

# **디데이 시계**

**oopt Stage 2050, 2060**

**progect Team : 3 team**

**date : 2020/06/08**

**201611188 김동곤**

**201711337 이희광**

**201413146 양영준**

**201614150 김지현**

# 2055. Write Unit Test Code

Unit Test <TimeKeeping> - 137개

```
class TimeKeepingTest {  
  
    @Test  
    void showTimeKeepingtest() {  
        TimeKeep tk = new TimeKeep();  
        tk.showTimeKeeping();  
        Calendar current_time = tk.gettime();  
        assertNotNull(current_time);  
    }  
  
    @Test  
    void getddaytest(){  
        TimeKeep tk = new TimeKeep();  
        assertEquals( expected: null,tk.getdday());  
    }  
  
    @Test  
    void setddaytest(){  
        TimeKeep tk = new TimeKeep();  
        tk.setdday(null);  
        assertEquals( expected: null,tk.getoriginTime());  
        assertEquals( expected: null,tk.getddaymemoTime());  
    }  
  
    @Test  
    void gettimetest(){  
        TimeKeep tk = new TimeKeep();  
        assertEquals( expected: null,tk.gettime());  
    }  
}
```

```

@Test
void getRealTimetest(){
    TimeKeep tk = new TimeKeep();
    assertEquals( expected: null,tk.getRealTime());
}

@Test
void setCurrentTimetest(){
    TimeKeep tk = new TimeKeep();
    tk.showTimeKeeping();
    tk.setCurrentTime();
    assertEquals( expected: 1,tk.get_flag());
    assertEquals( expected: 0,tk.getCursor());
}

@Test
void setActiveFunctiontest(){
    TimeKeep tk = new TimeKeep();
    tk.setActiveFunction();
    assertEquals( expected: 2,tk.get_flag());
}

@Test
void get_flagtest(){
    TimeKeep tk = new TimeKeep();
    assertEquals( expected: null,tk.gettime());
}

```

```

@Test
void getCursorstest(){
    TimeKeep tk = new TimeKeep();
    assertEquals( expected: null,tk.getRealTime());
}

@Test
void addsecondstest(){
    TimeKeep tk = new TimeKeep();
    tk.showTimeKeeping();
    Calendar current_time = (Calendar) tk.gettime().clone();
    System.out.println("currenttime " +current_time.get(Calendar.SECOND));
    tk.addseconds();
    assertEquals( expected: current_time.get(Calendar.SECOND)+1,
        |tk.gettime().get(Calendar.SECOND));
}

@Test
void moveCursorTimertest(){
    TimeKeep tk = new TimeKeep();
    tk.moveCursor_time();
    int cur_cursor = tk.getCursor();
    assertEquals( expected: 0,cur_cursor);
}

```

```

@Test
void plusTime_timetest(){
    TimeKeep tk = new TimeKeep();
    JButton button = new JButton();
    tk.showTimeKeeping();
    button.setText("Button1");//set모드
    tk.work(button);
    Calendar setting_time = (Calendar) tk.gettime().clone();
    button.setText("Button2");//1증가
    tk.work(button);
    button.setText("Button4");//확인
    tk.work(button);
    assertEquals( expected: setting_time.get(Calendar.MONTH)+1,
                 |tk.gettime().get(Calendar.MONTH));
}

@Test
void confirmTimetest(){
    TimeKeep tk = new TimeKeep();
    JButton button = new JButton();
    tk.showTimeKeeping();
    button.setText("Button1");//settime
    tk.work(button);
    button.setText("Button4");//confirm
    tk.work(button);
    assertEquals( expected: 0,tk.get_flag());
    assertEquals( expected: -1,tk.getCursor());
}

```

## Unit Test <Alarm> - 57개

```

@Test
void testCursor() {
    Alarm al = new Alarm();
    JButton btn1 = new JButton( text: "Button1");
    JButton btn4 = new JButton( text: "Button4");
    //초기 커서 위치 테스트
    assertEquals( expected: -1,al.getCursor());

    //커서 위치 이동 테스트
    al.work(btn1);
    assertEquals( expected: 3,al.getCursor());

    al.work(btn1);
    assertEquals( expected: 4,al.getCursor());

    al.work(btn1);
    assertEquals( expected: 5,al.getCursor());

    al.work(btn1);
    assertEquals( expected: 3,al.getCursor());

    //초기 커서 위치 돌아오는지 테스트
    al.work(btn4);
    assertEquals( expected: -1,al.getCursor());
}

```

```

@Test
void testsetAlarm() {
    Alarm al = new Alarm();
    JButton btn1 = new JButton( text: "Button1");
    JButton btn4 = new JButton( text: "Button4");
    //초기 플래그값 확인
    assertEquals( expected: 0,al.get_flag());

    //set 플래그 확인
    al.work(btn1);
    assertEquals( expected: 1,al.get_flag());
    //설정하면 나왔을때.
    al.work(btn4);
    assertEquals( expected: 0,al.get_flag());
}

```

```

@Test
void testconfirmAlarm(){
    Alarm al = new Alarm();
    JButton btn1 = new JButton( text: "Button1");
    JButton btn4 = new JButton( text: "Button4");

    //세팅 모드 진입
    al.work(btn1);
    assertEquals( expected: 1,al.get_flag());
    assertEquals( expected: 3,al.getCursor());

    //confirm 후
    al.work(btn4);
    assertEquals( expected: 0,al.get_flag());
    assertEquals( expected: -1,al.getCursor());
}

```

```

@Test
void testplusTime_alarm(){
    Alarm al = new Alarm();
    JButton btn1 = new JButton( text: "Button1");
    JButton btn2 = new JButton( text: "Button2");
    //세팅 모드 진입
    al.work(btn1);
    assertEquals( expected: 0,al.getAlarm().get(Calendar.HOUR_OF_DAY));
    assertEquals( expected: 0,al.getAlarm().get(Calendar.MINUTE));
    assertEquals( expected: 0,al.getAlarm().get(Calendar.SECOND));
    //시 plus 후
    al.work(btn2);
    assertEquals( expected: 1,al.getAlarm().get(Calendar.HOUR_OF_DAY));
    assertEquals( expected: 0,al.getAlarm().get(Calendar.MINUTE));
    assertEquals( expected: 0,al.getAlarm().get(Calendar.SECOND));

    //분 plus 후
    al.work(btn1);
    al.work(btn2);
    assertEquals( expected: 1,al.getAlarm().get(Calendar.HOUR_OF_DAY));
    assertEquals( expected: 1,al.getAlarm().get(Calendar.MINUTE));
    assertEquals( expected: 0,al.getAlarm().get(Calendar.SECOND));

    //초 plus 후
    al.work(btn1);
    al.work(btn2);
    assertEquals( expected: 1,al.getAlarm().get(Calendar.HOUR_OF_DAY));
    assertEquals( expected: 1,al.getAlarm().get(Calendar.MINUTE));
    assertEquals( expected: 1,al.getAlarm().get(Calendar.SECOND));
}

