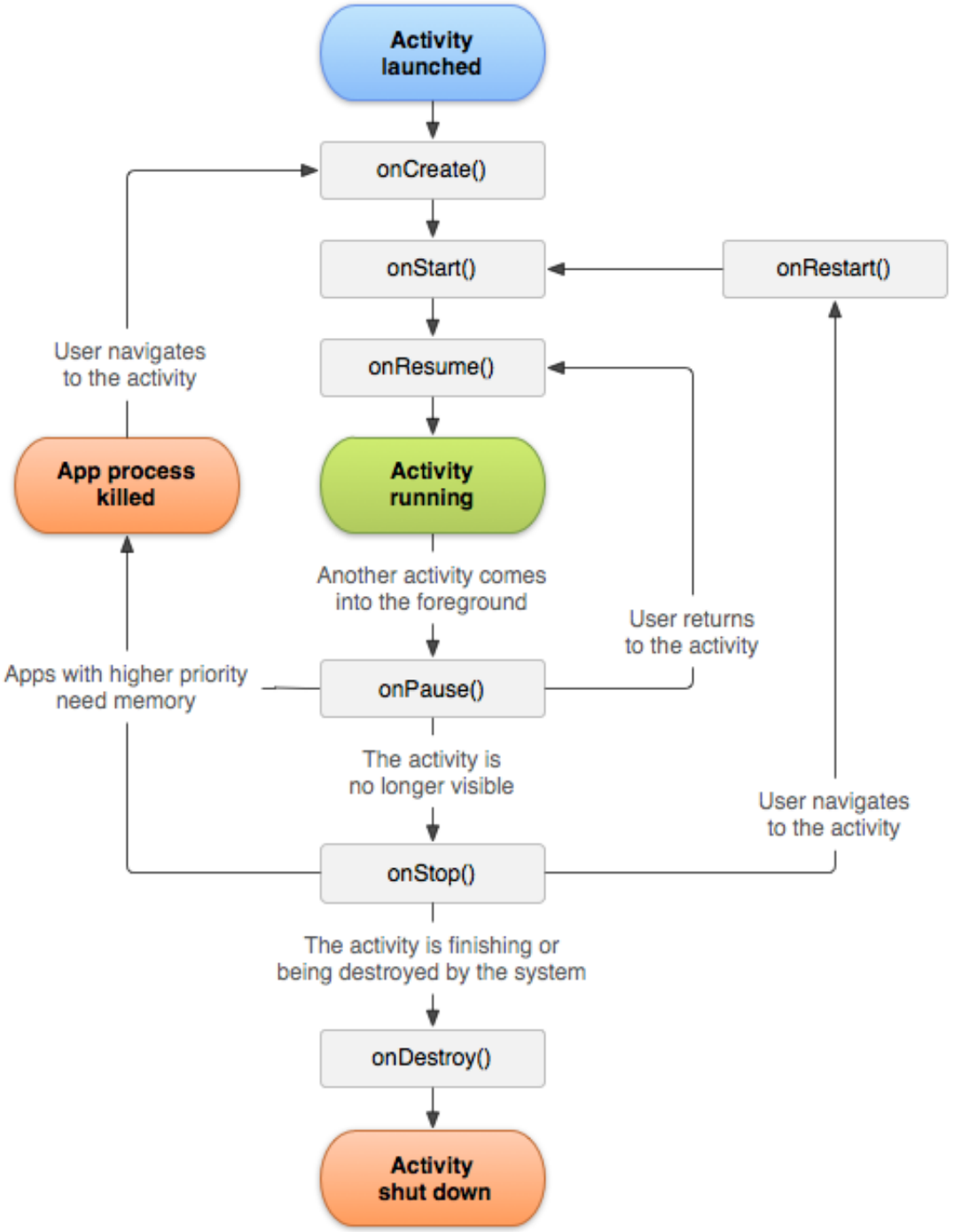
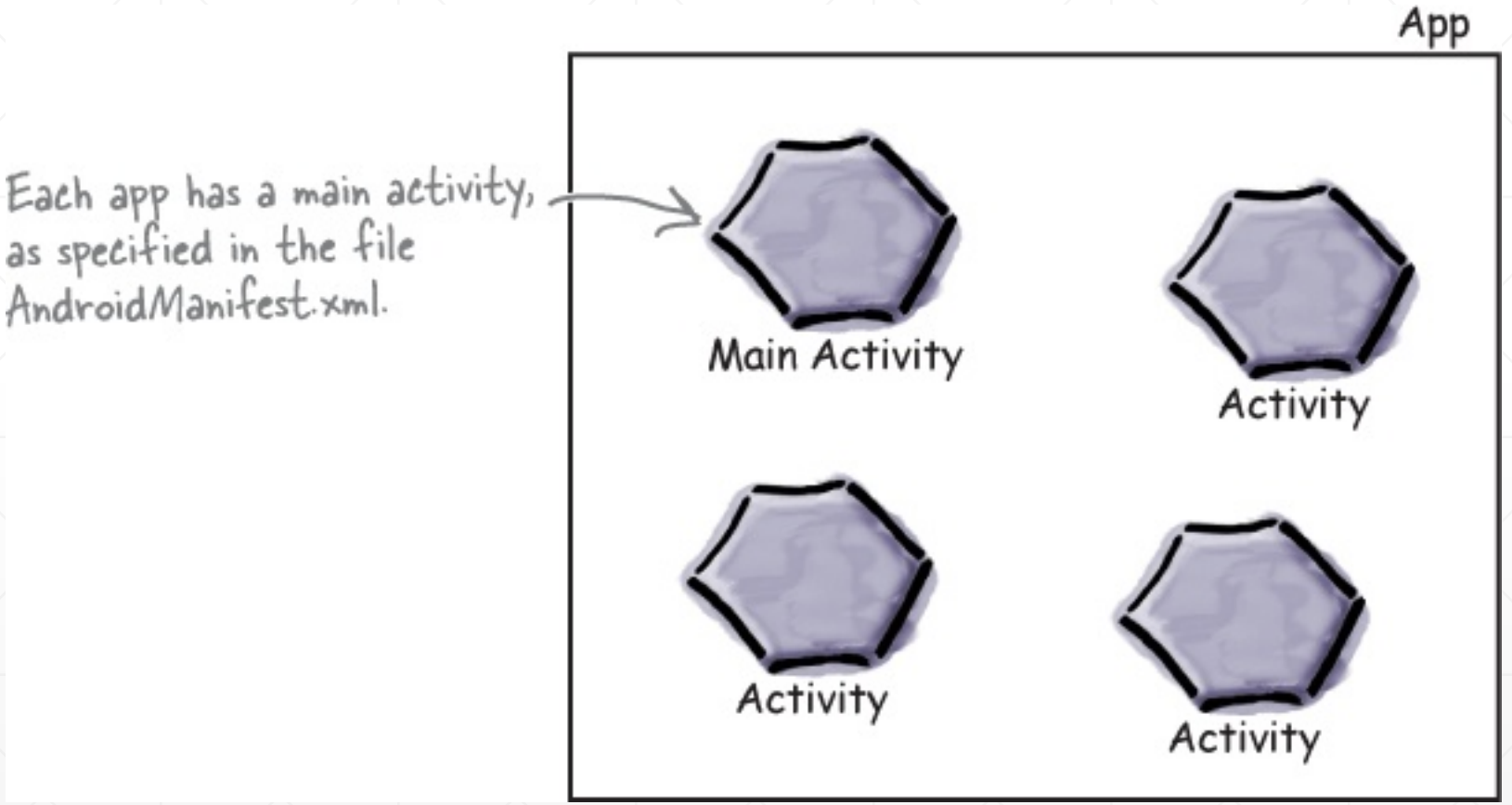
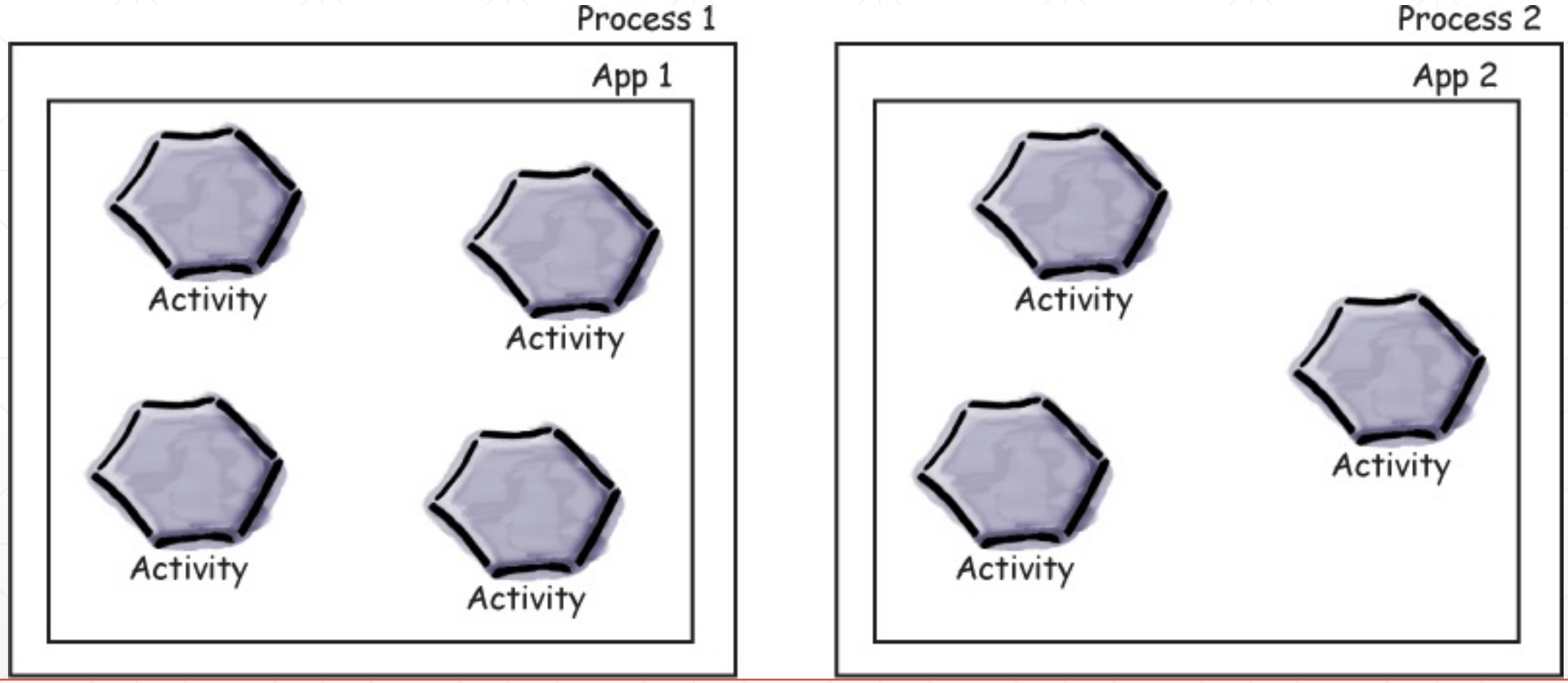
Chap 4 – Activity Lifecycle



