B5조

테마시계

김유진 201811241 이현우 201511288 류수진 201811249 서푸름 201811265

B5조
목차

1.1
Overview

Reliability

22 NEW Bugs

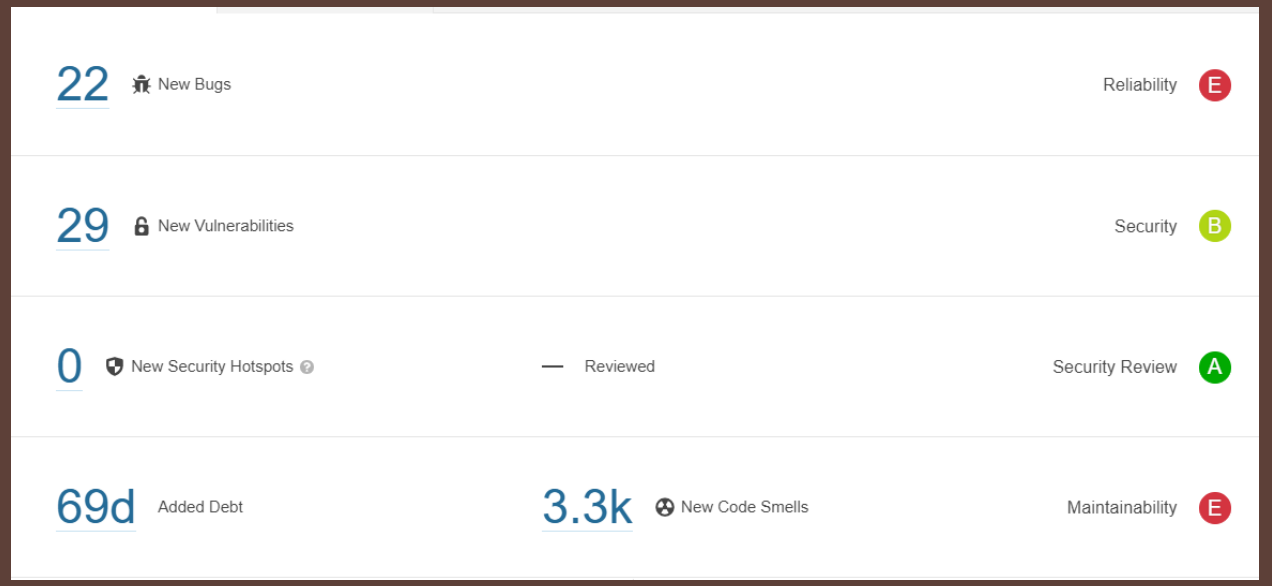
Security

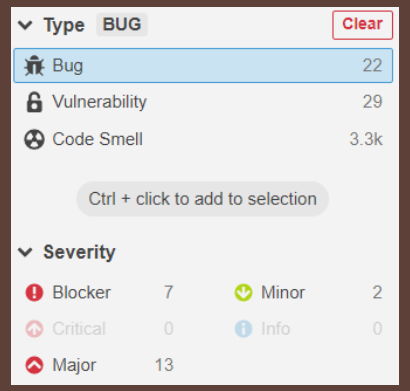
29 NEW Vulnerabilities

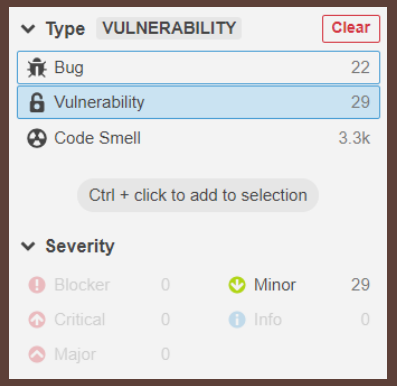
Security Review

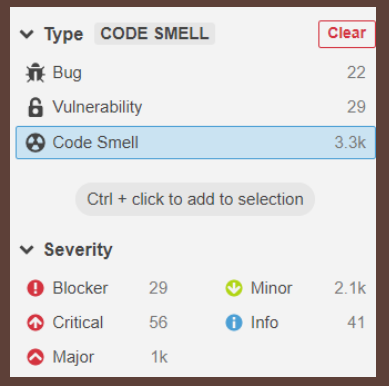
0 Security hotspot

1.1
Overview


22 New Bugs
Reliability E
29 New Vulnerabilities
Security B
0 New Security Hotspots
— Reviewed
Security Review A
69d Added Debt
3.3k New Code Smells
Maintainability E


Type BUG
Clear
Bug 22
Vulnerability 29
Code Smell 3.3k
Ctrl + click to add to selection
Severity
Blocker 7
Minor 2
Critical 0
Info 0
Major 13


Type VULNERABILITY
Clear
Bug 22
Vulnerability 29
Code Smell 3.3k
Ctrl + click to add to selection
Severity
Blocker 0
Minor 29
Critical 0
Info 0
Major 0


Type CODE SMELL
Clear
Bug 22
Vulnerability 29
Code Smell 3.3k
Ctrl + click to add to selection
Severity
Blocker 29
Minor 2.1k
Critical 56
Info 41
Major 1k

1.2
Bugs

Add an end condition to this loop. (False Alarms)

•모두 spining 하는 쓰레드에 대한 False Alarm임.