```

```

@Test
void testresetAlarm(){
    Alarm al = new Alarm();
    JButton btn1 = new JButton( text: "Button1");
    JButton btn4 = new JButton( text: "Button4");

    //세팅 호출 및 저장
    al.work(btn1);
    al.work(btn4);

    //저장 값 not null 확인
    assertNotNull(al.getAlarm());

    //리셋 호출
    al.work(btn4);
    //값 null 확인
    assertNull(al.getAlarm());
}

```

Unit Test <World Time> - 개

```

@Test
void changeCountryTest(){
    WorldTime worldTime = new WorldTime();
    JButton jbutton = new JButton( text: "");
    jbutton.setText("Button1");

    // init
    int cursor = -1;
    assertEquals(cursor, worldTime.getCursor());

    // after changeCountry() Method
    worldTime.work(jbutton);
    cursor = 2;
    assertEquals(cursor, worldTime.getCursor());
}

```

```

@Test
void nextCountryTest(){
    WorldTime worldTime = new WorldTime();
    JButton jbutton = new JButton( text: "");
    jbutton.setText("Button2");

    //init
    Queue<String> countriesQ = new LinkedList();
    countriesQ.offer( e: "LIS");
    countriesQ.offer( e: "LON");
    countriesQ.offer( e: "ROM");
    countriesQ.offer( e: "TYO");
    countriesQ.offer( e: "LAX");
    countriesQ.offer( e: "DEN");
    assertEquals(countriesQ, worldTime.get_countriesQ());

    String country = "SEL";
    assertEquals(country, worldTime.get_key());

    //after nextCountry() Method
    countriesQ.offer(country);
    country = countriesQ.poll();
    jbutton.setText("Button1");
    worldTime.work(jbutton);
    jbutton.setText("Button2");
    worldTime.work(jbutton);
    assertEquals(countriesQ, worldTime.get_countriesQ());
    assertEquals(country, worldTime.get_key());
}

```

```
@Test
void changeModeTest(){
    WorldTime worldTime = new WorldTime();
    JButton jbutton = new JButton( text: "");
    jbutton.setText("Button3");

    worldTime.work(jbutton);
    Queue<String> countriesQ = new LinkedList();
    countriesQ.offer( e: "LIS");
    countriesQ.offer( e: "LON");
    countriesQ.offer( e: "ROM");
    countriesQ.offer( e: "TYO");
    countriesQ.offer( e: "LAX");
    countriesQ.offer( e: "DEN");
    assertEquals(countriesQ, worldTime.get_countriesQ());

    String country = "SEL";
    assertEquals(country, worldTime.get_key());

    int cursor = -1;
    assertEquals(cursor, worldTime.getCursor());
}
```



```

@Test
void confirmCountryTest(){
    WorldTime worldTime = new WorldTime();
    JButton jbutton = new JButton( text: "");

    //changeCountry()
    jbutton.setText("Button1");
    worldTime.work(jbutton);

    //nexCountry()
    jbutton.setText("Button2");
    worldTime.work(jbutton);

    //confirmCountry
    jbutton.setText("Button4");
    worldTime.work(jbutton);

    int cursor = -1;
    assertEquals(cursor, worldTime.getCursor());

    Queue<String> countriesQ = new LinkedList();
    countriesQ.offer( e: "LON");
    countriesQ.offer( e: "ROM");
    countriesQ.offer( e: "TYO");
    countriesQ.offer( e: "LAX");
    countriesQ.offer( e: "DEN");
    countriesQ.offer( e: "SEL");
    assertEquals(countriesQ, worldTime.get_countriesQ_temp());

    String country = "LIS";
    assertEquals(country, worldTime.get_key());
}

```

```

@Test
void get_keyTest(){
    WorldTime worldTime = new WorldTime();
    String initCountry = "SEL";
    assertEquals(initCountry, worldTime.get_key());
}

```

```

@Test
void get_valueTest(){
    WorldTime worldTime = new WorldTime();
    Map<String, String> countries = new HashMap<>();
    countries.put("SEL", "Asia/Seoul");
    String value = countries.get("SEL");

    assertEquals(value, worldTime.get_value());
}

```

```

@Test
void getCursorTest(){
    WorldTime worldTime = new WorldTime();
    int cursor = -1;
    assertEquals(cursor, worldTime.getCursor());
}

```

## Unit Test <StopWatch> - 7개

```

class StopwatchTest {

    @Test
    void resetStopWatchTest() {
        Watch watch=new Watch();
        Stopwatch st=new Stopwatch();

        st.setStart(12);
        st.setEnd(22);
        st.setElapse(10);
        st.setElapsePrevious(1);
        st.setOn(true);

        Calendar cal=Calendar.getInstance();
        cal.set(Calendar.HOUR, 9);
        cal.set(Calendar.MINUTE, 2);
        cal.set(Calendar.SECOND, 5);

        st.setCal(cal);

        Calendar callap=Calendar.getInstance();
        callap.set(Calendar.HOUR, 5);
        callap.set(Calendar.MINUTE, 2);
        callap.set(Calendar.SECOND, 2);
    }
}

```

```

st.setCallLap(callLap);
assertEquals( expected: 12, st.getStart());
assertEquals( expected: 22, st.getEnd());
assertEquals( expected: 10, st.getElapse());
assertEquals( expected: 1, st.getElapsePrevious());
assertEquals( expected: true, st.isOn());

assertEquals( expected: 9, st.getCal().get(Calendar.HOUR));
assertEquals( expected: 2, st.getCal().get(Calendar.MINUTE));
assertEquals( expected: 5, st.getCal().get(Calendar.SECOND));
assertEquals( expected: 5, st.getCallLap().get(Calendar.HOUR));
assertEquals( expected: 2, st.getCallLap().get(Calendar.MINUTE));
assertEquals( expected: 2, st.getCallLap().get(Calendar.SECOND));

