Stopwatch Lifecycle

Create a new project named *Stopwatch* with a minimum API of *19*. Choose *Empty Activity* and uncheck “*Backward Compatibility (AppCompat)*”

strings.xml.txt

Add the following to strings.xml:

<string name="start">Start</string>

<string name="stop">Stop</string>

<string name="reset">Reset</string>

Change the main layout to *LinearLayout* and set *android:orientation="vertical"*

Replace contents of the layout file with the following widgets:

<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" />

layout.txt

<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" />

Add the following instance variables:

instance variables.txt

private int seconds = 0;

private boolean running;

Add the following click methods to the activity:

// Start the stopwatch running when the Start button is clicked

public void onClickStart(View v) {

onClick.txt

running = true;

}

// Stop the stopwatch running when the Stop button is clicked

public void onClickStop(View v) {

running = false;

}

// Reset the stopwatch when the Reset button is clicked

public void onClickReset(View v) {

running = false;

seconds = 0;

}

In Android, only the main thread can update the UI. The solution is to use a *Handler*.

We need to use a *Handler* class used to schedule code that should run as some time in the future. We will use a *Handler* to schedule the stopwatch code to run every second.

To use the *Handler*, wrap the code you wish to schedule in a *Runnable* object, then use the Handler’s *post()* and *postDelayed()* methods to specify when you want the code to run.

Add the following runTimer() method to update the UI every second:

private void runTimer() {

final TextView timeView = (TextView) findViewById(R.id.time\_view);

final Handler handler = new Handler();

handler.post(new Runnable() {

runTimer.txt

@Override

public void run() {

int hours = seconds/3600;

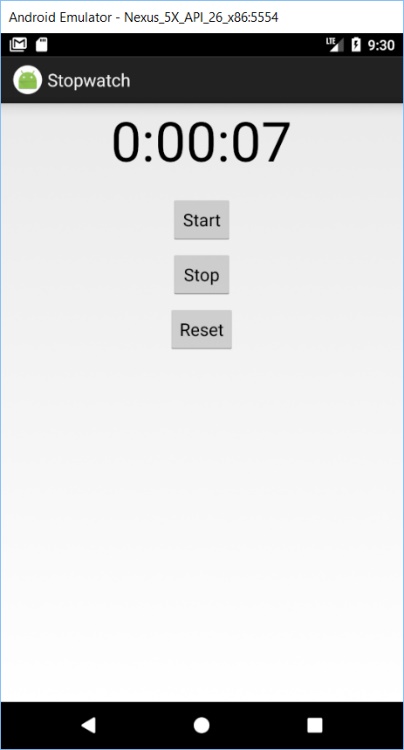
int minutes = (seconds%3600)/60;

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);

}

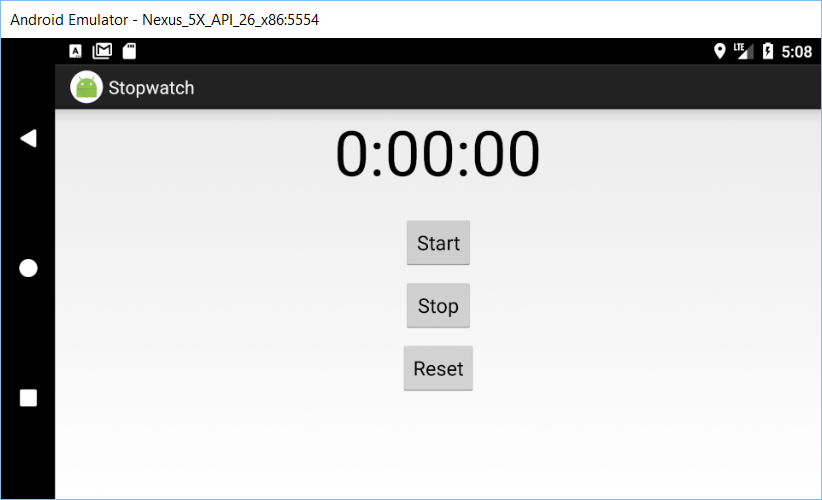
});

}

Call the timer by adding *runTimer()* in the *onCreate()* method.

