

# 졸업작품 교과목

## 포트폴리오

---



---

20170674 - 양재완




---

# 목차

- 자기소개
- 진행한 졸업작품
- 졸업작품 수행 중 학습내용
- 깃과 깃허브 학습내용
- 졸업작품 수행 후기

# 자기소개

## ■ 기본사항

|   |      |                               |     |               |
|---|------|-------------------------------|-----|---------------|
|  | 성명   | 양재완                           | 나이  | 24 세          |
|   | 생년월일 | 1998.03.23                    | 핸드폰 | 010-5348-3039 |
|   | 이메일  | Cinlu2976@m365.dongyang.ac.kr | 전화  | 010-5348-3039 |
|   | 학과   | 컴퓨터정보공학                       | 학번  | 20170674      |

## 성장배경

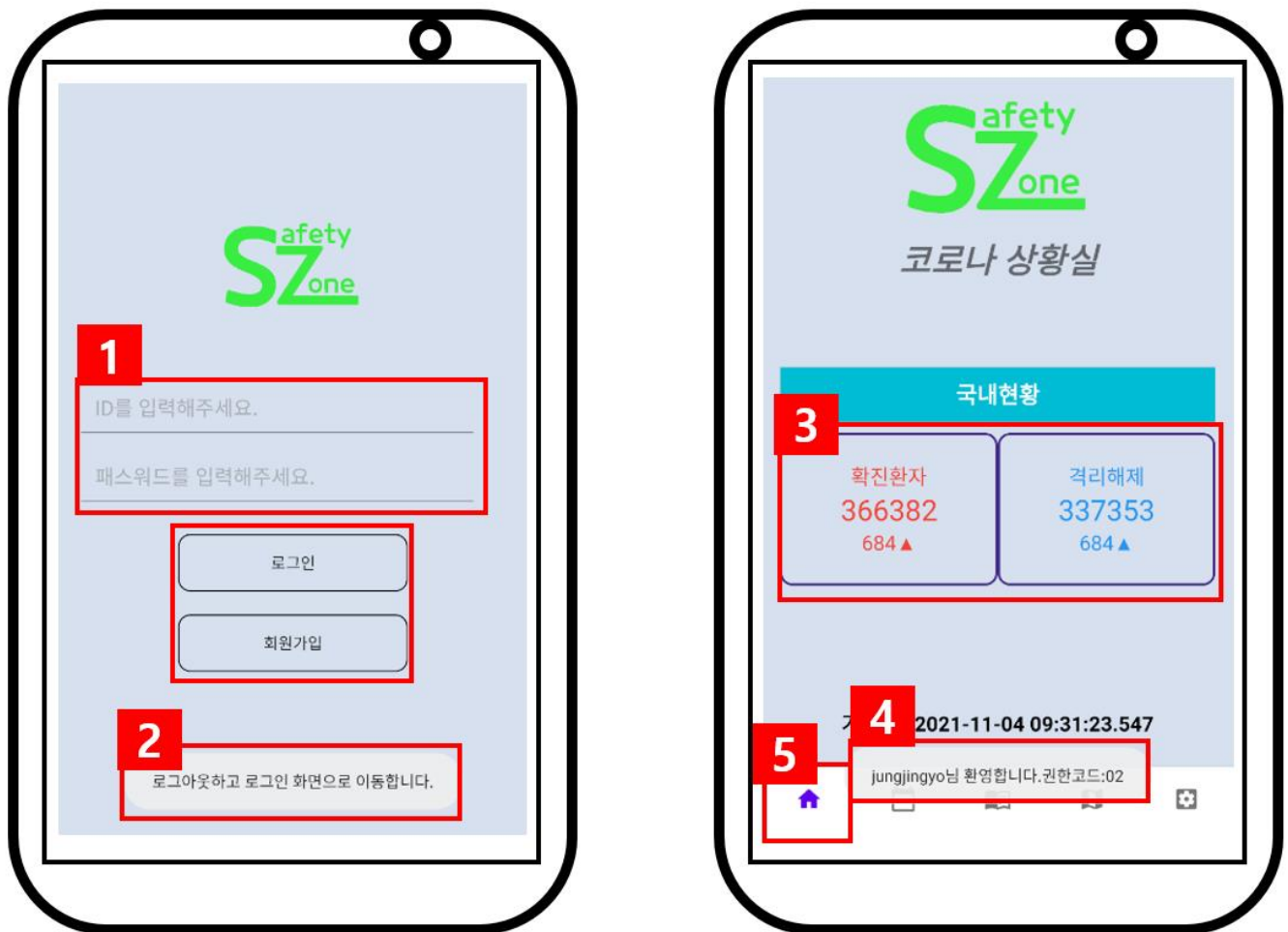
많지는 않지만 아르바이트와 사회복무를 통해 조금이나마 사회경험을 하면서 현실적으로 내가 할 수 있고, 내가 흥미를 느낄 수 있는 일에 최선을 다하자는 마음가짐을 갖게 되었습니다. 특히나 내성적인 성격을 고치기 위해 타인과의 대면이 많은 주차-차량 관리 업무를 자원 신청하여 복무하였는데, 이를 통해 많은 깨달음을 얻었습니다. 그 결과 학창시절에 비해 타인과의 소통에 있어서 좀 더 수월한 면을 갖추게 되었습니다. 최근에는 너무나 큰 목표를 잡아 목표 달성이 어렵다는 좌절감에 빠지기 보다는, 목표를 잘게 세분화하여 하나하나 이를 때 마다 개인적인 성취감을 느낄 수 있도록 인생목표를 바꿔나가고 있습니다.