```

```

JButton button=new JButton();
button.setText("Button4");

```

```

st.work(button);
st.work(button);

```

```

assertEquals( expected: 0, st.getStart());
assertEquals( expected: 0, st.getEnd());
assertEquals( expected: 0, st.getElapse());
assertEquals( expected: 0, st.getElapsePrevious());
assertEquals( expected: false, st.isOn());

```

```

assertEquals( expected: 0, st.getCal().get(Calendar.HOUR));
assertEquals( expected: 0, st.getCal().get(Calendar.MINUTE));
assertEquals( expected: 0, st.getCal().get(Calendar.SECOND));
assertEquals( expected: 0, st.getCallLap().get(Calendar.HOUR));
assertEquals( expected: 0, st.getCallLap().get(Calendar.MINUTE));
assertEquals( expected: 0, st.getCallLap().get(Calendar.SECOND));

```

```

}

```

```

@Test

```

```

void startStopWatchTest(){

```

```

    Watch watch=new Watch();
    Stopwatch st=new Stopwatch();

```

```

    st.setOn(false);

```

```

    JButton button=new JButton();
    button.setText("Button2");
    st.work(button);

```

```

    assertEquals( expected: true, st.isOn());

```

```

}

```

```

@Test
void stopStopWatchTest(){
    Watch watch=new Watch();
    Stopwatch st=new Stopwatch();

    st.setOn(true);

    JButton button=new JButton();
    button.setText("Button2");
    st.work(button);

    assertEquals( expected: false,st.isOn());
}

```

```

@Test
void stopStopWatchTest(){
    Watch watch=new Watch();
    Stopwatch st=new Stopwatch();

    st.setOn(true);

    JButton button=new JButton();
    button.setText("Button2");
    st.work(button);

    assertEquals( expected: false,st.isOn());
}

```

```

@Test
void storeLapTimeTest(){
    Watch watch=new Watch();
    Stopwatch st=new Stopwatch();

    st.setOn(true);

    st.setElapse(36000);

    JButton button=new JButton();
    button.setText("Button1");
    st.work(button);

    assertEquals( expected: 10,st.getCallLap().get(Calendar.HOUR));
    assertEquals( expected: 0,st.getCallLap().get(Calendar.MINUTE));
    assertEquals( expected: 0,st.getCallLap().get(Calendar.SECOND));
}

```

```

void getStopWatchTestInON(){
    Watch watch=new Watch();
    Stopwatch st=new Stopwatch();

    JButton button=new JButton();
    button.setText("Button4");
    st.work(button);

    st.setOn(true);

    //    this.end=calculateTime(this.end,this.cal);
    //    this.elapsed=this.elapsedPrevious+(this.end-this.start);
    //    splitElapsed(elapsed,this.cal);

    st.setElapsedPrevious(1800);
    st.setStart(0);
    st.setEnd(1800);
    st.calculateTime(st.getEnd(),st.getCal());
    st.setElapsed(st.getElapsedPrevious()+(st.getEnd()-st.getStart()));
    st.splitElapsed(st.getElapsed(),st.getCal());

    assertEquals( expected: 1,st.getCal().get(Calendar.HOUR));
    assertEquals( expected: 0,st.getCal().get(Calendar.MINUTE));
    assertEquals( expected: 0,st.getCal().get(Calendar.SECOND));|
}

```

```

@Test
void getStopWatchTestInOFF(){

    Watch watch=new Watch();
    Stopwatch st=new Stopwatch();

    st.setOn(false);

    Calendar cal=st.getStopWatch();

    assertEquals( expected: 0,cal.get(Calendar.HOUR));
    assertEquals( expected: 0,cal.get(Calendar.MINUTE));
    assertEquals( expected: 0,cal.get(Calendar.SECOND));
}

```

## Unit Test <D-day> - 6기

```

class DdayTest {

    @Test
    void setDdayTest() {
        Watch watch=new Watch();
        Dday d=new Dday();

        d.setFlag_set(0);
        d.setSetting_page(null);

        JButton button=new JButton();
        button.setText("Button1");
        d.work(button);

        assertEquals( expected: 1,d.getFlag_set());
        assertEquals( expected: 0,d.getCur_cursor());

        assertEquals( expected: "AAA",d.getSetting_page().get_memo());
    }
}

```

```

@Test
void moveCursor_Dday_Test(){
    Watch watch=new Watch();
    Dday d=new Dday();

    //if && if
    d.setCur_cursor(2);

    d.setString_cur(2);

    d.setFlag_set(1);
    d.setSetting_page(null);

    JButton button=new JButton();
    button.setText("Button1");
    d.work(button);

    assertEquals( expected: 0,d.getString_cur());
    assertEquals( expected: 0,d.getCur_cursor());
    //if&&else
    d.setCur_cursor(2);
    d.setString_cur(1);
    d.setFlag_set(1);
    d.setSetting_page(null);

    d.work(button);

    assertEquals( expected: 2,d.getString_cur());
    assertEquals( expected: 2,d.getCur_cursor());

    d.setString_cur(0);

    //else

    d.setCur_cursor(7);
    d.setFlag_set(1);
    d.setSetting_page(null);

    d.work(button);

    assertEquals( expected: 1,d.getCur_cursor());
}