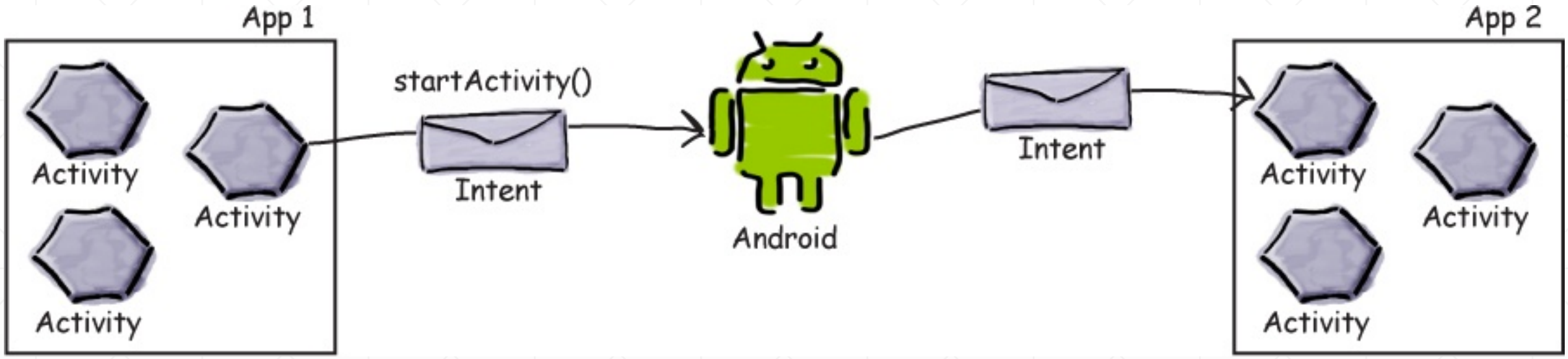
Chap 4 – Activity Lifecycle



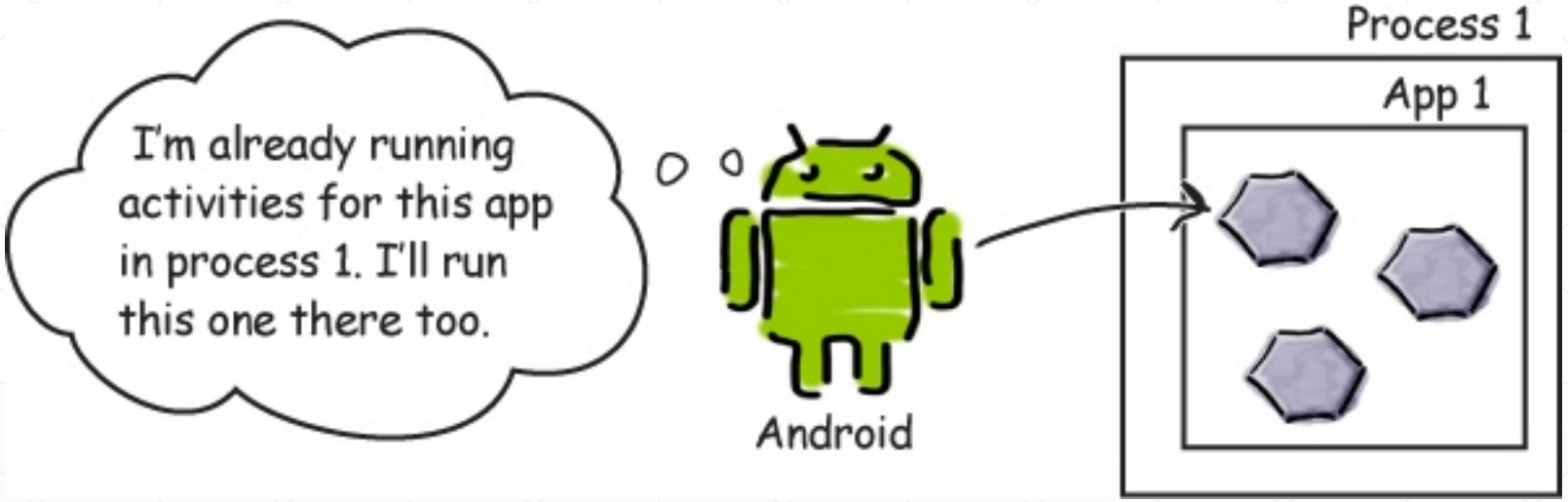
Chap 4 – Activity Lifecycle



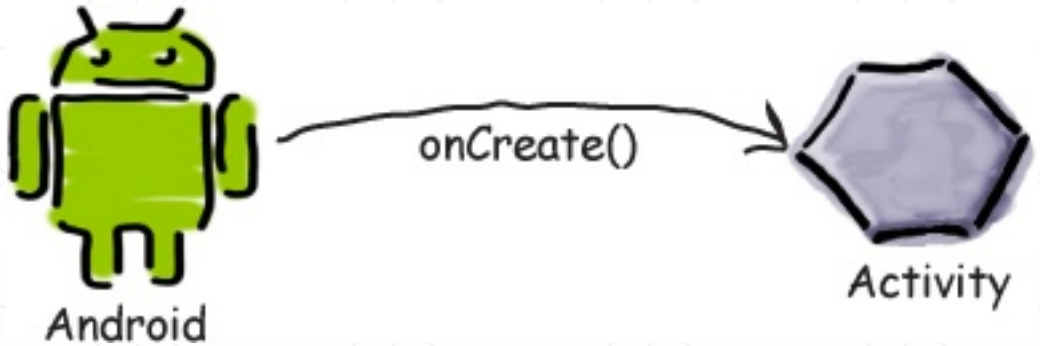
Chap 4 – Activity Lifecycle



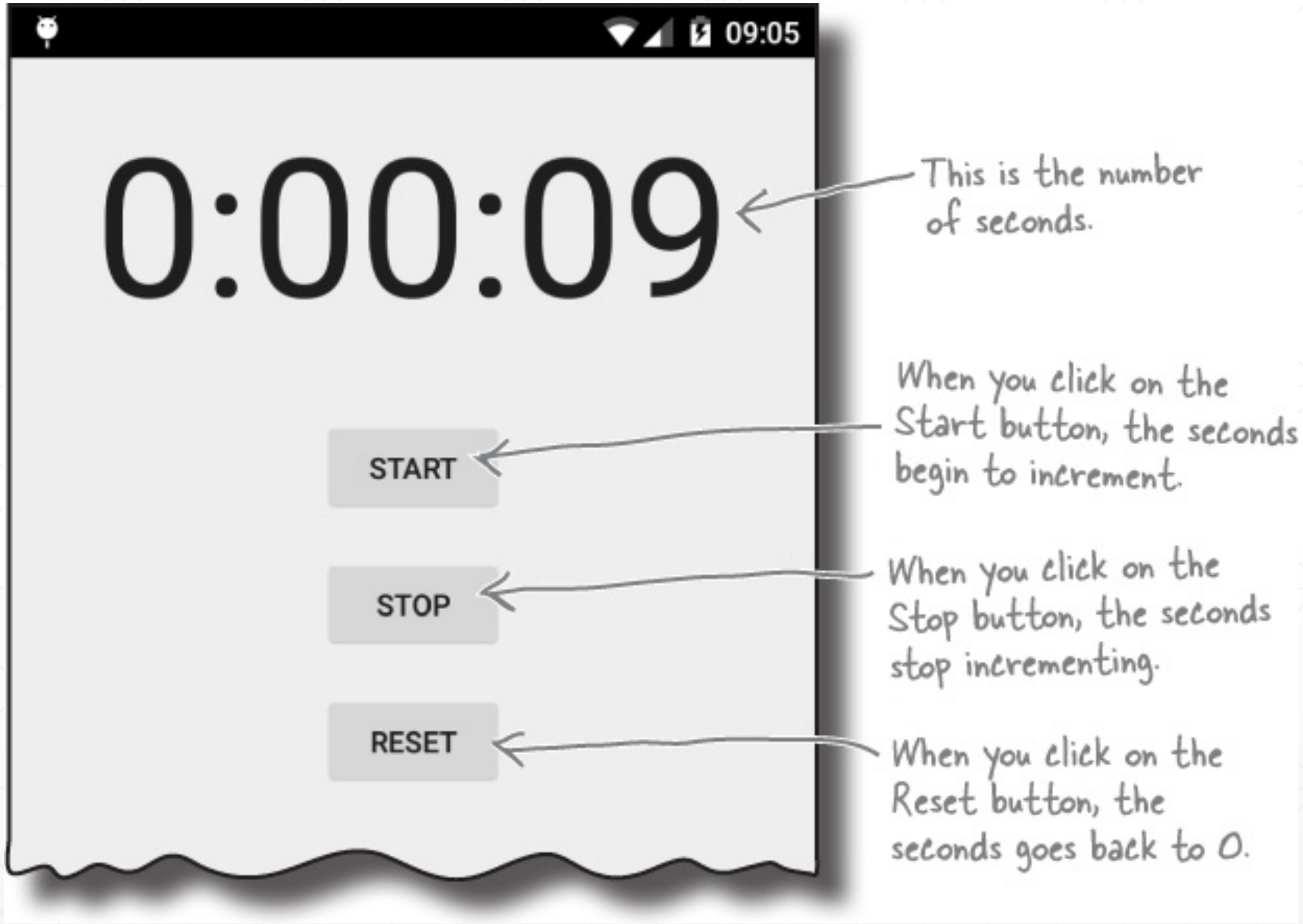
Chap 4 – Activity Lifecycle



Chap 4 – Activity Lifecycle



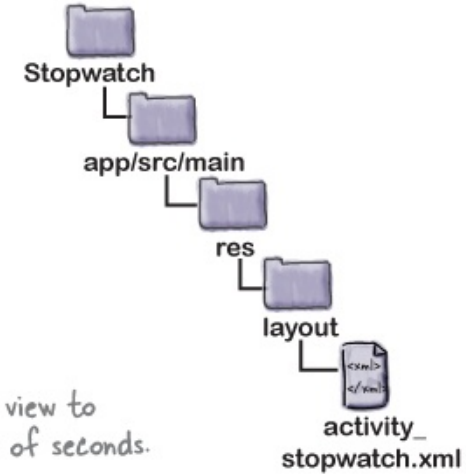
Chap 4 – Activity Lifecycle



Chap 4 – Activity Lifecycle - Layout

Chap 4 – Activity Lifecycle

```
<RelativeLayout 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:paddingBottom="16dp"
    android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:paddingTop="16dp"
    tools:context=".StopwatchActivity" >

    <TextView
        android:id="@+id/time_view"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="0dp"
        android:text=""
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:textSize="92sp" />

    <Button
        android:id="@+id/start_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/time_view"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="20dp"
        android:onClick="onClickStart"
        android:text="@string/start" />
```



We'll use this text view to display the number of seconds.

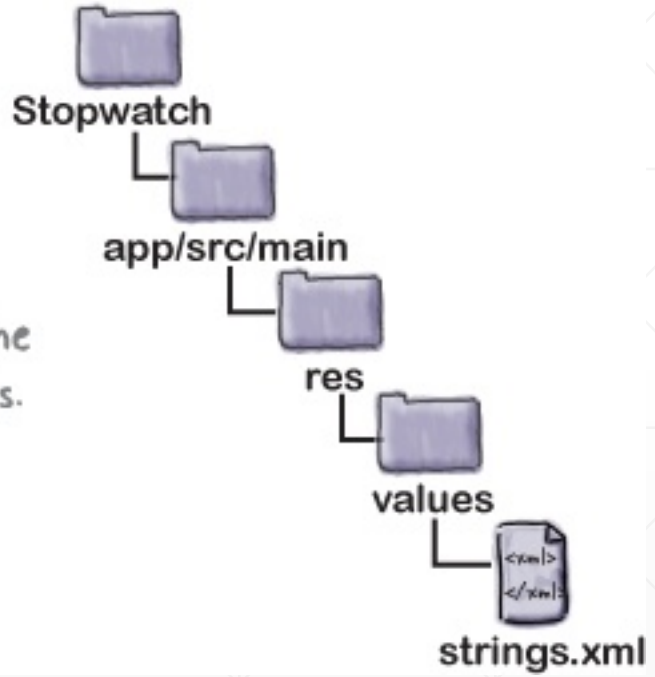
These attributes make the stopwatch timer nice and big.

This is for the Start button. It calls a method called onClickStart() when it gets clicked.

Chap 4 – Activity Lifecycle - Layout

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

These are the button labels.

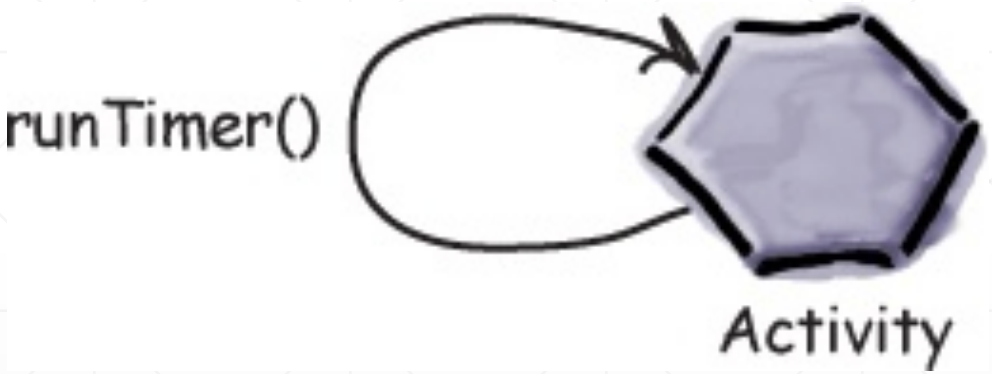


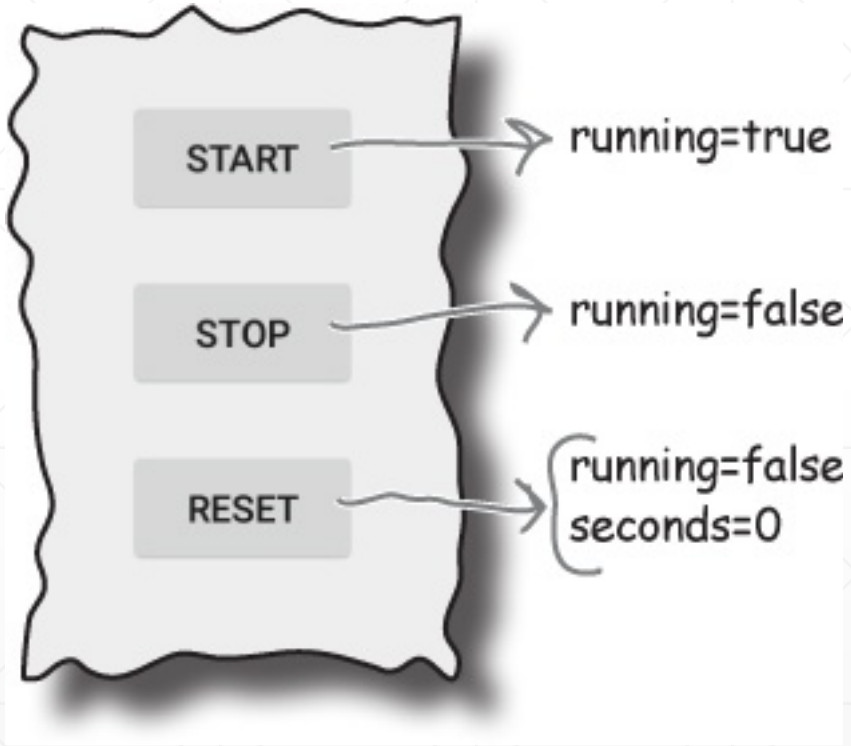
When you click on the Start button, the onClickStart() method is called.

When you click on the Stop button, the onClickStop() method is called.

