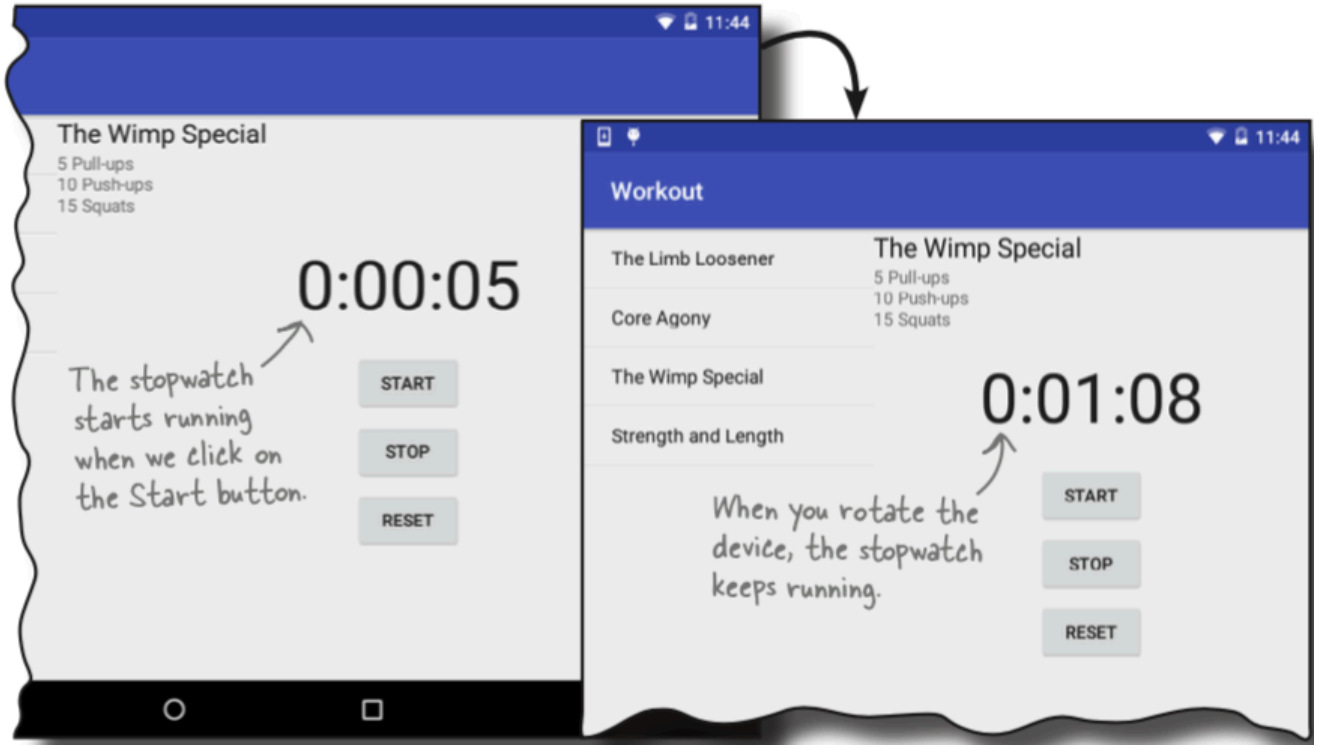
We didn't change any of the methods on this page.



