한판하시계

객체지향개발방법론 6팀
201611279 이동준
201611280 이동훈
201611298 정태민
201611266 성시진

CONTENTS

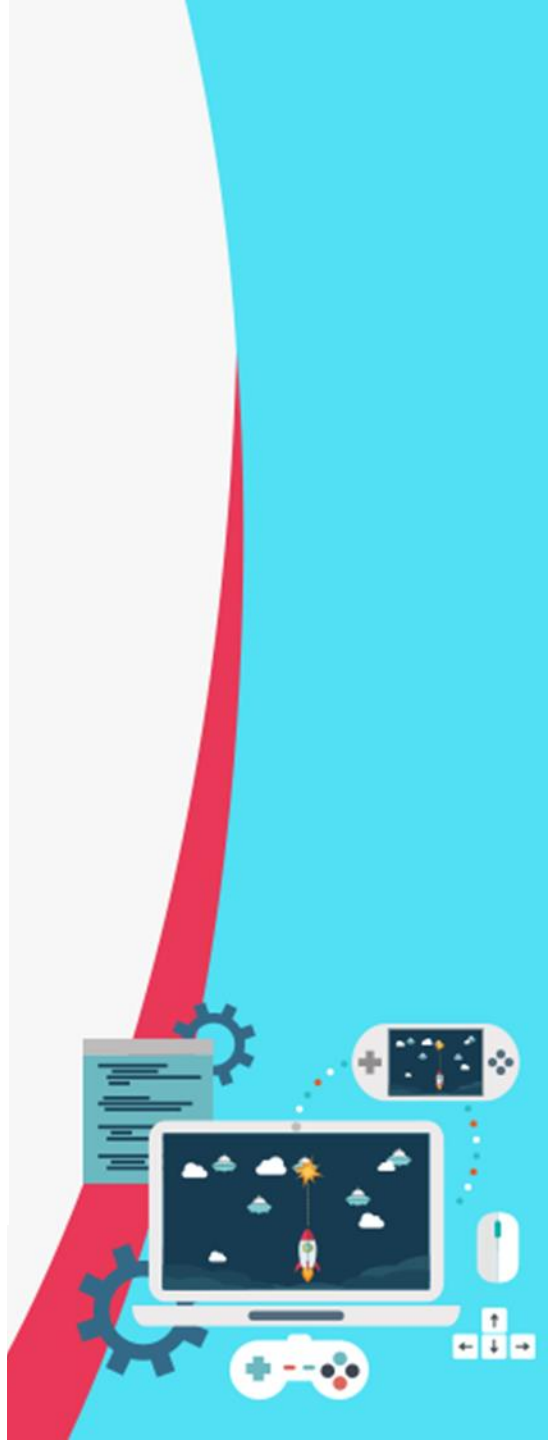
Real Product & Demo

Write Unit Test Code (Time)

```
class TimeTest {

    @Test
    void requestCurTime() {
        Time t = new Time();
        Calendar temp = (Calendar) t.curTime.clone(); // 현재시간을 저장한 임시 변수
        t.requestCurTime(); // 10ms 증가
        assertNotEquals(temp, t.curTime);// 10ms가 증가한 현재시간으로 이전의 시간과 다른지 확인
        t.WPressed(true);//setTime 할 때 현재시간은 증가하지 않음
        Calendar temp2 = (Calendar) t.curTime.clone(); // 현재시간을 저장한 임시 변수
        assertEquals(temp2, t.curTime);
    }
```

```
    @Test
    void changeTimeUnit() throws NoSuchFieldException {

        Time t = new Time();
        try {
            Field field = t.getClass().getDeclaredField("timeUnit");
            field.setAccessible(true);
            t.WPressed(true);
            for(int i =1 ; i<=5;i++) {
                t.changeTimeUnit();
                int temp = (int)(field.get(t));
                assertEquals(i+1, temp);
            }
            t.changeTimeUnit();
            int temp = (int)(field.get(t));
            assertEquals(1, temp);
        }
        catch(NoSuchFieldException e){
            e.printStackTrace();
        }
        catch (IllegalAccessException e) {
            e.printStackTrace();
        }


    }

}
```

```
@Test
void increaseTimeValue() {
    Time t = new Time();//isSetTime = false;
    t.WPressed(true);//setTime

    //test set second(59->0)
    t.curTime.set(Calendar.SECOND, 59);
    int temp1 = t.curTime.get(Calendar.MINUTE);
    t.increaseTimeValue();
    assertEquals(0, t.curTime.get(Calendar.SECOND));
    assertEquals(temp1, t.curTime.get(Calendar.MINUTE));

    //test set minute(59->0)
    t.curTime.set(Calendar.MINUTE, 59);
    int temp2 = t.curTime.get(Calendar.HOUR_OF_DAY);
    t.changeTimeUnit();
    t.increaseTimeValue();
    assertEquals(0, t.curTime.get(Calendar.MINUTE));
    assertEquals(temp2, t.curTime.get(Calendar.HOUR_OF_DAY));

    //test set hour(23->0)
    t.curTime.set(Calendar.HOUR_OF_DAY, 23);
    int temp3 = t.curTime.get(Calendar.DATE);
    t.changeTimeUnit();
    t.increaseTimeValue();
    assertEquals(0, t.curTime.get(Calendar.HOUR_OF_DAY));
    assertEquals(temp3, t.curTime.get(Calendar.DATE));

    //test set date(getLeastMaximum ->1)
    t.curTime.set(Calendar.DATE, t.curTime.getActualMaximum(Calendar.DATE));
    int temp4 = t.curTime.get(Calendar.MONTH);
    t.changeTimeUnit();
    t.increaseTimeValue();
    assertEquals(1, t.curTime.get(Calendar.DATE));
    assertEquals(temp4, t.curTime.get(Calendar.MONTH));

    //test set month(DECEMBER -> JANUARY)
    t.curTime.set(Calendar.MONTH, Calendar.DECEMBER);
    int temp5 = t.curTime.get(Calendar.YEAR);
    t.changeTimeUnit();
    t.increaseTimeValue();
    assertEquals(Calendar.JANUARY, t.curTime.get(Calendar.MONTH));
    assertEquals(temp5, t.curTime.get(Calendar.YEAR));

