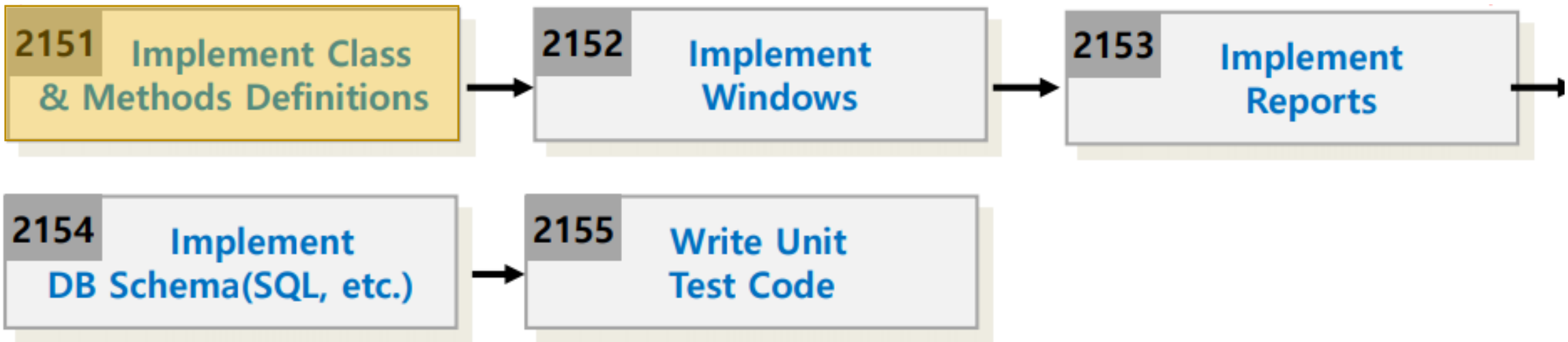
OOPT Stage 2050 Construct

당첨되시계

201411212 송인호
201611234 전재원
201611230 전계원
201711809 박수빈


2151
Implement Class
& Methods Definitions
2152
Implement
Windows
2153
Implement
Reports
2154
Implement
DB Schema(SQL, etc.)
2155
Write Unit
Test Code

Type	Class
Name	Timer
Purpose	Timer 기능 관리 클래스
Overview(Class)	
Cross Reference	R1.1, R1.2, R1.3, R1.4, R1.5, R1.6
Exceptional Courses Of Events	

Type	Class
Name	Stopwatch
Purpose	Stopwatch 기능 관리 클래스
Overview(Class)	
Cross Reference	R3.1, R3.2, R3.3, R3.4, R3.5
Exceptional Courses Of Events	

Type	Class
Name	Alarm
Purpose	AlarmInfo 객체의 관리 및 알람 기능
Overview(Class)	
Cross Reference	R2.1, R2.2, R2.3, R2.4, R2.5, R2.6
Exceptional Courses Of Events	

Type	Class
Name	AlarmInfo
Purpose	알람의 시간 정보 및 On/Off 정보를 가지고 있는 클래스
Overview(Class)	
Cross Reference	R2.1, R2.2, R2.3, R2.4, R2.5, R2.6
Exceptional Courses Of Events	

Type	Class
Name	WorldTime
Purpose	WorldTime 기능 관리 클래스
Overview(Class)	
Cross Reference	R4.1, R4.2, R4.3
Exceptional Courses Of Events	

Type	Class
Name	TimeKeeping
Purpose	TimeKeeping 기능 관리 클래스
Overview(Class)	
Cross Reference	R5.1, R5.2
Exceptional Courses Of Events	

Type	Class
Name	Lottery
Purpose	Lottery 기능 관리 클래스
Overview(Class)	
Cross Reference	R6.1, R6.2, R6.3
Exceptional Courses Of Events	

Type	Class
Name	ModeSelection
Purpose	6개의 모드 중 원하는 모드를 선택하는 기능을 위한 클래스
Overview(Class)	
Cross Reference	R7.1, R7.2
Exceptional Courses Of Events	

Type	Class
Name	TimeTicker
Purpose	TimeData 객체를 관리하는 클래스
Overview(Class)	
Cross Reference	R9.1
Exceptional Courses Of Events	

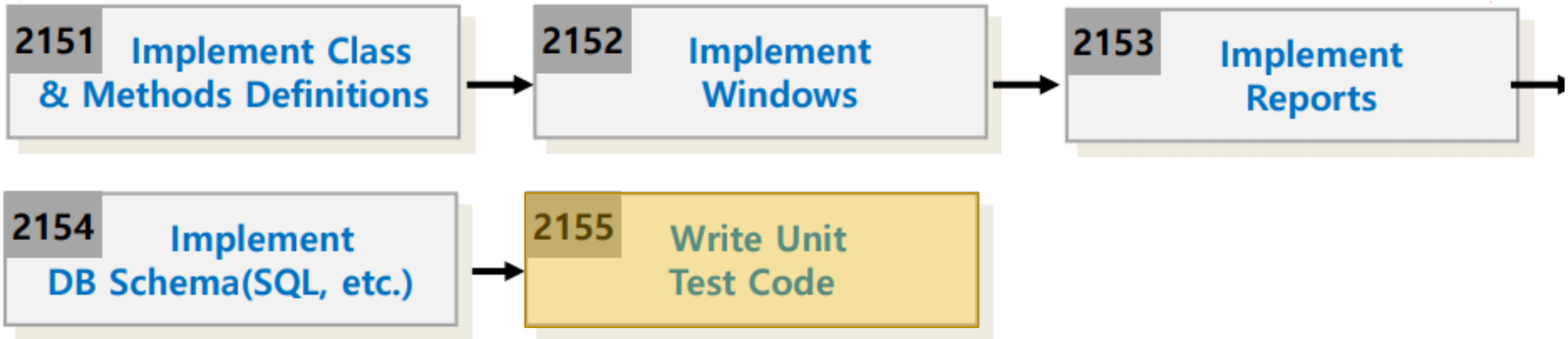
Type	Class
Name	TimeData
Purpose	현재 시간과, 사용자의 입력 없이 대기중인 시간을 가지는 클래스
Overview(Class)	
Cross Reference	R9.1
Exceptional Courses Of Events	

Type	Class
Name	Buzzer
Purpose	시계의 알람, 타이머 등의 모드에서 버저가 울리는 경우 알림음을 제어하기 위한 클래스
Overview(Class)	
Cross Reference	R1.5, R2.4
Exceptional Courses Of Events	

Type	Class
Name	Mode
Purpose	추상 클래스로, 시계의 각 모드들 6개의 클래스가 상속받는다. 해당 모드가 화면에 표시된 경우의 동작과, 백그라운드에서 처리할 동작 등이 추상 함수로 선언되어 있다.
Overview(Class)	
Cross Reference	
Exceptional Courses Of Events	

Type	Class
Name	Controller
Purpose	사용자로부터 들어오는 입력을 각 모드 클래스의 기능으로 매핑 시켜주는 클래스
Overview(Class)	
Cross Reference	R1~R9
Exceptional Courses Of Events	

Type	Class
Name	GlobalSettings
Purpose	시계의 기본적인 정보를 다루는 클래스로, 다른 클래스들이 사용하는 변수 및 함수들이 정의되어 있다.
Overview(Class)	
Cross Reference	
Exceptional Courses Of Events	