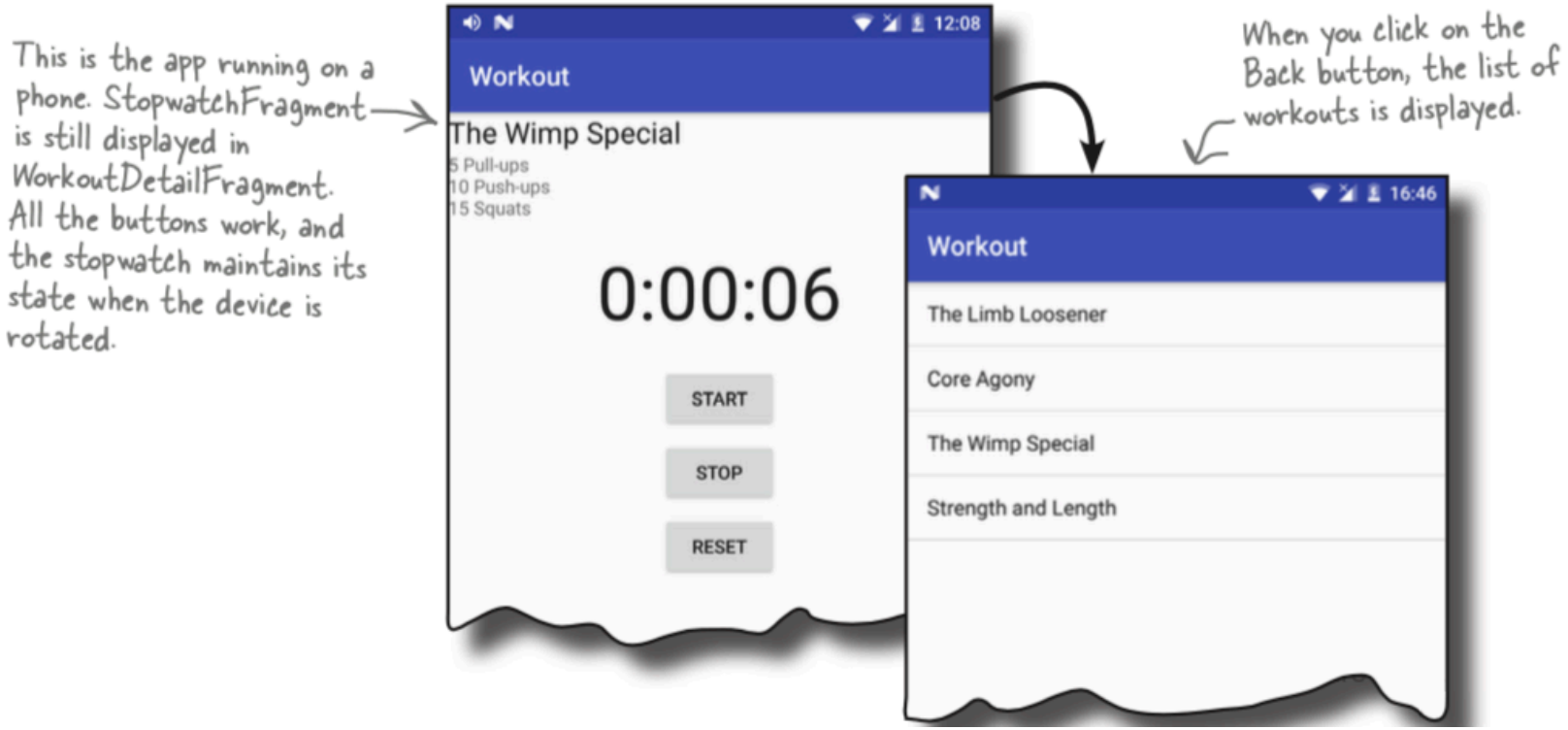
# Prueba en tableta



# Prueba en tableta



# Prueba en teléfono



## BULLET POINTS

- Fragments can contain other fragments.
- If you use the `android:onClick` attribute in a fragment, Android will look for a method of that name in the fragment's parent activity.
- Instead of using the `android:onClick` attribute in a fragment, make the fragment implement the `View.OnClickListener` interface and implement its `onClick()` method.
- If you use the `<fragment>` element in your layout, the fragment gets recreated when you rotate the device. If your fragment is dynamic, use a fragment transaction instead.
- Fragments contain two methods for getting a fragment manager, `getFragmentManager()` and `getChildFragmentManager()`.
- `getFragmentManager()` gets a reference to the fragment manager associated with the fragment's parent activity. Any fragment transactions you create using this fragment manager are added to the back stack as extra transactions.
- `getChildFragmentManager()` gets a reference to the fragment manager associated with the fragment's parent fragment. Any fragment transactions you create using this fragment manager are nested inside the parent fragment transaction.