    //test set year(2099 -> 1970)
    t.curTime.set(Calendar.YEAR, 2099);
    t.changeTimeUnit();
    t.increaseTimeValue();
    assertEquals(1970, t.curTime.get(Calendar.YEAR));

}
```



Write Unit Test Code (Stopwatch)

```java
class StopwatchTest {

    @Test
    void requestStopwTime() {
        Stopwatch st = new Stopwatch();
        try{
            Field field = st.getClass().getDeclaredField("isPaused");
            field.setAccessible(true);
            boolean value = (boolean)field.get(st);
            assertTrue(value); // isPaused = true 확인
        } catch (NoSuchFieldException e) {
            e.printStackTrace();
        } catch (IllegalAccessException e) {
            e. printStackTrace();
        }
    }
@Test
void resetStopw() {
    Stopwatch st = new Stopwatch();
    Calendar cal = st.stopwTime;
    try{
        Field field = st.getClass().getDeclaredField("isPaused");
        field.setAccessible(true);
        st.SPressed(false); //increase stopw
        st.SPressed(false); //pause stopw
        st.WPressed(false); //reset stopw
        boolean value = (boolean)field.get(st);
        assertTrue(value); //isPaused = true 확인
        assertEquals(st.stopwTime, cal); //reset 확인
    } catch (NoSuchFieldException e) {
        e.printStackTrace();
    } catch (IllegalAccessException e) {
        e.printStackTrace();
    }
}
```

```java
@Test
void increaseStopw() {
    Stopwatch st = new Stopwatch();
    Calendar cal = st.stopwTime;
    try{
        Field field = st.getClass().getDeclaredField("isPaused");
        field.setAccessible(true);
        st.SPressed(false); // increase stopw
        boolean value = (boolean)field.get(st);
        assertFalse(value); // isPaused = false 확인
        try {
            Thread.sleep(5000);
        } catch(InterruptedException e) {
            e.printStackTrace();
        }//stopw 5초
        cal.add(Calendar.SECOND, 5);
        assertEquals(st.stopwTime, cal); // stopw 5초 확인
    } catch (NoSuchFieldException e) {
        e.printStackTrace();
    } catch (IllegalAccessException e) {
        e. printStackTrace();
    }


}
```

```java
@Test
void pauseStopw() {
    Stopwatch st = new Stopwatch();
    Calendar cal = st.stopwTime;
    try{
        Field field = st.getClass().getDeclaredField("isPaused");
        field.setAccessible(true);
        st.SPressed(false); // increase stopw
        try {
            Thread.sleep(5000);
        } catch(InterruptedException e) {
            e.printStackTrace();
        }//stopw 5초
        st.SPressed(false); // pause stopw
        boolean value = (boolean)field.get(st);
        assertTrue(value); // isPaused = true 확인
        cal.add(Calendar.SECOND, 5);
        assertEquals(st.stopwTime, cal); // stopw 5초 확인
    } catch (NoSuchFieldException e) {
        e.printStackTrace();
    } catch (IllegalAccessException e) {
        e. printStackTrace();
    }
}
```

