

The screenshot shows the Android Studio interface with the following details:

- Project Structure:** The left sidebar shows the project structure with modules like `manifests`, `java` (containing `com`), `example` (containing `stopwatch` and `MainActivity`), and `res` (containing `drawable`, `layout`, `mipmap`, `values`, and `xml`). Below these are generated files like `java (generated)` and `res (generated)`. Scripts like `build.gradle` and `gradle-wrapper.properties` are also listed.
- Code Editor:** The main editor window displays the `MainActivity.java` code. The code defines a class `MainActivity` that extends `AppCompatActivity`. It includes imports for `Bundle`, `Handler`, `View`, and `TextView`. The `onCreate` method is overridden to handle ` savedInstanceState` and set the content view to `activity\_main`. It also initializes `seconds`, `running`, and `wasRunning` variables. The `runTimer` method is called at the end of `onCreate`.
- Toolbars and Status Bar:** The top bar shows tabs for `activity\_main.xml` and `MainActivity.java`. The bottom status bar indicates the code has been modified 8 minutes ago, the time is 31:51, the encoding is UTF-8, and there are 4 spaces.
- Sidemenu:** The right side features a vertical sidemenu with icons for Device Manager, Emulator, and Device File Explorer.

```
src/main/java/com/example/stopwatch/MainActivity.java
```

```
1 package com.example.stopwatch;
2
3 import androidx.appcompat.app.AppCompatActivity;
4
5 import android.os.Bundle;
6 import android.os.Handler;
7 import android.view.View;
8 import android.widget.TextView;
9
10 public class MainActivity extends AppCompatActivity {
11     private int seconds = 0;
12     private boolean running;
13     private boolean wasRunning;
14
15     @Override
16     protected void onCreate(Bundle savedInstanceState) {
17         super.onCreate(savedInstanceState);
18         setContentView(R.layout.activity_main);
19         if (savedInstanceState != null) {
20             seconds = savedInstanceState.getInt("seconds");
21             running = savedInstanceState.getBoolean("running");
22             wasRunning = savedInstanceState.getBoolean("wasRunning");
23         }
24         runTimer();
25     }
26 }
```

File Edit View Navigate Code Refactor Build Run Tools VCS Window Help STOPWATCH - MainActivity.java [STOPWATCH.app.main]

ATCH app src main java com example stopwatch MainActivity onSaveInstanceState

Android Project Resource Manager Structure Bookmarks Build Variants

activity\_main.xml MainActivity.java

```
25 }
26 }
27 @Override
28 public void onSaveInstanceState(Bundle savedInstanceState) {
29     super.onSaveInstanceState(savedInstanceState);
30     savedInstanceState.putInt("seconds", seconds);
31     savedInstanceState.putBoolean("running", running);
32     savedInstanceState.putBoolean("wasRunning", wasRunning);
33 }
34 @Override
35 protected void onStart() {
36     super.onStart();
37     if (wasRunning) {
38         running = true;
39     }
40 }
41 @Override
42 protected void onStop() {
43     super.onStop();
44     wasRunning = running;
45     running = false;
46 }
47 public void onClickStart(View view) {
48     running = true;
49 }
//Stop the stopwatch running when the Stop button is clicked.
```

Version Control Run TODO Problems Terminal Logcat App Inspection Build Profiler Event Log Layout Inspector

The screenshot shows the Android Studio interface with the following details:

- Project Structure:** The left sidebar displays the project structure under the **app** module. It includes the **src** folder containing **main**, **java**, and **res**. The **java** folder contains the **com.example.stopwatch** package with the **MainActivity** class selected.
- Code Editor:** The main editor window shows the **MainActivity.java** file. The code implements a stopwatch functionality with methods for stopping and resetting the timer, and a private method for running the timer.
- Code Snippet:** The code snippet below highlights the string format used for displaying time in hours, minutes, and seconds.

```
//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;  
}  
  
private void runTimer() {  
    final TextView timeView = (TextView)findViewById(R.id.textView);  
    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++;  
            }  
        }  
    });  
}
```

**Bottom Navigation Bar:** The navigation bar includes icons for Version Control, Run, TODO, Problems, Terminal, Logcat, App Inspection, Build, Profiler, Event Log, and Layout Inspector.

The screenshot shows the Android Studio interface with the code editor open to `MainActivity.java`. The code implements a timer functionality using a Handler.

```
private void runTimer() {
    final TextView timeView = (TextView)findViewById(R.id.textView);
    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, delayMillis: 1000);
        }
    });
}
```

The code is annotated with several highlights:

- Line 64: `String time = String.format("%d:%02d:%02d", hours, minutes, secs);` is highlighted in yellow.
- Line 74: `handler.postDelayed(this, delayMillis: 1000);` is highlighted in blue.
- Line 64: A green arrow icon is placed before the opening brace of the inner `Runnable` anonymous class, indicating a potential issue or warning.

The status bar at the top shows the device is connected as `emulator_5554 API 23 x86_64`.