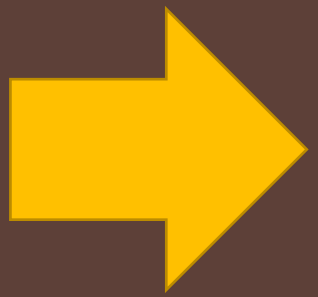
src/AlarmMode.java

Add an end condition to this loop. Why is this an issue? 13 days ago L49
Bug Blocker Open Not assigned 15min effort Comment cert

```
        alarmChecker = new Thread(new Runnable() {
            @Override
            public void run() {
                while(true) {
                    try {
                        Thread.sleep(6000);
                    } catch (InterruptedException e) {
                        e.printStackTrace();
                    }
                    if(checkRingAlarmExist()){
                        try {
                            Thread.sleep(60000);
                        } catch (InterruptedException e) {
                            e.printStackTrace();
                        }
                    }
                }
```

Add an end condition to this loop. Why is this an issue? 13 days ago L49
Bug Blocker Open Not assigned 15min effort Comment cert

```
            }
        });
        alarmChecker.start();
    }
```



생략

위의 무한 루프문은 새 쓰레드에서 수행되는 코드로 무한히 루프하는 것이 쓰레드의 목적에 적절하므로 버그라고 볼 수 없다

1.2
Bugs

Either re-interrupt this method or rethrow the "InterruptedException"

- 원인 : 쓰레드에서 발생한 exception은 단순 logging 처리로 끝나면 안되고, exception을 throw하거나 re-inturrupt 해야 한다는 경고문.

```
@Test public void buzzerMultifulRingTest()
{
    Buzzer buzzer = new Buzzer();
    buzzer.OnBuzzer();
    try {
        Thread.sleep(15000);
    } catch (InterruptedException e) {

Either re-interrupt this method or rethrow the "InterruptedException". Why is this an issue?   4 hours ago ▾ L80
Bug ▾ Major ▾ Open ▾ Not assigned ▾ 15min effort Comment   cwe, error-handling, multi-threading ▾

        e.printStackTrace();
    }
    buzzer.OnBuzzer();
```

Vulnerability #93

[Bug - Major] Either re-interrupt this method or rethrow the "InterruptedException".

[B5] Taehyeong Kim이(가) 3일 전에 추가함. 2일 전에 수정됨.

상태:	Closed	시작시간:	2020/06/19
우선순위:	Normal	완료기한:	
담당자:	-	진척도:	100%
목표버전:	1.0.2	추정시간:	

설명

해당 취약점을 개선합니다.

링크 : http://sonar.third.kr/project/issues?id=B5%3Atest&resolved=false&severities=MAJOR&sinceLeakPeriod=true&types=BUG

이력 | Notes | Property changes

푸름 서이(가) 2일 전에 변경

- 상태을(를) *New*에서 *Rejected*(으)로 변경되었습니다.

테스트 케이스 클래스에서 발생한 에러여서, 수정할 필요가 없을 것 같습니다.

[B5] Taehyeong Kim이(가) 2일 전에 변경

- 상태을(를) *Rejected*에서 *Feedback*(으)로 변경되었습니다.

Test클래스 이외에도 해당 취약점이 발생한 Class의 코드를 수정해주세요.

http://sonar.third.kr/project/issues?id=B5%3Atest&open=AXKYtCprkxvrUYqgCP85&resolved=false&severities=MAJOR&sinceLeakPeriod=true&types=BUG

푸름 서이(가) 2일 전에 변경

- 상태을(를) *Feedback*에서 *Resolved*(으)로 변경되었습니다.
- 진척도을(를) *0*에서 *100*(으)로 변경되었습니다.

해당 문제가 발생한 클래스의 목록을 모두 확인하였고,
해당 문제에 대한 픽스를 적용하였습니다.

[B5] Taehyeong Kim이(가) 2일 전에 변경

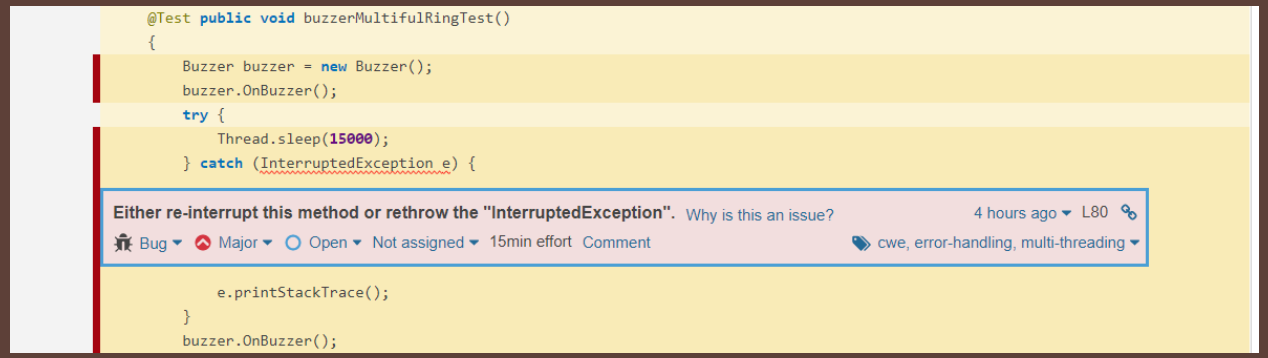
- 상태을(를) *Resolved*에서 *Closed*(으)로 변경되었습니다.

쓰레드에서 발생한 예외처리를 단순 logging으로 끝내고 있음. Thread.currentThread().interrupt(); 문을 추가하여 re-interrupting을 권장함.