```

```
void showNextDdayTest(){
    Watch watch=new Watch();
    Dday d=new Dday();

    d.setFlag_set(0);
    d.setSetting_page(null);

    JButton button=new JButton();
    button.setText("Button1");
    d.work(button);

    dday_data current_page;

    current_page = new dday_data();

    current_page.set_memo(d.getSetting_page().get_memo());

    d.setCurrent_page(current_page);

    button=new JButton();
    button.setText("Button2");
    d.work(button);

    assertEquals( expected: "AAA",d.getCurrent_page().get_memo());
}
}
```

```
@Test
void plusDayTest(){
    Watch watch=new Watch();
    Dday d=new Dday();

    d.setFlag_set(1);

    d.setCur_cursor(0);
    dday_data setting_page=new dday_data();
    Calendar cal= Calendar.getInstance();
    cal.set(Calendar.MONTH,10);

    setting_page.set_cal(cal);
    setting_page.set_memo("aaa");

    d.setSetting_page(setting_page);

    JButton button=new JButton();
    button.setText("Button2");
    d.work(button);

    assertEquals( expected: 11,d.getSetting_page().get_cal().get(Calendar.MONTH));
}
}
```

```
@Test
void deleteDdayTest(){

    Watch watch=new Watch();
    Dday d=new Dday();

    d.setFlag_set(0);

    dday_data current_page;

    current_page = new dday_data();

    current_page.set_memo("111");

    d.setCurrent_page(current_page);
    d.getDdayQ().clear();

    JButton button=new JButton();
    button.setText("Button4");
    d.work(button);

    assertEquals( expected: null,d.getCurrent_page());
}
}
```

```
@Test
void confirmDdayTest(){

    Watch watch=new Watch();
    Dday d=new Dday();

    d.setFlag_set(1);

    dday_data setting_page=new dday_data();
    Calendar cal= Calendar.getInstance();
    cal.set(Calendar.MONTH,10);

    setting_page.set_cal(cal);
    setting_page.set_memo("no1");
    d.setSetting_page(setting_page);

    JButton button=new JButton();
    button.setText("Button4");
    d.work(button);

    assertEquals( expected: "no1",d.getCurrent_page().get_memo());
}
```



## Unit Test <Timer> - 11개

```
class TimerTest {  
  
    @Test  
    void getCursorTest() {  
        mTimer mtimer = new mTimer();  
        int cursor = mtimer.getCursor();  
        assertEquals( expected: -1, cursor);  
    }  
  
    @Test  
    void getFlagTest(){  
        mTimer mtimer = new mTimer();  
        int flag = mtimer.get_flag();  
        assertEquals( expected: 0, flag);  
    }  
  
    @Test  
    void getPauseFlagTest(){  
        mTimer mtimer = new mTimer();  
        int flag = mtimer.getPauseFlag();  
        assertEquals( expected: 0, flag);  
    }  
  
    @Test  
    void getTimerTimeTest(){  
        mTimer mtimer = new mTimer();  
        Calendar c = mtimer.getTimerTime();  
        assertEquals( expected: null, c);  
    }  
}
```

```
    @Test  
    void setTimerTest(){  
        mTimer mtimer = new mTimer();  
        JButton button = new JButton();  
        button.setText("Button1");  
        mtimer.work(button);  
        int flag = mtimer.get_flag();  
        assertEquals( expected: 1, flag);  
    }  
  
    @Test  
    void moveCursorTimerTest(){  
        mTimer mtimer = new mTimer();  
        mtimer.moveCursor_timer();  
        int cur_cursor = mtimer.getCursor();  
        assertEquals( expected: 3, cur_cursor);  
    }  
}
```

```

@Test
void plusTimertest(){
    mTimer mtimer = new mTimer();
    JButton button = new JButton();
    button.setText("Button1");//set모드
    mtimer.work(button);
    button.setText("Button2");//1증가
    mtimer.work(button);
    button.setText("Button4");//확인
    mtimer.work(button);
    Calendar pre_time = mtimer.getTimer();
    assertEquals( expected: 1,pre_time.get(Calendar.HOUR_OF_DAY));
    assertEquals( expected: 0,pre_time.get(Calendar.MINUTE));
    assertEquals( expected: 0,pre_time.get(Calendar.SECOND));
}

@Test
void confirmTimertest(){
    mTimer mtimer = new mTimer();
    JButton button = new JButton();
    button.setText("Button1");//set모드
    mtimer.work(button);
    button.setText("Button4");//확인
    mtimer.work(button);
    int flag_set = mtimer.get_flag();
    int cur_cursor = mtimer.getCursor();
    assertEquals( expected: 0,flag_set);
    assertEquals( expected: -1,cur_cursor);
}

```

```

@Test
void stopTimertest(){
    mTimer mtimer = new mTimer();
    JButton button = new JButton();
    button.setText("Button1");//set모드
    mtimer.work(button);
    button.setText("Button2");//1증가
    mtimer.work(button);
    button.setText("Button4");//확인
    mtimer.work(button);
    button.setText("Button4");//stop
    mtimer.work(button);
    Calendar pre_time = mtimer.getTimer();
    assertEquals( expected: null,pre_time);
}

@Test
void startTimertest(){
    mTimer mtimer = new mTimer();
    JButton button = new JButton();
    button.setText("Button1");//set모드
    mtimer.work(button);
    button.setText("Button2");//1증가
    mtimer.work(button);
    button.setText("Button4");//확인
    mtimer.work(button);
    button.setText("Button2");//start
    mtimer.work(button);
    int flag = mtimer.get_flag_sp();
    assertEquals( expected: 1,flag);
}

```

```

@Test
void pauseTimertest(){
    mTimer mtimer = new mTimer();
    JButton button = new JButton();
    button.setText("Button1");//set모드
    mtimer.work(button);
    button.setText("Button2");//1증가
    mtimer.work(button);
    button.setText("Button4");//확인
    mtimer.work(button);
    button.setText("Button2");//start
    mtimer.work(button);
    button.setText("Button2");//pause
    mtimer.work(button);
    int flag = mtimer.get_flag_sp();
    assertEquals( expected: 0, flag);
}

```

## Unit Test <Function Activator> - 87개

```

@Test
void nextActivateFunctionTest() {
    Queue<Integer> modeQ = new LinkedList<>();
    modeQ.offer(watch_Type.ALARM.ordinal());
    modeQ.offer(watch_Type.WORLDTIME.ordinal());
    modeQ.offer(watch_Type.STOPWATCH.ordinal());
    FunctionActivator functionActivator = new FunctionActivator(modeQ);

    JButton jButton = new JButton( text: "");
    jButton.setText("Button1");

    //init
    //position == 0
    int position = 0;
    assertEquals(position, functionActivator.get_position());

    //position 0 -> 1
    functionActivator.work(jButton);
    position = 1;
    assertEquals(position, functionActivator.get_position());

    // position 5 -> 0
    functionActivator.work(jButton);
    functionActivator.work(jButton);
    functionActivator.work(jButton);
    functionActivator.work(jButton);

    position = 0;
    assertEquals(position, functionActivator.get_position());
}

