일정 관리 DWS

Team 1

201811169 김재현, 201611261 민지호
201611276 이규은, 201811301 한지희

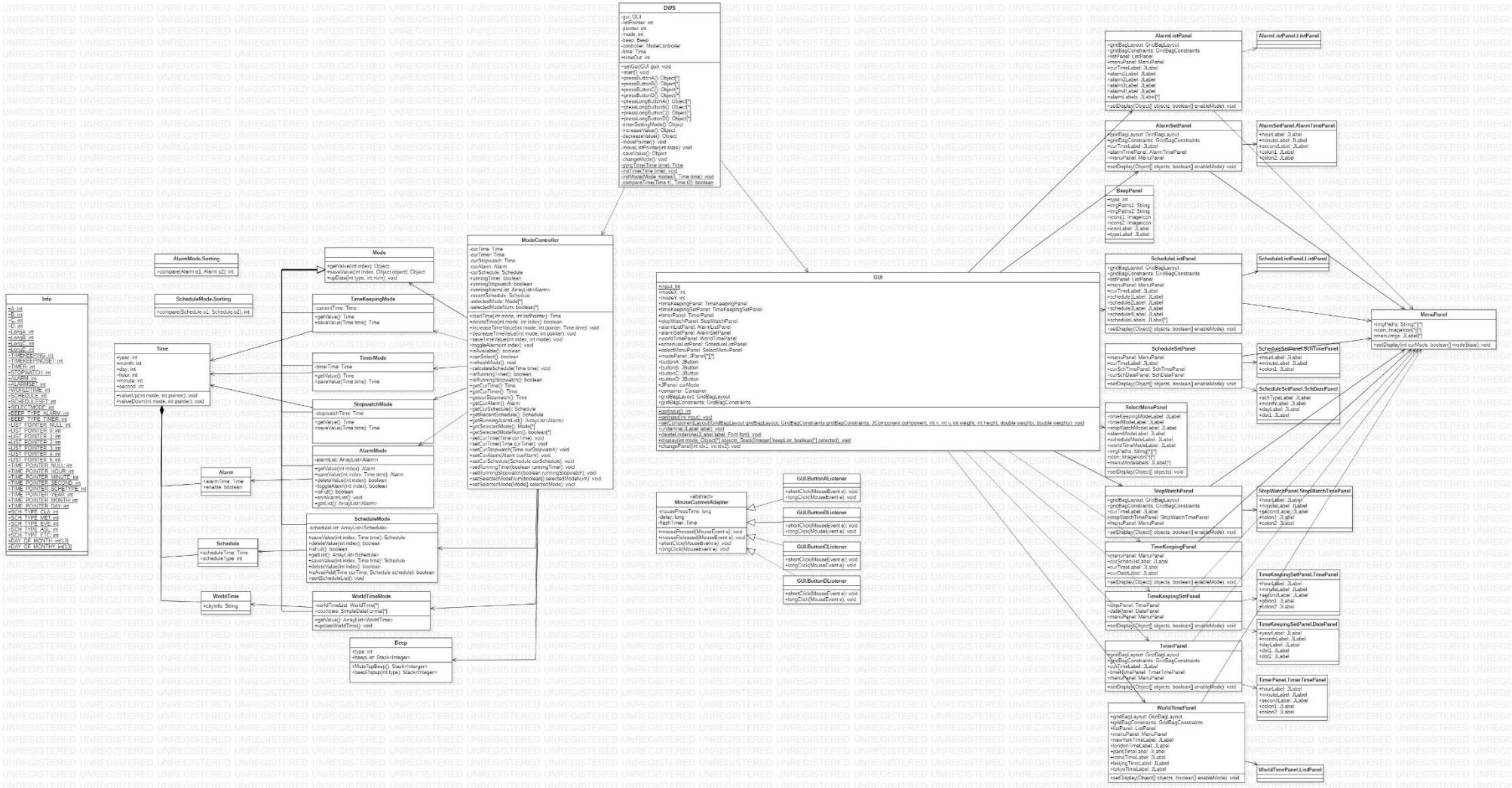
- 2051 Class, Method
- 2052 Reports, UI, Storyboards
- 2055 Interaction Diagrams
- 2061 Design Class Diagrams
- 2063 Traceability Analysis
- 2067 Traceability Analysis

INDEX

Index

2051

2051 Implement Class & Methods Definitions



2051 Implement Class & Methods Definitions

Name	DWS
Type	Class
Purpose	User와 상호작용하며 Digital Watch를 작동시키는 클래스
OverView	N/A
Cross Reference	R1, R2, R3, R4, R5, R6, R7, R8, R9 “Listing Schedule”, “Add Schedule”, “Modify Schedule”, “Delete Schedule”, “Calculate Recent Schedule”, “Set Current Time”, “Set Timer”, “Start Timer”, “Pause Timer”, “Reset Timer”, “Start Stopwatch”, “Pause Stopwatch”, “Reset Stopwatch”, “Listing Alarm”, “Set Alarm”, “Enable Alarm”, “Disable Alarm”, “Modify Alarm”, “Delete Alarm”, “Listing WorldTime”, “Change Mode”, “Select Mode”, “Time Out”, Beep”, “Mute Beep”, “Display”
Exceptional Courses of Events	N/A

2051 Implement Class & Methods Definitions

Name	ModeController
Type	Class
Purpose	6개의 Mode를 관리하는 클래스
OverView	N/A
Cross Reference	R7.1 “Change Mode”, R7.2 “Select Mode”
Exceptional Courses of Events	N/A

2051 Implement Class & Methods Definitions

Name	Mode
Type	Class
Purpose	modeController에서 모든 Mode들을 편리하게 관리하기 위해 만든 클래스
OverView	N/A
Cross Reference	N/A
Exceptional Courses of Events	N/A