1.2 Bugs

Either re-interrupt this method or rethrow the "InterruptedException"

• 원인 : 쓰레드에서 발생한 exception은 단순 logging 처리로 끝나면 안되고, exception을 throw하거나 re-inturrupt 해야 한다는 경고문.




```
world_time_updater = new Thread(new Runnable() {
    @Override
    public void run() {
        while (true) {
            if(is_world_time) {
                try {
                    Thread.sleep( millis: 1000);
                    if(is_world_time)
                    syncWorldTime();
                } catch (InterruptedException e) {
                    Thread.currentThread().interrupt();
                }
            }else{
                Thread.yield();
            }
        }
    }
});
```

```
public void initStopWatchMode() {
    stop_watch_running = new Thread(new Runnable() {
        @Override
        public void run() {
            try {
                while(true) {
                    if(!testStopWatchMax() && is_stop_watch_running) {
                        Thread.sleep( millis: 1000);
                        if(!testStopWatchMax() && is_stop_watch_running)
                        increaseStopWatchSeconds();
                    }else{
                        Thread.yield();
                    }
                }
            } catch (InterruptedException e) {
                Thread.currentThread().interrupt();
            }
        }
    });
```

쓰레드에서 발생한 예외처리를 단순 logging으로 끝내고 있음. Thread.currentThread().interrupt(); 문을 추가하여 re-interrupting을 권장함.

1.2
Bugs

```
void initAlarmMode(){
    buzzer = Buzzer.getInstance();
    timeManager = TimeManager.getInstance();
    MappingAlarmMode();
    for(int i=0;i<4;i++){
        alarms[i] = new Alarm( hour: 0, minute: 0, actived: -1);
    }
    //시간을 0으로 초기화 해주고, activated를 모두 false로 초기화해준다.
    //1분마다 checkringalarmExist 호출
    alarmChecker = new Thread(new Runnable() {
        @Override
        public void run() {
            while(true) {
                try {
                    Thread.sleep( millis: 6000);
                } catch (InterruptedException e) {
                    Thread.currentThread().interrupt();
                }
                if(checkRingAlarmExist()){
                    try {
                        Thread.sleep( millis: 60000);
                    } catch (InterruptedException e) {
                        Thread.currentThread().interrupt();
                    }
                }
            }
        }
    });
    alarmChecker.start();
}
```

```
private Buzzer(){
    sound_player = new Thread(new Runnable() {
        @Override
        public void run() {
            URL url  = Thread.currentThread().getContextClassLoader().getResource( name: "beep.wav");
            try {
                while(true){
                    if(is_buzzer_running){
                    AudioInputStream audioIn = AudioSystem.getAudioInputStream(url);
                    // Get a sound clip resource.
                    Clip clip = null;
                    clip = AudioSystem.getClip();
                    // Open audio clip and load samples from the audio input stream.
                    clip.open(audioIn);
                    clip.start();
                    Thread.sleep( millis: 300);
                    }else{
                        Thread.yield();
                    }
                }
            } catch (LineUnavailableException e) {
                java.lang.System.out.println("EX:SoundError");
            } catch (UnsupportedAudioFileException e) {
                java.lang.System.out.println("EX:AudioFileNotValid");
            } catch (IOException | InterruptedException e) {
                java.lang.System.out.println("EX:Inturrupted");
                if (e instanceof InterruptedException) {
                    Thread.currentThread().interrupt();
                }
            }
        }
    });
    sound_player.start();
    instance = ModeManagerInteracter.getInstance();
}
```

```
    Thread displayUpdater = new Thread(new Runnable() {
        @Override
        public void run() {
            while(true){
                upper_segment.setText(system.getSegmentContentUpper());
                lower_segment.setText(system.getSegmentContentLower());
                upper_segment.setForeground(system.getTextColor());
                lower_segment.setForeground(system.getTextColor());
                upper_panel.setBackground(system.getBackGroundColor());
                lower_panel.setBackground(system.getBackGroundColor());
                panel.setBackground(system.getBackGroundColor());
                try {
                    Thread.sleep( millis: 50);
                } catch (InterruptedException e) {
                    Thread.currentThread().interrupt();
                }
            }
        }
    });
    displayUpdater.start();
}
```

```
public void InitCurrentTimeMode() {
    timeManager = TimeManager.getInstance();
    currentTimeUpdater = new Thread(new Runnable() {
        @Override
        public void run() {
            try {
                while(true) {
                    if(is_watch_running) {
                        Thread.sleep( millis: 1000);
                        if(is_watch_running)
                        SyncWithCurrentTime();
                    }else{
                        Thread.yield();
                    }
                }
            } catch (InterruptedException e) {
                Thread.currentThread().interrupt();
            }
        }
    });
    currentTimeUpdater.start();
}
```

```
void initWorldTimeMode() {
    worlds[0] = new World(new Time( seconds: 0, minute: 0, hour: -17, day: 0, mounth: 0, year: 0), name: "WASHINGTON");
    worlds[1] = new World(new Time( seconds: 0, minute: 0, hour: -9, day: 0, mounth: 0, year: 0), name: "LONDON");
    worlds[2] = new World(new Time( seconds: 0, minute: 0, hour: -8, day: 0, mounth: 0, year: 0), name: "PARIS");
    worlds[3] = new World(new Time( seconds: 0, minute: 0, hour: -7, day: 0, mounth: 0, year: 0), name: "BERLIN");
    worlds[4] = new World(new Time( seconds: 0, minute: 0, hour: 0, day: 0, mounth: 0, year: 0), name: "TOKYO");
    worlds[5] = new World(new Time( seconds: 0, minute: 0, hour: -8, day: 0, mounth: 0, year: 0), name: "ROMA");
    worlds[6] = new World(new Time( seconds: 0, minute: 0, hour: -14, day: 0, mounth: 0, year: 0), name: "OTTAWA");
    worlds[7] = new World(new Time( seconds: 0, minute: 0, hour: -6, day: 0, mounth: 0, year: 0), name: "MOCKBA");
    worlds[8] = new World(new Time( seconds: 0, minute: 0, hour: -7, day: 0, mounth: 0, year: 0), name: "BRUSSELS");
    worlds[9] = new World(new Time( seconds: 0, minute: 0, hour: 0, day: 0, mounth: 0, year: 0), name: "SEOUL");
    worlds[10] = new World(new Time( seconds: 0, minute: 0, hour: -1, day: 0, mounth: 0, year: 0), name: "BEIJING");
    worlds[11] = new World(new Time( seconds: 0, minute: -30, hour: -3, day: 0, mounth: 0, year: 0), name: "DELHI");
    worlds[12] = new World(new Time( seconds: 0, minute: 0, hour: -2, day: 0, mounth: 0, year: 0), name: "ZAKARTA");
    worlds[13] = new World(new Time( seconds: 0, minute: 0, hour: -6, day: 0, mounth: 0, year: 0), name: "ANKARA");
    worlds[14] = new World(new Time( seconds: 0, minute: 0, hour: -6, day: 0, mounth: 0, year: 0), name: "RIYADH");
    worlds[15] = new World(new Time( seconds: 0, minute: 0, hour: -7, day: 0, mounth: 0, year: 0), name: "PRETORIA");
    worlds[16] = new World(new Time( seconds: 0, minute: 0, hour: -15, day: 0, mounth: 0, year: 0), name: "MAXICO");
    worlds[17] = new World(new Time( seconds: 0, minute: 0, hour: -12, day: 0, mounth: 0, year: 0), name: "BRASILIA");
    worlds[18] = new World(new Time( seconds: 0, minute: 0, hour: -12, day: 0, mounth: 0, year: 0), name: "BUENOS AIRES");
    worlds[19] = new World(new Time( seconds: 0, minute: 0, hour: 1, day: 0, mounth: 0, year: 0), name: "SYDNEY");
    world_time_updater = new Thread(new Runnable() {
        @Override
        public void run() {
            while (true) {
                if(is_world_time) {
                    try {
                        Thread.sleep( millis: 1000);
                        if(is_world_time)
                        syncWorldTime();
                    } catch (InterruptedException e) {
                        Thread.currentThread().interrupt();
                    }
                }else{
                    Thread.yield();
                }
            }
        }
    });
    world_time_updater.start();
}
```

