Alarm Customizing Digital Watch

3rd Cycle : Static Analysis

201511172 컴퓨터공학부 강민호
201511257 컴퓨터공학부 남관우
201511271 컴퓨터공학부 신윤섭
201810502 컴퓨터공학부 전현지

Index

001. Specification Revision

1. Stage 2050

2051	16	문서에서 다뤄진 class 의 method 가 실제 code 에서 사용되는 것에 비해 적다.
2051	18	문서에서 다뤄진 class 의 method 가 실제 code 에서 사용되는 것에 비해 적다.
2051	22	문서에서 다뤄진 class 의 method 가 실제 code 에서 사용되는 것에 비해 적다.
2051	26	문서에서 다뤄진 class 의 method 가 실제 code 에서 사용되는 것에 비해 적다.
2051	32	문서에서 다뤄진 class 의 method 가 실제 code 에서 사용되는 것에 비해 적다.
2051	41	문서에서 다뤄진 class 의 method 가 실제 code 에서 사용되는 것에 비해 적다.
2051	49	문서에서 다뤄진 class 의 method 가 실제 code 에서 사용되는 것에 비해 적다.

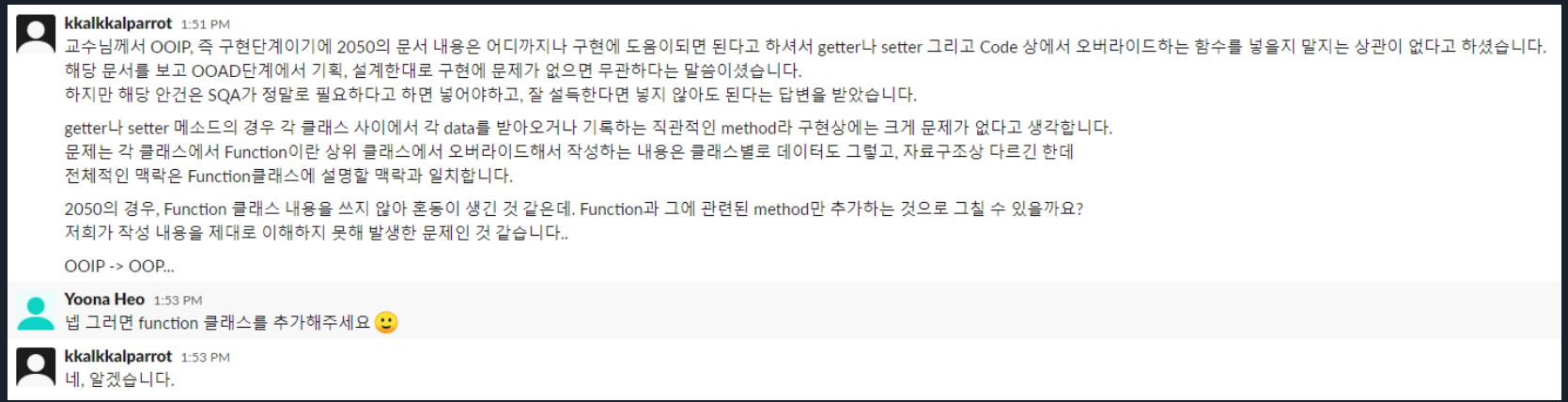
Function 클래스에서 오버라이드 하는 각 함수가 각 기능 별 클래스에 없어서 발생한 문제.

SQA와 잘 협의 하여 필요한 Function 클래스와 그에 대한 Method를 추가하는 것으로 합의함.

getter와 setter Method는 직관적인 Method라 굳이 추가하지 않는 것으로 합의함.

001. Specification Revision

1. Stage 2050



kkalkkalparrot 1:51 PM
교수님께서 OOIP, 즉 구현단계이기에 2050의 문서 내용은 어디까지나 구현에 도움이되면 된다고 하셔서 getter나 setter 그리고 Code 상에서 오버라이드하는 함수를 넣을지 말지는 상관이 없다고 하셨습니다.
해당 문서를 보고 OOAD단계에서 기획, 설계한대로 구현에 문제가 없으면 무관하다는 말씀이셨습니다.
하지만 해당 안건은 SQA가 정말로 필요하다고 하면 넣어야하고, 잘 설득한다면 넣지 않아도 된다는 답변을 받았습니다.

getter나 setter 메소드의 경우 각 클래스 사이에서 각 data를 받아오거나 기록하는 직관적인 method라 구현상에는 크게 문제가 없다고 생각합니다.
문제는 각 클래스에서 Function이란 상위 클래스에서 오버라이드해서 작성하는 내용은 클래스별로 데이터도 그렇고, 자료구조상 다르긴 한데
전체적인 맥락은 Function클래스에 설명할 맥락과 일치합니다.