```

```

@Test
void onOffFunctionTest() {
    Queue<Integer> modeQ = new LinkedList<>();
    modeQ.offer(watch_Type.ALARM.ordinal());
    modeQ.offer(watch_Type.WORLDTIME.ordinal());
    modeQ.offer(watch_Type.STOPWATCH.ordinal());
    FunctionActivator functionActivator = new FunctionActivator(modeQ);

    JButton jButton = new JButton( text: "");
    jButton.setText("Button2");

    int position = 0;
    boolean active_function = true;
    assertEquals(active_function, functionActivator.get_active_function(position));

    functionActivator.work(jButton);
    active_function = false;
    assertEquals(active_function, functionActivator.get_active_function(position));
}

```

```

@Test
void confirmActiveTest() {
    Queue<Integer> modeQ = new LinkedList<>();
    modeQ.offer(watch_Type.ALARM.ordinal());
    modeQ.offer(watch_Type.WORLDTIME.ordinal());
    modeQ.offer(watch_Type.STOPWATCH.ordinal());
    FunctionActivator functionActivator = new FunctionActivator(modeQ);
    JButton jButton = new JButton( text: "");

    //nextActivateFunction()
    //position 0 -> 1
    int position = 1;
    jButton.setText("Button1");
    functionActivator.work(jButton);
    assertEquals(position, functionActivator.get_position());

    //onOffFunction()
    //active_count 3 -> 2
    int active_count = 2;
    jButton.setText("Button2");
    functionActivator.work(jButton);
    assertEquals(active_count, functionActivator.get_active_count());

    //confirmActive - not confirm
    //position 1 -> 1
    position = 1;
    jButton.setText("Button4");
    functionActivator.work(jButton);
    assertEquals(active_count, functionActivator.get_active_count());
}

```

```

    assertEquals(position, functionActivator.get_position());

    //onOffFunction()
    //active_count 2 -> 3
    active_count = 3;
    jButton.setText("Button2");
    functionActivator.work(jButton);
    assertEquals(active_count, functionActivator.get_active_count());

    //confirmActive - confirm
    //position 1 -> 0
    position = 0;
    jButton.setText("Button4");
    functionActivator.work(jButton);
    assertEquals(active_count, functionActivator.get_active_count());
    assertEquals(position, functionActivator.get_position());
}

```

```

@Test
void get_activeTest() {
    Queue<Integer> modeQ = new LinkedList<>();
    modeQ.offer(watch_Type.ALARM.ordinal());
    modeQ.offer(watch_Type.WORLDTIME.ordinal());
    modeQ.offer(watch_Type.STOPWATCH.ordinal());
    FunctionActivator functionActivator = new FunctionActivator(modeQ);

    boolean active_function = true;
    int position = 0;
    assertEquals(active_function, functionActivator.get_active(position));
}

```

```

@Test
void get_active_nameTest() {
    Queue<Integer> modeQ = new LinkedList<>();
    modeQ.offer(watch_Type.ALARM.ordinal());
    modeQ.offer(watch_Type.WORLDTIME.ordinal());
    modeQ.offer(watch_Type.STOPWATCH.ordinal());
    FunctionActivator functionActivator = new FunctionActivator(modeQ);

    String active_function_name = "arm";
    int position = 0;
    assertEquals(active_function_name, functionActivator.get_active_name(position));
}

```

```

@Test
void get_active_countTest() {
    Queue<Integer> modeQ = new LinkedList<>();
    modeQ.offer(watch_Type.ALARM.ordinal());
    modeQ.offer(watch_Type.WORLDTIME.ordinal());
    modeQ.offer(watch_Type.STOPWATCH.ordinal());
    FunctionActivator functionActivator = new FunctionActivator(modeQ);

    int active_count = 3;
    assertEquals(active_count, functionActivator.get_active_count());
}

```

```

@Test
void get_modeQTest() {
    Queue<Integer> modeQ = new LinkedList<>();
    modeQ.offer(watch_Type.ALARM.ordinal());
    modeQ.offer(watch_Type.WORLDTIME.ordinal());
    modeQ.offer(watch_Type.STOPWATCH.ordinal());
    FunctionActivator functionActivator = new FunctionActivator(modeQ);

    assertEquals(modeQ, functionActivator.get_modeQ());
}
@Test
void get_positionTest() {
    Queue<Integer> modeQ = new LinkedList<>();
    modeQ.offer(watch_Type.ALARM.ordinal());
    modeQ.offer(watch_Type.WORLDTIME.ordinal());
    modeQ.offer(watch_Type.STOPWATCH.ordinal());
    FunctionActivator functionActivator = new FunctionActivator(modeQ);

    int position = 0;
    assertEquals(position, functionActivator.get_position());
}

```

#### Unit Test <Buzzer> - 47H

```

@Test
void testgetbuzzer() {
    Buzzer bz = new Buzzer();
    int result = bz.getbuzzer();
    assertEquals( expected: 0, result);
}
@Test
void testonBuzzer(){
    Buzzer bz = new Buzzer();
    bz.onBuzzer( type: 1);
    assertEquals( expected: 1, bz.getbuzzer());
    assertEquals( expected: 15, bz.getLeftTime());

    bz.onBuzzer( type: 2);
    assertEquals( expected: 2, bz.getbuzzer());
    assertEquals( expected: 15, bz.getLeftTime());
}
@Test
void testsubTimeBuzzer(){
    Buzzer bz = new Buzzer();
    bz.onBuzzer( type: 1);
    assertEquals( expected: 15, bz.getLeftTime());

    bz.subTimeBuzzer();
    assertEquals( expected: 14, bz.getLeftTime());
}

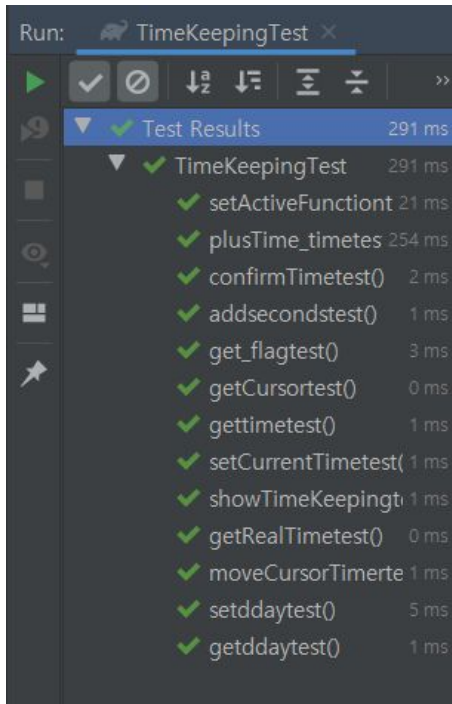
```

```
@Test
void turnOffBuzzer(){
    Buzzer bz = new Buzzer();
    bz.onBuzzer( type: 1);
    assertEquals( expected: 1, bz.getbuzzer());

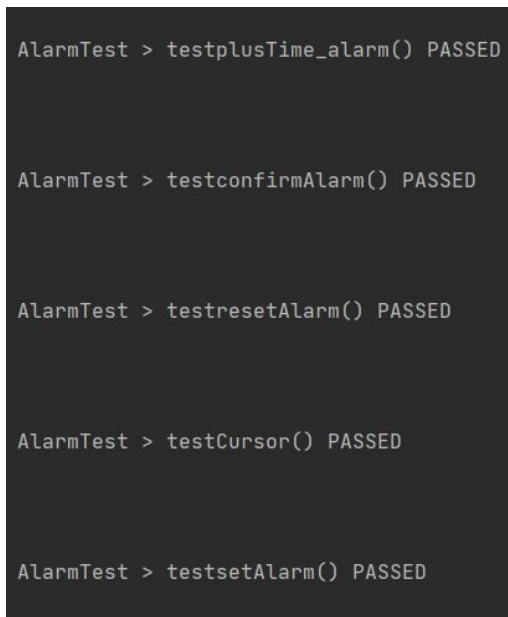
    bz.turnOffBuzzer();
    assertEquals( expected: 0, bz.getbuzzer());
}
```