Write Unit Test Code (Timer)

```
class TimerTest {
    @Test
    void requestTimerTime() {
        Timer temp = new Timer();
        Calendar tempCal = (Calendar)temp.timerTime.clone();

        temp.WPressed(true); // into settimer
        for(int i = 0; i < 5; i++)
            temp.SPressed(false); // timer set 5second
        temp.WPressed(false); // out settimer

        assertNotEquals(tempCal, temp.timerTime);
    }
```

```
@Test
void decreaseTimer(){
    Timer temp = new Timer();
    try {
        Field field = temp.getClass().getDeclaredField("isPaused");
        field.setAccessible(true);
        temp.WPressed(true); // into settimer
        for (int i = 0; i < 5; i++)
            temp.SPressed(false); // timer set 5second
        temp.WPressed(false); // out settimer
        temp.decreaseTimer(); // timer start => isPaused = false

        boolean ret = (boolean) field.get(temp);

        assertEquals(false, ret);
    }catch (Exception e){
        e.printStackTrace();
    }
}
```

```
@Test
void pauseTimer() {
    Timer temp = new Timer();
    try{
        Field field = temp.getClass().getDeclaredField("isPaused");
        field.setAccessible(true);
        temp.WPressed(true); // into settimer
        for(int i = 0; i < 5; i++)
            temp.SPressed(false); // timer set 5second
        temp.WPressed(false); // out settimer
        temp.decreaseTimer(); // timer start => isPaused = false

        boolean ret = (boolean) field.get(temp);

        assertEquals(false, ret);
    }catch(Exception e){
        e.printStackTrace();
    }
}
```

```
    @Test
    void resetTimer() {
        Timer temp = new Timer();
        Calendar tempCal = temp.timerTime;

        temp.WPressed(true); // into settimer
        for(int i = 0; i < 5; i++)
            temp.SPressed(false); // timer set 5second
        temp.WPressed(false); // out settimer
        temp.resetTimer(); // reset timer

        assertEquals(tempCal, temp.timerTime);

    }

}
```