2050의 경우, Function 클래스 내용을 쓰지 않아 혼동이 생긴 것 같은데. Function과 그에 관련된 method만 추가하는 것으로 그칠 수 있을까요?
저희가 작성 내용을 제대로 이해하지 못해 발생한 문제인 것 같습니다..

OOIP -> OOP...

Yoona Heo 1:53 PM
넵 그러면 function 클래스를 추가해주세요 🙂

kkalkkalparrot 1:53 PM
네, 알겠습니다.

001. Specification Revision

1. Stage 2050

Function Class 및 Method 추가.

2051. Implement Class & Method Definitions

Type	Class
Name	Function
Purpose	6개의 기능(Timekeeping, Stopwatch, Timer, D-day, Alarm, AlarmCustom)에서 공통적으로 사용되는 Method들 오버라이드 하도록한다.
Overview(Class)	각 기능에서 공통적으로 사용되는 Method가 입력되어있다.
Cross Reference	Function : R1.1, R1.3, R2.1, R2.2, R2.5, R3.1, R3.2, R3.6, R4.1, R4.6, R5.1, R6.1, R6.2, R6.3, R7.3 Use Cases : Set Time, Set Display, Set Timer, Set Alarm, Stop Alarm Buzzer, Set D-day, Set Alarm Interval, Set Alarm Volume, Cancel
Exceptional Courses of Events	N/A

001. Specification Revision

1. Stage 2050

Function Class 및 Method 추가.

2051. Implement Class & Method Definitions

Type	Method
Name	cancel
Purpose	진행중이던 내용을 취소하고 기본 모드로 돌아간다.
Cross Reference	Function : R7.3 Use Cases : Cancel
Input	void
Output	void
Abstract operation	N/A
Exceptional Courses of Events	N/A

001. Specification Revision

2. Stage 2060

2063	123~134	Test case 의 description 이 구체적이지 않다. ‘정상적’, ‘제대로’와 같은 명확한 설명이 없는 단어들이 사용된다.

‘정상적’, ‘제대로’라는 description은 실제로 구현한 사람들 외에는 알아보기 힘든 모호한 표현이다.

따라서 해당 표현을 삭제하고 좀더 면밀한 description으로 재작성하였다.

001. Specification Revision

2. Stage 2060

2063. System Testing

Test No.	Test 항목	Description	Use Case	System func
4-1	타이머 시간 설정 시험	시, 분, 초를 각각 설정할 수 있는지 확인한다.	Set Timer	R2.1
4-2	타이머 설정 저장 시험	타이머가 설정한 시각으로 설정 되었는지 확인한다.	Set Timer	R2.1
5-1	타이머 카운트다운 시험	타이머의 시간이 설정한 시각부터 1초씩 카운트 다운되는 지 확인한다.	Start Timer	R2.2
6-1	타이머 버저 시험	타이머가 끝났을 때 버저가 울리는 지 확인한다.	Beep Timer	R2.3
7-1	타이머 초기화 시험	타이머가 00 : 00 : 00으로 초기화하는지 확인한다.	Reset Timer	R2.4

002. System Test Report

1. Display StopWatch Record

Display Stopwatch Record	스톱워치가 일시정지된 상태에서 Select Button을 2초 이상 누르면 해당 화면 모드로 진입하는가	Pass
	기록이 최신순으로 출력되는가 →가장 오래된 것부터 출력된다	Fail
	한 화면에 최대 3개까지 출력되는가	Pass
	기록이 없으면 NONE이 출력되는가	Pass

1000~2040을 전체적으로 검토하면서 기록은 최신순으로 출력하지 않는 것으로 정정되었음.

따라서 TestCase역시 '기록이 최신순으로 출력되는가'에서 '스톱워치의 기록들이 정상적인지 확인한다.'라고 수정되었음.