2151
Implement Class
& Methods Definitions
2152
Implement
Windows
2153
Implement
Reports
2154
Implement
DB Schema(SQL, etc.)
2155
Write Unit
Test Code

```java
public class AlarmInfoTest {
    @Test
    public void checkInitialIsOn() {
        AlarmInfo alarmInfo = new AlarmInfo();

        assertEquals(alarmInfo.getIsOn(), actual: false);
    }

    @Test
    public void checkInitialTimestamp() {
        AlarmInfo alarmInfo = new AlarmInfo();

        assertEquals(alarmInfo.getTimestamp(), LocalTime.of( hour: 12,  minute: 0,  second: 0));
    }

    @Test
    public void checkInitialTimestampWithParameter() {
        AlarmInfo alarmInfo = new AlarmInfo( hour: 13,  minute: 54, second: 50);

        assertEquals(alarmInfo.getTimestamp(), LocalTime.of( hour: 13,  minute: 54,  second: 50));
    }

    @Test
    public void checkIsOnAfterSetIsOn() {
        AlarmInfo alarmInfo = new AlarmInfo();

        alarmInfo.setIsOn(true);

        assertEquals(alarmInfo.getIsOn(), actual: true);
    }

    @Test
    public void checkTimestampAfterSetTimestamp() {
        AlarmInfo alarmInfo = new AlarmInfo();
        LocalTime localTime = LocalTime.of( hour: 14,  minute: 25,  second: 43);

        alarmInfo.setTimestamp(localTime);

        assertEquals(alarmInfo.getTimestamp(), LocalTime.of( hour: 14,  minute: 25,  second: 43));
    }
}
```

```java
public class AlarmTest {
    @Test
    public void checkInitialAlarmTime() {
        Alarm alarm = new Alarm( top: null,  bottom: null);
        ArrayList<AlarmInfo> alarmInfoArr = alarm.getAlarmInfoArr();
        LocalTime expectedTime = LocalTime.of( hour: 12,  minute: 0,  second: 0);

        for (int i = 0; i < alarmInfoArr.size(); i++) {
            assertEquals(alarmInfoArr.get(i).getIsOn(), actual: false);
            assertEquals(alarmInfoArr.get(i).getTimestamp(), expectedTime);
        }
    }

    @Test
    public void checkInitialAlarmSize() {
        Alarm alarm = new Alarm( top: null,  bottom: null);
        ArrayList<AlarmInfo> alarmInfoArr = alarm.getAlarmInfoArr();

        assertEquals(alarmInfoArr.size(), actual: 4);
    }

    @Test
    public void checkIsOnAfterSetIsOn() {
        Alarm alarm = new Alarm( top: null,  bottom: null);
        AlarmInfo alarmInfo = alarm.getAlarmInfoArr().get(0);

        // true
        alarmInfo.setIsOn(true);
        assertEquals(alarmInfo.getIsOn(), actual: true);

        // false
        alarmInfo.setIsOn(false);
        assertEquals(alarmInfo.getIsOn(), actual: false);
    }
```

```java
@Test
public void checkTimestampAfterSetTimestamp() {
    Alarm alarm = new Alarm( top: null,  bottom: null);
    AlarmInfo alarmInfo = alarm.getAlarmInfoArr().get(0);

    LocalTime localTime = LocalTime.of( hour: 13,  minute: 43,  second: 49);
    alarmInfo.setTimestamp(localTime);
    assertEquals(alarmInfo.getTimestamp(), localTime);
}

@Test
public void checkCurAlarmIdxAfterPlusIdx() {
    Alarm alarm = new Alarm( top: null,  bottom: null);

    int random = (int) (Math.random() % 20);
    for (int i = 0 ; i < random; i++) {
        alarm.plusIdx();
    }

    assertEquals(alarm.getCurAlarmIdx(),  actual: random % 4);
}

@Test
public void checkTimestampAfterSetting() {
    Alarm alarm = new Alarm( top: null,  bottom: null);

    int random = (int) (Math.random() % 20);
    for (int i = 0 ; i < random; i++) {
        alarm.plusIdx();
    }

    alarm.setAlarm();

    // add values
    int hour = 7;
    int minute = 15;

    for (int i = 0; i < hour; i++)
        alarm.plusValue();
    alarm.plusIdx();

    for (int i = 0; i < minute / 10; i++)
        alarm.plusValue();
    alarm.plusIdx();

    for (int i = 0; i < minute % 10; i++)
        alarm.plusValue();
    alarm.plusIdx();

    assertEquals(alarm.getAlarmInfoArr().get(random % 4).getTimestamp()
            LocalTime.of( hour: hour + 12, minute));
}
```

```java
public class BuzzerTest {
    @Test
    public void checkInitialClip() {
        Buzzer buzzer = new Buzzer();

        Clip clip = buzzer.getClip();

        assertEquals( expected: clip != null,  actual: true);
    }

    @Test
    public void checkClipAfterStartBuzzer() throws InterruptedException {
        Buzzer buzzer = new Buzzer();

        buzzer.startBuzzer();
        Clip clip = buzzer.getClip();

        sleep( millis: 500);

        assertEquals(clip.isRunning(),  actual: true);
    }

    @Test
    public void checkClipAfterStopBuzzer() throws InterruptedException {
        Buzzer buzzer = new Buzzer();

        buzzer.startBuzzer();

        sleep( millis: 500);

        buzzer.stopBuzzer();
        Clip clip = buzzer.getClip();

        assertEquals(clip.isRunning(),  actual: false);
    }
}
```

```java
public class GlobalSettingsTest {
    @Test
    public void checkModesAfterSetModes() {
        ArrayList<ModeType> input = new ArrayList<~>();

        input.add(ModeType.LOTTERY);
        input.add(ModeType.TIMER);
        input.add(ModeType.TIMEKEEPING);
        input.add(ModeType.WORLDTIME);

        GlobalSettings.getInstance().setModes(input);

        ArrayList<ModeType> modes = GlobalSettings.getInstance().getModes();

        assertEquals(modes.get(0), ModeType.LOTTERY);
        assertEquals(modes.get(1), ModeType.TIMER);
        assertEquals(modes.get(2), ModeType.TIMEKEEPING);
        assertEquals(modes.get(3), ModeType.WORLDTIME);
    }

    @Test
    public void checkModesAfterSetCurMode() {
        GlobalSettings.getInstance().setCurMode(ModeType.LOTTERY);

        assertEquals(GlobalSettings.getInstance().getCurMode(), ModeType.LOTTERY);
    }
}
```

```java
public class LotteryTest {
    @Test
    public void checkInitialIsStarted() {
        Lottery lottery = new Lottery( top: null,  bottom: null);

        assertEquals(lottery.getIsStarted(),  actual: false);
    }

    @Test
    public void checkIsStartedAfterStartLottery() {
        Lottery lottery = new Lottery( top: null,  bottom: null);

        lottery.startLottery();

        assertEquals(lottery.getIsStarted(),  actual: true);
    }

    @Test
    public void checkLotteryCountAfterStartLottery() {
        Lottery lottery = new Lottery( top: null,  bottom: null);

        lottery.startLottery();

        assertEquals(lottery.getLotteryArr().size(),  actual: 6);
    }

    @Test
    public void checkLotteryCountAfterResetLottery() {
        Lottery lottery = new Lottery( top: null,  bottom: null);

        lottery.startLottery();
        lottery.resetLottery();

        assertEquals(lottery.getLotteryArr().size(),  actual: 0);
    }
}
```

```java
public class ModeSelectionTest {
    @Test
    public void checkInitialModeIdx() {
        ModeSelection modeSelection = new ModeSelection( top: null,  bottom: null);

        assertEquals(modeSelection.getModeIdx(),  actual: 0);
    }

    @Test
    public void checkTempModesCountAfterSetModeSelection() {
        ModeSelection modeSelection = new ModeSelection( top: null,  bottom: null);
        modeSelection.setModeSelection();

        assertEquals(modeSelection.getTempModes().size(),  actual: 4);
    }

    @Test
    public void checkTempModesCpyCountAfterSetModeSelection() {
        ModeSelection modeSelection = new ModeSelection( top: null,  bottom: null);
        modeSelection.setModeSelection();

        assertEquals(modeSelection.getTempModesCpy().size(),  actual: 4);
    }
```

```java
    @Test
    public void checkTempModesAfterSetting() {
        GlobalSettings.getInstance().resetSettings();
        ModeSelection modeSelection = new ModeSelection( top: null,  bottom: null);
        modeSelection.setModeSelection();

        for (int i = 0; i < 1; i++)
            modeSelection.plusValue();
        modeSelection.confirmValue();

        for (int i = 0; i < 2; i++)
            modeSelection.plusValue();
        modeSelection.confirmValue();

        for (int i = 0; i < 3; i++)
            modeSelection.plusValue();
        modeSelection.confirmValue();

        for (int i = 0; i < 4; i++)
            modeSelection.plusValue();
        modeSelection.confirmValue();

        ArrayList<ModeType> answer = new ArrayList<ModeType>();
        answer.add(ModeType.STOPWATCH);
        answer.add(ModeType.TIMER);
        answer.add(ModeType.TIMEKEEPING);