Write Unit Test Code (Alarm)

```
class AlarmTest {

    @Test
    void setAlarm() {
        Alarm temp = new Alarm();
        try{
            Field field = temp.getClass().getDeclaredField("alarms");
            field.setAccessible(true);
            Calendar tempCal = ((LinkedList<Calendar>)field.get(temp)).get(0);

            temp.WPressed(true); // into setalarm
            for(int i = 0; i < 5; i ++)
                temp.SPressed(false); // increase 5 minutes
            temp.WPressed(false); // out setalarm

            List<Calendar> ret = (LinkedList<Calendar>)field.get(temp);

            assertNotEquals(tempCal, ret.get(0));
        }catch (Exception e){
            e.printStackTrace();
            return;
        }
    }
    @Test
    void changeAlarmIndex() {
        Alarm temp = new Alarm();
        try{
            Field field = temp.getClass().getDeclaredField("index");
            field.setAccessible(true);
            int curIndex = (int)field.get(temp);

            temp.WPressed(false); // next alarm

            assertNotEquals(curIndex, (int)field.get(temp));

        }catch (Exception e){
            e.printStackTrace();
            return;
        }
    }
```

```
    @Test
    void changeAlarmToggle() {
        Alarm temp = new Alarm();
        try{
            Field field = temp.getClass().getDeclaredField("toggle");
            field.setAccessible(true);

            temp.SPressed(false); // alarm toggle

            LinkedList<Boolean> ret = (LinkedList<Boolean>)field.get(temp);

            assertEquals(false, ret.get(0));

        }catch (Exception e){
            e.printStackTrace();
            return;
        }
    }
    @Test
    void requestAlarm() {
        Alarm temp = new Alarm();
        try{
            String[] tempString = new String[3];
            tempString = temp.requestAlarm();

            temp.WPressed(true); // into setalarm
            for(int i = 0; i < 5; i ++)
                temp.SPressed(false); // increase 5 minutes
            temp.WPressed(false); // out setalarm

            assertNotEquals(tempString, temp.requestAlarm());
        }catch (Exception e){
            e.printStackTrace();
            return;
        }
    }

}
```