## 학교생활/대내외 활동

대학 생활을 통하여 동기 및 팀 프로젝트 팀원 간의 소통과 협력에 대해 느낀 바가 많습니다. 조직원들 간의 소통과 협력이 중요하다는 것을 깨달았으며, 조직의 문제를 혼자서만 해결하려 들어서는 안된다는 점도 알게 되었습니다. 교양 과목을 통해서도 작문 실력 향상 등의 성과가 있었다고 생각합니다. 사회 경험이 많으신 분들께서 우스갯소리로 저희 세대를 “마치 온실 속의 화초처럼 귀하게 자랐다”라고 하시곤 합니다. 저도 “온실 속의 화초” 같은 존재였습니다. 학창시절 학교생활에만 안주하면서, 사회를 경험해보지 못한 탓입니다. 그러나 군휴학으로 대학교에 휴학 신청을 했던 시기, 조금이지만 아르바이트 경험을 쌓으면서 사회가 어떻게 돌아가는지를 직접 보고 겪었습니다. 저는 제가 생각해왔던 사회와, 실제 사회경험의 차이를 느끼면서, 이상 속의 사회보다는 좀 더 현실적인, 현실 감각을 키우는 데에 노력해야겠다는 다짐을 갖게 되었습니다.

# 진행한 졸업작품 팀 프로젝트

## 4 조 '세이프티 존'



▲ 졸업작품 '세이프티 존' 메인화면

## ‘세이프티 존’ 앱 개요



아두이노

서버 및 DB

애플리케이션

6 명의 조원이 3 개 파트로 나뉘어 개발 진행

본인은 앱 파트 소속으로 안드로이드 스튜디오를 이용한 앱 개발 진행

건물 내 재난 상황 발생 시 빠른 초동대처를 위해 구상한 앱

1. 회원가입을 통해 회원정보를 서버에 저장
2. 일반 계정으로 로그인 시 코로나 상황판, 공지사항 확인, 재난 대처 요령 확인, 대피로 확인 등의 기능 제공
3. 관리자 계정으로 로그인 시 공지사항 작성, 푸시알림 전송, 대피로 업로드, 화재감지 센서 확인 등의 기능 이용 가능

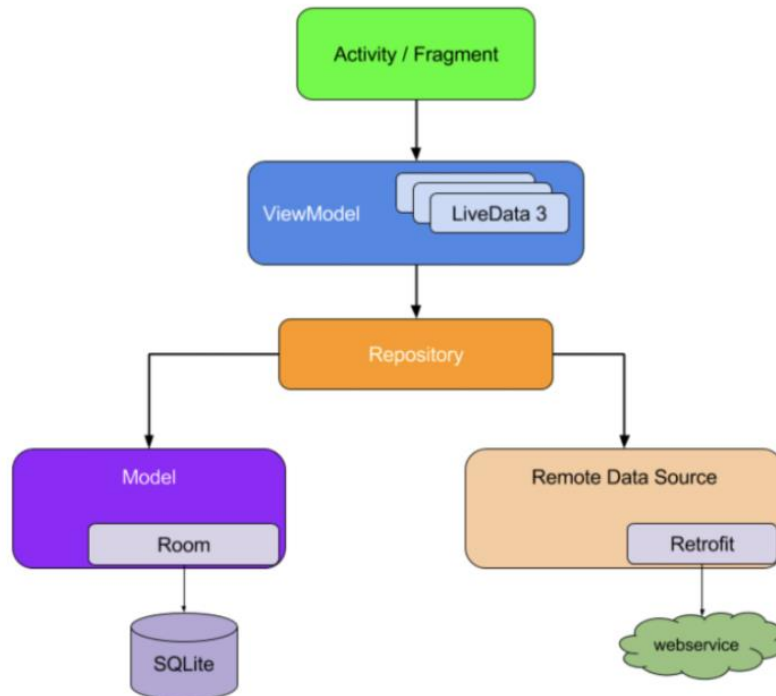
# 팀 프로젝트 중 학습내용

## ● 앱 개발 환경



- JDK (Java Development Kit)
- Android Studio
  - Android App Module
  - Android Library Module
  - Google App Engine Module
- Android SDK
- 공공 API