1.2 Bugs

Unused field: Alarm.ringTime

• 원인 : 원래 설계가 변경되어 사용하지 않는 필드가 생겼으나, 이를 삭제하지 않았음

```
public class Alarm{
    private int hour;
    private int minute;
    private int actived; // -1: none, 0: disable, 1:ensable
    private String[] state = {"NONE","OFF","ON"};

    public String getState() { return state[actived + 1]; }

    public void setHour(int hour) { this.hour = hour; }

    public int getHour() { return hour; }

    public void setMinute(int minute) { this.minute = minute; }

    public int getMinute() { return minute; }

    public void setActive(int active) { this.actived = active; }

    public int getActive() { return actived; }

    public Alarm(int hour, int minute, int actived){
        this.hour = hour;
        this.minute = minute;
        this.actived = actived;
    }


}
```

Alarm Class의 ring Time 필드는 사용되지 않았으므로 삭제.

new Buzzer() invokes Thread.start()

•생성자에서 쓰레드를 실행하는 것은 Buzzer 클래스를 상속받는 클래스의 생성자가 있을 경우, 해당클래스의 생성자가 실행되기 전에 스레드가 먼저 실행될 수 있고 오작동의 원인이 될 수 있음.

```
public final class Buzzer {

    private boolean is_buzzer_running = false;
    private Thread sound_player;
    private Thread reserve;
    private ModeManagerInteracter instance;
    private final static long RESERVE_OFF_TIME_IN_MIL = 30000;
    private static Buzzer singleton;

    public static Buzzer getInstance(){
        if(singleton == null){
            singleton = new Buzzer();
        }
        return singleton;
    }
```

Buzzer는 상속이 필요 없는 싱글톤 객체이므로, final키워드 추가

1.2
Vulnerability

Use a logger to log this exception.

•Throwable.printStackTrace 는 민감한 정보를 출력할 수 있으므로 개발자가 정의한 Logger를 사용하기를 권장하는 내용.

```
} catch (LineUnavailableException e) {
    java.lang.System.out.println("EX:SoundError");
} catch (UnsupportedAudioFileException e) {
    java.lang.System.out.println("EX:AudioFileNotValid");
} catch (IOException | InterruptedException e) {
    java.lang.System.out.println("EX:Inturrupted");
    if (e instanceof InterruptedException) {
        Thread.currentThread().interrupt();
    }
}
```

다른 방법으로 디버깅을 위한 최소한의 정보만 제공

Explicitly declare the visibility for "hour"

•일부 명시적으로 visibility (public, private ...)가 정의되지 않은 field에 대해
•명시적인 visibility 사용을 권장하는 에러임.

```
private TimeManager time_manager = TimeManager.getInstance();
private Segment segment;
private Thread world_time_updater = new Thread();
private World[] worlds = new World[20];
private int world_index = 0;
private boolean locked = false;     // 초기에는 locked 되어있지 않
private Time current_time = new Time();
private boolean is_world_time = false;
```

```
private int theme_index = 0;
private Theme[] themes = new Theme[8];
private Segment segment;
```

```
  private int hour;
  private int minute;
  private int actived; // -1: none, 0: disable, 1:ensable
  private String[] state = {"NONE","OFF","ON"};
```

Visibility를 모두 명시함

1.5
Code Coverage

Element	Class, %	Method, %	Line, %
Alarm	100% (1/1)	37% (3/8)	50% (8/16)
AlarmMode	62% (5/8)	47% (17/36)	55% (62/112)
ButtonActionCallback			
Buzzer	100% (4/4)	90% (9/10)	79% (38/48)
CurrentTimeMode	100% (2/2)	72% (8/11)	79% (23/29)
MethodCallback			
Mode			
ModeManager	100% (3/3)	91% (22/24)	87% (95/108)
ModeManagerInteracter	100% (1/1)	100% (7/7)	100% (17/17)
Segment	100% (1/1)	80% (8/10)	92% (25/27)
StandardCallback			
StopWatchMode	100% (14/14)	65% (29/44)	75% (88/117)
System	0% (0/1)	0% (0/10)	0% (0/18)
Theme	100% (1/1)	100% (4/4)	100% (8/8)
ThemeMode	100% (1/1)	53% (7/13)	73% (30/41)
Time	100% (1/1)	47% (11/23)	59% (62/104)
TimeManager	100% (1/1)	100% (5/5)	100% (9/9)
TimerMode	100% (14/14)	63% (30/47)	66% (95/143)
WatchGUI	0% (0/7)	0% (0/14)	0% (0/105)
World	100% (1/1)	100% (1/1)	100% (4/4)
WorldTimeMode	100% (2/2)	52% (9/17)	61% (47/76)