Write Unit Test Code (Worldtime-1)

```
class WorldtimeTest {

    @Test
    void requestWorldtime() {
        Time time = new Time();
        Worldtime worldtime = new Worldtime(time.curTime);
        try{
            Field field1 = worldtime.getClass().getDeclaredField("GMT9");
            Field field2 = worldtime.getClass().getDeclaredField("worldClock");
            Field field3 = worldtime.getClass().getDeclaredField("isSummerTime");
            Field field4 = worldtime.getClass().getDeclaredField("city");
            Field field5 = worldtime.getClass().getDeclaredField("curCity");
            worldtime.calWorldTime();
            Calendar temp1 = (Calendar) field1.get(worldtime);
            Calendar temp2 = (Calendar) field2.get(worldtime);
            String[] temp3 = (String[]) field4.get(worldtime);

            assertEquals(temp3[(int)field5.get(worldtime)], "TOKYO"); // 시간대 변경 전 GMT+9 대표 도시 도쿄 출력 확인
            assertEquals((boolean)field3.get(worldtime), false); // 서머타임 미적용 상태 출력 확인
            assertEquals((temp2.getTimeInMillis() - temp1.getTimeInMillis()) / (60 * 60 * 1000), 0); // *
            //assertEquals(temp2.get(Calendar.HOUR_OF_DAY), temp1.get(Calendar.HOUR_OF_DAY));
            assertEquals(temp2.get(Calendar.MINUTE), temp1.get(Calendar.MINUTE)); // *
            assertEquals(temp2.get(Calendar.SECOND), temp1.get(Calendar.SECOND)); // *: 올바른 시각 출력 확인

            worldtime.changeIsSummertime(); // 서머타임 적용 상태로 변경
            worldtime.calWorldTime();
            assertEquals(temp3[(int)field5.get(worldtime)], "TOKYO"); // 서머타임만 적용 후 현재 도시 도쿄 출력 확인
            assertEquals((boolean)field3.get(worldtime), true); // 서머타임 적용 상태 출력 확인
            assertEquals((temp2.getTimeInMillis() - temp1.getTimeInMillis()) / (60 * 60 * 1000), 1); // *
            //assertEquals(temp2.get(Calendar.HOUR_OF_DAY) - 1, temp1.get(Calendar.HOUR_OF_DAY));
            assertEquals(temp2.get(Calendar.MINUTE), temp1.get(Calendar.MINUTE)); // *
            assertEquals(temp2.get(Calendar.SECOND), temp1.get(Calendar.SECOND)); // *: 서머타임 적용 후 올바른 시각 출력 확인

            worldtime.changeIsSummertime(); // 서머타임 미적용 상태로 변경
            worldtime.changeCity(); // 시간대 GMT+10으로 변경
            worldtime.calWorldTime();
            assertEquals(temp3[(int)field5.get(worldtime)], "SYDNEY"); // GMT+10 대표 도시 시드니 출력 확인
            assertEquals((boolean)field3.get(worldtime), false); // 서머타임 미적용 상태 출력 확인
            assertEquals((temp2.getTimeInMillis() - temp1.getTimeInMillis()) / (60 * 60 * 1000), 1); // *
            //assertEquals(temp2.get(Calendar.HOUR_OF_DAY) - 1, temp1.get(Calendar.HOUR_OF_DAY));
            assertEquals(temp2.get(Calendar.MINUTE), temp1.get(Calendar.MINUTE)); // *
            assertEquals(temp2.get(Calendar.SECOND), temp1.get(Calendar.SECOND)); // *: 올바른 시각 출력 확인
            worldtime.changeIsSummertime(); // 서머타임 적용 상태로 변경
            worldtime.calWorldTime();
            assertEquals(temp3[(int)field5.get(worldtime)], "SYDNEY"); // 서머타임만 적용 후 현재 도시 시드니 출력 확인
            assertEquals((boolean)field3.get(worldtime), true); // 서머타임 적용 상태 출력 확인
            assertEquals((temp2.getTimeInMillis() - temp1.getTimeInMillis()) / (60 * 60 * 1000), 2); // *
            //assertEquals(temp2.get(Calendar.HOUR_OF_DAY) - 2, temp1.get(Calendar.HOUR_OF_DAY));
            assertEquals(temp2.get(Calendar.MINUTE), temp1.get(Calendar.MINUTE)); // *
            assertEquals(temp2.get(Calendar.SECOND), temp1.get(Calendar.SECOND)); // *: 서머타임 적용 후 올바른 시각 출력 확인
        } catch (NoSuchFieldException e) {
            e.printStackTrace();
        } catch (IllegalAccessException e) {
            e.printStackTrace();
        }
    }
```

```
    @Test
    void changeCity() {
        Time time = new Time();
        Worldtime worldtime = new Worldtime(time.curTime);
        try {
            Field field1 = worldtime.getClass().getDeclaredField("curCity");
            Field field2 = worldtime.getClass().getDeclaredField("timeDiff");
            Field field3 = worldtime.getClass().getDeclaredField("city");
            field1.setAccessible(true);
            field2.setAccessible(true);
            field3.setAccessible(true);
            int field2List[] = (int[]) field2.get(worldtime);
            String field3List[] = (String[]) field3.get(worldtime);

            worldtime.changeCity();
            worldtime.changeCity();
            worldtime.changeCity(); // GMT+9에서 GMT+12로 이동
            assertEquals((int) field1.get(worldtime), 23); // GMT-11이 0번째 시간대면 GMT+12는 23번째 시간대 확인
            assertEquals(field2List[(int) field1.get(worldtime)], 3); // GMT+9와 GMT+12는 3시간 차이 확인
            assertEquals(field3List[(int) field1.get(worldtime)], "WELLINGTON"); // GMT+12를 대표하는 도시 웰링턴 확인

            worldtime.changeCity(); // GMT+12에서 GMT-11로 이동
            assertEquals((int) field1.get(worldtime), 0); // GMT-11은 0번째 시간대 확인
            assertEquals(field2List[(int) field1.get(worldtime)], -20); // GMT+9와 GMT-11는 -20시간 차이 확인
            assertEquals(field3List[(int) field1.get(worldtime)], "PAGO PAGO"); // GMT-11를 대표하는 도시 파고파고 확인
        } catch (NoSuchFieldException e) {
            e.printStackTrace();
        } catch (IllegalAccessException e) {
            e.printStackTrace();
        }
    }
```