이는 앞선 2060 Specification Revision에 의해 '기록 버튼을 누른 시각에 맞는 기록이 저장되었는지 확인한다'로 수정함.

해당 사실을 SQA에게 알려 인지시킨 상황임.

002. System Test Report

2. Display Time

Display Time	시각이 초 단위로 변하는가	Pass
	Set Time을 통해 변경하고 난 뒤, 시각이 초 단위로 변하는가	Pass
	화면 좌측 상단부터 순서대로 설정한 D-day까지 남은 일 수, 알람 개수, 알람 아이콘, 시, 분, 초, 년, 월, 일, 요일이 표시되는가 →실제 설정한 알람 개수가 10개일 때 화면에는 9가 출력된다	Fail

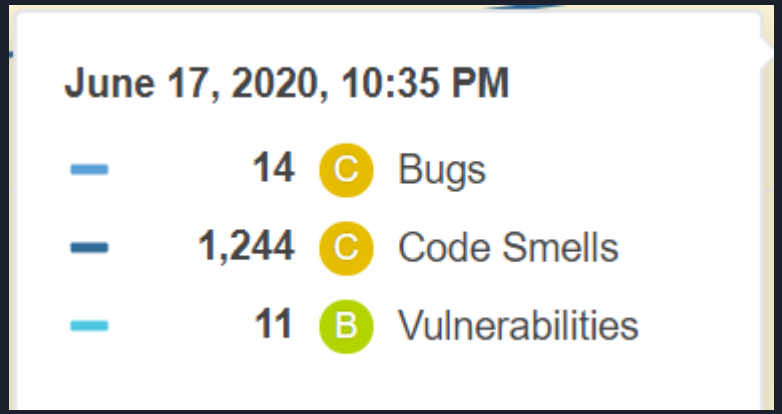
두 자리 수 이상의 수를 GUI로 표현할 경우 알람 개수 View가 좁아서 잘려서 보임을 확인했다.

이에 알람 개수가 10개일 때 '1' 이미지와 '0' 이미지가 보이는 것에서 '10' 이미지가 출력되도록 하였음.

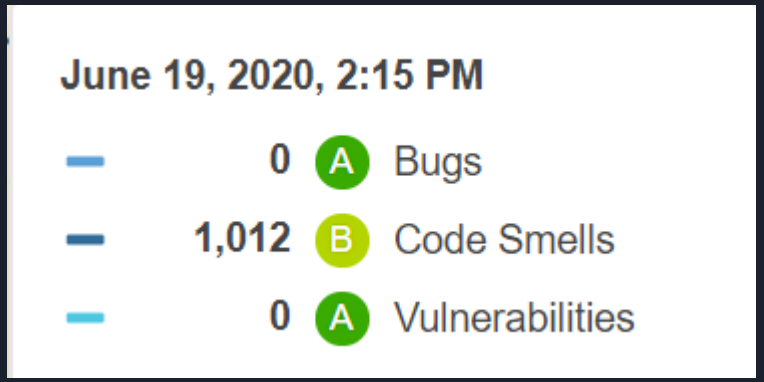
```
final static String TEN = "ten.jpg";
```

003. Static Analysis

Before



After



003. Static Analysis

공통 Code Smell

Code Smell	Blocker	상속 받아오는 class인 Function에서와 같은, 대문자/소문자 구분만 다른 FID라는 이름의 변수를 선언했다. 또한, 변수가 선언된 뒤 코드 내에서 사용된 적은 없고, Function class에서 상속받은 fid만 사용되었다.
Code Smell	Blocker	상속 받아오는 class인 Function에서와 같은 mode라는 변수를 선언했다. 아무리 private으로 선언한다고 해도 변수 이름에 차이를 두는 것이 좋다.
Code Smell	Major	상속 받아오는 class인 Function에서와 동일한 이름의 함수 changeMode를 선언했으므로 @Override를 추가해줘야 한다.

Function 클래스를 상속받는 클래스들에게 공통적으로 있었던 문제.

fid 변수가 중복되어 선언된 부분을 수정했고, 공통적으로 사용하는 type변수를 Function클래스에 protected 형으로 선언하여 코드 일관성을 향상시켰다.

changeMode() 함수도 상속받아 사용하는 구조이므로 @Override 어노테이션을 달아주는 것으로 수정했다.