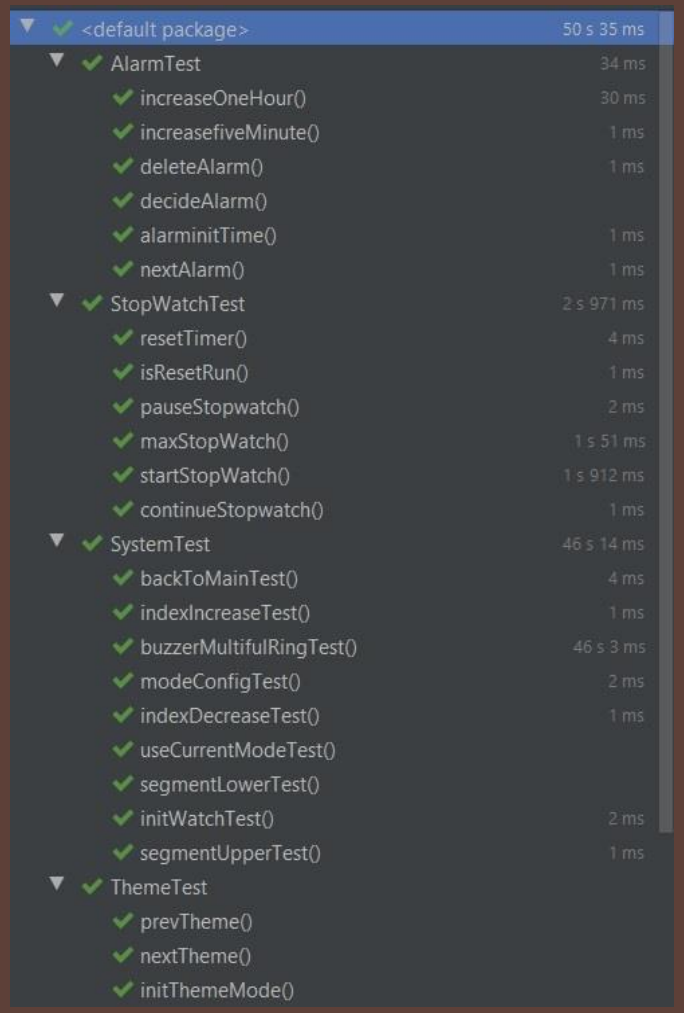
1.5
Code Coverage

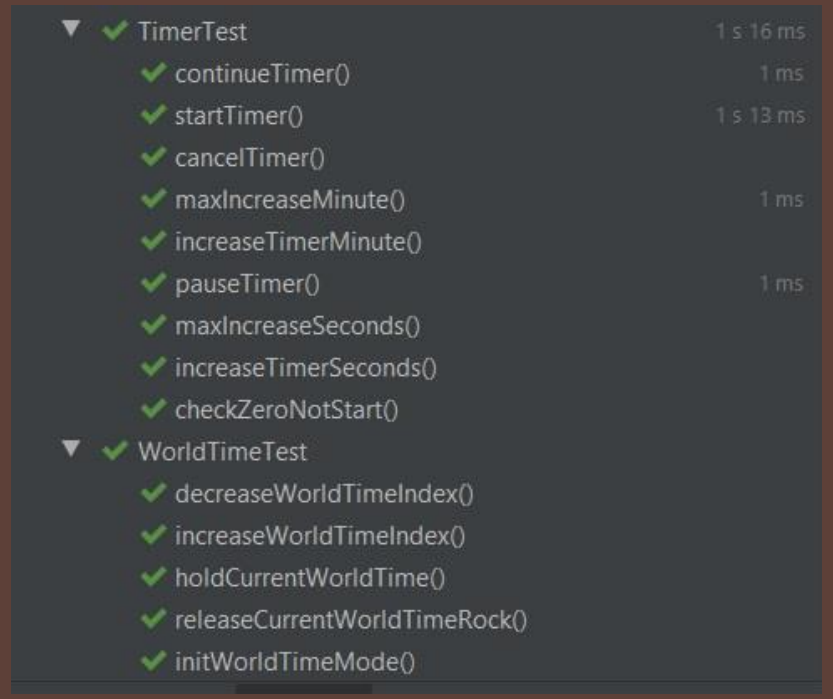
- GUI, Callback 클래스를 제외한 대부분의 Class 에 대해 50% 이상의 Method Coverage를 보였다.

- Time, Alarm Class의 Cover되지 않은 일부 Method를 위한 UnitTest 작성이 필요해보임.

- Time, Alarm클래스의 메소드들은 단순히 getter, sette뿐이므로, 굳이 테스트 할 필요가 없다고 판단, JUNIT Test를 작성하지 않았음.