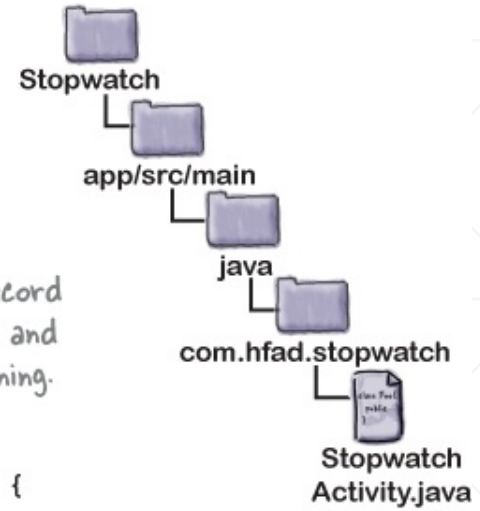
When you click on the Reset button, the onClickReset() method is called.

Chap 4 – Activity Lifecycle - Activity





Chap 4 – Activity




```
package com.hfad.stopwatch;

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

public class StopwatchActivity extends Activity {

    private int seconds = 0;
    private boolean running;

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


    //Start the stopwatch running when the Start button is clicked.
    public void onClickStart(View view) {
        running = true;
    }

    //Stop the stopwatch running when the Stop button is clicked.
    public void onClickStop(View view) {
        running = false;
    }

    //Reset the stopwatch when the Reset button is clicked.
    public void onClickReset(View view) {
        running = false;
        seconds = 0;
    }
}
```

Use seconds and running to record the number of seconds passed and whether the stopwatch is running.

This gets called when the Start button is clicked.

Start the stopwatch running.

This gets called when the Stop button is clicked.

Stop the stopwatch running.

This gets called when the Reset button is clicked.

Stop the stopwatch running and set the seconds to 0.

```
private void runTimer() {
    final TextView timeView = (TextView)findViewById(R.id.time_view);
    ...
    int hours = seconds/3600;
    int minutes = (seconds%3600)/60;
    int secs = seconds%60;
    String time = String.format("%d:%02d:%02d",
            hours, minutes, secs);
    timeView.setText(time);
    if (running) {
        seconds++;
    }
    ...
}
```

Get the text view.

We've left out a bit of code here. We'll look at that on the next page.

Format the seconds into hours, minutes, and seconds. This is plain Java code.

Set the text view text.

If running is true, increment the seconds variable.

Chap 4 – Activity

```
final Handler handler = new Handler();
handler.post(Runnable);
```

You put the code you want to run in the Handler's run() method.

Chap 4 – Activity

```
final Handler handler = new Handler();
handler.postDelayed(Runnable, long);
```

← Use this method to delay running code by a specified number of milliseconds.

```
private void runTimer() {
    final TextView timeView = (TextView)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("%d:%02d:%02d",
                    hours, minutes, secs);
            timeView.setText(time);
            if (running) {
                seconds++;
            }
            handler.postDelayed(this, 1000);
        }
    });
}
```

Create a new Handler.

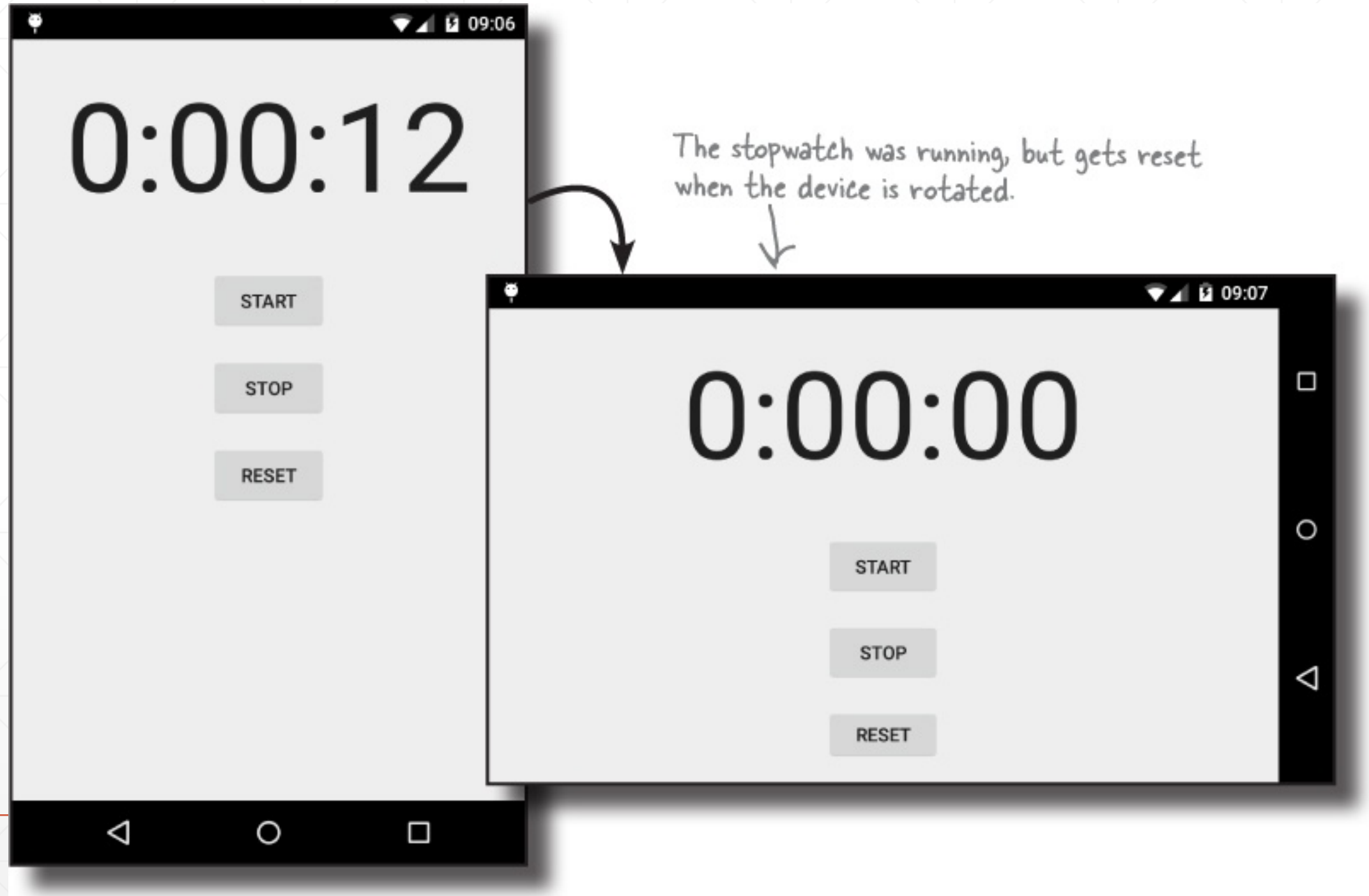
Call the post() method, passing in a new Runnable. The post() method processes codes without a delay, so the code in the Runnable will run almost immediately.

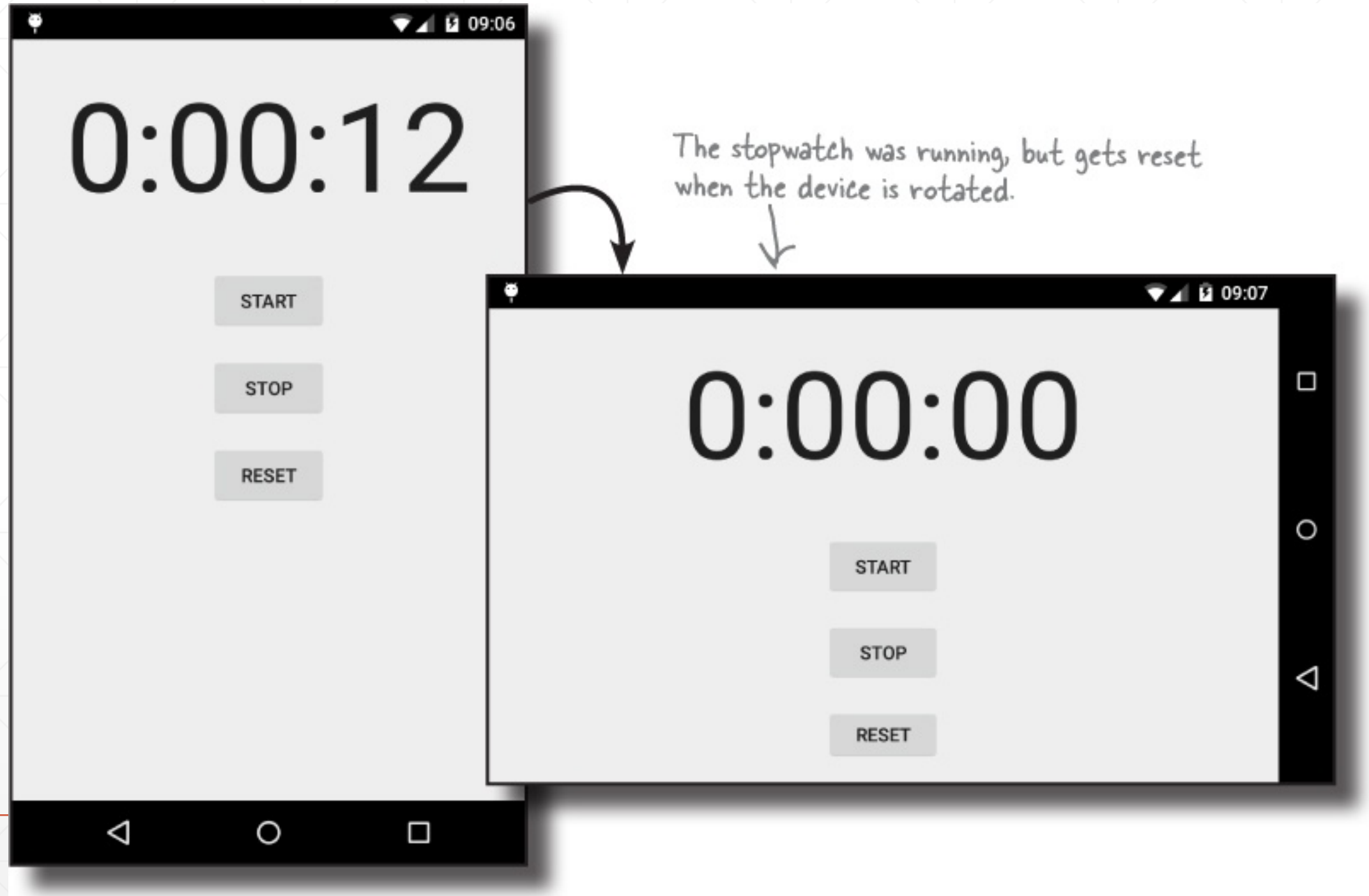
The Runnable run() method contains the code you want to be run—in our case, the code to update the text view.

Post the code in the Runnable to be run again after a delay of 1,000 milliseconds, or 1 second. As this line of code is included in the Runnable run() method, this will keep getting called.

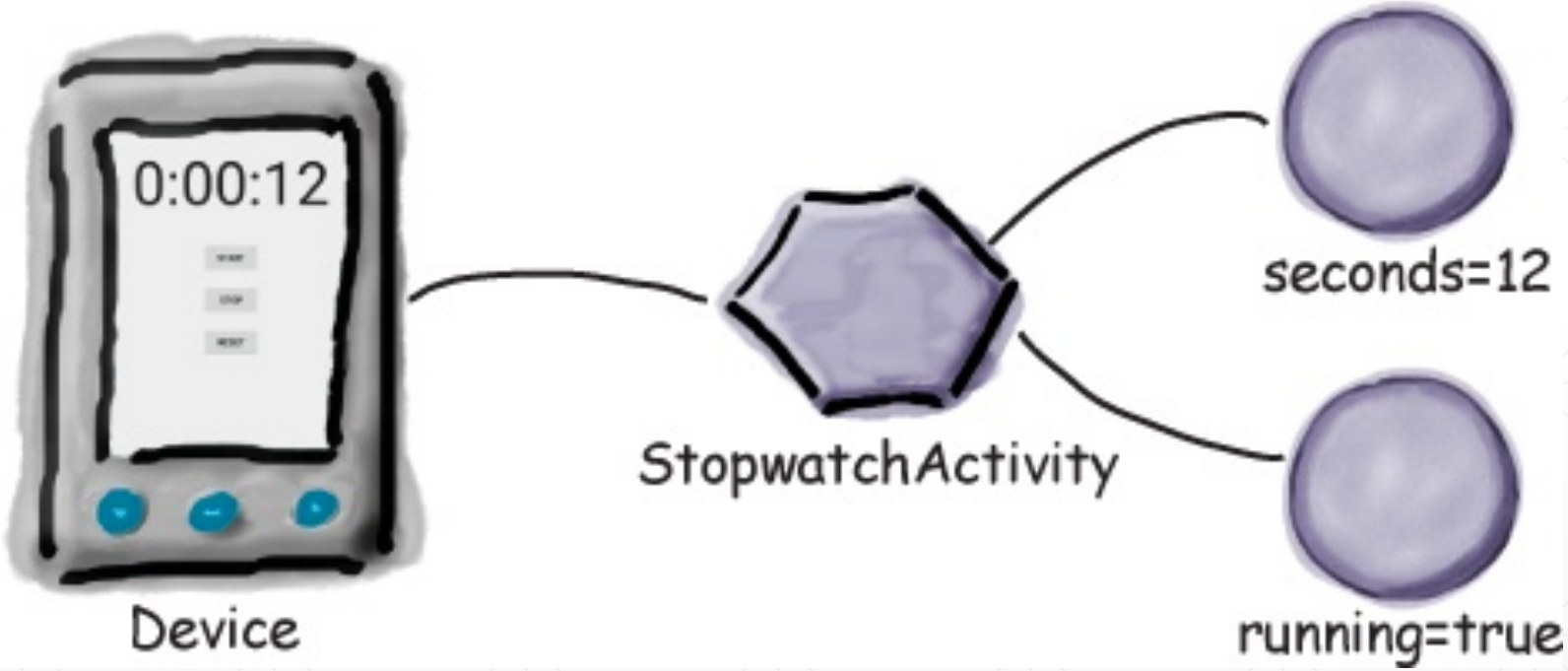
Chap 4 – Activity

```
protected void onCreate(Bundle savedInstanceState) {
...    runTimer();

}
```