# 2061. Unit Testing

## TimeKeep



## Alarm



## WorldTime



```
WorldTimeTest > changeModeTest() PASSED

WorldTimeTest > get_keyTest() PASSED

WorldTimeTest > confirmCountryTest() PASSED

WorldTimeTest > get_valueTest() PASSED

WorldTimeTest > getCursorTest() PASSED

WorldTimeTest > changeCountryTest() PASSED

WorldTimeTest > nextCountryTest() PASSED
```

## StopWatch

Test Name	Duration
StopWatchTest	3 s 149 ms
getStopWatchTestInOFF()	1 s 953 ms
stopStopWatchTest()	148 ms
resetStopWatchTest()	250 ms
startStopWatchTest()	192 ms
storeLapTimeTest()	381 ms
getStopWatchTestInON()	225 ms

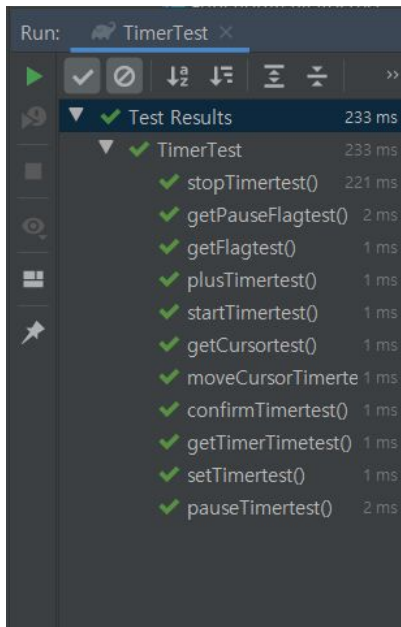
## Dday

Test Name	Duration
DdayTest	2 s 454 ms
confirmDdayTest()	1 s 687 ms
moveCursor_Dday_Test()	131 ms
showNextDdayTest()	124 ms
plusDayTest()	190 ms
setDdayTest()	159 ms
de	163 ms

Tests passed: 6

Run only tests that failed/crashed after last run; press Shift to choose Run/Debug mode

## mTimer



## FunctionActivator

```
FunctionActivatorTest > onOffFunctionTest() PASSED
```

```
FunctionActivatorTest > get_active_nameTest() PASSED
```

```
FunctionActivatorTest > get_modeQTest() PASSED
```

```
FunctionActivatorTest > get_active_countTest() PASSED
```

```
FunctionActivatorTest > get_positionTest() PASSED
```

```
FunctionActivatorTest > get_activeTest() PASSED
```

```
FunctionActivatorTest > confirmActiveTest() PASSED
```

```
FunctionActivatorTest > nextActivateFunctionTest() PASSED
```

## Buzzer

```
BuzzerTest > turnOffBuzzer() PASSED
```

```
BuzzerTest > testgetbuzzer() PASSED
```

```
BuzzerTest > testonBuzzer() PASSED
```

```
BuzzerTest > testsubTimeBuzzer() PASSED
```

## 2062 System Testing

Test No.	Test 항목	Description	Function
1	Show Time Test	현재 시간을 월, 일, 요일(디데이 라벨), 시, 분, 초, 연도 순서대로 화면에 표시하는지 test	showTimeKeeping
2	Show Time Test	사용자가 설정한 시간을 월, 일, 요일(디데이 라벨), 시, 분, 초, 연도 순서대로 화면에 표시하는지 test	showTimeKeeping
3	Show Time Test	현재 시간이 1초마다 1초씩 증가하는지 test	showTimeKeeping
4	Set Time Test	현재 시간 모드에서 setTime버튼을 눌렀을 때 setTime으로 진입되는지 test	setCurrentTime
5	Set Time Test	현재 시간 모드의 setTime에서 (+1)버튼을 눌렀을 때 월, 일, 요일(디데이 라벨), 시, 분, 초, 연도 중 커서가 위치한 값이 1씩 증가하는지 test	setCurrentTime
6	Set Time Test	현재 시간 모드의 setTime에서 next버튼을 눌렀을 때 커서가 다음 textview로 이동하는지 test	setCurrentTime
7	Set Alarm Test	알람 모드에서 setAlarm 버튼을 눌렀을때 setAlarm으로 진입되는지 test	setAlarm
8	Set Alarm Test	setAlarm 모드에서 버튼1를 누르면서 커서가 잘 이동하는지 test	setAlarm
9	Set Alarm Test	setAlarm 모드에서 버튼 2를 누르면 1씩 잘 증가되는지 test	setAlarm
10	Set Alarm test	setAlarm 모드에서 버튼4를 눌렀을때 데이터가 잘 저장 되었는지 테스트	setAlarm
11	Reset Alarm test	알람 모드에서 버튼4를 눌렀을때 기존 알람 데이터가 잘 지워지고 OFF로 표시 되는지 확인	ResetAlarm
12	Show Alarm Test	알람 화면이 올바르게 표시	showAlarm
13	Sound Buzzer Test	부저가 잘 울리고 시간이 설정 되는지 테스트	OnBuzzer
14	Turn Off Buzzer Test	함수를 호출하면 부저가 잘 종료되는지 테스트	TurnOffBuzzer
15	Buzzer Timeout	부저의 남은 시간이 잘 감소되고 0초에 부저가 종료되는지 테스트	getLeftTime
16	Watch World Time Test	나라를 변경하고자 버튼을 눌렀을 때, 나라를 출력해주는 화면이 1초마다 깜빡거리는지 Test	WatchWorldTime
17	Watch World Time Test	세계 시간을 월, 일, 지역, 시, 분, 초, 연도 순서대로 화면에 보여주는지 Test	WatchWorldTime, ChangeMode