2051 Implement Class & Methods Definitions

Name	TimeKeepingMode
Type	Class
Purpose	Time Keeping Mode와 관련된 데이터와 함수를 담은 클래스
OverView	N/A
Cross Reference	R2 “Set Current Time”
Exceptional Courses of Events	N/A

2051 Implement Class & Methods Definitions

Name	TimerMode
Type	Class
Purpose	Timer Mode와 관련된 데이터와 함수를 담은 클래스
OverView	N/A
Cross Reference	R3 “Set Timer”, “Start Timer”, “Pause Timer”, “Reset Timer”
Exceptional Courses of Events	N/A

2051 Implement Class & Methods Definitions

Name	StopwatchMode
Type	Class
Purpose	Stopwatch Mode와 관련된 데이터와 함수를 담은 클래스.
OverView	N/A
Cross Reference	R4 “Start Stopwatch”, “Pause Stopwatch”, “Reset Stopwatch”
Exceptional Courses of Events	N/A

2051 Implement Class & Methods Definitions

Name	AlarmMode
Type	Class
Purpose	Alarm Mode와 관련된 데이터와 함수를 담은 클래스
OverView	N/A
Cross Reference	R5 “Listing Alarm”, “Set Alarm”, “Enable Alarm”, “Disable Alarm”, “Modify Alarm”, “Delete Alarm”
Exceptional Courses of Events	N/A

2051 Implement Class & Methods Definitions

Name	ScheduleMode
Type	Class
Purpose	Schedule Mode와 관련된 데이터와 함수를 담은 클래스
OverView	N/A
Cross Reference	R1 “Listing Schedule”, “Add Schedule”, “Modify Schedule”, “Delete Schedule”, “Calculate Recent Schedule”
Exceptional Courses of Events	N/A

2051 Implement Class & Methods Definitions

Name	WorldTimeMode
Type	Class
Purpose	World Time Mode와 관련된 데이터와 함수를 담은 클래스
OverView	N/A
Cross Reference	R6 “Listing WorldTime”
Exceptional Courses of Events	N/A

2051 Implement Class & Methods Definitions

Name	Beep
Type	Class
Purpose	Alarm와 Timer의 시간이 되면 팝업 창을 띄우게 하는 데이터와 기능을 담은 클래스
OverView	N/A
Cross Reference	R8 “Beep”, “Mute Beep”
Exceptional Courses of Events	N/A

2051 Implement Class & Methods Definitions

Name	Time
Type	Class
Purpose	시간 정보를 담기 위한 클래스
OverView	N/A
Cross Reference	N/A
Exceptional Courses of Events	N/A

2051 Implement Class & Methods Definitions

Name	Alarm
Type	Class
Purpose	알람 정보를 담기 위한 클래스
OverView	N/A
Cross Reference	N/A
Exceptional Courses of Events	N/A

2051 Implement Class & Methods Definitions

Name	Schedule
Type	Class
Purpose	스케줄 정보를 담기 위한 클래스
OverView	N/A
Cross Reference	N/A
Exceptional Courses of Events	N/A

2051 Implement Class & Methods Definitions

Name	WorldTime
Type	Class
Purpose	세계 시각 정보를 담기 위한 클래스
OverView	N/A
Cross Reference	N/A
Exceptional Courses of Events	N/A

2051 Implement Class & Methods Definitions

Name	GUI
Type	Class
Purpose	모든 GUI를 출력하는 클래스
OverView	N/A
Cross Reference	R9 “Display”
Exceptional Courses of Events	N/A

2052 Implement Windows

Name	PressButtonA
Responsibilities	사용자가 버튼 A를 누름
Type	GUI
Cross Reference	R1.2, R1.3, R2.1, R3.1, R5.2, R5.5, R7.2, R8.1
Note	- 현재 모드가 Schedule Mode, Alarm Mode, Select Mode일 경우, 커서를 위로 이동 - 현재 모드가 Schedule Mode, Time Keeping Mode, Timer Mode, Alarm Mode이면서 Setting을 하는 상태일 경우, 커서가 가리키는 시간 단위의 값을 1 증가 - Beep이 발생했을 경우, Mute Beep
Pre-Conditions	N/A
Post-Conditions	N/A

2052 Implement Windows

Name	PressButtonB
Responsibilities	사용자가 버튼 B를 누름
Type	GUI
Cross Reference	R1.2, R1.3, R2.1, R3.2, R3.3, R4.1, R4.2, R5.2, R5.5, R7.2, R8.2
Note	- 현재 모드가 Time Keeping Mode일 경우, 현재 시간을 Setting하는 화면으로 전환 - 현재 모드가 Schedule Mode, Alarm Mode일 경우, 현재 포인팅 된 schedule이나 alarm을 수정(modify)하는 화면으로 전환 - 현재 모드가 Schedule Mode, Time Keeping Mode, Alarm Mode이면서 Setting을 하는 상태일 경우, 현재까지 입력된 정보를 저장 - 현재 모드가 Timer Mode, Stopwatch Mode이면서 각 Timer와 Stopwatch가 흐르지 않는 상태일 경우 시작, 각 Timer와 Stopwatch가 흐르는 상태일 경우 일시정지 - 현재 모드가 Select Mode일 경우, 현재 포인팅된 모드 항목의 활성화 상태를 선택 - Beep이 발생했을 경우, Mute Beep
Pre-Conditions	N/A
Post-Conditions	N/A

2052 Implement Windows

Name	PressButtonC
Responsibilities	사용자가 버튼 C를 누름
Type	GUI
Cross Reference	R1.2, R1.3, R2.1, R3.1, R5.2, R5.5, R7.2, R8.2
Note	- 현재 모드가 Schedule Mode, Alarm Mode, Select Mode일 경우, 커서를 아래로 이동 - 현재 모드가 Schedule Mode, Time Keeping Mode, Alarm Mode이면서 Setting을 하는 상태일 경우, 커서가 가리키는 시간 단위의 값을 1 감소 - 현재 모드가 Timer Mode일 경우 포인터를 다음 시간 항목으로 이동 - Beep이 발생했을 경우, Mute Beep
Pre-Conditions	N/A
Post-Conditions	N/A