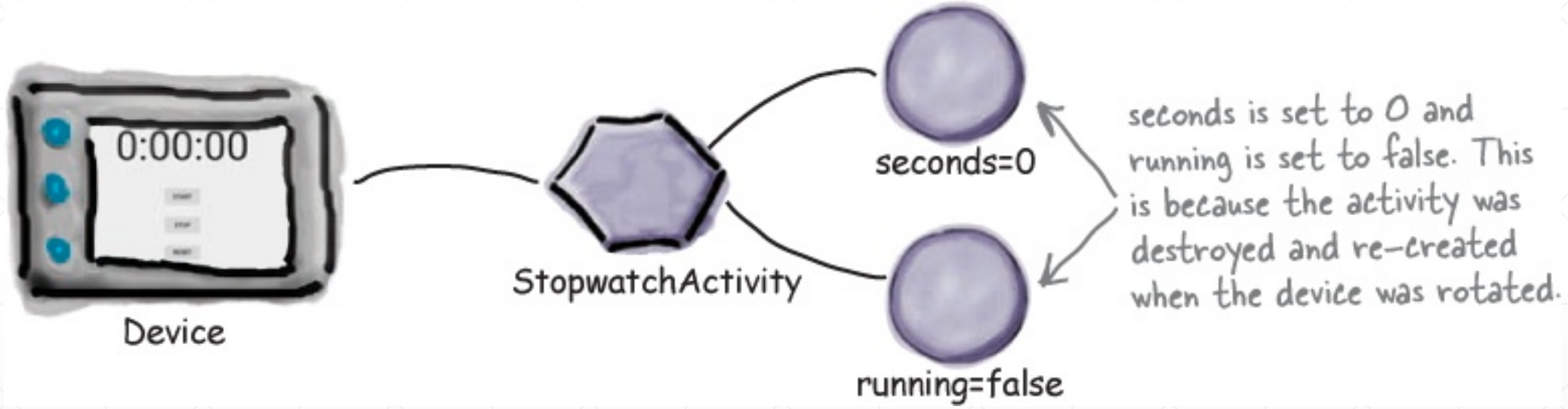

09:06
0:00:12
START
STOP
RESET
The stopwatch was running, but gets reset when the device is rotated.
09:07
0:00:00
START
STOP
RESET


09:06
0:00:12
START
STOP
RESET
The stopwatch was running, but gets reset when the device is rotated.
09:07
0:00:00
START
STOP
RESET

Chap 4 – Activity Lifecycle

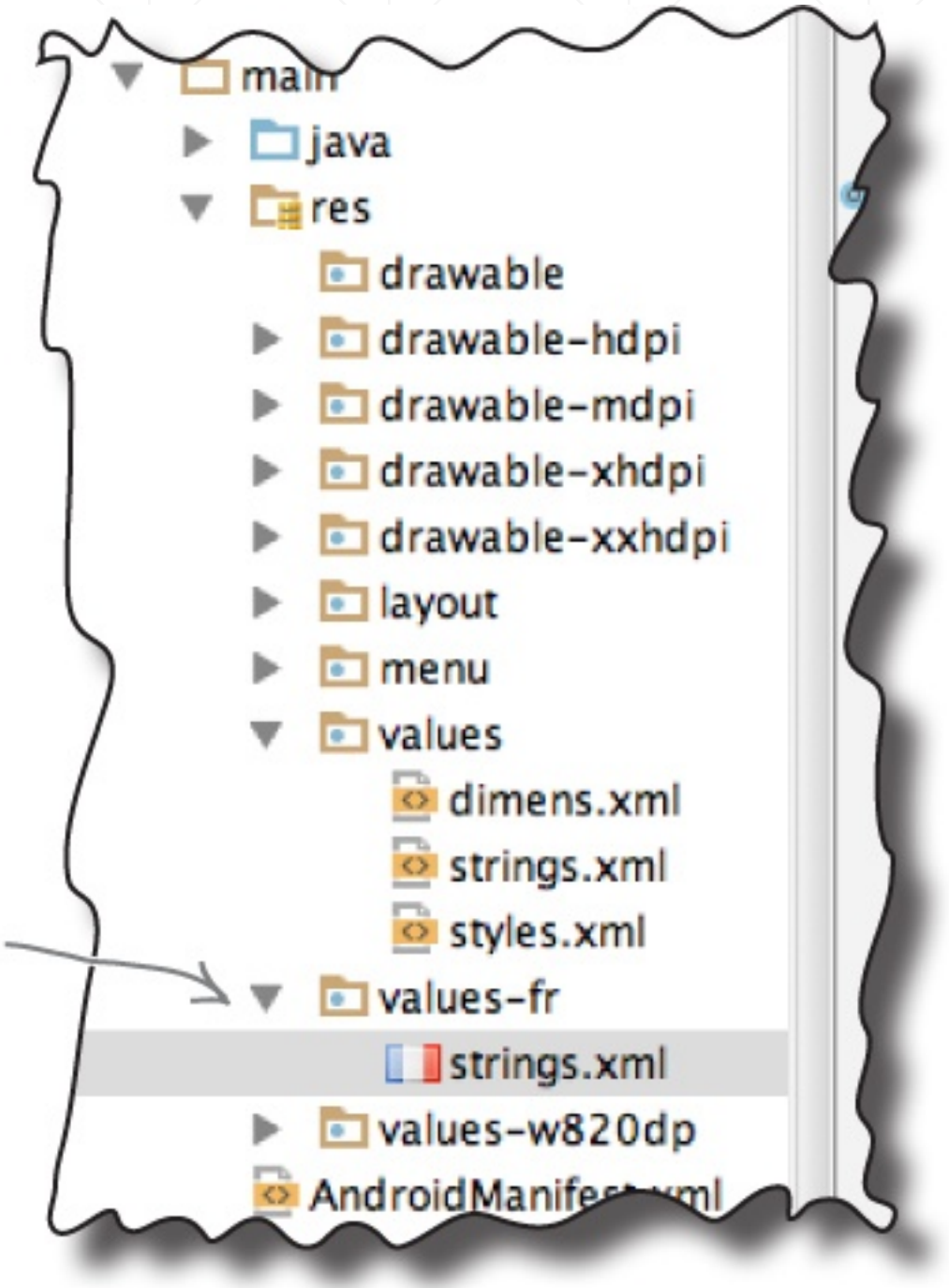


Chap 4 – Activity Lifecycle



Chap 4 – Activity Lifecycle

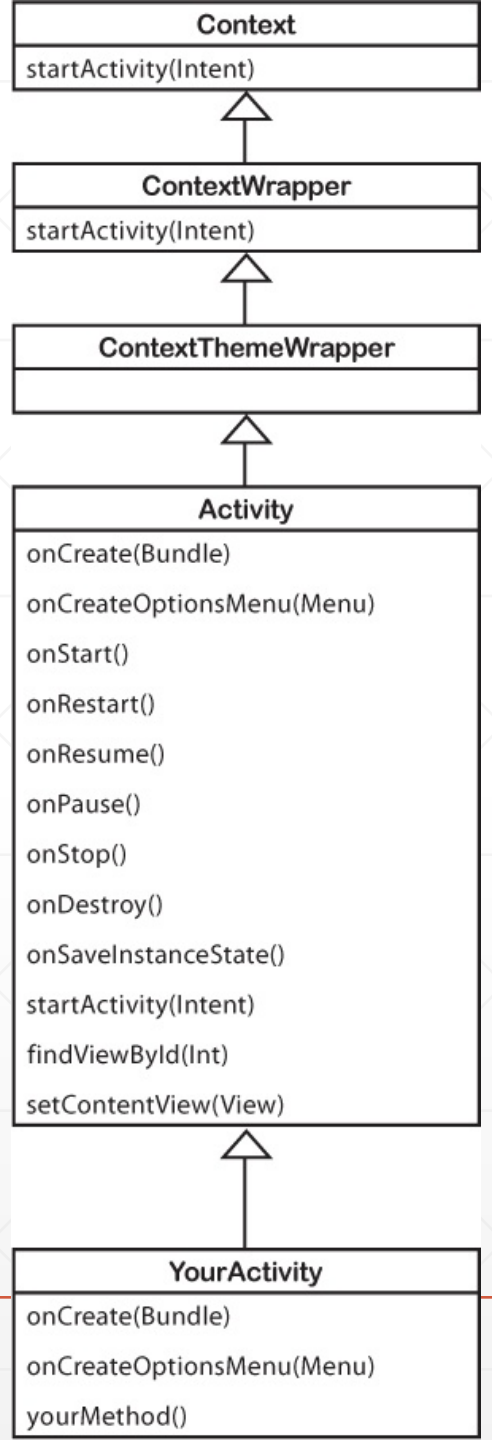
Android apps can contain multiple resource files in the app/src/main/res folder. If the device locale is set to France, Android will use the strings.xml file in the values-fr folder, for instance.

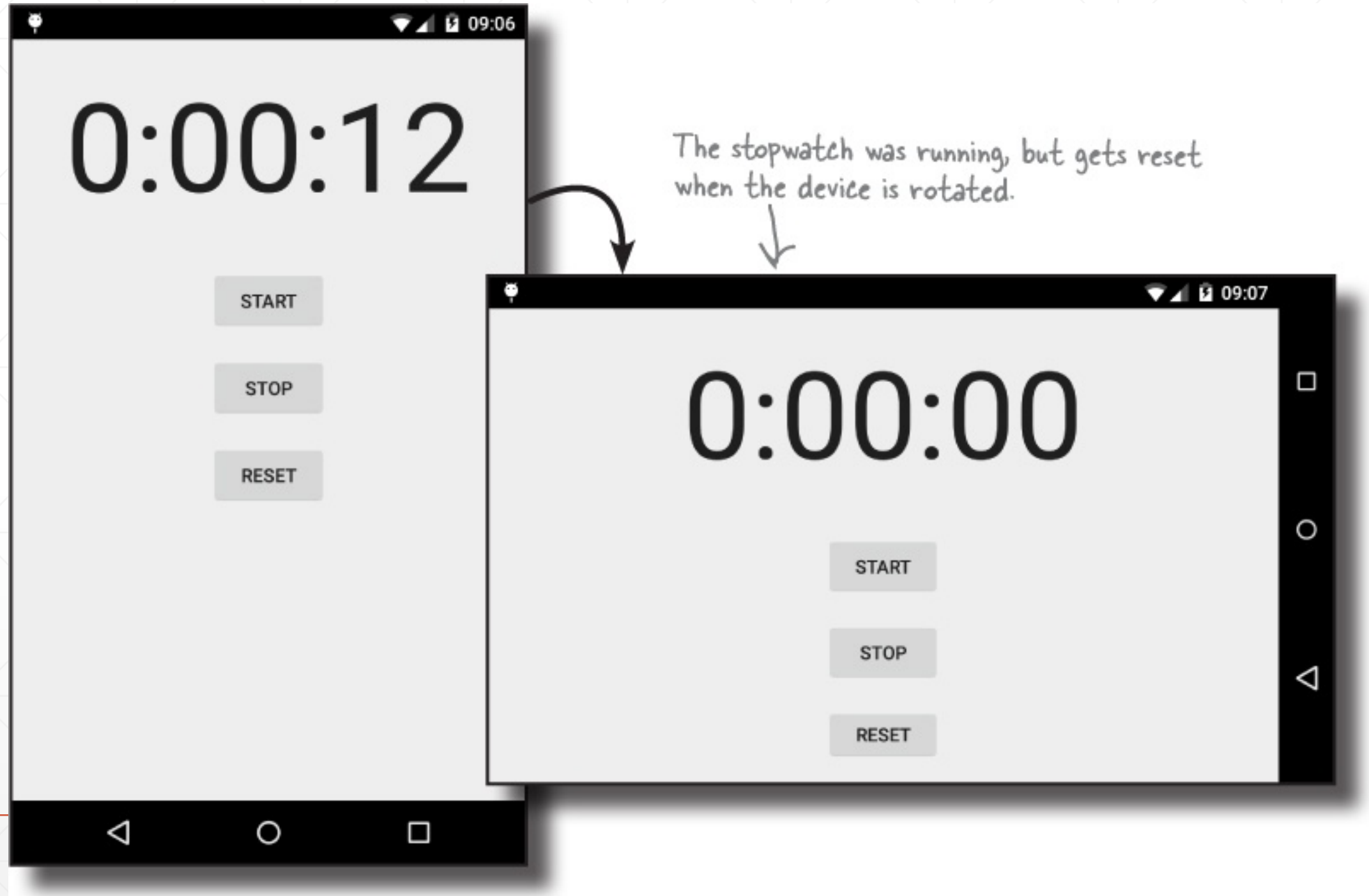


Chap 4 – Activity Lifecycle

Chap 4 – Activity Lifecycle

Chap 4 – Activity Lifecycle




09:06
0:00:12
START
STOP
RESET
The stopwatch was running, but gets reset when the device is rotated.
09:07
0:00:00
START
STOP
RESET