		나라를 설정하던 도중 버튼 3을 눌러 모드를 변경 하였을 때, 저장한 나라와 시간을 화면에 보여주는지 Test	
18	Watch World Time Test	나라를 설정하던 도중 버튼 4를 눌러 저장하였을 때, 저장한 나라와 시간을 화면에 보여주는지 Test	confirmCountry
19	Change Country Test	큐에 저장된 나라 순서대로 월, 일, 지역, 시, 분, 초, 연도 순서대로 화면에 보여주는지 Test	changeCountry
20	Reset Stopwatch test	스톱 워치와 관련된 변수들이 초기화 되는지 test	resetStopWatch
21	Start Stopwatch test	스톱 워치가 시작되는지 test	startStopWatch
22	Stop Stopwatch test	스톱 워치가 멈추는지 test	stopStopWatch
23	StoreLapTime test	랩 타임이 저장되는지, 랩 타임이 제대로 계산되는지 test	storeLapTime
24	Show stopWatch test	스톱 워치의 경과시간을 제대로 계산하여 화면에 표시 하는지 test	getStopwatch
25	Set Dday test	디데이와 디데이 메모가 설정되는지 test	setDday
26	Set Dday test	디데이 설정 시 커서가 작동하는지 test	setDday
27	Set Dday test	디데이 설정 시 시간이 증가하는지 test	setDday
28	Set Dday test	디데이가 설정 완료되는지 test	setDday
29	Show Next Dday test	다음 디데이가 출력되는지 test	showNextDday
30	Delete Dday test	디데이가 삭제되는지 test	deleteDday
31	Show Dday	타임 키프 모드 of 요일 textView에 현재 날짜로 설정된 디데이가 표시되는지 test	showDday
32	Show Timer Test	타이머의 시, 분, 초가 초기값대로 화면에 표시되는지 test	getTimer
33	Show Timer Test	타이머의 시, 분, 초가 설정한 시간대로 화면에 표시되는지 test	getTimer
34	Set Timer Test	타이머 모드에서 setTimer버튼을 눌렀을 때 setTimer로 진입되는지 test	setTimer
35	Set Timer Test	타이머 모드의 setTimer에서 (+1)버튼을 눌렀을 때 시, 분, 초 중 커서가 위치한 값이 1씩 증가하는지 test	setTimer
36	Set Timer Test	타이머 모드의 setTimer에서 next버튼을 눌렀을 때 커서가 다음	setTimer

		textView로 이동하는지 test	
37	Start Timer Test	타이머가 지정된 시간으로부터 1초씩 감소하는지 test	startTimer
38	Pause Timer Test	타이머가 시작 된 상태에서 Pause에 해당하는 버튼을 눌렀을 때 시, 분, 초가 일시정지하는지 test	pauseTimer
39	Stop Timer Test	타이머가 시작, 혹은 Pause된 상태에서 Stop에 해당하는 버튼을 눌렀을 때 0시0분0초로 초기화되는지 test	stopTimer
40	Set Active Function Test	on/off, 기능 이름을 순서대로 화면에 보여주는지 Test 버튼1을 눌러 다음 기능으로 넘어가면 다음 기능에 대한 on/off와 기능 이름을 순서대로 화면에 보여주는지 Test	setActiveFunction
41	Set Active Function Test	현재 기능에서 버튼 2를 눌러 on -> off 혹은 off -> on으로 전환할 때, 전환이 되는지 test 전환하고 나서 화면에 보여주는지 test	setActiveFunction
42	Set Active Function Test	3개의 기능을 활성화, 2개의 기능이 비활성화 되었을 때, 버튼 4를 누르면 저장이 되고 TimeKeeping 모드로 넘어가는지 Test 활성화된 기능이 3개가 아닐 때, 버튼 4를 누르면TimeKeeping 모드로 넘어가는지 Test	setActiveFunction
43	Change Mode Test	mode에 해당하는 버튼을 눌렀을 때 다음 모드로 변경되는지 test	changeMode

## 2063 Testing Traceability Analysis

System Function		Essential Use Case	Operation in sequence diagram
showTimeKeeping		Show Current Time	O1, O41, O42
setCurrentTime		Set Current Time	O1, O2, O3, O4
setAlarm		Set Alarm When I Want	O9, O10, O11,O12
OnBuzzer		Sound Buzzer	O6
TurnOffBuzzer		Turn Off Buzzer	O5
Reset Alarm		Reset Alarm	O14
Show Alarm		Show Alarm	O44

getLeftTime		Buzzer Timeout	O7, O8
WatchWorldTime		Watch World Time	O46, O47
changeCountry		Change Country	O34, O35
StartStopWatch		Start StopWatch	O30
pauseStopWatch		Pause StopWatch	O31
resetStopWatch		Reset StopWatch	O32
getStopWatch		ShowStopWatch	O33
getLaptime		WatchLapTime	O28
storeLapTime		StoreLapTime	O29
setDday		Set D-day	O22, O23, O25, O27
showDday		Show D-day	O45
deleteDday		Delete D-day	O26
showNextDday		Show Next D-day Calendar	O24
startTimer		Start Timer	O15, O40
setTimer		Set Timer	O16, O19, O20, O21
pauseTimer		Pause Timer	O17
stopTimer		Stop Timer	O18
getTimer		Show Timer	O40
setActiveFunction		Set Active Function	O48, O49
Change Mode		Change Mode	O43