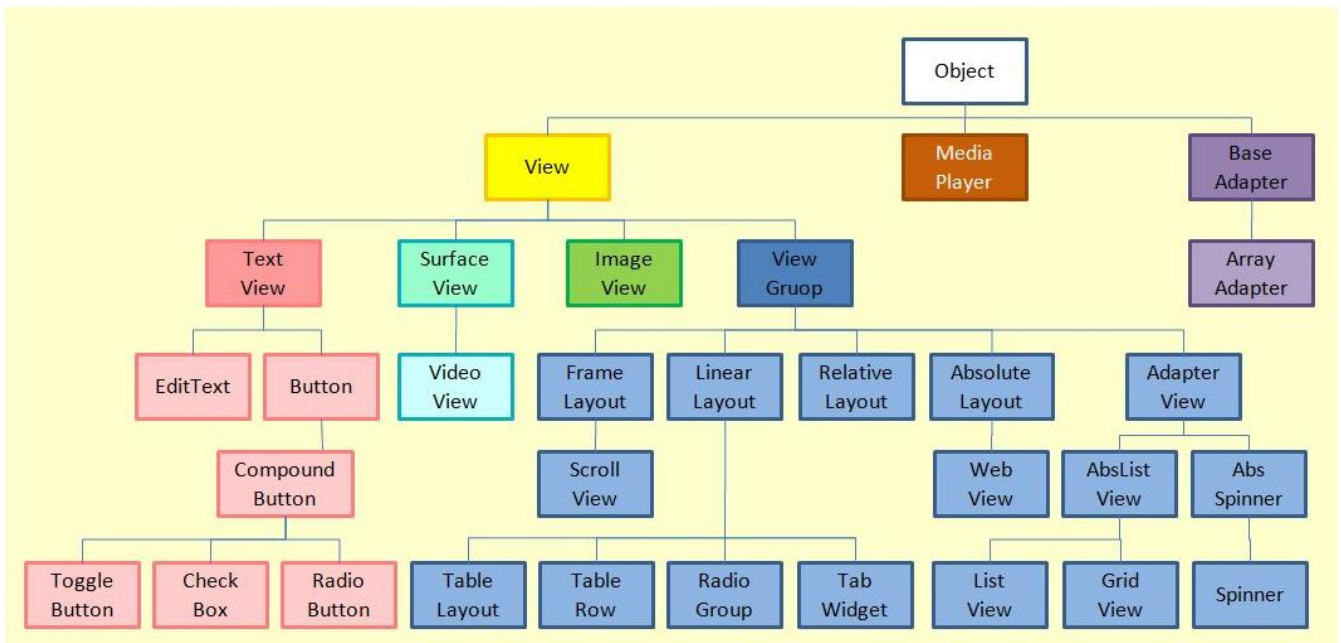
## ● 안드로이드 아키텍처



● 안드로이드 기반의 스마트폰은 안드로이드 아키텍처 운영 환경에서 작동한다.

● 안드로이드 아키텍처는 리눅스 커널을 기반으로 하는 라이브러리와 안드로이드 런타임이 있으며, 그 위에 애플리케이션 프레임워크가 있고 최상위에 우리가 실행하는 앱이 있다. 앱은 정보를 조회, 저장, 변경할 수 있고 무선 네트워크를 이용하여 다른 스마트폰과 통신하여 정보를 교환하기도 한다.

