Group1 Simple Timer Unit Test Design:

1. Activity Test

TimerActivityTest.java in the test/androidTest directory delegates to AbstractTimerActivityTest.java in java/test/ androidTest;   
its test methods don’t run in the UI thread, but instead use it:

* testActivityTestCaseSetUpProperly – Activity is launched OK
* testActivityScenarioInit – state display is “Stopped”, button display is "Start" and time display is 0.
* testActivityScenarioCancelRunning – verifies the following scenario:
  + time is 0, state display is “Stopped”, and button display "Start"
  + press button, expect state display is “Increment”, and button display "Increment"
  + press button 9 times, every press expect time display adding 1
  + wait 3 seconds, expect time display is 10, state display is “Running”, and button display "Cancel"
  + wait 5 seconds, expect time display is “5”
  + press button, expect time display is 0, state display is “Stopped”, and button display "Start"
* testActivityScenarioRepeatIncrementWithin3Seconds – verifies the following scenario:
  + time is 0, state display is “Stopped”, and button display "Start"
  + press button, expect state display is “Increment”, and button display "Increment"
  + press button 9 times, every press expect time display adding 1
  + wait 2 seconds, press button, expect time display is 11, and state display is “Increment”
  + wait 2 seconds, press button, expect time display is 12, and state display is “Increment”
  + wait 3 seconds, expect time display is 12, state display is “Running”, and button display "Cancel"
* testActivityScenarioStopAlarming – verifies the following scenario:
  + time is 0, state display is “Stopped”, and button display "Start"
  + press button, expect state display is “Increment”, and button display "Increment"
  + press button 9 times, every press expect time display adding 1
  + wait 3 seconds, expect time display is 10, state display is “Running”, and button display "Cancel"
  + wait 10 seconds, expect time display is 0, state display is “Alarming”, and expect button display "Stop"
  + wait 5 seconds, expect state display is “Alarming”
  + press button, expect time display is 0, state display is “Stopped”, and button display "Start"
* testActivityScenarioSetTimerValue performs these checks:
  + time is 0, state display is “Stopped”, and button display "Start"
  + Set EditText value to 10, press button, expect state display is “Running”, button display "Cancel”.
  + wait 10 seconds, expect time display is 0, state display is “Alarming”, and expect button display "Stop"
  + wait 5 seconds, expect state display is “Alarming”
  + press button, expect time display is 0, state display is “Stopped”, and button display "Start"
* testActivityScenarioIncUntilFull performs these checks:  
   display is 0, press button 99 times, expect state display is “Running”, button display "Cancel”, and time display is 99
* testActivityScenarioRotation performs these checks:

checks that the displayed value (state, button, and time) remains the same if the device is rotated

1. ActivityRobolectic Test

gradle testDebug runs TimerActivityRobolectic.java in src/test; it extends and runs AbstractTimerActivityTest.java

1. Model Test - Clock

Test.java in …/model/clock runs tests in DefaultClockModelTest, and DefaultClockModelTest delegate to AbstractClockModelTests:

* Both tests override the onTickListener to be an atomic int
* testStopped – After sleeping 5.5 seconds, that int is still 0
* testRunning – Set value is 5, after sleeping 5.5 seconds the int is 0

1. Model Test - Time

Test.java in …/model/time runs tests in DefaultTimeModelTest, and DefaultTimeModelTest delegate to AbstractTimeModelTests:

* testPreconditions – the model’s runtime == 0
* testIncrementRuntimeOne – time is correct after one second
* testIncrementRuntimeMany – time is 99 after 100 seconds
* testDecrementRuntimeOne – – set time value is 5, time is 4 after one second
* testDecrementRuntimeMany – – set value is 99, time is 0 after 100 seconds
* testDecrementTimeoutOne– timeout is decremented correctly after one second
* testDecrementTimeoutMany– timeout is decremented to 0 after 4 seconds

1. Model Test - Alarm

Test.java in …/model/alarm runs tests in DefaultAlarmModelTest, and DefaultAlarmModelTest delegate to AbstractAlarmModelTests:

* testPlayAlarm – check play status is playing.
* testStopAlarm – check play status is stopped.

1. Model Test - State

Test.java in …/model/state runs tests in DefaultTimerStateMachineTest   
(implemented in AbstractTimerStatemachineTest):

* There is a dependency on an inner mock class that implements TimeModel, ClockModel, AlarmModel, and TimerUIUpdateListener and replaces their methods with simple integer manipulations, etc.
* testPreconditions – tests that the initial state is Stopped
* testScenarioCancelRunning – verifies the following scenario:
  + time is 0, press button 10 times, expect time is 10
  + wait 3 seconds, expect time is 10
  + wait 5 seconds, expect time is “5”
  + press button, expect time is 0
* testActivityScenarioRepeatIncrementWithin3Seconds – verifies the following scenario:
  + time is 0, press button 10 times, expect time is 10
  + wait 2 seconds, press button, expect time is 11
  + wait 2 seconds, press button, expect time is 12
  + wait 2 seconds, expect time is 12
  + wait 2 second, expect time is 11
* testScenarioStopAlarming – verifies the following scenario:
  + time is 0, press button 10 times, expect time is 10
  + wait 3 seconds, expect time is 10
  + wait 10 seconds, expect time is 0
  + wait 5 seconds, expect time is 0
  + press button
  + wait 5 seconds, expect time is 0
* testScenarioSetTimerValue – verifies the following scenario:
  + time is 0, Set time to 10
  + press button
  + wait 10 seconds, expect time is 0
  + wait 5 seconds, expect time is 0
  + press button
  + wait 5 seconds, expect time is 0