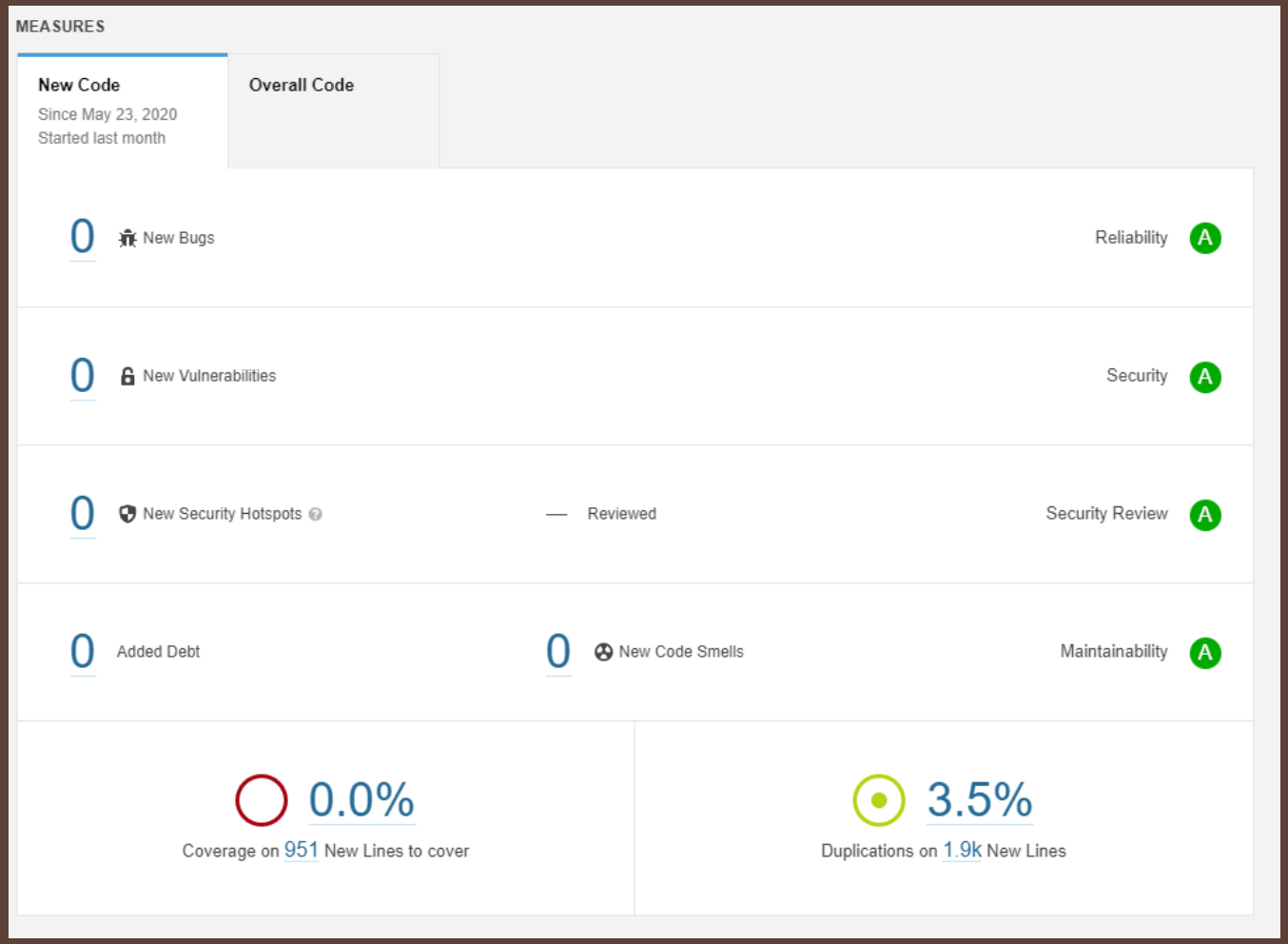
1.5
Code Coverage



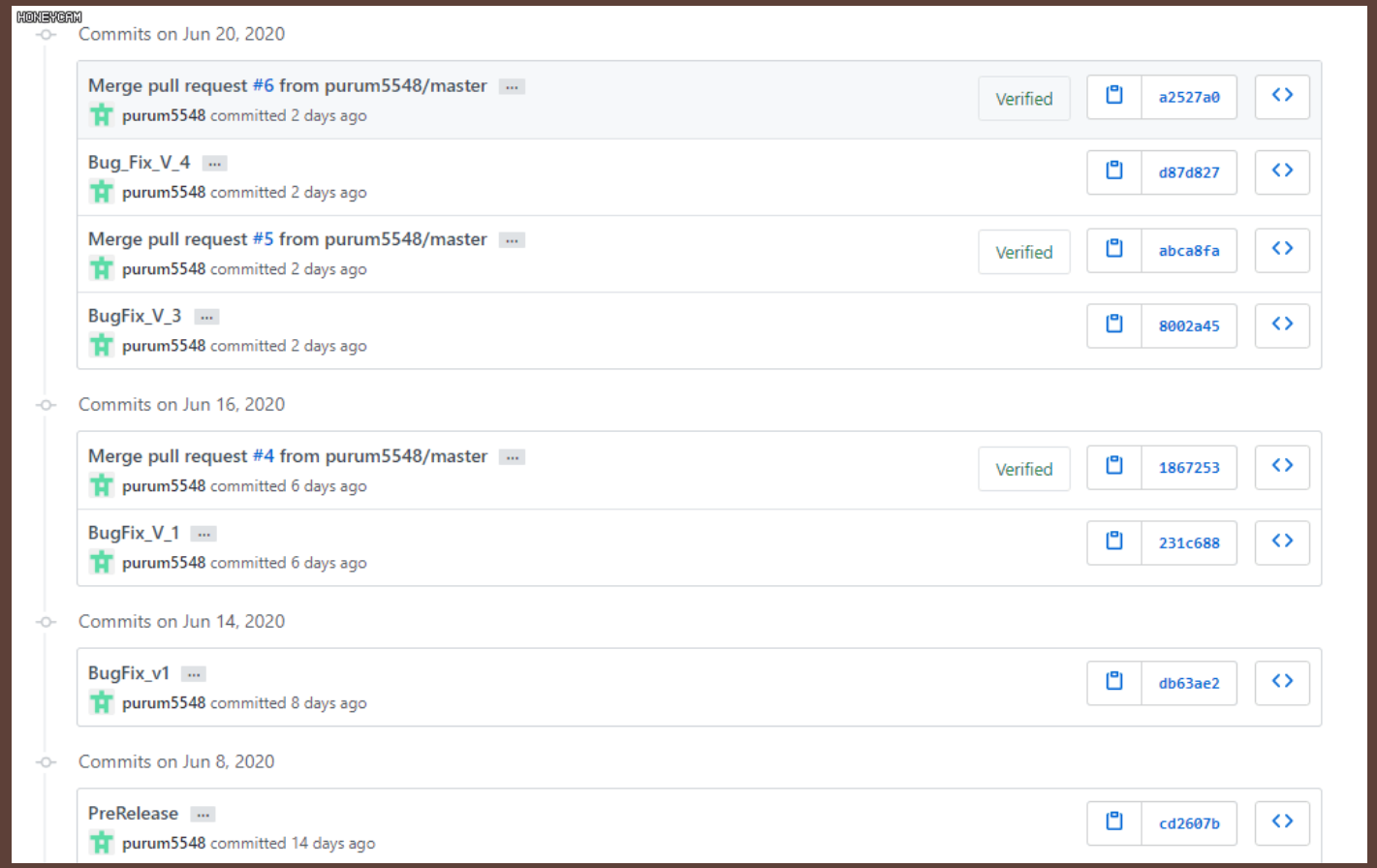


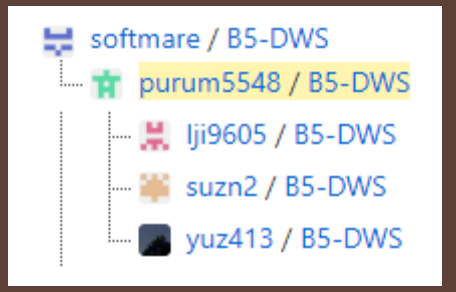
Test Pass Rate : 100.0%

1.6
After Fix


MEASURES
New Code
Since May 23, 2020
Started last month
Overall Code
0
New Bugs
Reliability
A
0
New Vulnerabilities
Security
A
0
New Security Hotspots
Reviewed
Security Review
A
0
Added Debt
0
New Code Smells
Maintainability
A