        answer.add(ModeType.WORLDTIME);

        assertEquals(modeSelection.getTempModes(), answer);
    }
}
```

```java
public class StopwatchTest {
    @Test
    public void checkInitialRunningFlag() {
        Stopwatch stopwatch = new Stopwatch( top: null,  bottom: null);

        assertEquals(stopwatch.getRunningFlag(),  actual: false);
    }

    @Test
    public void checkInitialTimestamp() {
        Stopwatch stopwatch = new Stopwatch( top: null,  bottom: null);

        assertEquals(stopwatch.getTimestamp(),  actual: 0);
    }

    @Test
    public void checkIsOnAfterStartStopwatch() {
        Stopwatch stopwatch = new Stopwatch( top: null,  bottom: null);

        stopwatch.startStopwatch();

        assertEquals(stopwatch.getRunningFlag(),  actual: true);
    }

    @Test
    public void checkTimestampAfterStartStopwatch() throws InterruptedException {
        Stopwatch stopwatch = new Stopwatch( top: null,  bottom: null);

        stopwatch.startStopwatch();
        stopwatch.setThreadMode(ThreadMode.BACKGROUND);
        stopwatch.start();

        sleep( millis: 500);

        assertEquals( expected: stopwatch.getTimestamp() > 0,  actual: true);
    }
```

```java
    @Test
    public void checkRunningFlagAfterPauseStopwatch() throws InterruptedException {
        Stopwatch stopwatch = new Stopwatch( top: null,  bottom: null);

        stopwatch.startStopwatch();
        stopwatch.setThreadMode(ThreadMode.BACKGROUND);
        stopwatch.start();

        sleep( millis: 500);

        stopwatch.pauseStopwatch();

        assertEquals(stopwatch.getRunningFlag(),  actual: false);
    }

    @Test
    public void checkTimestampAfterResetStopwatch() throws InterruptedException {
        Stopwatch stopwatch = new Stopwatch( top: null,  bottom: null);

        stopwatch.startStopwatch();
        stopwatch.setThreadMode(ThreadMode.BACKGROUND);
        stopwatch.start();

        sleep( millis: 500);

        stopwatch.pauseStopwatch();
        stopwatch.resetStopwatch();

        assertEquals(stopwatch.getTimestamp(),  actual: 0);
    }
}
```

```java
public class TimeDataTest {
    @Test
    public void checkInitialNoActionTime() {
        TimeData timeData = new TimeData();

        assertEquals(timeData.getNoActionTime(), actual: 0);
    }

    @Test
    public void checkTimestampAfterAddTimestamp() {
        TimeData timeData = new TimeData();

        long prev = timeData.getTimestamp();
        timeData.addTimestamp( timeDiff: 1000);

        assertEquals(timeData.getTimestamp(), actual: prev + 1000);
    }

    @Test
    public void checkTimestampAfterSetTimestamp() {
        TimeData timeData = new TimeData();

        timeData.setTimestamp(1000000);

        assertEquals(timeData.getTimestamp(), actual: 1000000);
    }

    @Test
    public void checkTimestampAfterAddNoActionTime() {
        TimeData timeData = new TimeData();

        long prev = timeData.getNoActionTime();
        timeData.addNoActionTime( timeDiff: 1000);

        assertEquals(timeData.getNoActionTime(), actual: prev + 1000);
    }

    @Test
    public void checkTimestampAfterSetNoActionTime() {
        TimeData timeData = new TimeData();

        timeData.addNoActionTime( timeDiff: 1000000);

        assertEquals(timeData.getNoActionTime(), actual: 1000000);
    }
}
```

```java
public class TimeKeepingTest {
    @Test
    public void checkInitialEditingIdx() {
        TimeKeeping timeKeeping = new TimeKeeping( top: null, bottom: null);

        assertEquals(timeKeeping.getEditingIdx(), actual: 0);
    }

    @Test
    public void checkIsEditingAfterSetTimeKeeping() {
        TimeKeeping timeKeeping = new TimeKeeping( top: null, bottom: null);

        timeKeeping.setTimeKeeping();

        assertEquals(timeKeeping.getIsEditing(), actual: true);
    }

    @Test
    public void checkTempTimeAfterSetting() {
        TimeKeeping timeKeeping = new TimeKeeping( top: null, bottom: null);

        timeKeeping.setTimeKeeping();

        LocalDateTime initialData = timeKeeping.getTempTime();

        // add values
        int hour = 7;
        int minute = 15;
        int second = 14;

        timeKeeping.plusIdx();
        timeKeeping.plusIdx();
        timeKeeping.plusIdx();
        timeKeeping.plusIdx();
        for (int i = 0; i < hour; i++)
            timeKeeping.plusValue();
        timeKeeping.plusIdx();

        for (int i = 0; i < minute / 10; i++)
            timeKeeping.plusValue();
        timeKeeping.plusIdx();

        for (int i = 0; i < minute % 10; i++)
            timeKeeping.plusValue();
        timeKeeping.plusIdx();

        for (int i = 0; i < second / 10; i++)
            timeKeeping.plusValue();
        timeKeeping.plusIdx();

        for (int i = 0; i < second % 10; i++)
            timeKeeping.plusValue();
        timeKeeping.plusIdx();

        int initMinute = initialData.getMinute();
        int initSecond = initialData.getSecond();

        assertEquals(timeKeeping.getTempTime(),
                initialData.withHour((initialData.getHour() + hour) % 24)
                    .withMinute((initMinute / 10 * 10 + minute / 10 * 10 + (initMinute + minute) % 10) % 60)
                    .withSecond((initSecond / 10 * 10 + second / 10 * 10 + (initSecond + second) % 10) % 60));
    }
}
```

```java
public class TimerTest {
    @Test
    public void checkInitialTimestamp() {
        Timer timer = new Timer( top: null,  bottom: null);

        assertEquals(timer.getTimestamp(),  actual: 10 * 1000);
    }

    @Test
    public void checkRunningFlagAfterStartTimer() {
        Timer timer = new Timer( top: null,  bottom: null);

        timer.startTimer();

        assertEquals(timer.getRunningFlag(),  actual: true);
    }

    @Test
    public void checkRunningFlagAfterPauseTimer() {
        Timer timer = new Timer( top: null,  bottom: null);

        timer.startTimer();
        timer.pauseTimer();

        assertEquals(timer.getRunningFlag(),  actual: false);
    }

    @Test
    public void checkRunningFlagAfterResetTimer() {
        Timer timer = new Timer( top: null,  bottom: null);

        timer.setSettingTime(20 * 1000);
        timer.startTimer();
        timer.pauseTimer();
        timer.resetTimer();

        assertEquals(timer.getRunningFlag(),  actual: false);
    }

    @Test
    public void checkTimestampAfterResetTimer() {
        Timer timer = new Timer( top: null,  bottom: null);

        timer.setSettingTime(20 * 1000);
        timer.startTimer();
        timer.pauseTimer();
        timer.resetTimer();

        assertEquals(timer.getTimestamp(),  actual: 20 * 1000);
    }
}
```

```java
public class TimeTickerTest {
    @Test
    public void checkInitialNoActionTime() {
        TimeTicker timeTicker = new TimeTicker(new TimeData(), controller: null);

        TimeData timeData = timeTicker.getTimeData();

        assertEquals(timeData.getNoActionTime(), actual: 0);
    }

    @Test
    public void checkNoActionTimeAfterRun() throws InterruptedException {
        TimeTicker timeTicker = new TimeTicker(new TimeData(), controller: null);

        timeTicker.start();

        sleep( millis: 500);

        TimeData timeData = timeTicker.getTimeData();

        assertEquals( expected: timeData.getNoActionTime() > 0, actual: true);
    }
}
```

```java
public class WorldTimeTest {
    @Test
    public void checkInitialTimezoneIdx() {
        WorldTime worldTime = new WorldTime( top: null, bottom: null);

        assertEquals(worldTime.getTimezoneIdx(), actual: 0);
    }

    @Test
    public void checkTimezoneIdxAfterTimezoneToRight() {
        WorldTime worldTime = new WorldTime( top: null, bottom: null);

        int random = (int) (Math.random() % 100);

        for (int i =0; i < random; i++) {
            worldTime.timezoneToRight();
        }

        assertEquals(worldTime.getTimezoneIdx(), actual: random % 24);
    }

    @Test
    public void checkTimezoneIdxAfterTimezoneToLeft() {
        WorldTime worldTime = new WorldTime( top: null, bottom: null);

        int random = (int) (Math.random() % 100);

        for (int i = 0; i < random; i++) {
            worldTime.timezoneToLeft();
        }

        assertEquals(worldTime.getTimezoneIdx(), actual: (24 - (random % 24)) % 24);
    }
```