003. Static Analysis

공통 Code Smell

```
5   1   public class Alarm extends Function {
6   2
7   3       final static int FID = 5;
8   4
9       -   public Alarm(System system) {
10      -       fid = 5;
    5   +   public Alarm() {
11  6           curAlarm = new AlarmData();
12  7           alarmList = new AlarmData[10];
13  8           for(int i=0; i<10; ++i)
14  9               alarmList[i] = null;
15  10          mode = 0; // 기본 모드
16  11          segmentPointer = new int[2];
17      -       typeindex = 0;
18      -       this.system = system;
    12  +       type = 0;
    13  +   }
```

2 src/main/java/Function.java

```
    @@ -3,7 +3,7 @@
3       public Function() {}
4
5       protected int mode;
    -   protected int fid;
6   +   protected int type;
7
8       abstract protected void cancel();
9       protected void changeMode(int mode) {}
```

```
+   @Override
    public void changeMode(int _mode) {
        mode ^= 1;
        if (mode == 0) {
```

003. Static Analysis

1. Alarm

28	Code Smell	Major	private로 선언된 변수 system은 선언 이후로 사용되지 않았다. 현재는 주석 처리된 함수에서 사용되던 변수이므로 삭제할 필요가 있다.

48	Code Smell	Major	if문 선언 후 {} 내부가 비어있다.
127	Code Smell	Major	상속 받아오는 class인 Function에서와 동일한 이름의 함수 changeMode를 선언했으므로 @Override를 추가해줘야 한다.

28 : System 클래스에서 각 기능을 생성할 때 일관성과 확장성을 위해 생성자에 system 인스턴스를 넘겨주는 것으로 통일하여 따로 수정하지는 않았다.

48, 127 : 코드 수정 완료.

003. Static Analysis

1. Alarm

```
46  41      public void addTimeToAlarmList(Time alarmTime) {
47  42          int size = getSize();
48      -       if (size >= 10) {}// 꽉 찼으니 저장 안함.
49      -       else {
50      -
        43  +       if (size < 10) {
51  44              for (int i = 0; i < size; i++) {
52  45                  if (alarmList[i].getTime().equals(alarmTime))
53  46                      return;
```

```
24  22          private int[] segmentPointer;
25      -       private int mode;
26      -       private int typeindex; // 시분초 구분
27  23          private int alarmSettingValue[] = {-1,-1,-1};
28      -       private System system;
29  24
```

003. Static Analysis

2. AlarmView

108	Code Smell	Blocker	Switch문에서 case의 break;가 존재하지 않는다. 이럴 경우 아래의 default와 break되지 않은 case가 동시에 실행되게 된다. Break하지 않을 것이라면, continue/throw/return 등의 다른 명령어를 추가해야 한다.

default 경우 예외처리함.

```
59  +   public void setAlarmList(String str) {
60  60      if (str.substring(0, 6).equals("      "))
61  61          displaySegment(350, 240, ALARM_WIDTH, ALARM_HEIGHT, "  NONE", layer++);
62  62      else
63  63          displaySegment(350, 240, ALARM_WIDTH, ALARM_HEIGHT,
64  64                  str.substring(0, 6), layer++);
65  65
66  66      if (str.substring(6, 12).equals("      "))
67  67          displaySegment(350, 240 + ALARM_HEIGHT, ALARM_WIDTH, ALARM_HEIGHT, "  NONE", layer++);
68  68      else
69  69          displaySegment(350, 240 + ALARM_HEIGHT, ALARM_WIDTH, ALARM_HEIGHT,
70  70                  str.substring(6, 12), layer++);
71  71
72  72      if (str.substring(12, 18).equals("      "))
73  73          displaySegment(350, 240 + ALARM_HEIGHT * 2, ALARM_WIDTH, ALARM_HEIGHT, "  NONE", layer++);
74  74      else
75  75          displaySegment(350, 240 + ALARM_HEIGHT * 2, ALARM_WIDTH, ALARM_HEIGHT,
76  76                  str.substring(12, 18), layer++);
77  77  }
78  78
79      -   public void setAlarmList(AlarmData[] alarmData, int pointer, int size) {
80      -
81      -       Time alarmData1;
82      -
83      -       switch (size) {
84      -           case 0:
```

003. Static Analysis

3. Border

Code Smell	Major	선언된 Border 함수의 parameter인 system은 해당 함수 내에서 사용되지 않는 parameter이다.