0.0%
Coverage on 951 New Lines to cover
3.5%
Duplications on 1.9k New Lines

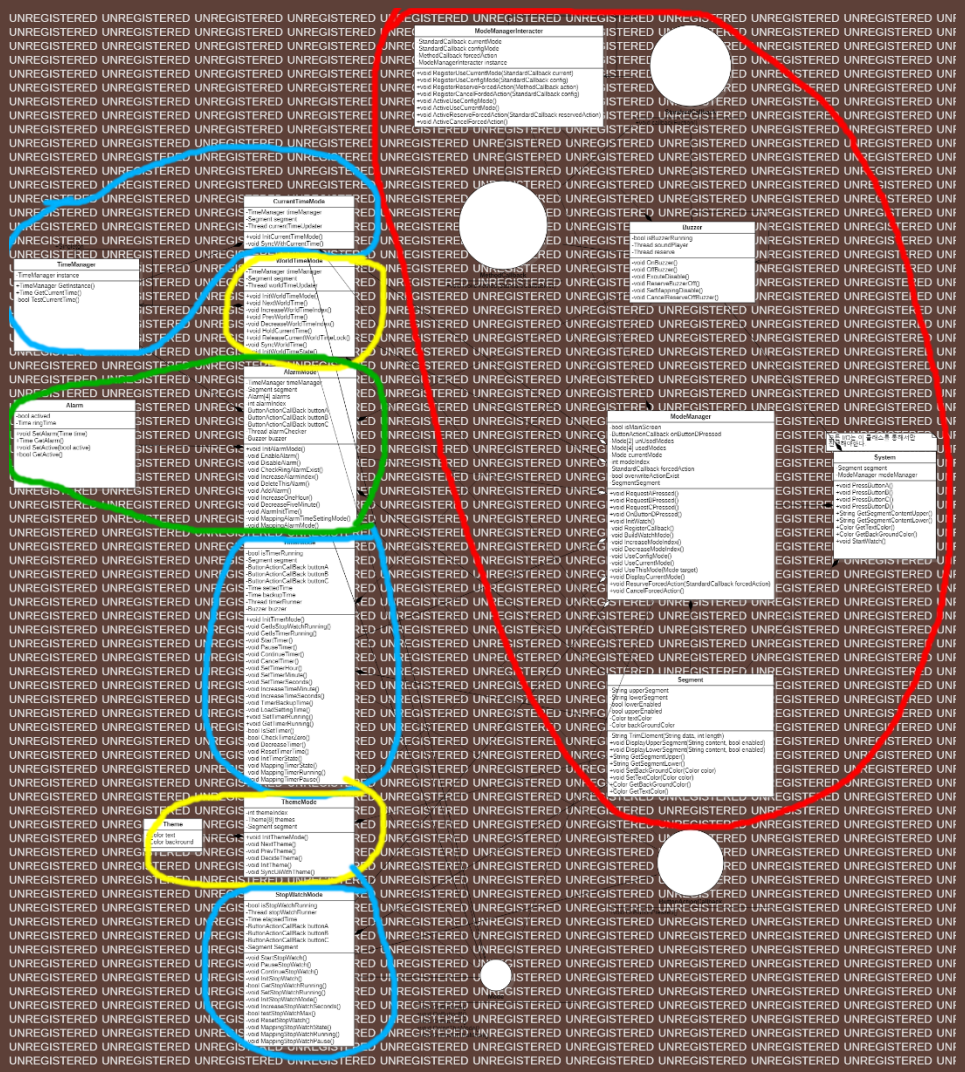
additional
Review





- GitHub를 통해서 협업 진행, Fork를 하였고, 브루트 포스 테스트는 브렌치를 따서 진행함

additional
Review



additional
Review

- Dependency 를 극단적으로 낮춰서 여러 이점이 생겼음

- 1. 버그가 발생 했을 때, 발생한 위치를 특정하기가 매우 쉬웠음, 버그가 발생하면, 무조건 해당 클래스의 책임이기 때문에, 해당 클래스의 코드만 검사하면 되고, 실제로도, 브루트 포스 테스팅과, 소검팀 테스트에서도, 극단적으로 낮은 버그율을 보여줌