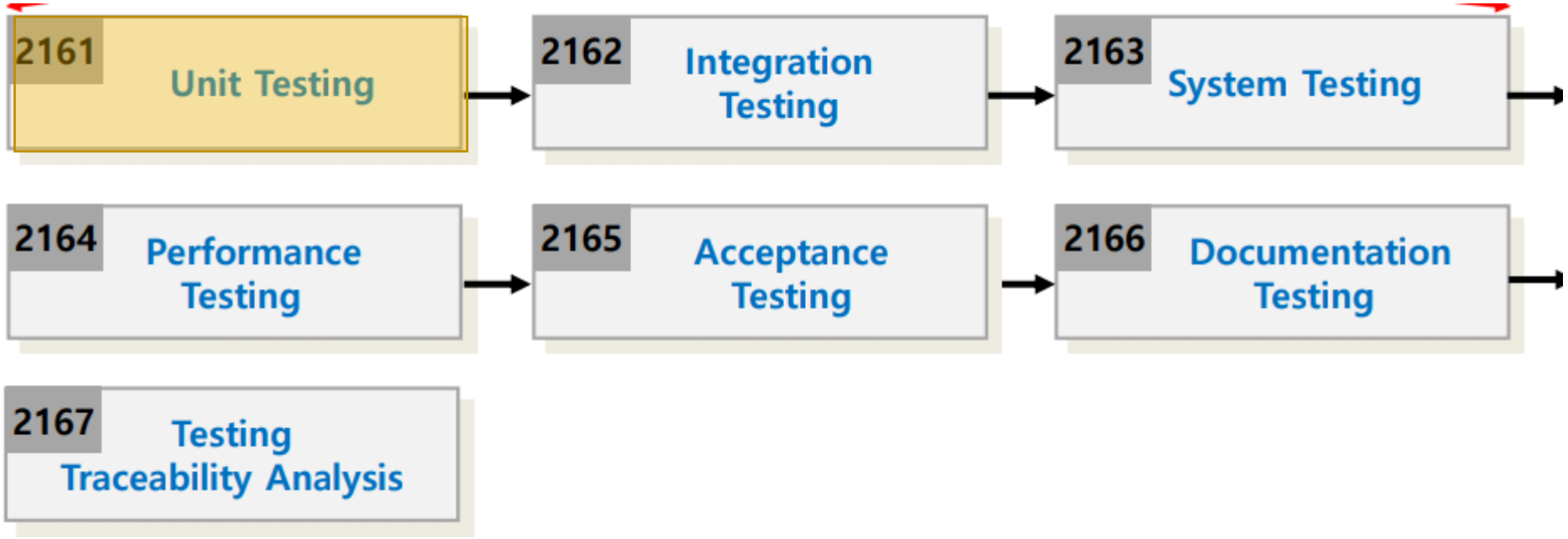
```java
    @Test
    public void checkTimezoneIdxAfterSetting() {
        WorldTime worldTime = new WorldTime( top: null,  bottom: null);

        int randomRight = (int) (Math.random() % 100);

        for (int i = 0; i < randomRight; i++) {
            worldTime.timezoneToRight();
        }

        int randomLeft = (int) (Math.random() % 100);

        for (int i = 0; i < randomLeft; i++) {
            worldTime.timezoneToLeft();
        }

        // 120 is bigger than 100 and 24 * 5
        assertEquals(worldTime.getTimezoneIdx(),  actual: (120 + randomRight - randomLeft) % 24);
    }
}
```