## ● Android Java 클래스 계층

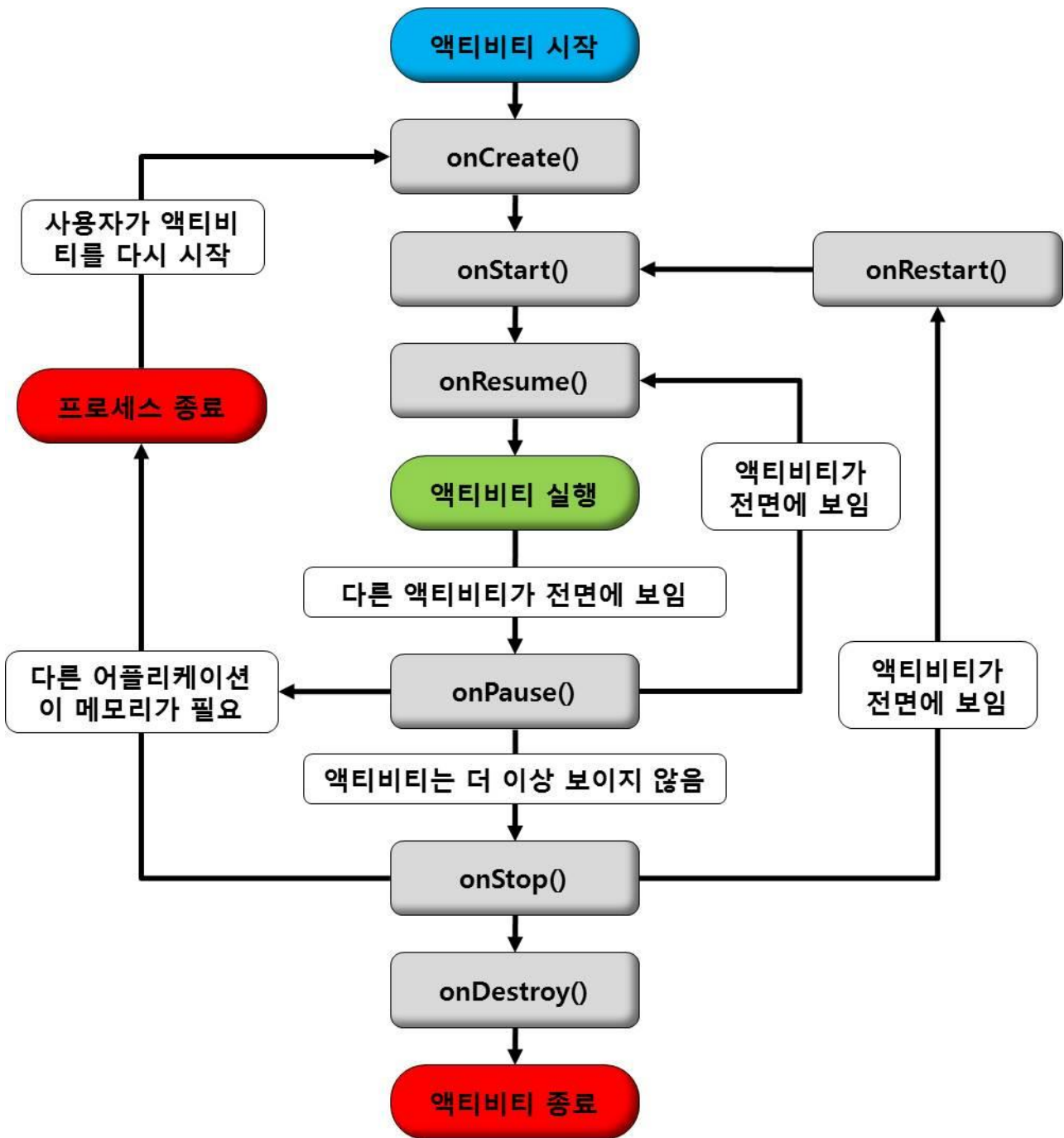


## ▲ Android Java 클래스 계층도

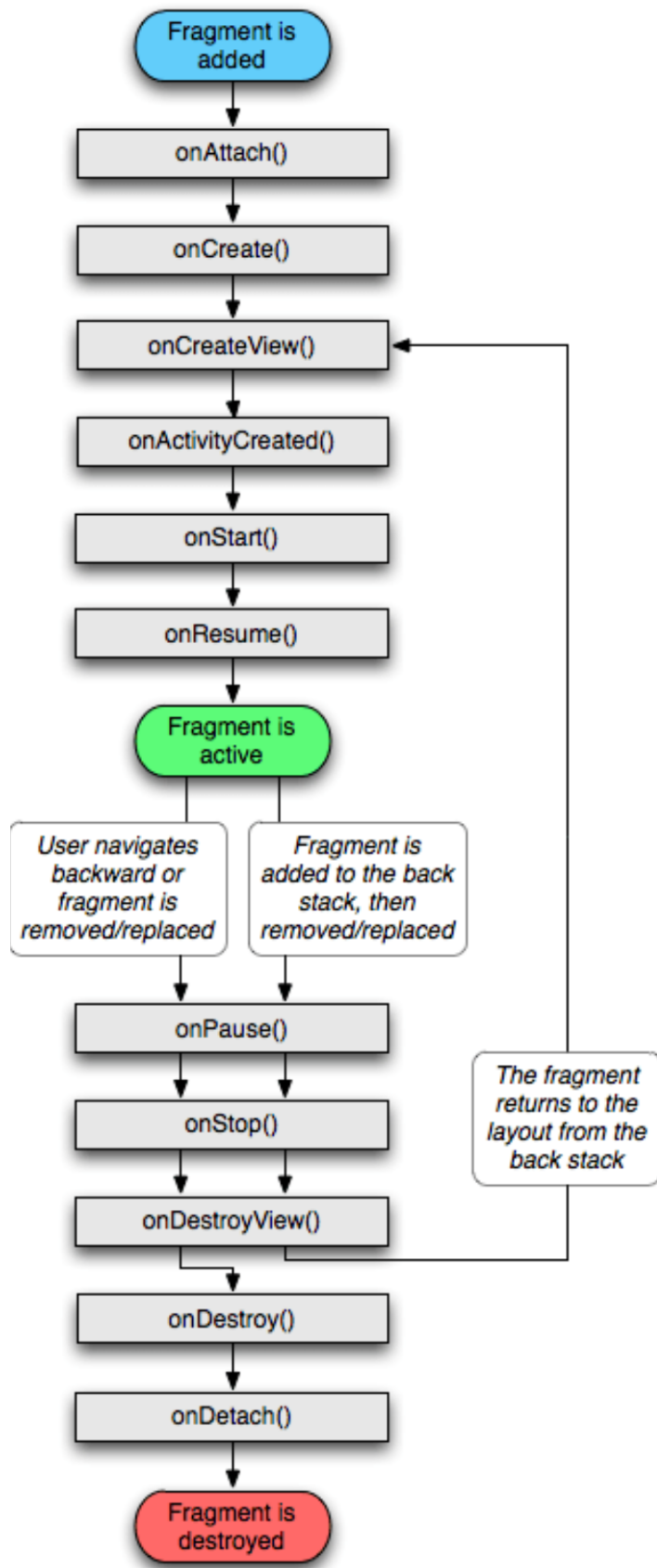
- Android Java 는 최상위 클래스인 Object 를 기준으로 서브 클래스인 뷰 (View)가 존재하며, 뷰 클래스는 UI 컴포넌트들을 위한 기본적인 구현 영역을 기술하고 있다.
- 하나의 뷰는 화면 상의 사각 영역을 차지한다. 또한 그리기와 이벤트 처리를 담당한다. 뷰는 위젯 (Widget)의 기반이 되는 클래스이다.
- 위젯 (Widget)은 버튼, 텍스트 필드 또는 사용자와 상호 작용하는 UI 컴포넌트이다. 뷰 그룹은 다른 뷰나 뷰 클래스를 담고 레이아웃 특성을 정의하며 눈에 보이지 않는 레이아웃의 기반이 되는 클래스이다.