Write Unit Test Code (Worldtime-2)

```java
@Test
void calWorldTime() {
    Time time = new Time();
    Worldtime worldtime = new Worldtime(time.curTime);
    try {
        Field field1 = worldtime.getClass().getDeclaredField("GMT9");
        Field field2 = worldtime.getClass().getDeclaredField("worldClock");
        worldtime.calWorldTime();
        Calendar temp1 = (Calendar) field1.get(worldtime);
        Calendar temp2 = (Calendar) field2.get(worldtime);
        assertEquals(temp1, temp2); // 현재 시각은 GMT+9

        worldtime.changeCity();
        worldtime.changeCity();
        worldtime.changeCity(); // GMT+12로 변경
        worldtime.calWorldTime();
        temp2 = (Calendar) field2.get(worldtime);
        assertEquals((temp2.getTimeInMillis() - temp1.getTimeInMillis()) / (60 * 60 * 1000), 3); // GMT+9와 GMT+12는 3시간 차이 확인

        worldtime.changeIsSummertime(); // 서머타임 적용
        worldtime.calWorldTime();
        temp2 = (Calendar) field2.get(worldtime);
        assertEquals((temp2.getTimeInMillis() - temp1.getTimeInMillis()) / (60 * 60 * 1000), 4); // 서머타임 적용 시 GMT+9와 GMT+12는 4시간 차이 확인

        worldtime.changeIsSummertime(); // 서머타임 미적용
        worldtime.changeCity(); // GMT+12 -> GMT-11 변경
        worldtime.calWorldTime();
        temp2 = (Calendar) field2.get(worldtime);
        assertEquals((temp2.getTimeInMillis() - temp1.getTimeInMillis()) / (60 * 60 * 1000), -20); // GMT+9와 GMT-11은 -20시간 차이 확인

        worldtime.changeIsSummertime(); // 서머타임 적용
        worldtime.calWorldTime();
        temp2 = (Calendar) field2.get(worldtime);
        assertEquals((temp2.getTimeInMillis() - temp1.getTimeInMillis()) / (60 * 60 * 1000), -19); // 서머타임 적용 시 GMT+9와 GMT-11은 -19시간 차이 확인
    } catch (NoSuchFieldException e) {
        e.printStackTrace();
    } catch (IllegalAccessException e) {
        e.printStackTrace();
    }
}
```

```java
@Test
void changeIsSummertime() {
    boolean temp;
    Time time = new Time();
    Worldtime worldtime = new Worldtime(time.curTime);
    try {
        Field field = worldtime.getClass().getDeclaredField("isSummerTime");
        field.setAccessible(true);
        temp = (boolean) field.get(worldtime); // 처음 서머타임 미적용 상태
        worldtime.changeIsSummertime(); // 서머타임 미적용 -> 적용
        assertNotEquals((boolean) field.get(worldtime), temp); // 서머타임 적용 상태 확인
        worldtime.changeIsSummertime(); // 서머타임 적용 -> 미적용
        assertEquals((boolean) field.get(worldtime), temp); // 서머타임 미적용 상태 확인
    } catch (NoSuchFieldException e) {
        e.printStackTrace();
    } catch (IllegalAccessException e) {
        e.printStackTrace();
    }
}
```

Write Unit Test Code (Game)

```
class GameTest {

    @Test
    void start() {
        Game game = new Game();
        try{
            Field field = game.getClass().getDeclaredField("gameState");
            field.setAccessible(true);
            int value = (int)field.get(game);
            assertEquals(value, 0); // gameState = START_GAME_STATE 확인
            game.buttonPressed(); // gmae start
            value = (int)field.get(game);
            assertEquals(value, 1); // gameState = GAME_PLAYING_STATE
        } catch (NoSuchFieldException e) {
            e.printStackTrace();
        } catch (IllegalAccessException e) {
            e. printStackTrace();
        }
    }
}
```