생성자에서 System을 제거하고
기본 생성자로 바꿈.

2 src/main/java/Border.java

```
@@ -5,7 +5,7 @@
5  5
6  6       private boolean borderState;
7  7
8     -    public Border(System system) {
   8  +    public Border() {
9  9            borderState = false;
10 10       }
11 11
```

003. Static Analysis

4. D-day

108	Code Smell	Blocker	Switch문에서 case의 break;가 존재하지 않는다. 이럴 경우 아래의 default와 break되지 않은 case가 동시에 실행되게 된다. Break하지 않을 것이라면, continue/throw/return 등의 다른 명령어를 추가해야 한다.

default 경우 예외처리함

2 src/main/java/D_day.java

```
@@ -183,6 +183,8 @@ else if(dateSettingValue[type] > Date.numOfDays[dateSettingValue[1]])
183 183                         dateSettingValue[type] = Date.numOfDays[dateSettingValue[1]];
184 184                     }
185 185                     break;
    186 +               default:
    187 +                   break;
186 188             }
187 189         }
188 190
```

003. Static Analysis

4. Date

15	Vulnerability	Minor	public으로 선언된 array numOfDays[]는 상속받는 class에서만 사용되는 변수이므로 protected로 속성을 변경하는 것이 좋다.

protected로 변경함.

2 src/main/java/Date.java

```
@@ -12,7 +12,7 @@
12        public final int MONTH_TOP_LIMIT = 12;
13        public final int MONTH_BOTTON_LIMIT = 1;
14        // 인덱스 1 ~ 12가 월에 대응하는 일 수. 인덱스 0는 TimeSettingMode일 때 순환적인 처리
-         public final static int numOfDays[] = {1,31,28,31,30,31,30,31,31,30,31,30,31};
15      + public final static int[] numOfDays = {1,31,28,31,30,31,30,31,31,30,31,30,31};
16
17        private int year;
18        private int month;
```

003. Static Analysis

5. DefaultLayout

3~4	Code Smell	Minor	이미 java.awt.*를 import 했으므로 불필요하다.
7	Code Smell	Minor	import한 IOException은 사용되지 않았으므로 불필요하다.

사용되지 않는 클래스 제거

003. Static Analysis

6. Stopwatch

133	Code Smell	Major	선언된 함수 changeMode의 else 구문 내에 아무것도 입력되어 있지 않다.

코드 수정 완료.

```
124 +   @Override
125     public void changeMode(int mode) {
126         this.mode = mode;
127
130 -       if(this.mode == 0) {
128 +       if (this.mode == 0) {
129             recordPointer = 0;
130         }
133 -       else {
134 -
135 -       }
131     }
132
```

003. Static Analysis

7. System

> 전반적인 code의 complexity가 너무 높고, 반복되는 code들이 많다.

System 클래스는 컨트롤러 역할을 동반하기 때문에 GUI 관련 코드로 인해 반복되는 코드가 존재한다. 이에 SQA와 의논하여 수정하지 않는 것으로 합의했다.

003. Static Analysis

8. Time

2~3	Code Smell	Minor	굳이 java.lang class를 import할 필요는 없다.
38	Bug	Minor	선언한 boolean 함수 equals의 parameter object t가 유효한 값을 가지는지 확인하는 조건문이 있는 것이 좋다.
40	Bug	Major	변수 time이 null일 수도 있는 경우를 대비해 NullPointerException 등의 예외처리를 거치는 것이 좋다.

2~3: System 클래스 이름이 java.lang.System 과 동일하여 java.lang을 붙여주지 않으면 intelliJ가 인식하지 못하는 문제가 있었다. SQA와 의논 완료된 사항.

38, 40: Time의 인스턴스인지 확인함으로써 해결했다.

003. Static Analysis

9. Timer

39	Bug	Major	&&의 양 옆에 같은 code가 작성되어 있다.
63	Bug	Major	&&의 양 옆에 같은 code가 작성되어 있다.
	Code Smell	Major	if문 선언 후 {} 내부가 비어있다.

39, 63: 코드 리팩토링함.