## ● 앱의 기본 화면 구성 및 화면 이동



## ▲ 안드로이드 액티비티 생명주기



## ▲ 프래그먼트 생명주기

---

## ● 프래그먼트 생명주기 설명

### 1. OnAttach()

프래그먼트가 액티비티에 붙을 때 호출

### 2. OnCreate()

프래그먼트가 액티비티에 호출되어 생성되는 단계

### 3. OnCreateView()

프래그먼트에 속한 View 와 ViewGroup 에 대한 ui 바인딩 가능

### 4. OnActivityCreated()

액티비티에서 프래그먼트를 모두 생성하고 난 다음 호출, 액티비티와 프래그먼트를 연결하는 역할

### 5. OnStart()

프래그먼트가 사용자에게 보여지기 전에 호출되는 함수

### 6. OnResume()

프래그먼트가 화면에 보여지는 단계, 사용자와의 상호작용 가능

| 메소드         | 설명  | 다음 메소드                     |
|-------------|---|----------------------------|
| onCreate()  | 액티비티가 생성될 때 호출되며 사용자 인터페이스 초기화에 사용됨.                                      | onStart()                  |
| onRestart() | 액티비티가 멈췄다가 다시 시작되기 바로 전에 호출됨.   | onStart()                  |
| onStart()   | 액티비티가 사용자에게 보여지기 바로 직전에 호출됨.  | onResume() 또는 onStop()     |
| onResume()  | 액티비티가 사용자와 상호작용하기 바로 전에 호출됨.  | onPause()                  |
| onPause()   | 다른 액티비티가 보여질 때 호출됨. 데이터 저장, 스레드 중지 등의 처리를 하기에 적당한 메소드.                    | onResume() 또는 onStop()     |
| onStop()    | 액티비티가 더이상 사용자에게 보여지지 않을 때 호출됨. 메모리가 부족할 경우에는 onStop() 메소드가 호출되지 않을 수도 있음. | onRestart() 또는 onDestroy() |
| onDestroy() | 액티비티가 소멸될 때 호출됨. finish() 메소드가 호출되거나 시스템이 메모리 확보를 위해 액티비티를 제거할 때 호출됨.     | 없음                         |

## ▲ API 액티비티 생명주기

액티비티와 프래그먼트는 생명주기에 따라 적절한 메소드가 호출되므로 개발 전에 숙지한다면 액티비티 구성에 있어 더 원활한 작업이 가능하다.

졸업작품에서 첫 시작으로 앱 화면 구성을 다루게 되었는데, 생명주기를 기반으로 액티비티 및 프래그먼트 전환에 대해 학습하고 인텐트, 네비게이션 등의 기능을 통해 화면 전환을 구현하여 이를 프로젝트에 적용하였다.