Unit Testing

Tests passed: 20 of 20 tests – 10 s 331 ms

<default package>	10 s 331 ms
AlarmTest	156 ms
changeAlarmIndex()	134 ms
requestAlarm()	5 ms
setAlarm()	13 ms
changeAlarmToggle()	4 ms
GameTest	62 ms
start()	62 ms
StopwatchTest	10 s 5 ms
resetStopw()	2 ms
increaseStopw()	5 s 1 ms
pauseStopw()	5 s 1 ms
requestStopwTime()	1 ms
TimerTest	56 ms
resetTimer()	27 ms
requestTimerTime()	10 ms
decreaseTimer()	9 ms
pauseTimer()	10 ms
TimeTest	25 ms
requestCurTime()	12 ms
increaseTimeValue()	7 ms
changeTimeUnit()	6 ms
WorldtimeTest	27 ms
changeCity()	7 ms
requestWorldtime()	9 ms
calWorldTime()	5 ms
changeIsSummertime()	6 ms

System Testing (Time)

Time Test		
No.	Name	Test Description
1-1	requestCurTime	현재 시간이 흐르는지, 시각 설정 시 시간 흐름이 멈추는지 확인
1-2	changeTimeUnit	시각 설정 시 해당 함수를 실행시켰을 때 선택된 시각 단위가 초->분->시->일->월->년->초 순으로 변경 되는지 확인
1-3	increaseTimeValue	시각 설정 시 해당 함수를 실행시켰을 때 선택된 시각 단위 값이 증가하는지, 그 단위 최댓값에서 실행 시 최솟값으로 변경되는지 확인

System Testing (Stopwatch)

Stopwatch Test		
No.	Name	Test Description
2-1	requestStopwTime	처음 시계 전원을 켤 시 스톱워치가 정지 상태인지 확인
2-2	resetStopw	스톱워치 시작한 후 일시정지 했을 때 함수 실행 시 0분 0초 0센티초로 초기화 되는지 확인
2-3	increaseStopw	스톱워치 시작 후 스톱워치 시간이 잘 증가하는지 확인
2-4	pauseStopw	스톱워치 시작하고 일정 시간 이후 일시정지 했을 시 시간 증가가 멈추고 흐른 시간에 맞게 화면에 표시되는지 확인

System Testing (Timer)

Timer Test		
No.	Name	Test Description
3-1	requestTimerTime	타이머 시간 설정 시 시간이 잘 설정되고 설정한 시간이 화면에 잘 보이는지 확인
3-2	decreaseTimer	타이머 시작 시 타이머 시간이 일시정지나 초기화 또는 0시 0분 0초가 될 때까지 1초 간격으로 잘 감소하는지 확인
3-3	pauseTimer	타이머 시작 이후 함수 호출 시 일시정지 되는지 확인
3-4	resetTimer	함수 호출 시 타이머 시간이 0시 0분 0초로 초기화 되는지 확인

System Testing (Alarm)

Alarm Test		
No.	Name	Test Description
4-1	setAlarm	설정한 알람 시각이 제대로 입력되었는지 확인
4-2	changeAlarmIndex	함수 호출 시 다음 알람으로 넘어가지는지 확인 4번째 알람에서 호출 시 1번째 알람으로 넘어가지는지 확인
4-3	changeAlarmToggle	함수 호출 시 알람이 활성화(on)상태일 때 비활성화(off)로, 비활성화(off)상태일 때 활성화(on)로 전환되는지, 비활성화(off)된 알람이 설정한 시각에 울리지 않는지 확인
4-4	requestAlarm	알람 시각 설정 시 시각이 잘 설정되고 설정한 시각이 화면에 잘 보이는지 확인