Chap 4 – Activity Lifecycle


```
<activity
    android:name="com.hfad.stopwatch.StopwatchActivity"
    android:label="@string/app_name"
    android:configChanges="orientation|screenSize" >
```

The | means we need to bypass both configuration changes. This is because most devices have a rectangular screen, so rotating the device changes both the orientation and the screen size.

```
public void onConfigurationChanged(Configuration config) {
        // do nothing here
}
```

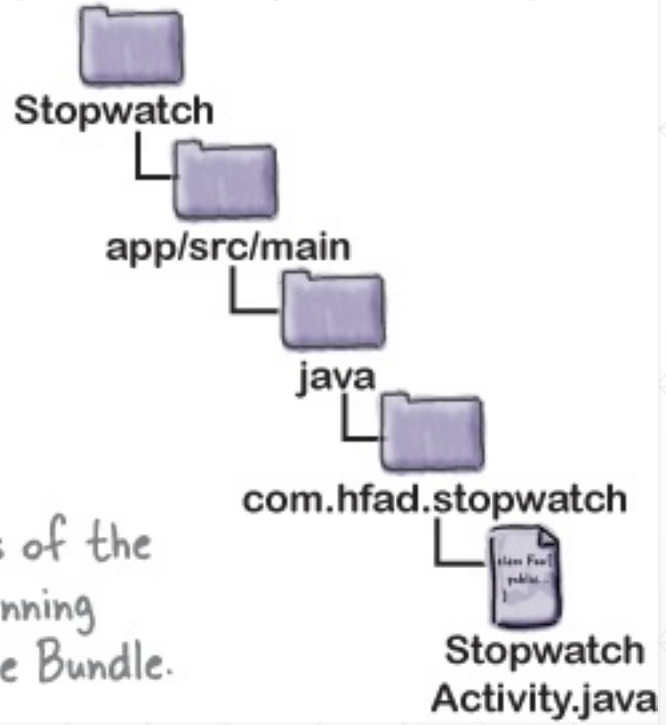
Chap 4 – Activity Lifecycle

bundle.put*("name", value)

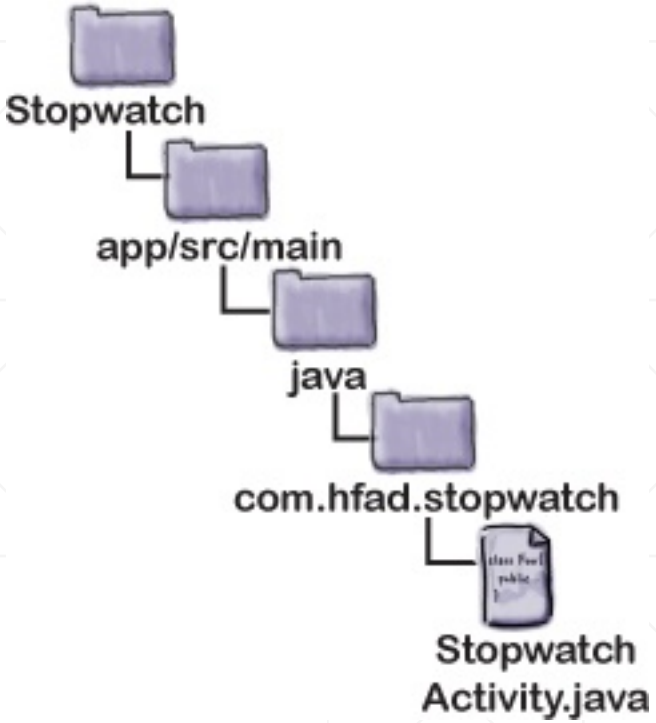
Chap 4 – Activity Lifecycle

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

Save the values of the seconds and running variables to the Bundle.

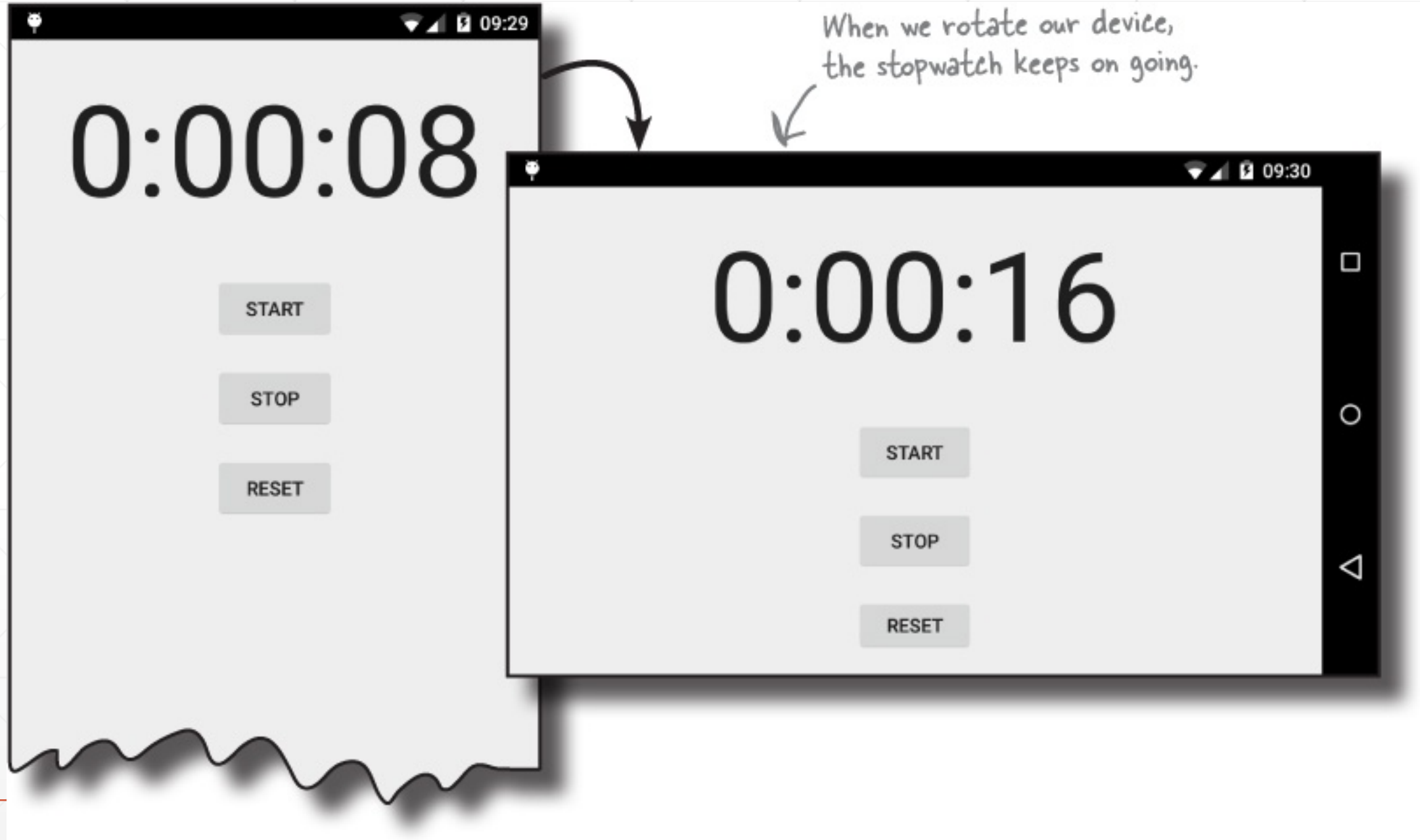


Chap 4 – Activity Lifecycle

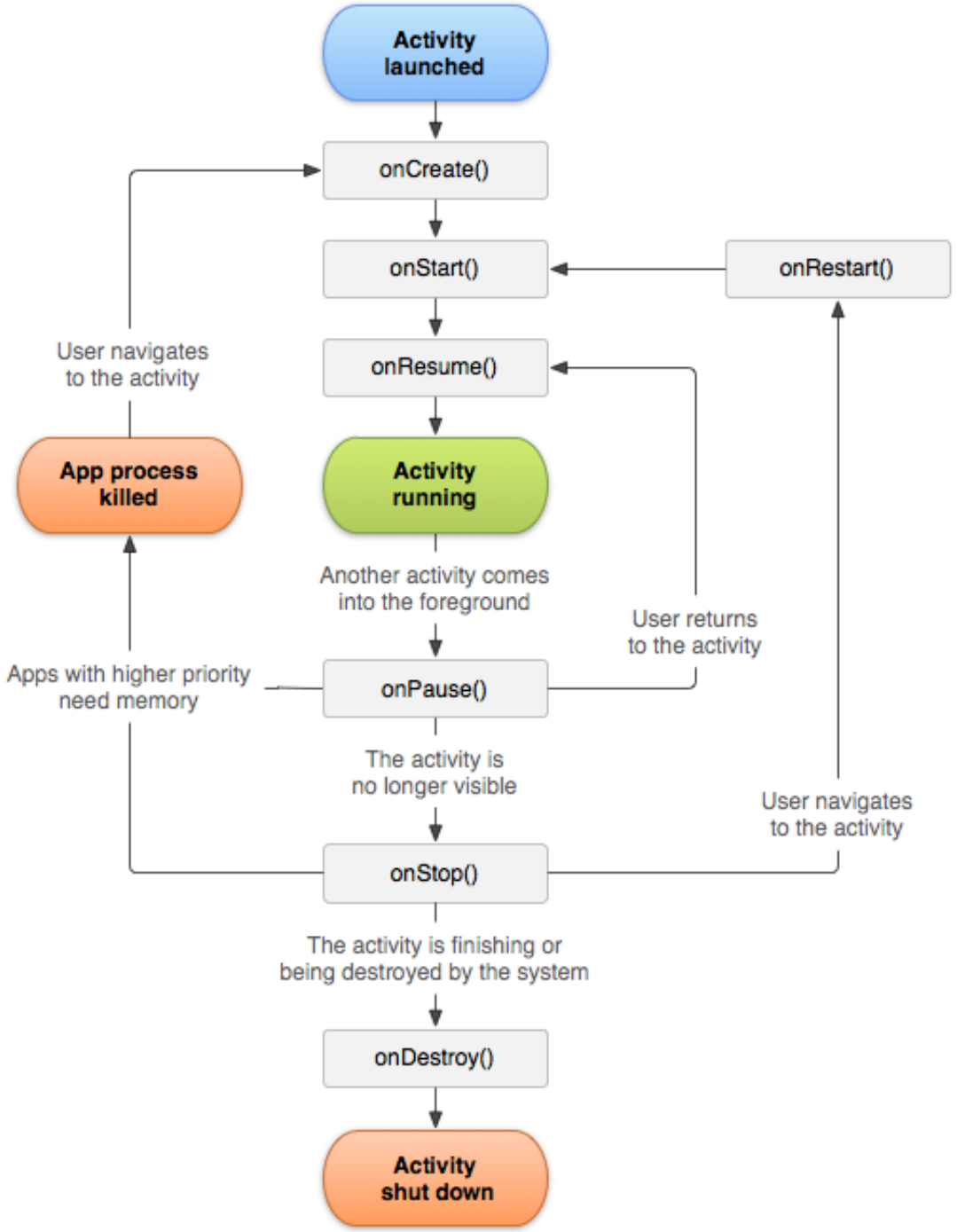



```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_stopwatch);
    if (savedInstanceState != null) {
        seconds = savedInstanceState.getInt("seconds");
        running = savedInstanceState.getBoolean("running");
    }
    runTimer();
}
```

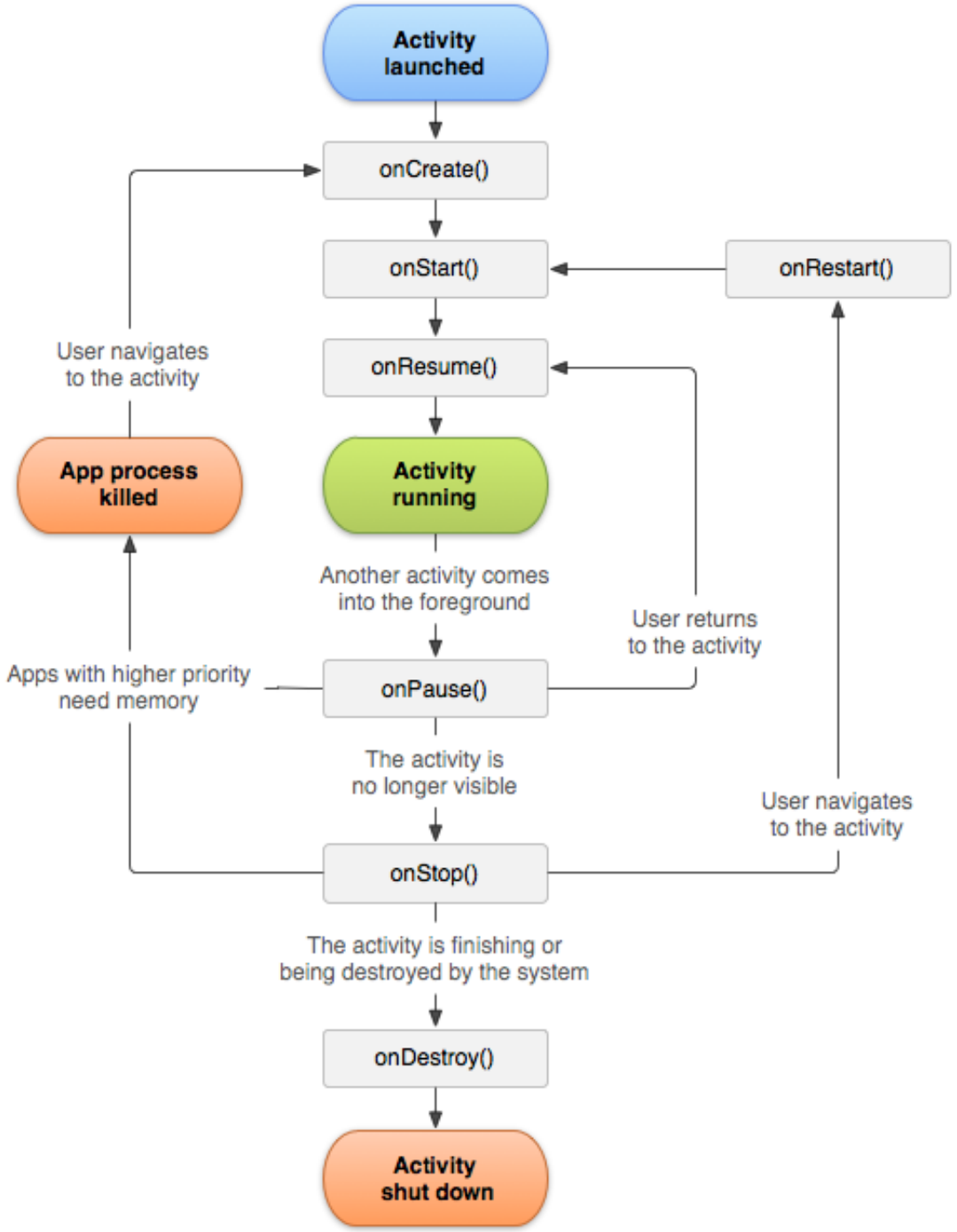
Retrieve the values of the seconds and running variables from the Bundle.