2052 Implement Windows

Name	PressButtonD
Responsibilities	사용자가 버튼 D를 누름
Type	GUI
Cross Reference	R1.2, R1.3, R5.2, R5.5, R7.1, R7.2, R8.2
Note	• 현재 모드가 Time Keeping Mode, Alarm Mode, Schedule Mode이면서 Setting하고 있는 상태일 경우, 포인터를 다음 시간 항목으로 이동 • 현재 모드가 Select Mode일 경우, 사용자가 선택한 모드 4개를 활성화하고 저장 • 현재 모드가 Time Keeping Mode, Timer Mode, Stopwatch Mode, Alarm Mode, World Time Mode, Schedule Mode일 경우, 사용자가 선택한 모드 4개를 토대로 다음 모드로 전환 • Beep이 발생했을 경우, Mute Beep
Pre-Conditions	N/A
Post-Conditions	N/A

2052 Implement Windows

Name	PressLongButtonA
Responsibilities	사용자가 버튼 A를 2초 이상 누름
Type	GUI
Cross Reference	R5.3, R5.4, R8.2
Note	- 현재 모드가 Alarm Mode 상태면서 alarm이 1개 이상 있을 경우 현재 포인팅 된 알람 활성화 상태를 선택 - Beep이 발생했을 경우, Mute Beep
Pre-Conditions	N/A
Post-Conditions	N/A

2052 Implement Windows

Name	PressLongButtonB
Responsibilities	사용자가 버튼 B를 2초 이상 누름
Type	GUI
Cross Reference	R1.2, R3.4, R4.3, R5.2, R8.2
Note	• 현재 모드가 Schedule Mode, Alarm Mode일 경우 각 schedule 또는 alarm을 추가(add)하는 화면으로 전환 • 현재 모드가 Timer Mode, Stopwatch Mode이면서 현재 Timer와 Stopwatch가 일시정지 상태일 경우 이를 Reset • Beep이 발생했을 경우, Mute Beep
Pre-Conditions	N/A
Post-Conditions	N/A

2052 Implement Windows

Name	PressLongButtonC
Responsibilities	사용자가 버튼 C를 2초 이상 누름
Type	GUI
Cross Reference	R1.4, R5.6, R8.2
Note	• 현재 모드가 Schedule Mode, Alarm Mode일 경우 현재 포인팅 된 Schedule 또는 Alarm을 삭제 • Beep이 발생했을 경우, Mute Beep
Pre-Conditions	N/A
Post-Conditions	N/A

2052 Implement Windows

Name	PressLongButtonD
Responsibilities	사용자가 버튼 D를 2초 이상 누름
Type	GUI
Cross Reference	R7.2, R8.2
Note	• 현재 모드가 Time Keeping Mode일 경우 Select Mode 화면으로 전환 • Beep이 발생했을 경우, Mute Beep
Pre-Conditions	N/A
Post-Conditions	N/A

2055 Write Unit Test Code – Alarm Mode Test

```java
@Test
public void makingAlarmMode(){
    assertEquals(new ArrayList<Alarm>(), alarmMode.getList());
}
@Test
public void setAlarm(){
    Time tmp = new Time();

    assertEquals(tmp, alarmMode.saveValue( index: -1, tmp).alarmTime);
    assertEquals( expected: 1, alarmMode.getList().size());
}

@Test
public void modifyAlarm(){
    Time tmp = new Time();
    alarmMode.saveValue( index: -1, tmp);

    tmp.hour = 1;
    assertEquals(tmp, alarmMode.saveValue( index: 0, tmp).alarmTime);
}

@Test
public void toggleAlarm(){
    Time tmp = new Time();
    alarmMode.saveValue( index: -1, tmp);

    //turn off a alarm
    alarmMode.toggleAlarm( index: 0);
    assertFalse(alarmMode.getValue( index: 0).enable);

    //turn on a alarm
    alarmMode.toggleAlarm( index: 0);
    assertTrue(alarmMode.getValue( index: 0).enable);
}
```

```java
@Test
public void isFull(){
    for(int i=0; i<4; i++){
        assertFalse(alarmMode.isFull());
        Time tmp = new Time();
        tmp.hour = i;
        alarmMode.saveValue( index: -1, tmp);
    }

    assertTrue(alarmMode.isFull());
}

@Test
public void sortAlarm() {
    //insert alarms in reverse order. (4,3,2,1)
    for (int i = 0; i < 4; i++) {
        Time tmpTime = new Time();
        tmpTime.hour = 4 - i;
        alarmMode.saveValue( index: -1, tmpTime);
    }

    //checking alarms are in regular order.
    for(int i=0; i<4; i++)
        assertEquals( expected: i+1, alarmMode.getList().get(i).alarmTime.hour);
}

@Test
public void deleteAlarm(){
    assertFalse(alarmMode.deleteValue( index: 0));

    Time tmp = new Time();
    alarmMode.saveValue( index: -1, tmp);

    assertTrue(alarmMode.deleteValue( index: 0));
}
```

