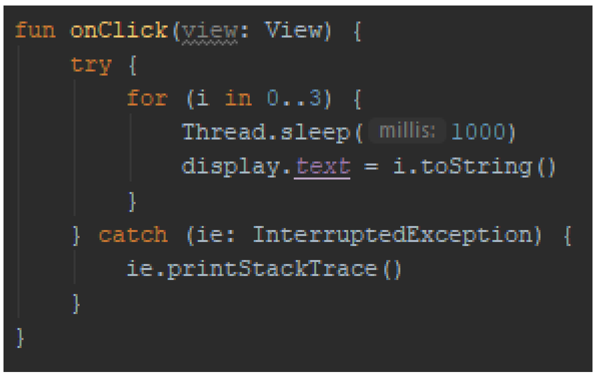
Mobile Dev 6.1P

Daniel Coady (102084174)

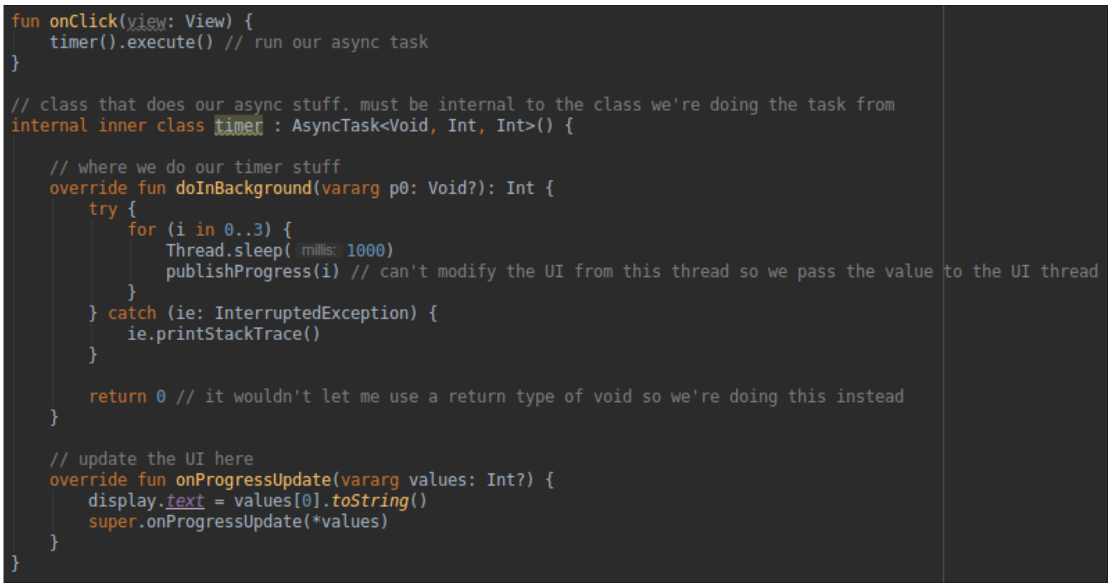
13/10/2019



```
fun onClick(view: View) {
    try {
        for (i in 0..3) {
            Thread.sleep( millis: 1000)
            display.text = i.toString()
        }
    } catch (ie: InterruptedException) {
        ie.printStackTrace()
    }
}
```

Figure 1: Original code

The problem with this original code is that it would hold up the main thread of the application. Because of this, all other operations were also halted and the application became completely unresponsive. To fix this we can create an AsyncTask to handle this operation in the background.



```kotlin
fun onClick(view: View) {
    timer().execute() // run our async task
}

// class that does our async stuff. must be internal to the class we're doing the task from
internal inner class timer : AsyncTask<Void, Int, Int>() {

    // where we do our timer stuff
    override fun doInBackground(vararg p0: Void?): Int {
        try {
            for (i in 0..3) {
                Thread.sleep( millis: 1000)
                publishProgress(i) // can't modify the UI from this thread so we pass the value to the UI thread
            }
        } catch (ie: InterruptedException) {
            ie.printStackTrace()
        }

        return 0 // it wouldn't let me use a return type of void so we're doing this instead
    }

    // update the UI here
    override fun onProgressUpdate(vararg values: Int?) {
        display.text = values[0].toString()
        super.onProgressUpdate(*values)
    }
}
```

Figure 2: New code

How the code works

onClick()

When the user clicks the button, we construct a new instance of the timer class and tell it to execute it's task.

doInBackground()

This is where the countdown code is placed since anything running here will be run on a separate thread to the main one, allowing for normal operation to resume even when the countdown is occuring. There is one change that was made to the code however, and that is where we change the text inside of our text view. To do this we actually need to publish that progress has been made so that we may update the UI accordingly. The reason why this change was made is because the UI may not be modified from outside of the UI thread.

onProgressUpdate()

Finally here we can update our UI since from within this method we have access to the UI thread.