When we rotate our device,
the stopwatch keeps on going.
09:29
0:00:08
START
STOP
RESET
09:30
0:00:16
START
STOP
RESET

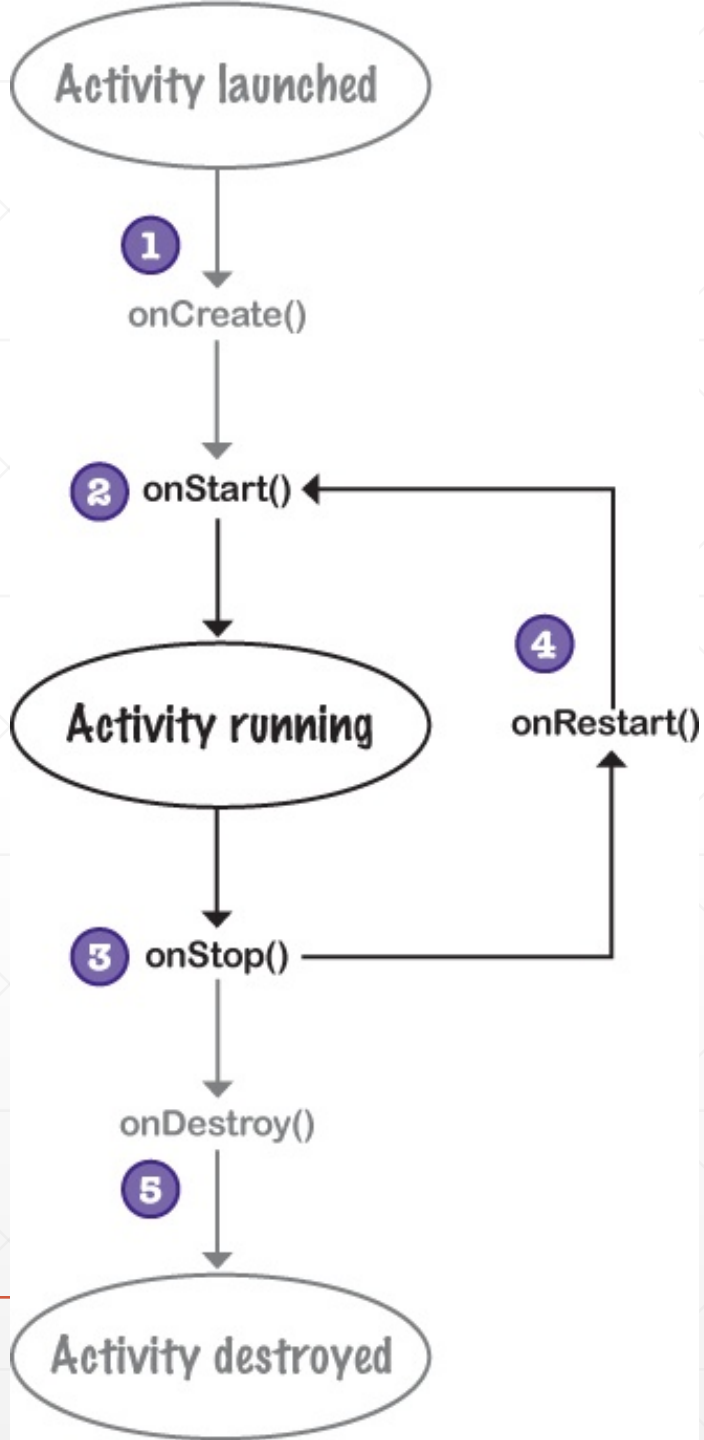
Chap 4 – Activity Lifecycle



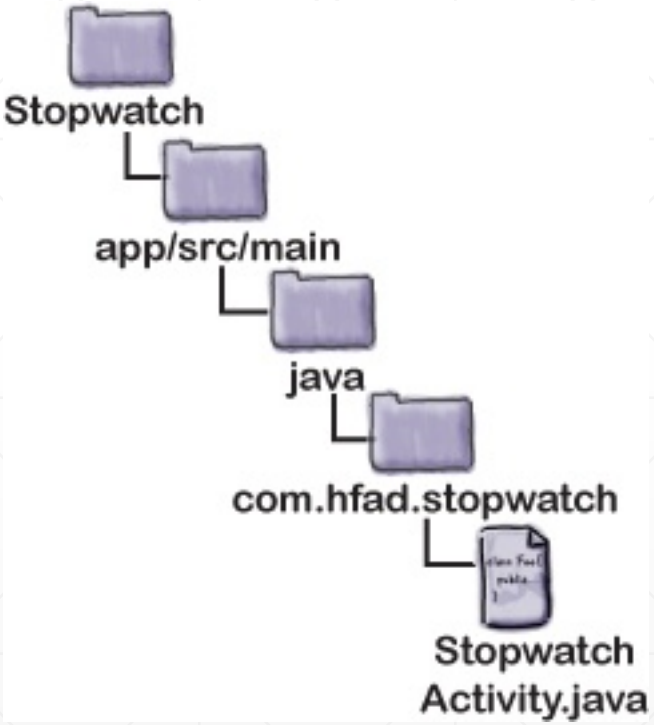
Chap 4 – Activity Lifecycle



Chap 4 – Activity Lifecycle



Chap 4 – Activity Lifecycle



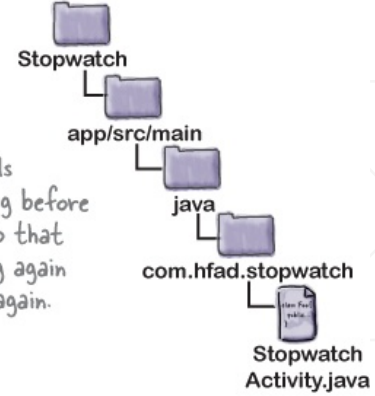
Chap 4 – Activity Lifecycle

```
@Override
protected void onStop() {
        super.onStop();
        running = false;
}
```