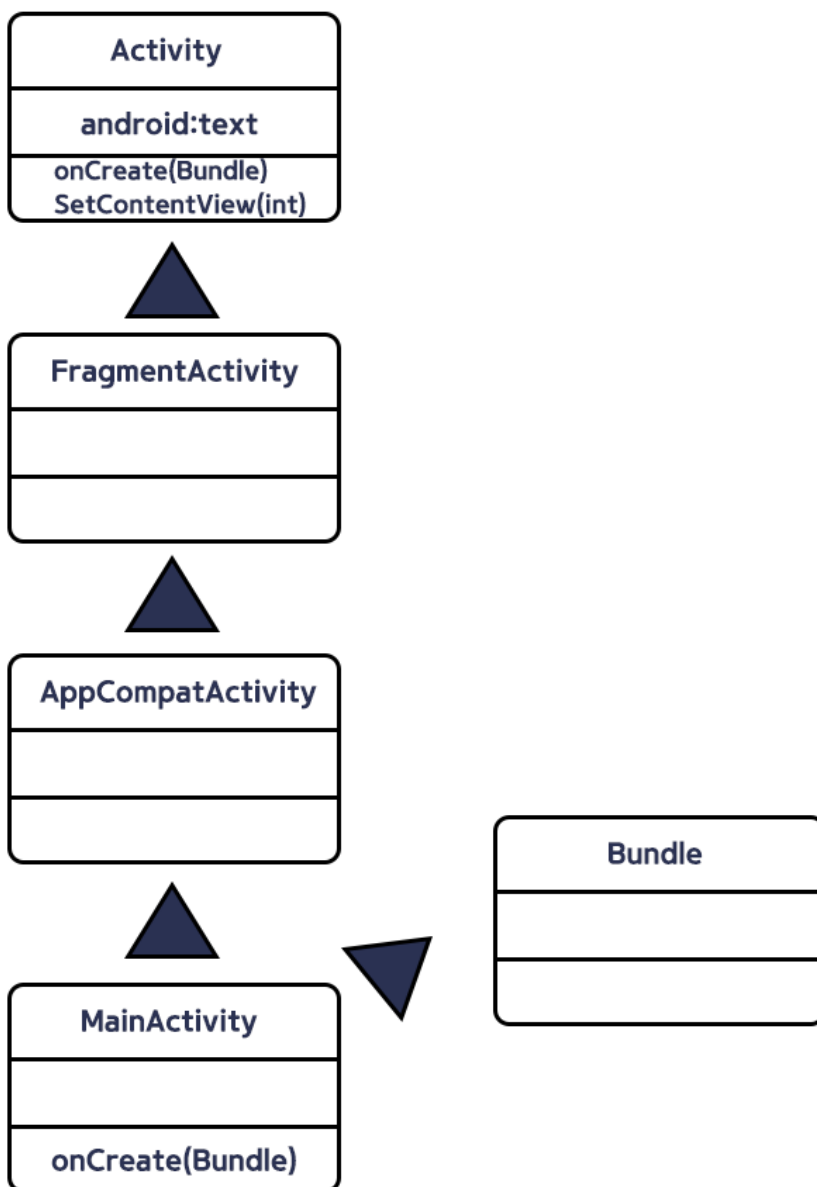
---

## ● 안드로이드 스튜디오의 레이아웃 구조

- **ConstraintLayout (제약 레이아웃)**
  - 제약 기반 모델
  - 제약 조건을 사용하여 화면 구성
  - 안드로이드 스튜디오에서 자동으로 설정되는 디폴트 레이아웃
- **LinearLayout (리니어 레이아웃)**
  - 박스 모델
  - 한쪽 방향으로 차례대로 뷰를 추가하여 화면 구성
  - 뷰가 차지할 수 있는 사각형 영역을 할당
- **RelativeLayout (상대 레이아웃)**
  - 규칙 모델
  - 부모 컨테이너나 다른 뷰와의 상대적 위치로 화면 구성
  - 제약 레이아웃을 사용하게 되면서 권장되지 않는 추세
- **FrameLayout (프레임 레이아웃)**
  - 싱글 모델
  - 가장 상위에 있는 하나의 뷰 또는 뷰그룹만 보여주는 구성
  - 여러 개의 뷰가 들어갈 경우 중첩해서 쌓임
  - 가장 단순하지만 여러 개의 뷰를 중첩한 후 뷰를 전환하여 보여주는 방식이므로 자주 사용

- **TableLayout** (테이블 레이아웃)
  - 격자 모델
  - 격자 모양의 배열을 사용하여 화면 구성
  - HTML 에서 많이 사용되는 정렬 방식과 유사

## ● 자바 클래스 다이어그램



## ● 프로젝트에 사용된 보건복지부 공공 API

### b) 요청 메시지 명세

| 항목명(영문)       | 항목명(국문)         | 항목크기 | 항목구분 | 샘플데이터               | 항목설명                  |
|---------------|-----------------|------|------|---------------------|-----------------------|
| serviceKey    | 인증키             | 100  | 1    | 인증키<br>(URL Encode) | 공공데이터포털에서<br>발급받은 인증키 |
| numOfRows     | 한 페이지 결과<br>수   | 4    | 0    | 10                  | 한 페이지 결과 수            |
| pageNo        | 페이지 번호          | 4    | 0    | 1                   | 페이지 번호                |
| startCreateDt | 데이터 생성일<br>시작범위 | 30   | 0    | 20200310            | 검색할 생성일 범위의<br>시작     |
| endCreateDt   | 데이터 생성일<br>종료범위 | 30   | 0    | 20200315            | 검색할 생성일 범위의<br>종료     |

