

# TASK GUIDE (B2.05)

## A. Objectives.

Student will understand how Countdowntimer works.

## B. Requirements.

Hardware:

- 2 GB RAM minimum, 8 GB RAM recommended
- 2 GB of available disk space minimum, 4 GB Recommended (500 MB for IDE + 1.5 GB for Android SDK and emulator system image)
- 1280 x 800 minimum screen resolution
- Intel processor with support for Intel VT-x, Intel EM64T (Intel 64), and Execute Disable (XD) Bit functionality

Software:

- Microsoft Windows 7/8/10 (32-bit or 64-bit)
- JDK 8
- Android Studio IDE (Minimum 3.2)

## C. Resources.

Documents:

- Guide

Supplement files:

- -

Test code:

- TestB2AdvancedWidgetsKT051.java

## D. Task Description.


Student start to declare Countdowntimer and utilize it to update timer.

## E. Specification.

1. Open task B2.04 (ColorGame project) that already test passed.
2. Open “MyActivity.kt” file and add new fields in MyActivity class, with this description.

Name	Data type	Modifiers access	Initial value
countDown	CountDownTimer	-	-
FORMAT	String	final	%d:%d

How to declare?



```
public class MyActivity extends AppCompatActivity {  
    .  
    .  
    <Modifier> <Field name>: <Data type>? = <initial value>;  
    .  
    .  
}
```

3. Create a new private void method with name “initTime” and blank parameters.

```
private fun initTimer() {  
}
```

4. In the “initTimer” method, define the countdown field with CountDownTimer definition, like below.

```
countDown = object : CountDownTimer(millisInFuture, countDownInterval)  
{  
    override fun onTick(millisUntilFinished: Long) {  
    }  
    override fun onFinish() {  
    }  
};
```

The code shows that the definition of CountDownTimer has 2 parameters. millisInFuture is the value (in milliseconds) of end time and countDownInterval is the value of step of time (in milliseconds).

Change the value of **millisInFuture** become

```
(resources.getInteger(R.integer.maxtimer) * 1000).toLong()
```

Change the value of **countDownInterval** become

1

The definition also produced 2 methods, onTick and onFinish. onTick is an event that will call every time step and onFinish is an event that will call if time reached **millisInFuture**.

5. In the “onTick” method, define the text will be shown in timer (TextView). Put this code in the “onTick” method.

```
timer!!.text = "" + String.format(FORMAT,
    TimeUnit.MILLISECONDS.toSeconds(millisUntilFinished) -
    TimeUnit.MINUTES.toSeconds(TimeUnit.MILLISECONDS.toMinutes(
        millisUntilFinished)),
    TimeUnit.MILLISECONDS.toMillis(millisUntilFinished) -
    TimeUnit.SECONDS.toMillis(TimeUnit.MILLISECONDS.toSeconds(
        millisUntilFinished)))
```

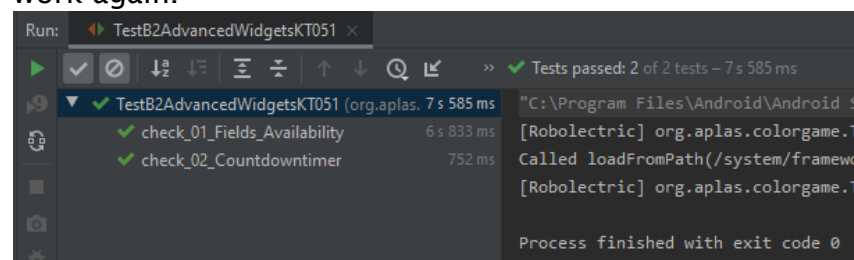
This code will update the text of timer every milliseconds.

6. Call “initTimer” method in the “onCreate” method, like below.

```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_layout)
    .
    .
    .
    initTimer();
}
```

## F. Testing.

1. Copy “TestB2AdvancedWidgetsKT051.java” file to “org.aplas.colorgame (test)” folder.
2. Right click on the “TestB2AdvancedWidgetsKT051.java” file then choose Run ‘TestB2AdvancedWidgetsKT051’ and click it. It may take long time to execute.
3. Get the result of your task. If passed you will get green check like below. If the test failed, you will get orange check get the messages and you must check your work again.



**You have to try until get all green checkes and continue to the next task.**

## Run the App

If you have passed the test, no change in the UI.