Chap 4 – Activity Lifecycle

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

Chap 4 – Activity Lifecycle




```
public class StopwatchActivity extends Activity {
    private int seconds = 0;
    private boolean running;
    private boolean wasRunning;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_stopwatch);
        if (savedInstanceState != null) {
            seconds = savedInstanceState.getInt("seconds");
            running = savedInstanceState.getBoolean("running");
            wasRunning = savedInstanceState.getBoolean("wasRunning");
        }
        runTimer();
    }

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

    @Override
    protected void onStop() {
        super.onStop();
        wasRunning = running;
        running = false;
    }

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

A new variable, wasRunning, records whether the stopwatch was running before the onStop() method was called so that we know whether to set it running again when the activity becomes visible again.

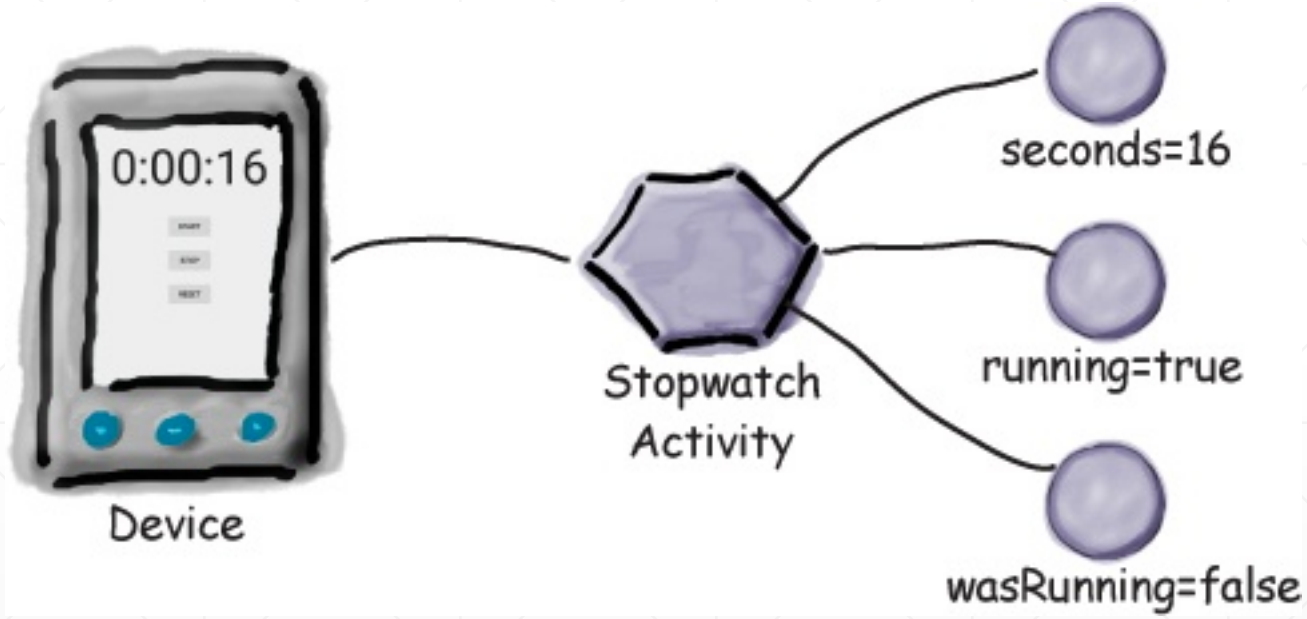
Restore the state of the wasRunning variable if the activity is re-created.

Save the state of the wasRunning variable.

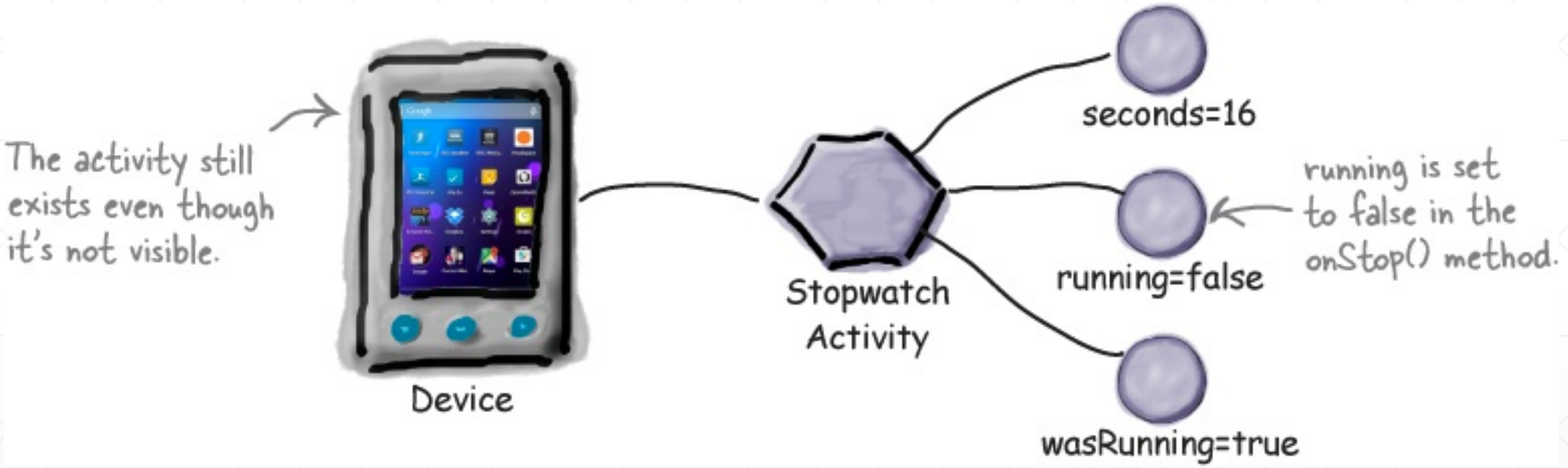
Record whether the stopwatch was running when the onStop() method was called.

Implement the onStart() method. If the stopwatch was running, set it running again.

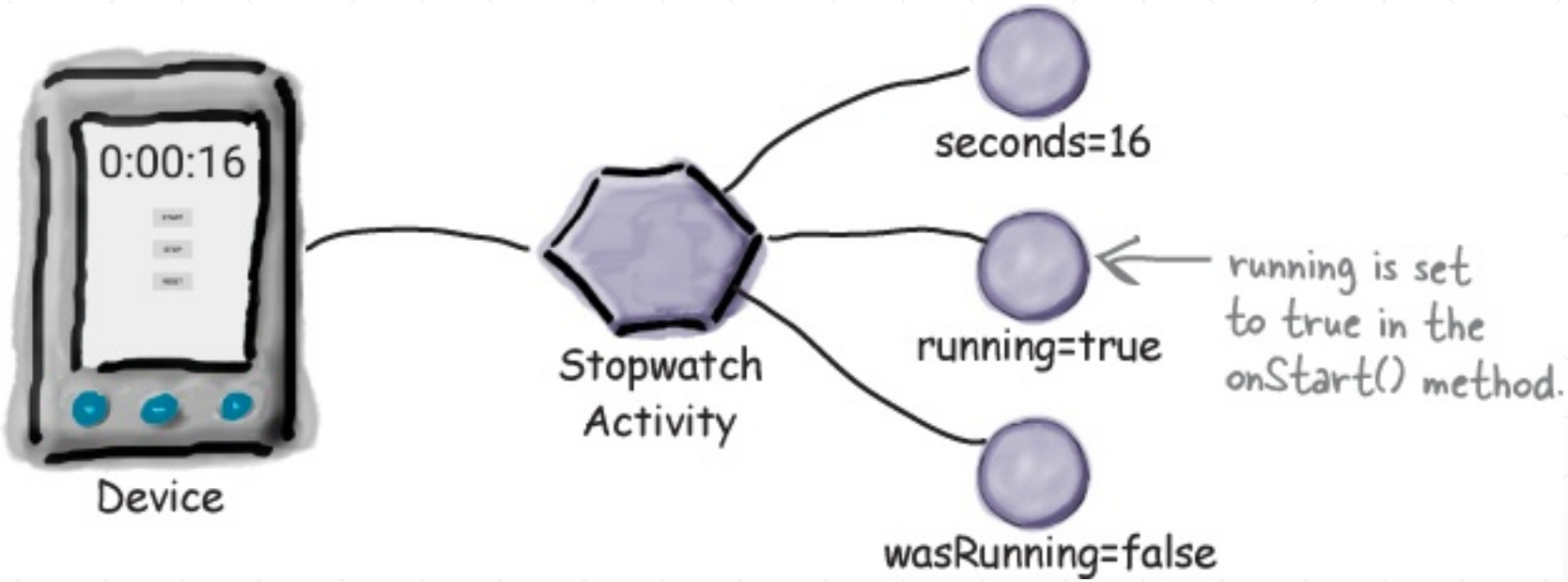
Chap 4 – Activity Lifecycle



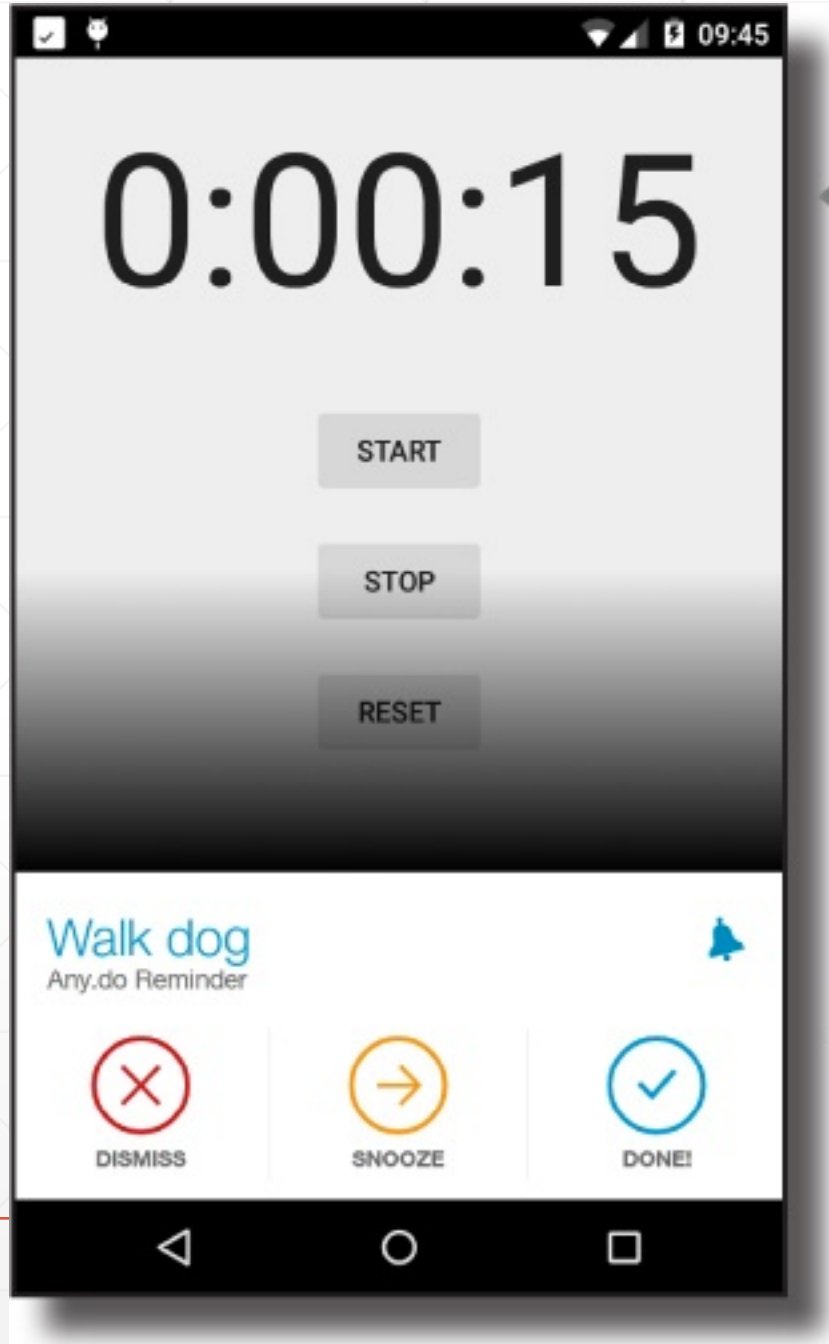
Chap 4 – Activity Lifecycle



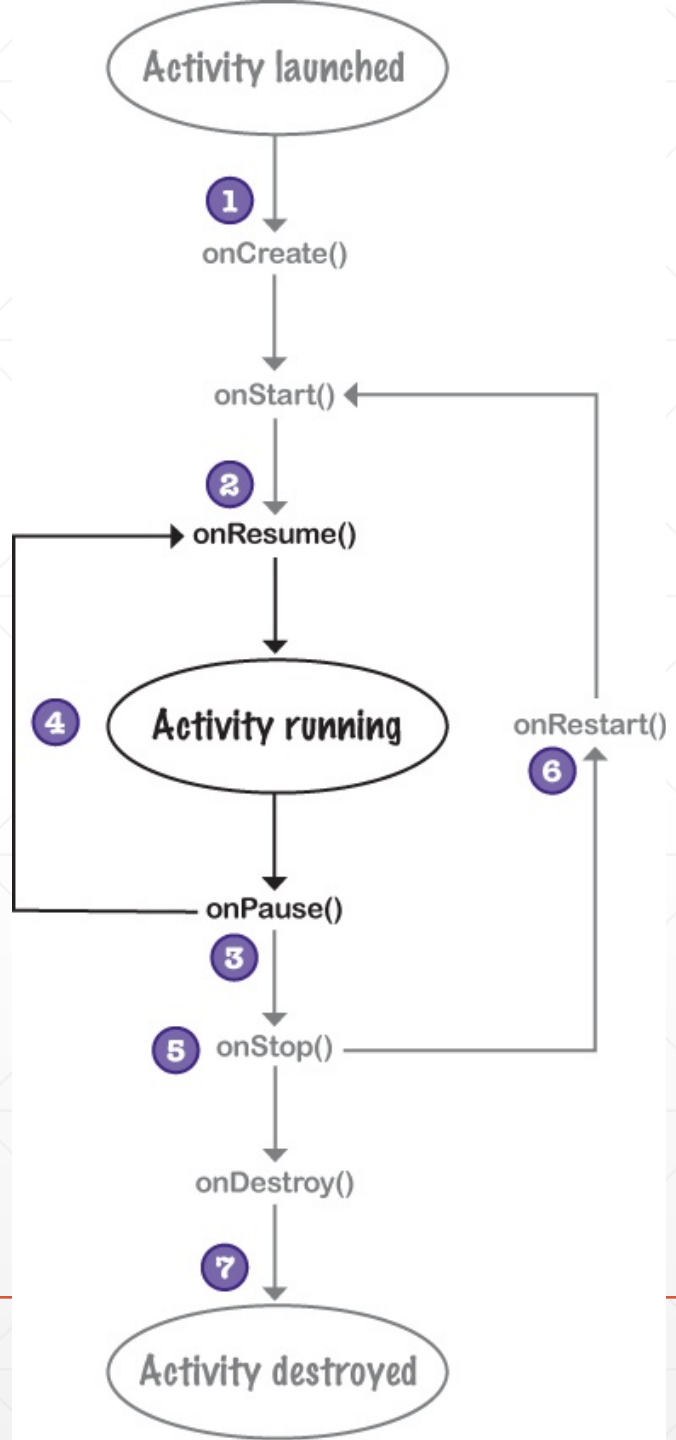
Chap 4 – Activity Lifecycle



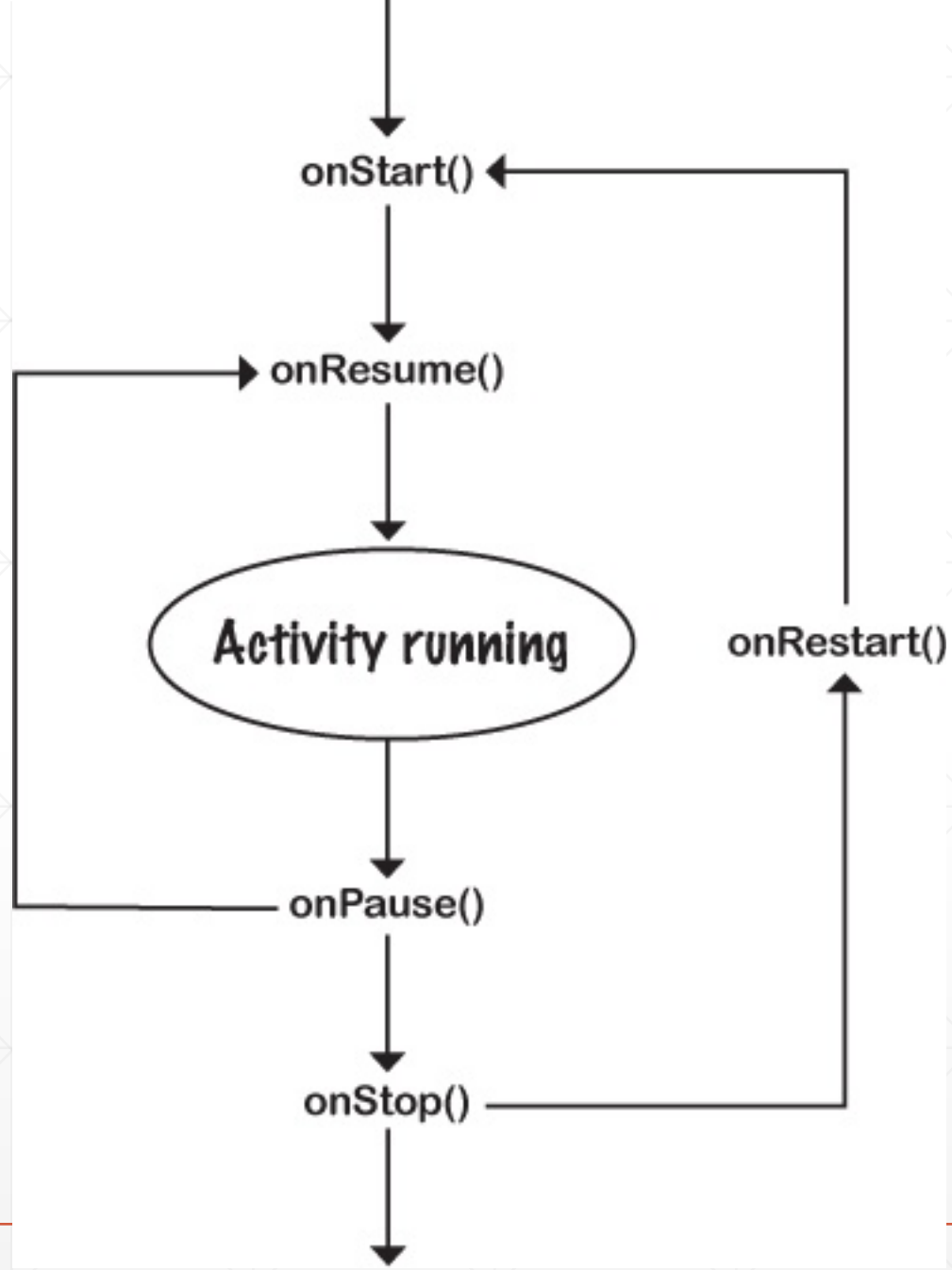
Chap 4 – Activity Lifecycle



Chap 4 –
Activity Lifecycle



Chap 4 –
Activity Lifecycle



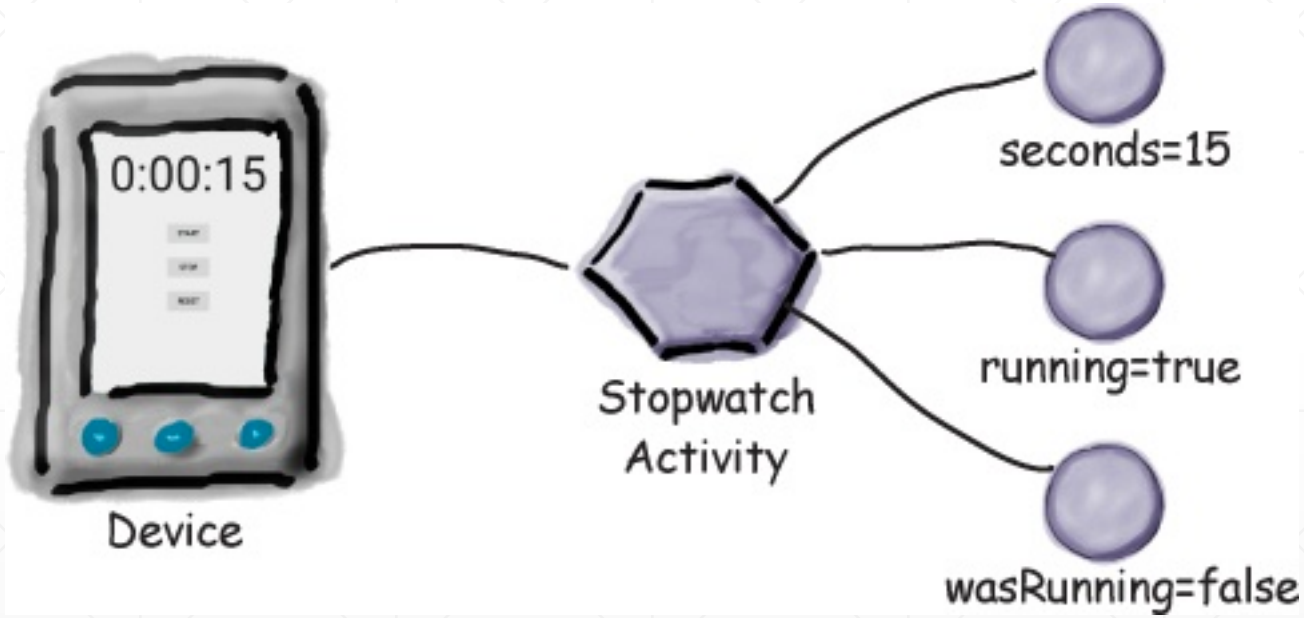
Chap 4 – Activity Lifecycle

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

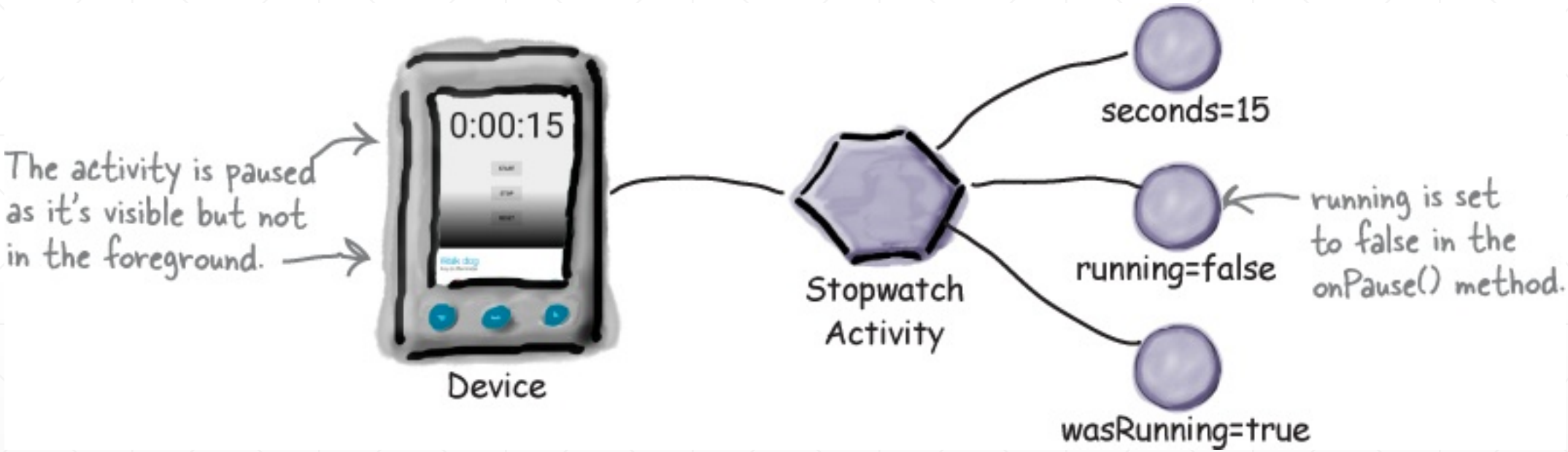
Chap 4 – Activity Lifecycle

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

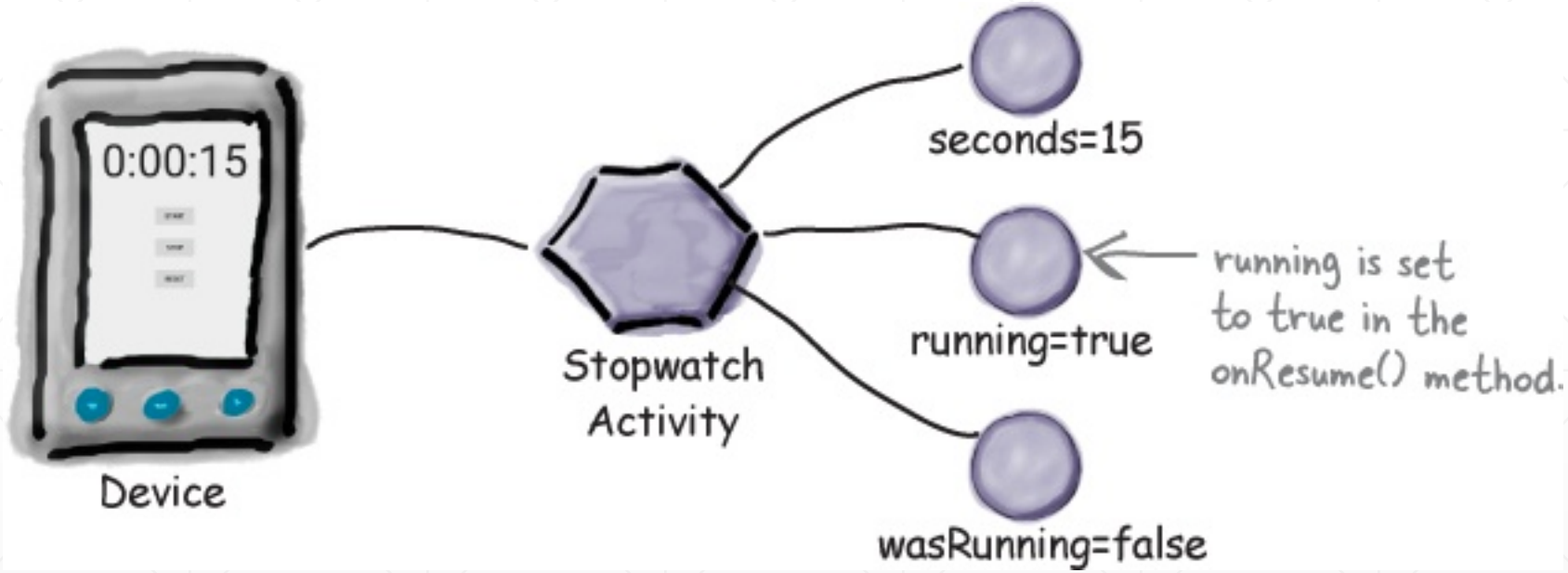
Chap 4 – Activity Lifecycle



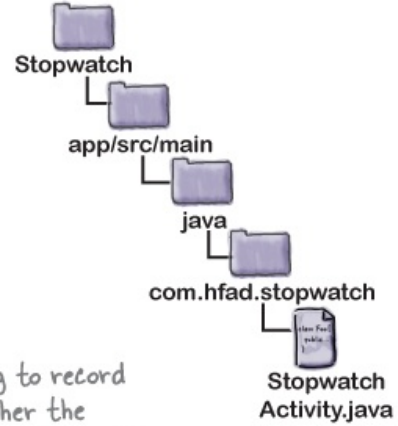
Chap 4 – Activity Lifecycle



Chap 4 – Activity Lifecycle