2055 Write Unit Test Code – Schedule Mode Test

```java
@Test
public void makingScheduleMode() { assertEquals(new ArrayList<Schedule>(), scheduleMode.getList()); }
@Test
public void setSchedule(){
    Time tmp = new Time();
    Schedule tmpSchedule = scheduleMode.saveValue( index: -1, tmp);

    assertEquals(tmp, tmpSchedule.scheduleTime);
    assertEquals(Info.SCH_TYPE_CLA, tmpSchedule.scheduleType);
    assertEquals( expected: 1, scheduleMode.getList().size());
}

@Test
public void modifySchedule(){
    Time tmp = new Time();
    Schedule tmpSchedule = scheduleMode.saveValue( index: -1, tmp);

    tmpSchedule.scheduleTime.hour = 1;
    Schedule tmpSchedule2 = scheduleMode.saveValue( index: 0, tmpSchedule.scheduleTime);
    assertEquals( expected: 1, tmpSchedule2.scheduleTime.hour);
}

@Test
public void deleteSchedule(){
    assertFalse(scheduleMode.deleteValue( index: 0));

    Time tmp = new Time();
    Schedule tmpSchedule = scheduleMode.saveValue( index: -1, tmp);
    assertTrue(scheduleMode.deleteValue( index: 0));
}
```

```java
@Test
public void isAvailableAdd(){
    Time tmpTime = new Time();
    tmpTime.month = 1;
    tmpTime.day = 1;
    tmpTime.hour = 1;

    Schedule schedule = new Schedule(); //01.01 00:00:00
    assertFalse(scheduleMode.isAvailAdd(tmpTime, schedule));

    Schedule schedule2 = new Schedule();
    schedule2.scheduleTime.hour = 2;    //01.01 02:00:00
    assertTrue(scheduleMode.isAvailAdd(tmpTime, schedule2));
}

@Test
public void sortSchedule() {
    //insert schedules in reverse order. (4,3,2,1)
    for (int i = 0; i < 4; i++) {
        Time tmpTime = new Time();
        tmpTime.hour = 4 - i;
        scheduleMode.saveValue( index: -1, tmpTime);
    }

    //checking schedules are in regular order.
    for(int i=0; i<4; i++)
        assertEquals( expected: i+1, scheduleMode.getList().get(i).scheduleTime.hour);
}
```

2055 Write Unit Test Code – Timer Mode Test

```java
@Test
public void makeTimerMode(){
    assertEquals( expected: 0, timerMode.getValue().year);
    assertEquals( expected: 0, timerMode.getValue().month);
    assertEquals( expected: 0, timerMode.getValue().day);
    assertEquals( expected: 0, timerMode.getValue().hour);
    assertEquals( expected: 0, timerMode.getValue().minute);
    assertEquals( expected: 0, timerMode.getValue().second);
}

@Test
public void setTimerMode(){
    Time tmp = new Time();
    tmp.hour = 1;
    timerMode.saveValue(tmp);

    assertEquals( expected: 0, timerMode.saveValue(tmp).year);
    assertEquals( expected: 0, timerMode.saveValue(tmp).month);
    assertEquals( expected: 0, timerMode.saveValue(tmp).day);
    assertEquals( expected: 1, timerMode.saveValue(tmp).hour);
    assertEquals( expected: 0, timerMode.saveValue(tmp).minute);
    assertEquals( expected: 0, timerMode.saveValue(tmp).second);
}
```

2055 Write Unit Test Code – Stopwatch Mode Test

```java
@Test
public void makeStopwatchMode(){
    assertEquals( expected: 0, stopwatchMode.getValue().year);
    assertEquals( expected: 0, stopwatchMode.getValue().month);
    assertEquals( expected: 0, stopwatchMode.getValue().day);
    assertEquals( expected: 0, stopwatchMode.getValue().hour);
    assertEquals( expected: 0, stopwatchMode.getValue().minute);
    assertEquals( expected: 0, stopwatchMode.getValue().second);
}

@Test
public void setStopwatchMode(){
    Time tmp = new Time();
    tmp.hour = 1;
    stopwatchMode.saveValue(tmp);

    assertEquals( expected: 0, stopwatchMode.saveValue(tmp).year);
    assertEquals( expected: 0, stopwatchMode.saveValue(tmp).month);
    assertEquals( expected: 0, stopwatchMode.saveValue(tmp).day);
    assertEquals( expected: 1, stopwatchMode.saveValue(tmp).hour);
    assertEquals( expected: 0, stopwatchMode.saveValue(tmp).minute);
    assertEquals( expected: 0, stopwatchMode.saveValue(tmp).second);
}
```

2055 Write Unit Test Code – Time Keeping Mode Test

```java
@Test
public void makeTimeKeepingMode(){
    Time time = new Time();
    TimeKeepingMode timeKeepingMode = new TimeKeepingMode(time);
    assertEquals(time, timeKeepingMode.getValue());
}

@Test
public void setTimeKeepingMode(){
    Time time = new Time();
    TimeKeepingMode timeKeepingMode = new TimeKeepingMode(time);

    time.hour = 1;
    assertEquals(time, timeKeepingMode.saveValue(time));
}
```

2055 Write Unit Test Code – World Time Mode Test

```java
@Test
public void makeWorldTimeMode(){
    worldTimeMode.updateWorldTime();
    Calendar calendar = Calendar.getInstance();
    Date date = calendar.getTime();
    SimpleDateFormat tmp = new SimpleDateFormat( pattern: "HH:mm:ss");

    String[] worldName = {
            "America/New_York",
            "Europe/London",
            "Europe/Paris",
            "Europe/Rome",
            "Asia/Shanghai",
            "Asia/Tokyo"
    };

    for(int i=0; i<6; i++) {
        tmp.setTimeZone(TimeZone.getTimeZone(worldName[i]));
        String worldTimeValue = worldTimeMode.getValue()[i].cityInfo;
        worldTimeValue.substring(worldTimeValue.length() - 8);
        assertEquals(tmp.format(date), worldTimeValue.substring(worldTimeValue.length() - 8));
    }
}
```