Run the application to see how it performs.

There is a problem. When you rotate the phone from portrait to landscape it is set back to zero.



When the orientation has changed then screen size also changes. The activity is destroyed, including any variables sued by the *runTimer()* method. The *onCreate()* method runs again and when *runTimer()* gets called again the running and seconds variables are set to their default values.

When the device configuration changes, anything that displays a UI needs to be updated to match the new configuration.

When Android creates and destroys an activity, the activity moves from being launched >> running >> destroyed

onCreate()

onDestroy()

The onDestroy() method is called before the activity is destroyed.

The activity lifecycle methods are:

onCreate(Bundle)

onStart()

onRestart()

onPause()

onStop()

onDestroy()

Other very useful activity methods are:

onSaveInstanceState(Bundle)

startActivity(Intent)

findViewById(int)

setContentView(View)

To save the current state of your activity, you need to implement the *onSaveInstanceState(Bundle)* method. This method gets called before the activity gets destroyed. Bundle allows you to different types of data into a single object.

Add the following method to the activity to save the state of variables seconds & running:

@Override

public void onSaveInstanceState(Bundle savedInstanceState) {

savedInstanceState.putInt("seconds", seconds);

onSaveInstanceState.txt

savedInstanceState.putBoolean("running", running);

}

Then restore in the *onCreate()* method. Add the following code to *onCreate(Bundle)* right before *runTimer()*:

onCreate.txt

if (savedInstanceState != null) {

seconds = savedInstanceState.getInt("seconds");

running = savedInstanceState.getBoolean("running");

}

Run the app and you will determine that changing orientation does not reset the timer.

Other activity lifecycle methods that deal with an activity’s visibility are:

|  |  |
| --- | --- |
| onStart() | Called when an activity becomes visible to the user. |
| onStop() | Called when an activity has stopped being visible to the user. Possible causes are:   * Activity is hidden by another activity that has appeared on top of it * Activity is about to be destroyed. In this case, onSaveInstanceState() gets called before onStop() |
| onRestart() | After activity has been made invisible and before it gets made visible again. |

onCreate()

onStart()

onDestroy()

onStop()

onRestart()

We need to implement the activity’s *onStop()* method so that the stopwatch stops running when the activity is not visible. We then implement the *onStart()* method so that the stopwatch starts again when the app is visible.

HINT: In Android Studio, to let the IDE add override methods: right-click >> Generate >> Override Methods.

We need to introduce another variable wasRunning to record 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.

Add this instance variable to the activity class:

private boolean wasRunning;

Add the following code to the onSaveInstanceState(Bundle savedInstanceState) method:

savedInstanceState.putBoolean("wasRunning", wasRunning);

Add the following code to the onCreate() method:

wasRunning = savedInstanceState.getBoolean("wasRunning");

Add the following *onStop()* method to the activity:

protected void onStop() {

onStop.txt

super.onStop();

wasRunning = running;

running = false;

}

Add the following onStart() method to the activity:

@Override

protected void onStart() {

onStart.txt

super.onStart();

if (wasRunning) {

running = true;

}

}

Test the application. After you start the timer, start google chrome. Then come back to the timer. It should continue from where it left off.

What id the app is partially visible. This happens when an app is visible but does not have the focus. This situation causes the activity to be paused. Two activities are needed to solve this problem: *onPause()* and *onResume()*.

|  |  |
| --- | --- |
| onPause() | called when activity is visible but another activity has focus |
| onResume() | called immediately before activity is about to start interacting with the user |

onCreate()

onStart()

onDestroy()

onStop()

onRestart()

onResume()

onPause()

The *onResume()* method is called whether or not the activity is started or resumed. This means that we can move our code from *onStart()* to *onResume()*.

Similarly, the *onPause()* method is called whether or not the activity is paused or stopped. This means that we can move our code from *onStop()* to *onPause()*.

Comment out the *onStop()* and *onStart()* methods and replace them with the following.

@Override

protected void onPause() {

onPause.txt

super.onPause();

wasRunning = running;

running = false;

}

@Override

onResume.txt

protected void onResume() {

super.onResume();

if (wasRunning) {

running = true;

}

}

Run the application.