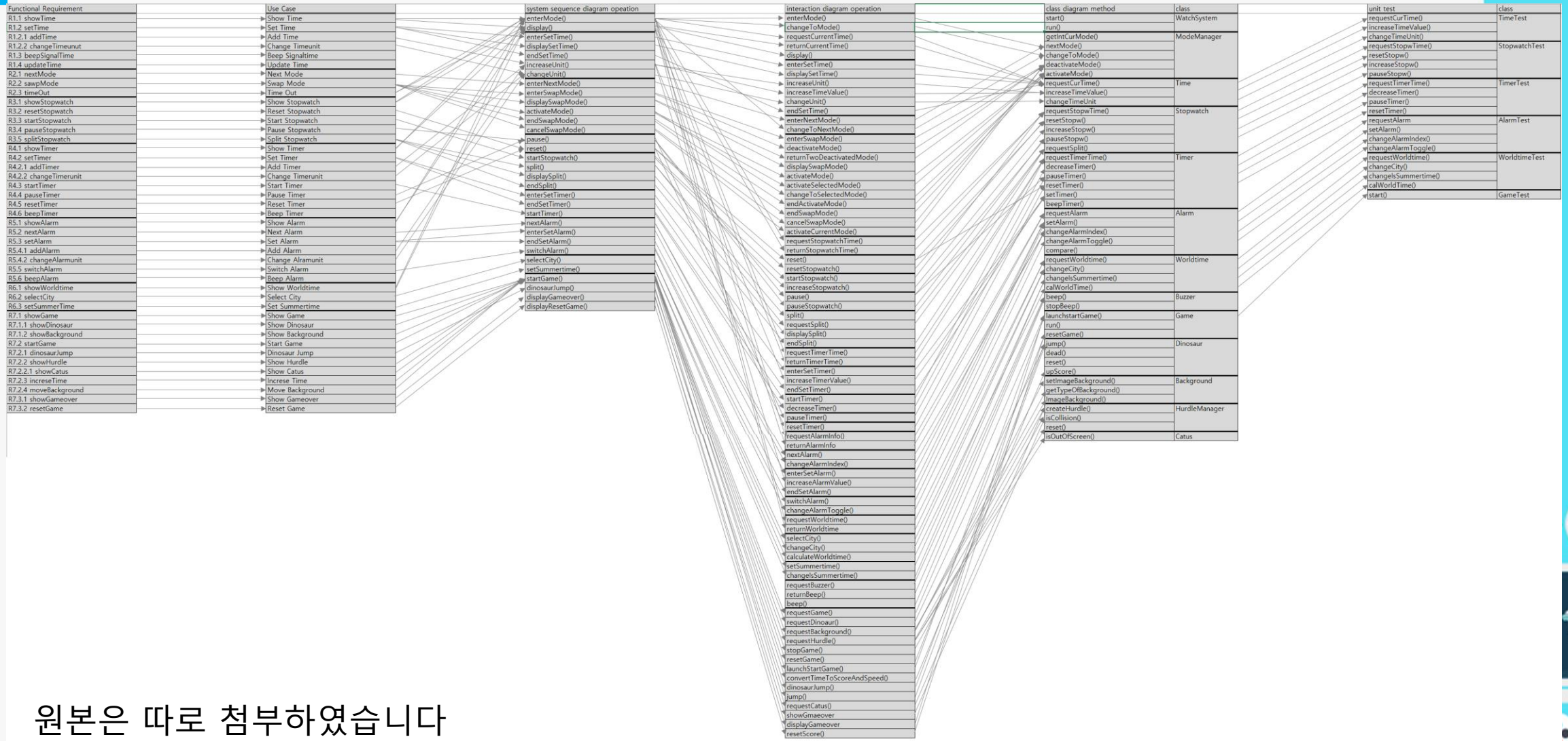
System Testing (Worldtime)

Worldtime Test		
No.	Name	Test Description
5-1	requestWorldtime	현재 보고있는 도시 이름과 그 도시의 시각을 잘 불러오는지 확인
5-2	changeCity	현재 보고있는 도시가 다음 시간대의 도시로 바뀌는지 확인
5-3	calWorldTime	현재 보고있는 도시와 서머타임 여부에 따라 세계 시각 계산이 제대로 되는지 확인
5-4	changeIsSummertime	서머타임 활성화 시에는 비활성화로 비활성화 시에는 활성화로 서머타임 여부를 전환하는지 확인

System Testing (Game)

Game Test		
No.	Name	Test Description
6-1	start	게임을 플레이 하고 있지 않을 시 게임 플레이를 시작하고 게임 플레이 상태를 게임 중으로 바꾸는지 확인

Testing Traceability Analysis



원본은 따로 첨부하였습니다

Q&A

THANK YOU!