```
package com.hfad.stopwatch;

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

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

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_stopwatch);
        if (savedInstanceState != null) {
            seconds = savedInstanceState.getInt("seconds");
            running = savedInstanceState.getBoolean("running");
            wasRunning = savedInstanceState.getBoolean("wasRunning");
        }
        runTimer();
    }

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

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



Use seconds, running, and wasRunning to record the number of seconds passed, whether the stopwatch is running, and whether the stopwatch was running before the activity was paused.

Get the previous state of the stopwatch if the activity's been destroyed and re-created.

If the activity's paused, stop the stopwatch.

If the activity's resumed, start the stopwatch again if it was running previously.

Save the state of the stopwatch if it's about to be destroyed.

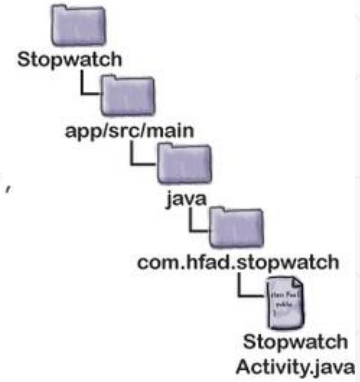
```
    @Override
    public void onSaveInstanceState(Bundle savedInstanceState) {
        savedInstanceState.putInt("seconds", seconds);
        savedInstanceState.putBoolean("running", running);
        savedInstanceState.putBoolean("wasRunning", wasRunning);
    }

    //Start the stopwatch running when the Start button is clicked.
    public void onClickStart(View view) {
        running = true;
    }

    //Stop the stopwatch running when the Stop button is clicked.
    public void onClickStop(View view) {
        running = false;
    }

    //Reset the stopwatch when the Reset button is clicked.
    public void onClickReset(View view) {
        running = false;
        seconds = 0;
    }

    //Sets the number of seconds on the timer.
    private void runTimer() {
        final TextView timeView = (TextView)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("%d:%02d:%02d",
                        hours, minutes, secs);
                timeView.setText(time);
                if (running) {
                    seconds++;
                }
                handler.postDelayed(this, 1000);
            }
        });
    }
}
```

This gets called when the Start button is clicked.

This gets called when the Stop button is clicked.

This gets called when the Reset button is clicked.

The runTimer() method uses a Handler to increment the seconds and update the text view.



Method	When it's called	Next method
onCreate()	When the activity is first created. Use it for normal static setup, such as creating views. It also gives you a Bundle giving the previously saved state of the activity.	onStart()
onRestart()	When your activity has been stopped just before it gets started again.	onStart()
onStart()	When your activity is becoming visible. It's followed by onResume() if the activity comes into the foreground, or onStop() if the activity is made invisible.	onResume()or onStop()
onResume()	When your activity is in the foreground.	onPause()
onPause()	When your activity is no longer in the foreground because another activity is resuming. The next activity isn't resumed until this method finishes, so any code in this method needs to be quick. It's followed by onResume() if the activity returns to the foreground, or onStop() if it becomes invisible.	onResume()or onStop()
onStop()	When the activity is no longer visible. This can be because another activity is covering it, or because the activity's being destroyed. It's followed byonRestart() if the activity becomes visible again, or onDestroy() if the activity is going to be destroyed.	onRestart() or onDestroy()
onDestroy()	When your activity is about to be destroyed or because the activity is finishing.	None


09:52
0:00:41
START
STOP
RESET
We started our stopwatch. It paused when the activity was partially obscured.
09:55
0:03:01
START
STOP
RESET
Walk dog
Any.do Reminder
DISMISS
SNOOZE
DONE!
09:55
0:03:26
START
STOP
RESET
The stopwatch started again when the activity came back into the foreground.

Chap 4 – Bullet Points

- Each app runs in its own process by default.
- Only the main thread can update the user interface.
- Use a Handler to schedule code, or post code to a different thread.
- A device configuration change results in the activity being destroyed and re-created.
- Your activity inherits the lifecycle methods from the Android Activity class. If you override any of these methods, you need to call up to the method in the superclass.
- onSaveInstanceState(Bundle) enables your activity to save its state before the activity gets destroyed.

Chap 4 – Bullet Points

- You can use the Bundle to restore state in onCreate().
- You add values to a Bundle using bundle.put*("name", value). You retrieve values from the bundle using bundle.get*("name").
- onCreate() and onDestroy(), deal with the birth and death of the activity.
- onRestart(), onStart() and onStop() deal with the visibility of the activity.
- onResume() and onPause() deal with when the activity gains and loses the focus.