- 2. 작업 로드 조절이 간단 해졌다. 클래스 단위로 작업하다 보니, 만약, 클래스가 생각보다 복잡해서, 예상보다 오래 걸린다면, 다른 사람이, 그 사람이 할 클래스를 대신 받아서 작업 해줄 수 있음

- 3. 작업 속도가 증가함, 서로가 만드는 클래스 간의 디펜던시가 낮아서, 작업하는 중에, "xx메소드(또는 클래스)가 없어서, 더 이상 작업이 불가능 하다, 기다려 " 같은 상황이 거의 발생하지 않았음(버저와 같은 코어 클래스가 없는 경우가 아니라면) 따라서 작업 중, 필요한 코드가 없어서 막히는 경우가 상당히 줄어듦

- 4. 프로세스와 운영체제의 관계를 모방한 클래스 구조로써, 프로그램이라는 것의 이점 처럼, 모드를 새로 만들고 교체하기가 매우 수월하다.

additional
Review

로드맵

1.0.0 진행

모듈 개발, 구동 확인

100%

10 일감 (10 건 완료 — 모두 완료)

연결된 일감

- ~~Feature #7~~: 1. System 클래스 개발
- ~~Feature #8~~: 2-1. CurrentTimeMode 클래스 개발
- ~~Feature #9~~: 2-2. WorldTimeMode 클래스 개발
- ~~Feature #10~~: 2-3. AlarmMode 클래스 개발
- ~~Feature #11~~: 2-4. TimerMode 클래스 개발
- ~~Feature #12~~: 2-5. ThemeMode 클래스 개발
- ~~Feature #13~~: 2-6. StopWatchMode 개발
- ~~Feature #14~~: 3-1. TimeManager 클래스 개발
- ~~Feature #15~~: 3-2. Buzzer 클래스 개발
- ~~Feature #16~~: 3-3. Segment 클래스 개발

1.0.1 진행

1차 버그 fix

100%

1 일감 (한 건 완료 — 모두 완료)

1.0.2 진행

Vulnerability fix

100%

6 일감 (6 건 완료 — 모두 완료)

연결된 일감

- ~~Vulnerability #93~~: [Bug - Major] Either re-interrupt this method or rethrow the "InterruptedException".
- ~~Vulnerability #94~~: [Bug - Major] Unused field: Alarm.ringTime
- ~~Vulnerability #95~~: [Bug - Major] new Buzzer() invokes Thread.start()
- ~~Vulnerability #96~~: [Vulnerability - Minor] Use a logger to log this exception.
- ~~Vulnerability #97~~: [Vulnerability - Minor] Explicitly declare the visibility for "hour"
- ~~Vulnerability #98~~: [Vulnerability - Minor] Make this "public static instance" field final

Friday, June 19th

[B5] 김태형 8:40 PM
이번 분석에서 나온 취약점 일감으로 만들어놓았습니다.
수정이 불필요하다고 생각되는 취약점에 대해서는 Reject 하시고 이유 써주시면 됩니다.
http://redmine.third.kr/projects/b5/issues

> **redmine.third.kr**
> Issues - B5 DWS - Redmine
> Redmine

Saturday, June 20th

[B5]서푸름 6:09 PM
RedMine에 수정 내용들에 대한 코멘트를 추가했고, 수정된 코드는 git의 master브랜치에 푸쉬 했습니다

[B5] 김태형 7:12 PM
수정 확인하였습니다. 보고서에 첨부된 코드들은 특정 문제가 발생한 파일들의 일부만 예시로 첨부한 것입니다.
취약점 분석결과 의 Bug와 Vulnerability 항목에서 아직 해결되지 않은 중복되는 취약점에 대해서 수정이 필요합니다.

[B5] 김태형 7:20 PM
소나큐브상에서 해당 이슈를 클릭하면 파일별로 취약점 발생코드의 위치와 취약점이 나와있습니다. 해당 항목들에 대해서 명확하게 안내드리지 못해서 죄송합니다.

[B5] 김태형 7:30 PM
아직 취약점이 남아있는 일감에 대해서 레드마인의 코멘트로 Feedback 하였습니다.

[B5]서푸름 10:37 PM
일단 소나 큐브상에 표기된 이슈들은 한번씩 다 확인하고 수정했습니다. 마찬가지로, MasterBranch에 푸쉬했습니다!

Yesterday

[B5] 김태형 1:39 AM
확인하였습니다. 수고하셨습니다!

additional
Review

- 실제로 개발 방법론에 따라서 개발을 진행 함으로써, 이러한 개발 과정에 대한 경험을 얻음
- 예상하지 못한 취약점을 정적 분석으로 찾아 낼 수 있다는 점

- 오프라인 수업이 아니여서, 작년과는 다른 방식의 수업 진행으로, 못한 것들이 많다고 느껴짐
- 개발 방법론은 배웠지만, 실제로 실무에서 사용할 때는 어떻게 변형되고, 어떻게 관리를 해야 하는지는 궁금함

감사합니다!