003. Static Analysis

9. Timer

```
                st = new StringTokenizer(str, " ");
-               if (st.nextToken().equals("0") && st.nextToken().equals("0") && st.nextToken().equals("0")) {
-
+               boolean isZero = true;
+               while (st.hasMoreTokens()) {
+                   if (!st.nextToken().equals("0")) {
+                       isZero = false;
+                       break;
+                   }
+               }
+               if (isZero) {
```

```
        StringTokenizer st = new StringTokenizer(str, " ");
-       if (st.nextToken().equals("0") && st.nextToken().equals("0") && st.nextToken().equals("0")) {
+       boolean isZero = true;
+       while (st.hasMoreTokens()) {
+           if (!st.nextToken().equals("0")) {
+               isZero = false;
+               break;
+           }
        }
-       else {
+
+       if (!isZero) {
            changeMode(2);
            timer.startTime();
        }
```

004. Review

신윤섭

- 문서화가 중요하다고 느꼈다. 요구사항을 이해하기 좋게 문서화하는 것이 솔직히 너무 시간을 많이 들고 귀찮다고 느꼈는데 협업 과정에서 요구사항이 잘 정리된 문서가 있어야 프로젝트가 원활하게 진행될 수 있음을 느꼈다.

- CTIP 환경을 구축하고 개발을 하는 것이 높은 수준의 소프트웨어를 만드는데 도움이 된다고 느꼈다. 테스트 코드를 짜는 것은 귀찮은 일이지만 여유가 있으면 프로그램의 품질을 위해 꼭 도입하는 것이 좋다고 생각한다.

004. Review

남관우

실무에서 협업 시 어떻게 코드를 관리하나 궁금했는데, 이번에 맛보기로라도 그 흐름을 체험해 볼 수 있어서 값진 경험이 되었습니다.
functional requirement를 만들어 use case로 발전시키고, system operation을 뽑아내고 이를 시퀀스 다이어그램에 적용하여 내부 메서드들을 논리적으로 만들어내는 과정이 개발자의 부담을 훨씬 줄여주는 일이라는 걸 새삼 깨달았습니다.
또한 implementation 단계에서 구축한 CTIP 환경을 경험하면서 젠킨스와 소나큐브가 코드 관리에 있어서 굉장히 큰 도움을 주는 툴이라는 것을 느꼈습니다. 기회가 된다면 4학년 때 소프트웨어V&V 수업을 들어보고 싶습니다. 한 학기 동안 고생하셨습니다.

004. Review

- 강민호 :

  바로 구현에 들어가는 것이 아니라 기획 단계부터 구현까지 차근차근 그 단계를 밟아갈 수 있었던 수업이었습니다. 보통 프로그램은 팀 프로젝트의 형태로 진행되는데, SQA와의 원활한 소통과 팀 내부의 구현 방향을 확인하기 위해서 기획서 및 설계서의 작성이 매우 중요하다는 것을 알게 된 수업이었습니다.

  또한 CTIP 환경을 세팅함으로서 불안한 코드를 수정할 수 있는 환경을 마련하는 것이, 버그를 빠르게 확인 및 수정하는데 큰 도움이 된다는 것을 깨달았습니다. 그래서 추후에 소프트웨어 검증 수업도 듣고 싶습니다.

004. Review

- 전현지 : 그동안 팀프로젝트로 여러번 프로그램을 만들었지만, 이번처럼 체계적으로 한 단계씩 만들었던 경험은 부족했기 때문에 좋은 경험이 되었습니다. 프로그램을 만들 때 코드 말고도 생각해야 하는 것들이 많은 것을 깨달았습니다. 문서를 만드는 과정에서 팀원들과 자세히 논의하지 않으면 나중에 결국 다시 돌아와서 다시 논의하게 되었고, 세세한 것들까지 결정해야 개발 단계에서 더욱 편한 것을 알게 되었습니다. 4학년 SQA팀에서 해 주신 검증도 저희 팀원끼리는 발견해내기 어려웠던 관점으로 제시해주셔서 많은 도움이 되었습니다. 일정도 빠듯하고 힘들었지만 그래도 좋은 팀원들과 만나 좋은 경험을 하게 되었습니다.

QnA