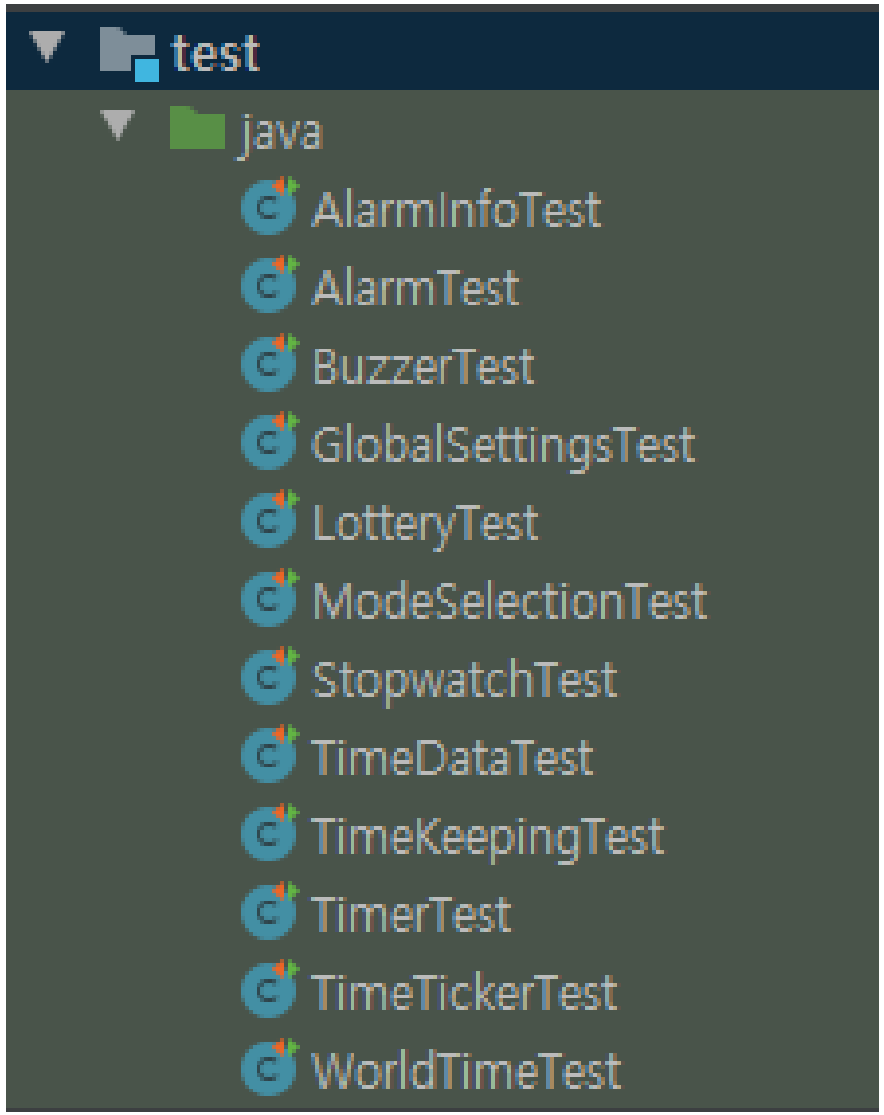
OOPT Stage 2060 Test

당첨되시계

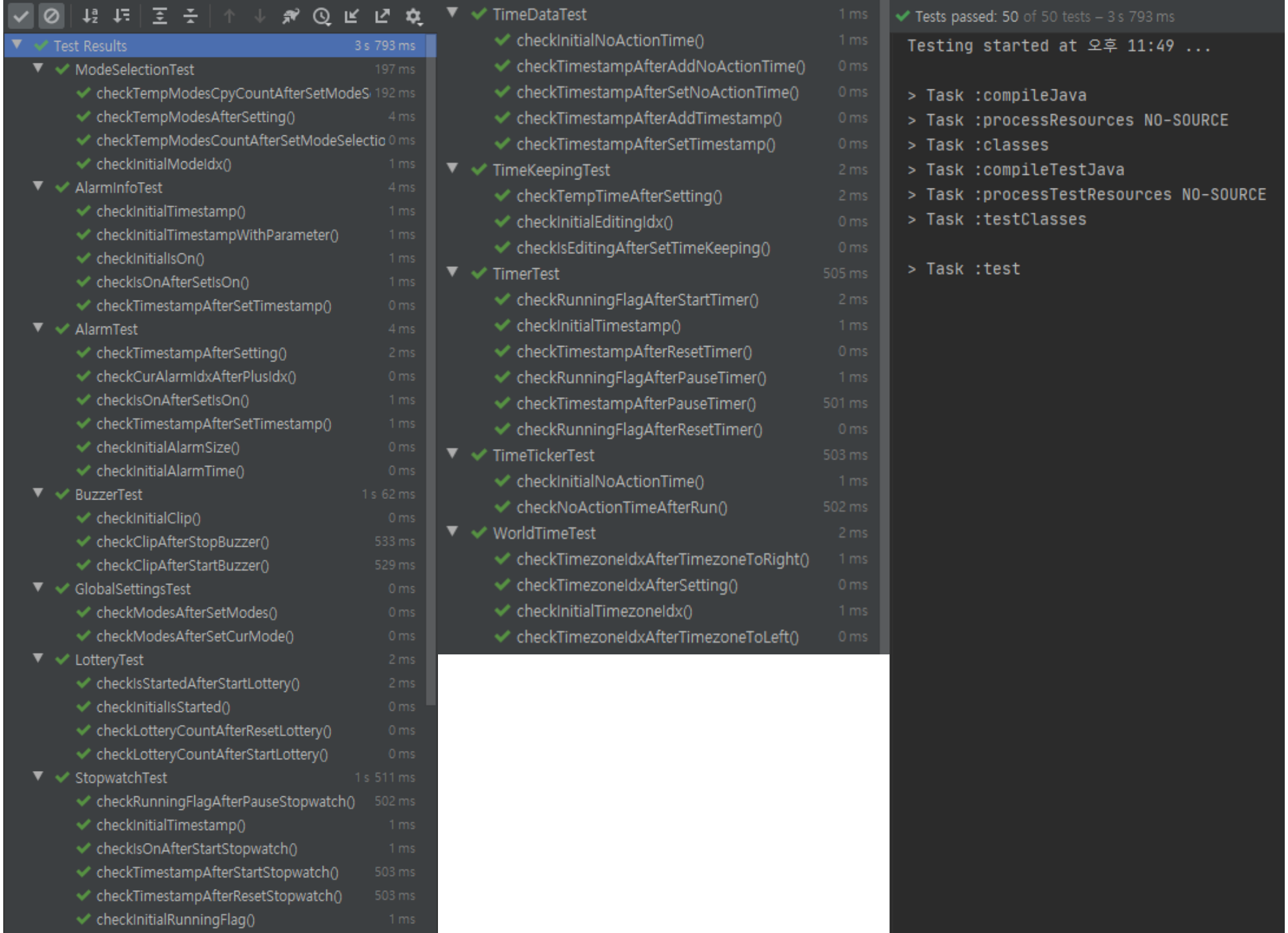
201411212 송인호
201611234 전재원
201611230 전계원
201711809 박수빈

[2067] Testing Traceability Analysis

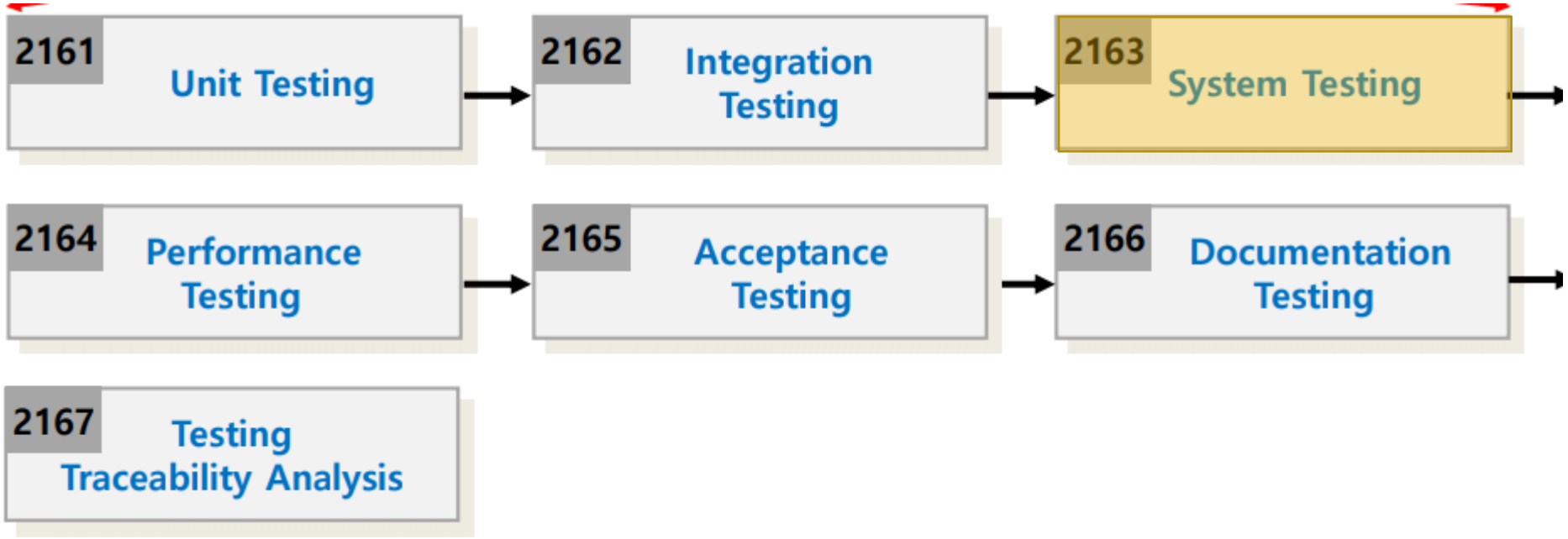



test
java
AlarmInfoTest
AlarmTest
BuzzerTest
GlobalSettingsTest
LotteryTest
ModeSelectionTest
StopwatchTest
TimeDataTest
TimeKeepingTest
TimerTest
TimeTickerTest
WorldTimeTest