O - number	Operation in sequence diagram
O1	setCurrentTime()
O2	plusTime_time()
O3	moveCursor_time()
O4	confirmTime()
O5	turnOffBuzzer()
O6	onBuzzer()
O7	getleftTime()
O8	subTimeBuzzer()
O9	moveCursor_alarm()
O10	plusTime_alarm()
O11	confirmAlarm()
O12	setAlarm()
O13	getAlarm()
O14	resetAlarm()
O15	startTimer()
O16	setTimer()
O17	pauseTimer()
O18	stopTimer()
O19	confirmTimer()
O20	moveCursor_timer()
O21	plusTimer()
O22	setDday()
O23	moveCursor_Dday()
O24	showNextDday()
O25	plusDay()
O26	deleteDday()



O27	confirmDday()
O28	getLapTime()
O29	storeLapTime()
O30	startStopWatch()
O31	stopStopWatch()
O32	resetStopWatch()
O33	getStopWatch()
O34	changeCountry()
O35	confirmCountry()
O36	nextActivateFunction()
O37	confirmActive()
O38	onOffFunction()
O39	get_active()
O40	getTimer()
O41	showTimerKeeping()
O42	gettime()
O43	changeMode()
O44	showAlarm()
O45	showDday()
O46	get_key()
O47	get_value()
O48	setActivateFunction()

Operation in sequence diagram	M-Link
setCurrentTime()	M2, M68, M71
plusTime_time()	M8, M68, M71
moveCursor_time()	M6, M68, M71

confirmTime()	M9, M68, M71
turnOffBuzzer()	M19, M24, M68
onBuzzer()	M20
getleftTime()	M19
subTimeBuzzer()	M21
moveCursor_alarm()	M16, M17, M68, M71
plusTime_alarm()	M14, M68, M71
confirmAlarm()	M13, M68, M71
setAlarm()	M12, M68, M71
getAlarm()	M18
resetAlarm()	M15, M68, M71
startTimer()	M50, M68, M71
setTimer()	M55, M68, M71
pauseTimer()	M54, M68, M71
stopTimer()	M53, M68, M71
confirmTimer()	M58, M68, M71
moveCursor_timer()	M56, M68, M71
plusTimer()	M57, M68, M71
setDday()	M10, M68, M71
moveCursor_Dday()	M45, M68, M71
showNextDday()	M44, M68, M71
plusDday()	M46, M68, M71
deleteDday()	M43, M68, M71
confirmDday()	M47, M68, M71
getLapTime()	M37, M68, M71
storeLapTime()	M36, M68, M71
startStopWatch()	M30,M33
stopStopWatch()	M31, M68, M71

resetStopWatch()	M35, M68, M71
getStopWatch()	M32,M33,M34
changeCountry()	M25, M68, M71
confirmCountry()	M29, M68, M71
nextActivateFunction()	M60, M68, M71
confirmActive()	M67, M68, M71
onOffFunction	M66, M68, M71
get_active()	M64
getTimer()	M51
showTimerKeeping()	M1, M5,
gettime()	M5,
changeMode()	M70, M68
showAlarm()	M11
showDday()	M42,
get_key()	M28
get_value()	M26
setActivateFunction()	M59, M68, M71

MID	Method	Unit Test	Class
M1	showTimeKeeping()	showTimeKeepingtest	TimeKeeping
M2	setCurrentTime()	setCurrentTimetest	
M3	addseconds()	addsecondstest	

M4	getdday()	getddaytest	
M5	gettime()	gettimetest	
M6	moveCursortime()	moveCursorTimeTest	
M7	getCursor()	getCursortest	
M8	plusTimetime()	plustimetest	
M9	confirmTime()	confirmTimeTest	
M10	setdday()	setddaytest	
M11	showAlarm()		Alarm
M12	setAlarm()	testsetAlarm	
M13	confirmAlarm()	testcomfirmAlarm	
M14	plusTimealarm()	testplusTime_alarm	
M15	resetAlarm()	testResetAlarm	
M16	moveCursorAlarm()	testCursor	
M17	getCursor()	testCursor	
M18	getAlarm()	testplusTime_alarm	
M19	getLeftTime()	testonBuzzer	Buzzer
M20	onBuzzer()	testonBuzzer	
M21	subTimeBuzzer()	testsubTimeBuzzer	
M22	soundBuzzer()		
M23	turnoffBuzzer()	turnOffBuzzer	
M24	watchWorldTime()		WorldTime
M25	changeCountry()	changeCountrytest	
M26	getValue()	get_valuetest	
M27	nextCountry()	nextCountryTest	
M28	getKey()	get_keytest	
M29	confirmCountry()	confirmCountryTest	
M30	startStopWatch()	startStopWatchtest	StopWatch
M31	stopStopWatch()	stopStopWatchtest	

M32	getStopWatch()	getStopWatchtestinoff getStopWatchtestinon	
M33	calculatorTime()	startStopWatchtest	
M34	splitElapse()	getStopWatchtestinon	
M35	resetStopWatch()	resetStopWatchTest	
M36	storeLapTime()	storeLaptimeTest	
M37	getLapTime()	storeLaptimeTest	
M38	showStopWatch()		
M39	setDday()	setDdaytest	Dday
M40	getDday()		
M41	getCursor()	moveCursor_DdayTest	
M42	ShowDday()		
M43	deleteDday()	deleteDdayTest	
M44	showNextDday()	showNextDdayTest	
M45	moveCursor_Dday()	moveCursor_DdayTest	
M46	plusDday()	plusDayTest	
M47	confirmDday()	confirmDdayTest	
M48	cmpday()		
M49	showTimer()		Timer
M50	startTimer()	startTimertest	
M51	getTimer()	getTimertest	
M52	get_flag()	getFlagTest	
M53	stopTimer()	stopTimertest	
M54	pauseTimer()	pauseTimertest	
M55	setTimer()	setTimertest	
M56	moveCursor_timer()	moveCursor_timertest	
M57	plusTimer()	plusTimertest	
M58	confirmTimer()	confirmTimertest	

M59	setActivateFunction()	setActiveFunctiontest	FunctionActivator
M60	nextActivateFunction()	nextActivateFunctiontest	
M61	get_active_count()	get_active_counttest	
M62	get_modeQ()	get_modeQtest	
M63	get_position()	get_position_test	
M64	get_active()	get_activetest	
M65	get_active_name()	get_active_name_test	
M66	onOffFunction()	onOffFuctionTest	
M67	confirmActive()	confirmActivateTest	
M68	pressButton()		
M69	display()		
M70	changeMode()	changeModetest	
M71	work()		Mode