2055 Write Unit Test Code – Mode Controller Test

```
@Before
public void init(){
    Time time = new Time();

    SimpleDateFormat format = new SimpleDateFormat( pattern: "yyyy:MM:dd:HH:mm:ss");
    Date date = new Date();
    String stringTimes[] = format.format(date).split( regex: ":");

    time.year = Integer.parseInt(stringTimes[0]);
    time.month = Integer.parseInt(stringTimes[1]);
    time.day = Integer.parseInt(stringTimes[2]);
    time.hour = Integer.parseInt(stringTimes[3]);
    time.minute = Integer.parseInt(stringTimes[4]);
    time.second = Integer.parseInt(stringTimes[5]);

    Mode[] modes = new Mode[6];

    modes[0] = new TimeKeepingMode(time);
    modes[1] = new TimerMode();
    modes[2] = new StopwatchMode();
    modes[3] = new AlarmMode();
    modes[4] = new WorldTimeMode();
    modes[5] = new ScheduleMode();

    modeCon = new ModeController(time, modes);
}

@Test
public void loadTime(){
    assertNotNull(modeCon.loadTime(Info.TIMEKEEPINGSET, listPointer: 0));
    assertNotNull(modeCon.loadTime(Info.TIMER, listPointer: 0));
    assertNotNull(modeCon.loadTime(Info.STOPWATCH, listPointer: 0));

    modeCon.saveTimeValue( index: -1, Info.ALARMSET);
    assertNotNull(modeCon.loadTime(Info.ALARMSET, listPointer: 0));

    boolean[] tmpModes = {true, true, false, false, true, true};
    modeCon.setSelectedModeNum(tmpModes);
    modeCon.saveTimeValue( index: -1, Info.SCHEDULESET);
    assertNotNull(modeCon.loadTime(Info.SCHEDULESET, listPointer: 0));
}
```

```
@Test
public void deleteTime(){
    modeCon.saveTimeValue( index: -1, Info.ALARMSET);
    assertNotNull(modeCon.loadTime(Info.ALARMSET, listPointer: 0));
    modeCon.deleteTime(Info.ALARM, index: 0);
    assertTrue(((AlarmMode) modeCon.getSelectedMode()[Info.ALARM/10]).getList().isEmpty());

    boolean[] tmpModes = {true, true, false, false, true, true};
    modeCon.setSelectedModeNum(tmpModes);
    modeCon.saveTimeValue( index: -1, Info.SCHEDULESET);
    assertNotNull(modeCon.loadTime(Info.SCHEDULESET, listPointer: 0));
    modeCon.deleteTime(Info.SCHEDULE, index: 0);
    assertTrue(((ScheduleMode) modeCon.getSelectedMode()[Info.SCHEDULE/10]).getList().isEmpty());
}
```

2055 Write Unit Test Code – Mode Controller Test

```
@Test
public void increaseTimeValue(){
    Time testTime = new Time();
    testTime.year = 2020;
    testTime.month = 2;
    testTime.day = 28;
    testTime.hour = 23;
    testTime.minute = 59;
    testTime.second = 59;

    modeCon.setCurTime(testTime);
    modeCon.increaseTimeValue(Info.TIMEKEEPINGSET, Info.TIME_POINTER_HOUR,  time: null);
    assertEquals( expected: 0, modeCon.getCurTime().hour);

    testTime.hour = 23;
    modeCon.setCurTimer(testTime);
    modeCon.increaseTimeValue(Info.TIMER, Info.TIME_POINTER_HOUR,  time: null);
    assertEquals( expected: 0, modeCon.getCurTimer().hour);

    testTime.hour = 23;
    Alarm testAlarm = new Alarm();
    testAlarm.alarmTime = testTime;
    modeCon.setCurAlarm(testAlarm);
    modeCon.increaseTimeValue(Info.ALARMSET, Info.TIME_POINTER_HOUR,  time: null);
    assertEquals( expected: 0, modeCon.getCurAlarm().alarmTime.hour);

    testTime.hour = 23;
    Schedule testSchedule = new Schedule();
    testSchedule.scheduleTime = testTime;
    modeCon.setCurSchedule(testSchedule);
    modeCon.increaseTimeValue(Info.SCHEDULESET, Info.TIME_POINTER_HOUR,  time: null);
    assertEquals( expected: 0, modeCon.getCurSchedule().scheduleTime.hour);

    testTime.hour = 23;
    modeCon.setCurStopwatch(testTime);
    modeCon.increaseTimeValue(Info.STOPWATCH, Info.TIME_POINTER_NULL,  time: null);
    assertEquals( expected: 0, testTime.hour);
```

```
    testTime.hour = 23;
    testTime.minute = 59;
    testTime.second = 59;
    modeCon.increaseTimeValue(Info.TIMEKEEPING, Info.TIME_POINTER_NULL, testTime);
    assertEquals( expected: 29, testTime.day);

    testTime.year = 2099;
    testTime.month = 12;
    testTime.day = 31;
    testTime.hour = 23;
    testTime.minute = 59;
    testTime.second = 59;
    modeCon.increaseTimeValue(Info.TIMEKEEPING, Info.TIME_POINTER_NULL, testTime);
    assertEquals( expected: 1900, testTime.year);
}
```