### c) 응답 메시지 명세

| 항목명(영문)           | 항목명(국문)             | 항목크기 | 항목구분 | 샘플데이터                      | 항목설명                |
|-------------------|---------------------|------|------|----------------------------|---------------------|
| resultCode        | 결과코드                | 2    | 1    | 00                         | 결과코드                |
| resultMsg         | 결과메시지               | 50   | 1    | NORMAL<br>SERVICE          | 결과메시지               |
| numOfRows         | 한 페이지 결과 수          | 4    | 1    | 10                         | 한 페이지당 표출<br>데이터 수  |
| pageNo            | 페이지 수               | 4    | 1    | 1                          | 페이지 수               |
| totalCount        | 전체 결과 수             | 4    | 1    | 6                          | 전체 결과 수             |
| SEQ               | 게시글 번호(감염현황<br>고유값) | 30   | 0    | 74                         | 게시글 번호(감염현황<br>고유값) |
| STATE_DT          | 기준일                 | 30   | 0    | 20200315                   | 기준일                 |
| STATE_TIME        | 기준시간                | 30   | 0    | 00:00                      | 기준시간                |
| DECIDE_CNT        | 확진자 수               | 15   | 0    | 8162                       | 확진자 수               |
| CLEAR_CNT         | 격리해제 수              | 15   | 0    | 834                        | 격리해제 수              |
| EXAM_CNT          | 검사진행 수              | 15   | 0    | 16272                      | 검사진행 수              |
| DEATH_CNT         | 사망자 수               | 15   | 0    | 75                         | 사망자 수               |
| CARE_CNT          | 치료중 환자 수            | 15   | 0    | 7253                       | 치료중 환자 수            |
| RESUTL_NEG_CNT    | 결과 음성 수             | 15   | 0    | 243778                     | 결과 음성 수             |
| ACC_EXAM_CNT      | 누적 검사 수             | 15   | 0    | 268212                     | 누적 검사 수             |
| ACC_EXAM_COMP_CNT | 누적 검사 완료 수          | 15   | 0    | 251940                     | 누적 검사 완료 수          |
| ACC_DEF_RATE      | 누적 확진률              | 30   | 0    | 3.2396602365               | 누적 확진률              |
| CREATE_DT         | 등록일시분초              | 30   | 0    | 2020-03-15<br>10:01:22.000 | 등록일시분초              |
| UPDATE_DT         | 수정일시분초              | 30   | 0    | null                       | 수정일시분초              |



▲ 보건복지부 코로나 19 감염현황 API 를 활용한 확진자 현황 화면



## ● XmlPullParser 를 이용한 공공 API 활용

```
//////////외부 url 연결시//////////
String data = editLoc.getText().toString();

//한글을 인코딩해서 인터넷으로 내보내기
data = URLEncoder.encode(data, "UTF-8");

String requestURL = "http://www.google.co.kr/ig/api?weather="+data;
URL url = new URL(requestURL);

InputStream is = url.openStream();
//////////

//////////내부 xml 파일이용시//////////
InputStream is = getResources().openRawResource(R.raw.person);
//////////

XmlPullParserFactory factory = XmlPullParserFactory.newInstance();
XmlPullParser parser = factory.newPullParser();

parser.setInput(is, "UTF-8");
int eventType = parser.getEventType();

while(eventType != XmlPullParser.END_DOCUMENT) {

switch(eventType) {

case XmlPullParser.START_TAG:

String startTag = parser.getName();

if(startTag.equals("item")) { search = new SearchData(); }

//방법 1. 속성값일때 얻기
p.setName(parser.getAttributeValue(0));
p.setAge(parser.getAttributeValue(1));
p.setAddress(parser.getAttributeValue(2));

//방법 2. 태그값일때 얻기
if(search != null) {
```

---

```
if(startTag.equals("title")) {
String temp = parser.nextText();
temp=temp.replace(" ", "");
temp=temp.replace(" ", "");
search.setTitle(temp);
}
if(startTag.equals("link")) { search.setLink(parser.nextText()); }
if(startTag.equals("description")) { search.setDescription(parser.nextText()); }
if(startTag.equals("telephone")) { search.setTelephone(parser.nextText()); }
if(startTag.equals("address")) { search.setAddress(parser.nextText()); }
if(startTag.equals("mapx")) { search.setMapx(parser.nextText()); }
if(startTag.equals("mapy")) { search.setMapy(parser.nextText()); }
}

break;

case XmlPullParser.END_TAG:

String endTag = parser.getName();
if(endTag.equals("item")) { list.add(search); }

} //end switch

eventType = parser.next();

} //end while

}
```