- 12 modules


Test Results 3 s 793 ms
ModeSelectionTest 197 ms
checkTempModesCpyCountAfterSetModeS 192 ms
checkTempModesAfterSetting() 4 ms
checkTempModesCountAfterSetModeSelectio 0 ms
checkInitialModeIdx() 1 ms
AlarmInfoTest 4 ms
checkInitialTimestamp() 1 ms
checkInitialTimestampWithParameter() 1 ms
checkInitialIsOn() 1 ms
checkIsOnAfterSetIsOn() 1 ms
checkTimestampAfterSetTimestamp() 0 ms
AlarmTest 4 ms
checkTimestampAfterSetting() 2 ms
checkCurAlarmIdxAfterPlusIdx() 0 ms
checkIsOnAfterSetIsOn() 1 ms
checkTimestampAfterSetTimestamp() 1 ms
checkInitialAlarmSize() 0 ms
checkInitialAlarmTime() 0 ms
BuzzerTest 1 s 62 ms
checkInitialClip() 0 ms
checkClipAfterStopBuzzer() 533 ms
checkClipAfterStartBuzzer() 529 ms
GlobalSettingsTest 0 ms
checkModesAfterSetModes() 0 ms
checkModesAfterSetCurMode() 0 ms
LotteryTest 2 ms
checkIsStartedAfterStartLottery() 2 ms
checkInitialIsStarted() 0 ms
checkLotteryCountAfterResetLottery() 0 ms
checkLotteryCountAfterStartLottery() 0 ms
StopwatchTest 1 s 511 ms
checkRunningFlagAfterPauseStopwatch() 502 ms
checkInitialTimestamp() 1 ms
checkIsOnAfterStartStopwatch() 1 ms
checkTimestampAfterStartStopwatch() 503 ms
checkTimestampAfterResetStopwatch() 503 ms
checkInitialRunningFlag() 1 ms
TimeDataTest 1 ms
checkInitialNoActionTime() 1 ms
checkTimestampAfterAddNoActionTime() 0 ms
checkTimestampAfterSetNoActionTime() 0 ms
checkTimestampAfterAddTimestamp() 0 ms
checkTimestampAfterSetTimestamp() 0 ms
TimeKeepingTest 2 ms
checkTempTimeAfterSetting() 2 ms
checkInitialEditingIdx() 0 ms
checkIsEditingAfterSetTimeKeeping() 0 ms
TimerTest 505 ms
checkRunningFlagAfterStartTimer() 2 ms
checkInitialTimestamp() 1 ms
checkTimestampAfterResetTimer() 0 ms
checkRunningFlagAfterPauseTimer() 1 ms
checkTimestampAfterPauseTimer() 501 ms
checkRunningFlagAfterResetTimer() 0 ms
TimeTickerTest 503 ms
checkInitialNoActionTime() 1 ms
checkNoActionTimeAfterRun() 502 ms
WorldTimeTest 2 ms
checkTimezoneIdxAfterTimezoneToRight() 1 ms
checkTimezoneIdxAfterSetting() 0 ms
checkInitialTimezoneIdx() 1 ms
checkTimezoneIdxAfterTimezoneToLeft() 0 ms
Tests passed: 50 of 50 tests – 3 s 793 ms
Testing started at 오후 11:49 ...
> Task :compileJava
> Task :processResources NO-SOURCE
> Task :classes
> Task :compileTestJava
> Task :processTestResources NO-SOURCE
> Task :testClasses
> Task :test

[2067] Testing Traceability Analysis



TimeKeeping, Lottery, ModeSelection, Stopwatch

TID	name	Description
T1	알람이 세팅한 시간대로 저장되는지 확인	알람을 설정한 후, 보기 모드에서 설정한 시간으로 보이는지 확인
T2	알람이 세팅한 시간에 울리는지 확인	알람을 설정한 후, On 상태로 만들고 지정한 시간에 울리는지 확인. 다시 Off상태로 만들었다가 다시 On 상태로 만들었는지도 확인
T3	6개 중 4개 모드에 포함하지 않았을 때 울리는지 확인	알람을 설정한 후, On 상태로 만든 다음 ModeSelection에서 Alarm을 선택하지 않는다. 울리지 않아야 통과
T4	알람이 울릴 때, 아무 버튼이나 누르면 알람이 꺼지는지 확인	알람을 설정한 후, On 상태로 만들고 지정한 시간에 울렸을 때 아무 버튼이나 눌렀을 때 버저가 꺼지는지 확인
T5	+버튼을 눌렀을 때 로또 번호가 나오는지 확인	초기 상태에서 +버튼을 눌러서 숫자가 6개 나오는지 확인. 이 숫자는 1이상 45 이하이며, 겹치지 않아야 하며 오름차순으로 보여야 한다.
T6	로또 번호가 나왔을 때, -버튼을 눌러 초기화 되는지 확인	로또 번호를 출력한 상황에서, -버튼을 눌러서 초기 상태로 만들고 다시 로또 번호를 출력했을 때, 번호가 바뀌는지 확인
T7	아무 모드에서나 Mode버튼을 길게 눌렀을 때(Long Mode버튼을 눌렀을 때) ModeSelection이 되는지 확인	기타 다른 모드에서, LongMode 버튼을 눌렀을 때 ModeSelection이 실행되는지 확인
T8	ModeSelection에서 설정이 끝난 후, 모드가 잘 설정되는지 확인	ModeSelection에서 설정을 마친 다음, 첫 모드로 이동되며, 설정한 4개의 모드가 순서대로 설정되었는지 확인
T9	Stopwatch에서 정지상태에서 잘 시작되는지 확인	정지 상태(초기 상태, 일시정지 상태)에서 +버튼을 누른다.
T10	Stopwatch에서 실행상태에서 일시정지 되는지 확인	실행 상태에서 +버튼을 누른다.
T11	Stopwatch에서 Lap이 기록되는지 확인	Adjust버튼을 눌러서 Lap이 기록되는지 확인한다. (실행, 정지 상태 둘 다)
T12	Stopwatch에서 일시 정지상태에서 초기화 되는지 확인	일시 정지 상태에서 –버튼을 눌러서 현재 시간과 Lap이 초기화되는지 확인