2055 Write Unit Test Code – Mode Controller Test

```
@Test
public void decreaseTimeValue(){
    Time testTime = new Time();
    testTime.year = 1900;
    testTime.month = 1;
    testTime.day = 1;

    modeCon.setCurTime(testTime);
    modeCon.decreaseTimeValue(Info.TIMEKEEPINGSET, Info.TIME_POINTER_HOUR);
    assertEquals( expected: 23, modeCon.getCurTime().hour);

    testTime.hour = 0;
    Alarm testAlarm = new Alarm();
    testAlarm.alarmTime = testTime;
    modeCon.setCurAlarm(testAlarm);
    modeCon.decreaseTimeValue(Info.ALARMSET, Info.TIME_POINTER_HOUR);
    assertEquals( expected: 23, modeCon.getCurAlarm().alarmTime.hour);

    testTime.hour = 0;
    Schedule testSchedule = new Schedule();
    testSchedule.scheduleTime = testTime;
    modeCon.setCurSchedule(testSchedule);
    modeCon.decreaseTimeValue(Info.SCHEDULESET, Info.TIME_POINTER_HOUR);
    assertEquals( expected: 23, modeCon.getCurSchedule().scheduleTime.hour);

    testTime.hour = 0;
    testTime.minute = 0;
    testTime.second = 2;
    modeCon.setCurTimer(testTime);
    modeCon.decreaseTimeValue(Info.TIMER, Info.TIME_POINTER_NULL);
    assertEquals( expected: 1, modeCon.getCurTimer().second);
    modeCon.decreaseTimeValue(Info.TIMER, Info.TIME_POINTER_NULL);
    assertEquals( expected: 0, modeCon.getCurTimer().second);
    modeCon.decreaseTimeValue(Info.TIMER, Info.TIME_POINTER_NULL);
    assertEquals( expected: 0, modeCon.getCurTimer().second);
}
```

2055 Write Unit Test Code – Mode Controller Test

```
@Test
public void saveTimeValue(){
    Time testTime = new Time();
    testTime.hour = 1;

    modeCon.setCurTime(testTime);
    modeCon.saveTimeValue( index: -1, Info.TIMEKEEPINGSET);
    assertEquals( expected: 1, ((TimeKeepingMode)(modeCon.getSelectedMode()[Info.TIMEKEEPINGSET/10])).getValue().hour);

    modeCon.setCurTimer(testTime);
    modeCon.saveTimeValue( index: -1, Info.TIMER);
    assertEquals( expected: 1, ((TimerMode)(modeCon.getSelectedMode()[Info.TIMER/10])).getValue().hour);

    modeCon.setCurStopwatch(testTime);
    modeCon.saveTimeValue( index: -1, Info.STOPWATCH);
    assertEquals( expected: 1, ((StopwatchMode)(modeCon.getSelectedMode()[Info.STOPWATCH/10])).getValue().hour);

    Alarm testAlarm = new Alarm();
    testAlarm.alarmTime = testTime;
    modeCon.setCurAlarm(testAlarm);
    modeCon.saveTimeValue( index: -1, Info.ALARMSET);
    assertEquals( expected: 1, ((AlarmMode)(modeCon.getSelectedMode()[Info.ALARMSET/10])).getValue( index: 0).alarmTime.hour);

    boolean[] tmpModes = {true, true, false, false, true, true};
    modeCon.setSelectedModeNum(tmpModes);
    Schedule testSchedule = new Schedule();
    testSchedule.scheduleTime = testTime;
    modeCon.setCurSchedule(testSchedule);
    modeCon.saveTimeValue( index: -1, Info.SCHEDULESET);
    assertEquals( expected: 1, ((ScheduleMode)(modeCon.getSelectedMode()[Info.SCHEDULESET/10])).getValue( index: 0).scheduleTime.hour);
}
```

2055 Write Unit Test Code – Mode Controller Test

```
@Test
public void canSelect(){
    boolean[] tmpModes = {true, true, false, false, true, true};
    modeCon.setSelectedModeNum(tmpModes);
    assertTrue(modeCon.canSelect());

    boolean[] tmpModes2 = {true, true, false, false, false, true};
    modeCon.setSelectedModeNum(tmpModes2);
    assertFalse(modeCon.canSelect());

    boolean[] tmpModes3 = {true, true, false, true, true, true};
    modeCon.setSelectedModeNum(tmpModes3);
    assertFalse(modeCon.canSelect());
}
```

2055 Write Unit Test Code – **Beep Test**

```java
@Test
public void Beep(){
    Beep beep = new Beep( gui: null);
    assertEquals( expected: 1, beep.beepPopup(Info.ALARM).size());
    assertEquals( expected: 2, beep.beepPopup(Info.TIMER).size());
}

@Test
public void MuteBeep(){
    Beep beep = new Beep( gui: null);
    assertEquals( expected: 1, beep.beepPopup(Info.ALARM).size());
    assertEquals( expected: 2, beep.beepPopup(Info.TIMER).size());
    assertEquals( expected: 1, beep.muteTopBeep().size());
    assertEquals( expected: 0, beep.muteTopBeep().size());
}
```

2055 Write Unit Test Code

Alarm Mode Test

Test	Duration	Result
deleteAlarm	0s	passed
isFull	0.015s	passed
makingAlarmMode	0.001s	passed
modifyAlarm	0.001s	passed
setAlarm	0s	passed
sortAlarm	0s	passed
toggleAlarm	0s	passed

Schedule Mode Test

Test	Duration	Result
deleteSchedule	0s	passed
isAvailableAdd	0s	passed
makingScheduleMode	0.001s	passed
modifySchedule	0s	passed
setSchedule	0s	passed
sortSchedule	0.002s	passed

2055 Write Unit Test Code

Timer Mode Test

Test	Duration	Result
makeTimerMode	0.001s	passed
setTimerMode	0s	passed

Stopwatch Mode Test

Test	Duration	Result
makeStopwatchMode	0.001s	passed
setStopwatchMode	0s	passed

Time Keeping Mode Test

Test	Duration	Result
makeTimeKeepingMode	0.004s	passed

2055 Write Unit Test Code

World Time Mode Test

Test	Duration	Result
makeWorldTimeMode	0.002s	passed

Mode Controller Test

Test	Duration	Result
canSelect	0.001s	passed
decreaseTimeValue	0.001s	passed
deleteTime	0.001s	passed
increaseTimeValue	0.002s	passed
loadTime	0.001s	passed
saveTimeValue	0.010s	passed

Beep Test

Test	Duration	Result
Beep	0.003s	passed
MuteBeep	0s	passed

2063

2063 System Testing

Test Number	Test	Description	Use Case	System Function	Pass
1-1	Listing Schedule Test	일정이 변동(추가/수정/삭제)되었을 때 현재에서 가까운 순서대로(월-일-시-분-타입) 재정렬되는가	Listing Schedule	R1.1	
1-2	Listing Schedule Test	현재 시간이 재설정되었을 때, 해당 시간에 따라 지난 일정이 삭제되고 저장된 일정이 재정렬되는가	Listing Schedule	R1.1	
1-3	Listing Schedule Test	일정의 세부사항(월/일/시/분/타입)이 모두 같을 경우, 일정의 등록 혹은 수정된 시간 순서대로 정렬되는가	Listing Schedule	R1.1	
1-4	Listing Schedule Test	시간이 흐름에 따라 일정이 자동으로 삭제되는가	Listing Schedule	R1.1	
2-1	Add Schedule Test	일정이 리스트에 정상적으로 추가되는가	Add Schedule	R1.2	
2-2	Add Schedule Test	일정 추가 시 세부사항이 입력되지 않으면 초기값으로 저장되는가	Add Schedule	R1.2	
2-3	Add Schedule Test	현재 시간보다 이전 시간의 일정을 추가하려고 시도하면 요청이 거부되는가	Add Schedule	R1.2	

2063 System Testing

Test Number	Test	Description	Use Case	System Function	Pass
3-1	Modify Schedule Test	수정하려고 시도할 때, 원래의 정보를 정상적으로 가져오는가	Modify Schedule	R1.3	
3-2	Modify Schedule Test	수정 후 수정된 세부사항이 정상적으로 적용되는가	Modify Schedule	R1.3	
3-3	Modify Schedule Test	일정의 시간을 수정할 때, 현재보다 이전의 시간으로 수정하려고 시도하면 요청이 거부되는가	Modify Schedule	R1.3	
4-1	Delete Schedule Test	일정이 정상적으로 삭제되는가	Delete Schedule	R1.4	
4-2	Delete Schedule Test	일정 리스트가 비어 있을 때 삭제 시도를 하면 에러가 발생하지 않는가	Delete Schedule	R1.4	
4-3	Delete Schedule Test	일정을 삭제하려고 시도했을 때, 현재 시간과 일정 시간이 정확히 일치 해 두 번의 삭제 요청이 들어갔을 때 에러가 발생하지 않는가	Delete Schedule	R1.4	

2063 System Testing

Test Number	Test	Description	Use Case	System Function	Pass
5-1	Calculate Recent Schedule Test	가장 가까운 일정만 정확하게 반환하는가	Calculate Recent Schedule	R1.5	
5-2	Calculate Recent Schedule Test	남은 일정이 존재하지 않을 때, NULL을 반환하는가	Calculate Recent Schedule	R1.5	
5-3	Calculate Recent Schedule Test	일정 리스트가 변동되었을 때, 오늘의 일정을 정확하게 반환하는가	Calculate Recent Schedule	R1.5	
5-4	Calculate Recent Schedule Test	반환되는 일정이 현재 시간과 가장 가까운 미래의 일정이 맞는가	Calculate Recent Schedule	R1.5	
6-1	Set Current Time Test	입력한 시간이 현재 시간으로 정확히 반영되는가	Set Current Time	R2.1	P

2063 System Testing

Test Number	Test	Description	Use Case	System Function	Pass
7-1	Set Timer Test	입력한 시간이 Timer 시간으로 정확히 반영되는가	Set Timer	R3.1	P
7-2	Set Timer Test	시간이 입력된 채로, 모드가 변경되면 Reset 되는가	Set Timer	R3.1	
7-3	Set Timer Test	Timer 시간을 설정할 때, 시간 설정 범위가 최소 0시0분0초, 최대 23시59분59초인가	Set Timer	R3.1	P
8-1	Start Timer Test	버튼을 눌렀을 때, 입력된 시간으로 timer가 시작되는가	Start Timer	R3.2	P
8-2	Start Timer Test	시간이 초 단위로 감소되는가	Start Timer	R3.2	P
8-3	Start Timer Test	Timer 시작 후, 다른 모드로 전환되었을 때에도 시간이 돌아온 시점-전환 시점 만큼 시간이 감소되었는가	Start Timer	R3.2	P
8-4	Start Timer Test	입력된 시간이 0시 0분 0초이면 타이머가 시작되지 않는가	Start Timer	R3.2	

2063 System Testing

Test Number	Test	Description	Use Case	System Function	Pass
8-5	Start Timer Test	일시정지 후 모드를 변경하지 않고 다시 시작 버튼을 눌렀을 때, 중단된 시간부터 Timer 시간이 작동하는가 (resume)	Start Timer	R3.2	P
9-1	Pause Timer Test	버튼을 눌렀을 때, Timer 시간이 감소되지 않고 정지하는가	Pause Timer	R3.3	P
9-2	Pause Timer Test	일시정지 후 모드를 변경한 뒤 다시 Timer 모드로 돌아왔을 때, Timer 시간이 0으로 재설정되는가	Pause Timer	R3.3	
10-1	Reset Timer Test	버튼을 눌렀을 때, Timer 시간이 0시 0분 0초로 재설정되는가	Reset Timer	R3.4	P

2063 System Testing

Test Number	Test 항목	Description	Use Case	System Function	Pass
11-1	Start Stopwatch Test	버튼을 눌렀을 때, 입력된 시간으로 stopwatch가 시작되는가	Start Stopwatch	R4.1	P
11-2	Start Stopwatch Test	시간이 초 단위로 증가되는가	Start Stopwatch	R4.1	P
11-3	Start Stopwatch Test	Stopwatch 시작 후, 다른 모드로 전환 되었을 때에도 시간이 돌아온 시 점-전환 시점 만큼 증가되었는가	Start Stopwatch	R4.1	P
11-4	Start Stopwatch Test	일시정지 후 모드를 변경하지 않고 다시 시작 버튼을 누르면 중단된 시 간부터 Stopwatch 시간이 작동되는가 (resume)	Start Stopwatch	R4.1	P