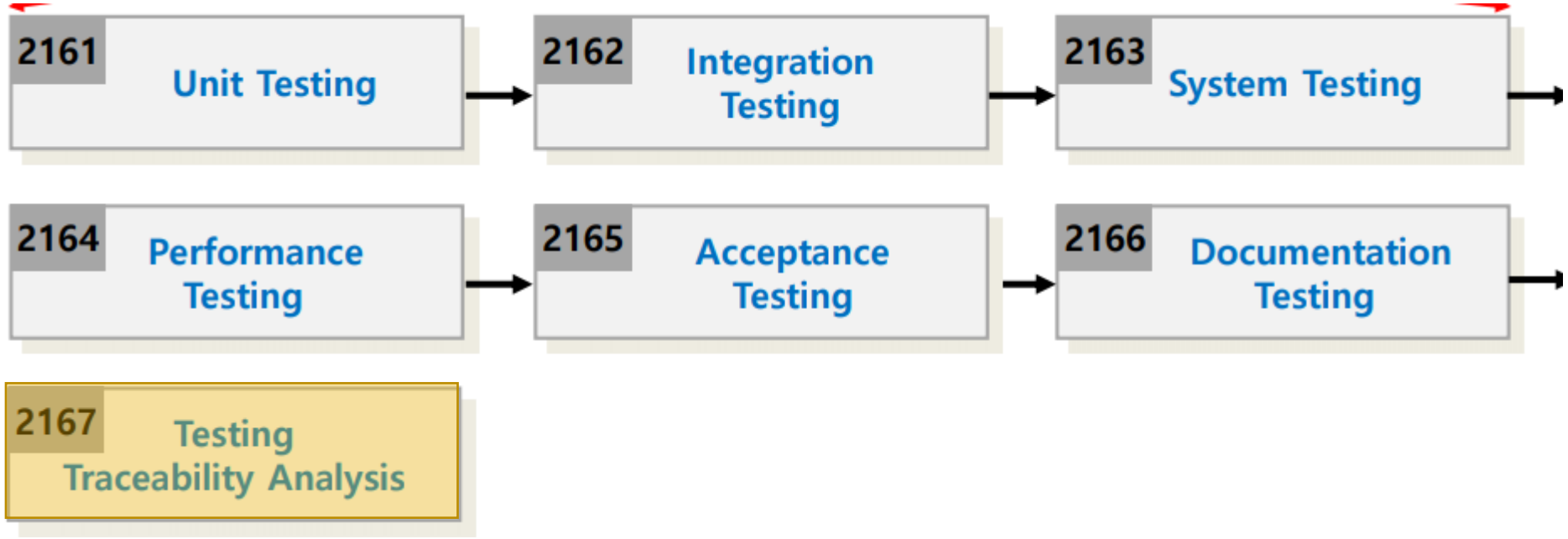
TimeKeeping, Timer

TID	name	Description
T13	시계를 처음 실행했을 때, 현재 시간 확인	초기 시간은 실행시킨 디바이스의 시스템 시간과 같아야 한다.
T14	시간을 설정했을 때, 현재 시간이 바뀌는지 확인	시간 설정 후, 보기 모드에서 설정한대로 시간이 바뀌었는지 확인
T15	시간을 설정했을 때, 변경된 시간에 따라 알람이 울리는지 확인	시간 설정 후, 설정한 시간에 맞춰 알람이 울리는지 확인
T16	타이머를 설정이 정상적으로 되는지 확인	타이머를 설정했을 때, 설정한 대로 시간이 맞춰지는지 확인
T17	타이머가 정상 시작하는지 확인	설정 모드가 아닐 때, +버튼을 눌러서 시간이 감소하는지 확인
T18	타이머가 정상 정지하는지 확인	실행 상태일 때, +버튼을 눌러서 시간 감소를 멈추는지 확인
T19	타이머가 종료되었을 때, 버저가 울리는지 확인	타이머의 시간이 0초일 때, 버저가 울리는지 확인
T20	버저가 울릴 때, 아무 버튼이나 눌렀을 때 꺼지는 확인	타이머의 시간이 0초라서 버저가 울릴 때, 아무 버튼이나 눌렀을 때 꺼지는 확인
T21	버저를 껐을 때, 시간이 설정 시간으로 바뀌는지 확인	버저를 껐을 때, 설정했던 시간으로 초기화 되는지 확인
T22	타이머를 시작하고, 다른 모드로 이동해도 시간이 가는지 확인	타이머를 시작하고 다른 모드로 이동했다가 다시 타이머로 돌아왔을 때, 시간이 감소하는지 확인
T23	타이머를 시작하고, ModeSelection에서 타이머를 제거했을 때, 시간이 0이 되었을 타이밍에 버저가 울리는지 확인	타이머를 시작하고, ModeSelection에서 타이머를 제거했을 때, 시간이 0이 되었을 타이밍에 버저가 울리는지 확인

WorldTime, Timeout, Mode Change, etc

TID	name	Description
T24	초기 화면에서 시간대가 -11로 시작하는지 확인	WorldTime에 처음 접근했을 때, 시간대가 -11로 되어있는지 확인
T25	시간대가 -11일 때, -버튼을 누르면 시간대가 +12로 이동하는지 확인	시간대가 -11일 때, -버튼을 누르면 시간대가 +12로 이동하는지 확인
T26	시간대가 +12일 때, +버튼을 누르면 시간대가 -11로 이동하는지 확인	시간대가 +12일 때, +버튼을 누르면 시간대가 -11로 이동하는지 확인
T27	24개의 시간대에서 각각 대표 도시와 시간이 잘 표시되는가 확인	24개의 시간대에서 각각 대표 도시와 시간이 잘 표시되는가 확인
T28	TimeKeeping에서 현재 시간을 변경했을 때, WorldTime에서 변경된 시간에 따라 출력하는지 확인	TimeKeeping모드에서 현재 시간을 변경 후, WorldTime에서 각 시간대를 확인
T29	20초동안 아무 입력도 하지 않으면 첫 모드로 이동하는지 확인	20초동안 아무 입력도 하지 않으면 첫 모드로 이동하는지 확인
T30	ModeSelection에서 모드 변경 후, 20초동안 아무 입력도 하지 않으면 변경한 첫 모드로 이동하는지 확인	ModeSelection에서 모드 변경 후, 20초동안 아무 입력도 하지 않으면 변경한 첫 모드로 이동하는지 확인
T31	ModeSelection을 실행한 다음, Mode버튼을 누르면 첫 모드로 이동하는가	LongMode 버튼을 눌러 ModeSelection을 실행한 다음, Mode버튼을 눌러 설정을 취소한다.
T32	Mode 버튼을 눌렀을 때, 다음 모드로 이동하는가	Mode버튼을 눌러서 ModeSelection에서 설정한 대로 이동하는지 확인
T33	알람이 Off인 상태에서 세팅한 시간에 안 울리는지 확인	알람에서 시간을 설정하고, 해당 알람을 Off로 바꾼다. 정해진 시간에 울리지 않아야 통과
T34	시간 설정중인 상황에서 ModeSelection에 진입이 가능한가	알람, 스탑워치, 타임키핑의 시간을 설정하는 도중에 LongMode버튼을 눌렀을 때, ModeSelection이 실행되지 않아야 통과

[2067] Testing Traceability Analysis



Use Case	S-Link
Set Timer	S1, S2, S3, S20, S33, S34, S35
Start Timer	S2, S23
Pause Timer	S2, S24
Reset Timer	S3, S25
Check Timer	S21
Ring Timer	S27, S39, S40
Stop Ring Timer	S26, S28, S39, S41
Show Timer	S22
Set Alarm	S1, S2, S3, S6, S7, S8 S33, S34, S35
On Alarm	S3, S37, S42
Off Alarm	S3, S37, S43
Next Alarm	S2, S38
Check Alarm	S7, S37, S44, S45
Ring Alarm	S27, S39, S40
Stop Ring Alarm	S9, S28, S39, S41
Show Alarm	S7, S44, S46

Use Case	S-Link
Start Stopwatch	S2, S12
Pause Stopwatch	S2, S13
Reset Stopwatch	S3, S14
Record Lap	S1, S15
Run Stopwatch	S47
Show Stopwatch	S46
Timezone to left	S3, S32
Timezone to right	S2, S31
Show Worldtime	S47
Set TimeKeeping	S1, S2, S3, S8, S33, S34, S35, S36, S48
Show TimeKeeping	S46
Start Lottery	S2, S9
Reset Lottery	S3, S10
Show Lottery	S46
Set Mode Selection	S1, S2, S3, S5, S16, S17, S18, S19, S33, S34, S35
Show Mode Selection	S46
Change Mode	S4, S29, S49, S50
Time Checkout	S30, S48, S50, S51, S52

SID	Operation in sequence diagram	M-Link
S1	pressAdjustButton()	M1, M6
S2	pressPlusButton()	M2, M6
S3	pressMinusButton()	M3, M6
S4	pressModeButton()	M4, M6
S5	pressLongModeButton()	M5, M6
S6	setAlarm()	M24
S7	getTimeStamp()	M35, M55
S8	setTimeStamp()	M36, M52
S9	stopRingAlarm()	M29, M23
S10	startLottery()	M49
S11	resetLottery()	M50
S12	startStopwatch()	M37
S13	pauseStopwatch()	M38
S14	resetStopwatch()	M39

SID	Operation in sequence diagram	M-Link
S15	recordLap()	M40
S16	getModes()	M65
S17	confirmValue()	M44, M47
S18	setModes()	M67
S19	setModeSelection()	M43
S20	setTimer()	M13
S21	checkTimer()	M22, M58
S22	showTimer()	M57, M61, M70, M71
S23	startTImer()	M14
S24	pauseTimer()	M15
S25	resetTimer()	M16
S26	stopRingTimer()	M18
S27	ringBuzzer()	M17, M22
S28	stopRingBuzzer()	M18, M23
S29	changeMode()	M7, M59, M60, M69
S30	timeCheckOut()	M8
S31	moveTimezoneRight()	M41, M63
S32	moveTimezoneLeft()	M42, M63