2063 System Testing

Test Number	Test 항목	Description	Use Case	System Function	Pass
12-1	Pause Stopwatch Test	버튼을 눌렀을 때, Stopwatch 시간이 증가되지 않는가	Pause Stopwatch	R4.2	P
12-2	Pause Stopwatch Test	일시정지 후 모드를 변경하면, 다시 Stopwatch 모드로 돌아왔을 때 Stopwatch 시간이 0으로 재설정되는가	Pause Stopwatch	R4.2	
13-1	Reset Stopwatch Test	버튼을 눌렀을 때, Stopwatch 시간이 0시 0분 0초로 재설정되는가	Reset Stopwatch	R4.3	P

2063

2063 System Testing

Test Number	Test	Description	Use Case	System Function	Pass
14-1	Listing Alarm Test	알람이 변동(추가/수정/삭제)되었을 때 시간 순서대로(시-분-초) 재정렬되는가	Listing Alarm	R5.1	
14-2	Listing Alarm Test	알람 시간이 같을 경우, 알람의 등록 혹은 수정된 시간 순서대로 정렬되는가	Listing Alarm	R5.1	
15-1	Set Alarm Test	알람이 리스트에 정상적으로 추가되는가	Set Alarm	R5.2	
15-2	Set Alarm Test	알람 추가 시 시간이 입력되지 않으면 초기값으로 추가되는가	Set Alarm	R5.2	
15-3	Set Alarm Test	알람 개수가 4개로 꽉 차있을 때 추가를 위해 버튼을 누르면 거부되는가	Set Alarm	R5.2	P
15-4	Set Alarm Test	알람을 설정했을 때 자동적으로 활성화되는가	Set Alarm	R5.2	P

2063 System Testing

Test Number	Test	Description	Use Case	System Function	Pass
16-1	Enable Alarm Test	알람이 비활성화 상태일 때, 버튼을 누르면 활성화 상태로 전환되는가	Enable Alarm	R5.3	
17-1	Disable Alarm Test	알람이 활성화 상태일 때, 버튼을 누르면 비활성화 상태로 전환되는가	Disable Alarm	R5.4	
18-1	Modify Alarm Test	수정 시에 원래의 정보를 정상적으로 가져오는가	Modify Alarm	R5.5	P
18-2	Modify Alarm Test	수정 후 수정된 시간이 정상적으로 적용되는가	Modify Alarm	R5.5	P
19-1	Delete Alarm Test	알람이 정상적으로 바로 삭제되는가	Delete Alarm	R5.6	P
19-2	Delete Alarm Test	알람 리스트가 비어있을 때 삭제 시도를 하면 에러가 발생하지 않는가	Delete Alarm	R5.6	P

2063 System Testing

Test Number	Test 항목	Description	Use Case	System Function	Pass
20-1	Listing World Time Test	나라별 세계 시간이 GMT 순서대로 제공되는가	Listing World Time	R6.1	P
21-1	Change Mode Test	버튼을 눌렀을 때, 모드가 지정된 순서대로 변경되는가	Change Mode	R7.1	P
22-1	Select Mode Test	사용자가 선택한 모드 설정 값이 정상적으로 반영되는가	Select Mode	R7.2	P
22-2	Select Mode Test	모드를 설정하기 전, 초기 모드 4가지가 선택되어 있는가	Select Mode	R7.2	P
23-1	Time Out Test	특정 화면에서 일정 시간이 지나면 Default Mode로 돌아가는가	Time Out	R7.3	P
23-2	Time Out Test	Timer와 Stopwatch가 작동 중일 때는 Default Mode로 돌아가지 않는가	Time Out	R7.3	P

2063 System Testing

Test Number	Test 항목	Description	Use Case	System Function	Pass
24-1	Display Test	Schedule Mode에서 일정 리스트 화면이 정렬된 순서대로 알맞게 출력되는가	Display	R9.1	
24-2	Display Test	각 Mode(Schedule, Time keeping, Timer, Stopwatch, Alarm, World Time)의 화면이 정상적으로 출력되는가	Display	R9.1	
24-3	Display Test	Time Keeping Mode에서 설정된 현재 시간이 정확히 출력되는가	Display	R9.1	P
24-4	Display Test	Time Keeping Mode에서 오늘의 일정이 화면 상단에 정확히 출력되는가	Display	R9.1	
24-5	Display Test	Beep, Mute Beep이 호출되었을 때 정상적으로 팝업이 띄워지는가	Display	R9.1	P

2063

2063 System Testing

Test Number	Test 항목	Description	Use Case	System Function	Pass
25-1	Beep Test	알람 시간과 현재 시간이 일치할 때 알람이 정상적으로 울리는가	Beep	R8.1	P
25-2	Beep Test	타이머 시간이 00이 되었을 때 알람이 정상적으로 울리는가	Beep	R8.1	P
25-3	Beep Test	두 가지 이상의 Beep이 동시에 호출되었을 때 두 개 이상의 팝업이 정상적으로 띄워지는가	Beep	R8.1	P
26-1	Mute Beep Test	알람이 울리고 있을 때 어떤 버튼을 누르든지 알람이 정상적으로 종료되는가	Mute Beep	R8.2	P
26-2	Mute Beep Test	알람이 울리고 있을 때 어떤 버튼을 누르든지 다른 기능에 영향을 미치지 않는가	Mute Beep	R8.2	P

2067

2067 Test Traceability Analysis