SID	Operation in sequence diagram	M-Link
S33	plusValue()	M10, M19, M30, M45
S34	minusValue()	M11, M20, M31, M46
S35	plusIdx()	M12, M21, M32 M48
S36	setTimeKeeping()	M9
S37	setIsOn()	M34
S38	nextAlarm()	M27
S39	getBuzzer()	M64
S40	startBuzzer()	M22
S41	stopBuzzer()	M23
S42	onAlarm()	M25
S43	offAlarm()	M26
S44	getIsOn()	M33
S45	ringAlarm()	M28
S46	processOnScreen()	M57
S47	processBackground()	M51, M53, M58
S48	getTimeData()	M62
S49	getCurMode()	M66
S50	setCurMode()	M68
S51	setNoActionTime()	M54
S52	getNoActionTime()	M56

MID	Method	T-Link	Class
M1	pressAdjustButton()	T1,T2,T3,T4,T5,T6	Controller
M2	pressPlusButton()	T1,T2,T3	
M3	pressMinusButton()	T1,T2,T3	
M4	pressModeButton()	T22,T31,T32	
M5	pressLongModeButton()	T3,T7,T8,T34	
M6	pressAnyButton()	T3,T4	
M7	changeMode()	T22,T32	
M8	timeCheckOut()	T29,T30	
M9	setTimeKeeping()	T14,T15,T28	TimeKeeping
M10	plusValue()	T14,T15,T28	
M11	minusValue()	T14,T15,T28	
M12	plusIdx()	T14,T15,T28	
M13	setTimer()	T16,T17,T18,T19,T20,T21,T22,T23,T34	Timer
M14	startTimer()	T17,T18,T19,T20,T21,T22,T23	
M15	pauseTimer()	T18	
M16	resetTimer()	T21	
M17	ringTImer()	T19,T20,T21,T23	
M18	stopRingTimer()	T20,T21	
M19	plusValue()	T16,T17,T18,T19,T20,T21,T22,T23	
M20	minusValue()	T16,T17,T18,T19,T20,T21,T22,T23	
M21	plusIdx()	T16,T17,T18,T19,T20,T21,T22,T23	
M22	startBuzzer()	T3,T4,T19,T20,T21	Buzzer
M23	stopBuzzer()	T4,T20,T21	

MID	Method	T-Link	Class
M24	setAlarm()	T1,T2,T3,T15,T33,T34	Alarm
M25	onAlarm()	T2,T3,T4,T15,T33	
M26	offAlarm()	T2,T33	
M27	nextAlarm()	T1,T2,T3,T4	
M28	ringAlarm()	T3,T4,T15	
M29	stopRingAlarm()	T4	
M30	plusValue()	T1,T2,T3,T4,T15,T33	
M31	minusValue()	T1,T2,T3,T4,T15,T33	
M32	plusIdx()	T1,T2,T3.T4,T15,T33	
M33	getIsOn()	T1,T2,T3,T4,T15,T33	AlarmInfo
M34	setIsOn()	T1,T2,T3,T4,T15,T33	
M35	getTimeStamp()	T1,T2,T3,T4,T15,T33	
M36	setTimeStamp()	T1,T2,T3,T4,T15,T33	
M37	startStopwatch()	T9,T10,T11,T12	Stopwatch
M38	pauseStopwatch()	T10,T11,T12	
M39	resetStopwatch()	T12	
M40	recordLap()	T11,T12	
M41	timezoneToRight	T26,T27,T28	WorldTime
M42	timezoneToLeft	T25,T27,T28	
M43	setModeSelection()	T3,T7,T8,T34	ModeSelection
M44	cancelModeSelection()	T31	
M45	plusValue()	T3,T8	
M46	minusValue()	T3,T8	
M47	confirmValue()	T3,T8	
M48	plusIdx()	T3,T8	

MID	Method	T-Link	Class
M49	startLottery	T5,T6	Lottery
M50	resetLottery	T6	
M51	addTimestamp	T14,T15	TimeData
M52	setTimestamp	T14,T15	
M53	addNoActionTime()	T29,T30	
M54	setNoActionTime()	T29,T30	
M55	getTimestamp()	T14,T15	
M56	getNoActionTime	T29,T30	
M57	processOnScreen()	Every Tests	Mode
M58	processBackground()	Every Tests	
M59	getCurrentEvent()	Every Tests	
M60	setThreadMode()	T3,T8,T30	
M61	getInstance()	Every Tests	GlobalSettings
M62	getTimeData()	T14,T15,T28	
M63	getTimeZone()	Every Tests	
M64	getBuzzer()	T3,T4,T19,T20,T21	
M65	getModes()	T3,T7,T8	
M66	getCurMode()	Every Tests	
M67	setModes()	T3,T8	
M68	setCurMode()	T8,T29,T30,T31	
M69	changeMode()	T32	
M70	longToLocalDateTime()	Every Tests	
M71	isTicking()	Every Tests	

TID	name
T1	알람이 세팅한 시간대로 저장되는지 확인
T2	알람이 세팅한 시간에 울리는지 확인
T3	6개 중 4개 모드에 포함하지 않았을 때 울리는지 확인
T4	알람이 울릴 때, 아무 버튼이나 누르면 알람이 꺼지는지 확인
T5	+버튼을 눌렀을 때 로또 번호가 나오는지 확인
T6	로또 번호가 나왔을 때, -버튼을 눌러 초기화 되는지 확인
T7	아무 모드에서나 Mode버튼을 길게 눌렀을 때(Long Mode버튼을 눌렀을 때) ModeSelection이 되는지 확인
T8	ModeSelection에서 설정이 끝난 후, 모드가 잘 설정되는지 확인
T9	Stopwatch에서 정지상태에서 잘 시작되는지 확인
T10	Stopwatch에서 실행상태에서 일시정지 되는지 확인
T11	Stopwatch에서 Lap이 기록되는지 확인
T12	Stopwatch에서 일시 정지상태에서 초기화 되는지 확인

TID	name
T13	시계를 처음 실행했을 때, 현재 시간 확인
T14	시간을 설정했을 때, 현재 시간이 바뀌는지 확인
T15	시간을 설정했을 때, 변경된 시간에 따라 알람이 울리는지 확인
T16	타이머를 설정이 정상적으로 되는지 확인
T17	타이머가 정상 시작하는지 확인
T18	타이머가 정상 정지하는지 확인
T19	타이머가 종료되었을 때, 버저가 울리는지 확인
T20	버저가 울릴 때, 아무 버튼이나 눌렀을 때 꺼지는 확인
T21	버저를 껐을 때, 시간이 설정 시간으로 바뀌는지 확인
T22	타이머를 시작하고, 다른 모드로 이동해도 시간이 가는지 확인
T23	타이머를 시작하고, ModeSelection에서 타이머를 제거했을 때, 시간이 0이 되었을 타이밍에 버저가 울리는지 확인

TID	name
T24	초기 화면에서 시간대가 -11로 시작하는지 확인
T25	시간대가 -11일 때, -버튼을 누르면 시간대가 +12로 이동하는지 확인
T26	시간대가 +12일 때, +버튼을 누르면 시간대가 -11로 이동하는지 확인
T27	24개의 시간대에서 각각 대표 도시와 시간이 잘 표시되는가 확인
T28	TimeKeeping에서 현재 시간을 변경했을 때, WorldTime에서 변경된 시간에 따라 출력하는지 확인
T29	20초동안 아무 입력도 하지 않으면 첫 모드로 이동하는지 확인
T30	ModeSelection에서 모드 변경 후, 20초동안 아무 입력도 하지 않으면 변경한 첫 모드로 이동하는지 확인
T31	ModeSelection을 실행한 다음, Mode버튼을 누르면 첫 모드로 이동하는가
T32	Mode 버튼을 눌렀을 때, 다음 모드로 이동하는가
T33	알람이 Off인 상태에서 세팅한 시간에 안 울리는지 확인
T34	시간 설정중인 상황에서 ModeSelection에 진입이 가능한가

OOPT Stage 2060 Test

감사합니다