---

- Firebase DataBase 를 이용한 FCM 푸시알림 구현

FirebaseMessagingService 오버라이드 메소드

**OnMessageReceived**

메시지가 수신되면 호출

**OnDeletedMessages**

FireBase Cloud Messaging Server 가 대기 중인 메시지를 삭제할 때 호출

**OnMessageSent**

메시지가 서버로 전송 성공 했을 때 호출

**OnSendError**

메시지가 서버로 전송 실패 했을 때 호출

**OnNewToken**

새로운 토큰이 생성될 때 호출

## ● FirebaseMessagingService 구현 예제

```
class MyFirebaseMessagingService : FirebaseMessagingService() {
    private val TAG = "FirebaseTest"

    // 메시지가 수신되면 호출
    override fun onMessageReceived(remoteMessage: RemoteMessage) {
        if(remoteMessage.data.isNotEmpty()){
            sendNotification(remoteMessage.notification?.title,
                remoteMessage.notification?.body!!)
        }
        else{

        }
    }

    // Firebase Cloud Messaging Server 가 대기중인 메시지를 삭제 시 호출
    override fun onDeletedMessages() {
        super.onDeletedMessages()
    }

    // 메시지가 서버로 전송 성공 했을때 호출
    override fun onMessageSent(p0: String) {
        super.onMessageSent(p0)
    }

    // 메시지가 서버로 전송 실패 했을때 호출
    override fun onSendError(p0: String, p1: Exception) {
        super.onSendError(p0, p1)
    }

    // 새로운 토큰이 생성 될 때 호출
    override fun onNewToken(token: String) {
        super.onNewToken(token)
        sendRegistrationToServer(token)
    }
}
```

```

private fun sendNotification(title: String?, body: String){
    val intent = Intent(this,MainActivity::class.java)
    intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP) // 액티비티 중복 생성 방지
    val pendingIntent = PendingIntent.getActivity(this, 0 , intent,
        PendingIntent.FLAG_ONE_SHOT) // 일회성

    val channelId = "channel" // 채널 아이디
    val defaultSoundUri = RingtoneManager.getDefaultUri(RingtoneManager.TYPE_NOTIFICATION) // 소리
    val notificationBuilder = NotificationCompat.Builder(this, channelId)
        .setContentTitle(title) // 제목
        .setContentText(body) // 내용
        .setAutoCancel(true)
        .setSound(defaultSoundUri)
        .setContentIntent(pendingIntent)

    val notificationManager = getSystemService(Context.NOTIFICATION_SERVICE) as NotificationManager

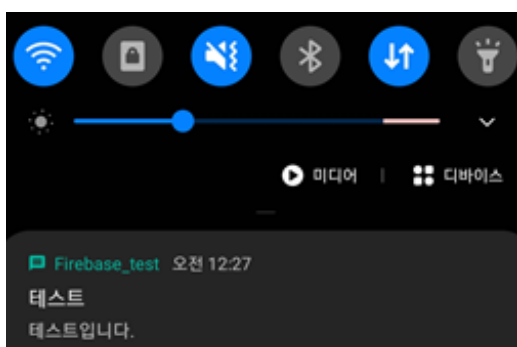
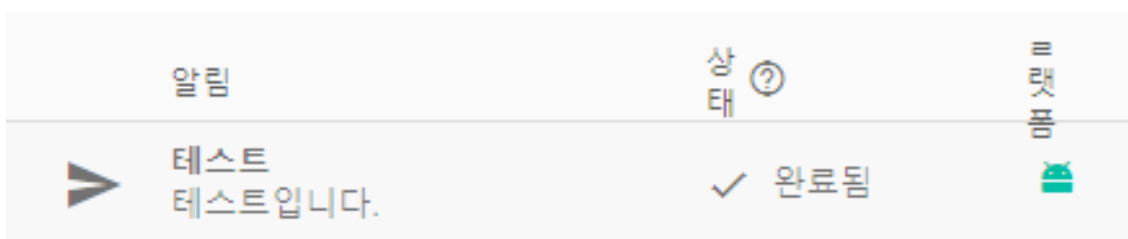
    // 오레오 버전 예외처리
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
        val channel = NotificationChannel(channelId,
            "Channel human readable title",
            NotificationManager.IMPORTANCE_DEFAULT)
        notificationManager.createNotificationChannel(channel)
    }

    notificationManager.notify(0 , notificationBuilder.build()) // 알림 생성
}

// 받은 토큰을 서버로 전송
private fun sendRegistrationToServer(token: String){

}
}

```



## ● 깃과 깃허브



● 깃(git)이란 형상 관리 도구 중 하나로, 컴퓨터 파일의 변경사항을 추적하고 여러 명의 사용자들 간에 해당 파일들의 작업을 조율하기 위한 분산 버전 관리 시스템이다.

● 깃(git)은 소스 코드를 따로 주고 받을 필요 없이, 하나의 프로젝트, 같은 파일을 여러 사람이 동시에 작업하는 병렬 개발이 가능하다는 장점이 있다.

● 깃허브(github)는 분산 버전 관리 툴인 깃(git)을 사용하는 프로젝트를 지원하는 웹 호스팅 서비스이다. 버전 관리 및 협업을 위한 코드 웹 호스팅 플랫폼으로 언제 어디서나 협업 프로젝트를 쉽게 진행할 수 있도록 돕는 역할을 한다.

## ● 깃허브 용어

| 용어                         | 기능   |
|----------------------------|--|
| 커밋 (Commit)                | 파일 추가 또는 변경 내용 저장                                |
| 푸시 (Push)                  | 파일 추가 또는 변경 내용<br>원격 저장소에 업로드                    |
| 브랜치 (Branch)               | 작업 중인 파일의 분기 생성                                  |
| 포크 (Fork)                  | 다른 원격 저장소의 파일<br>(프로젝트)를 나의 github 원격<br>저장소에 복사 |
| 클론 (Clone)                 | 다른 원격 저장소의 파일 복제 또는<br>참여                        |
| 작업 저장소 (Working Directory) | 내가 작업하려는 PC 내의 디렉토리                              |
| Staging Area               | 커밋 예정인 파일, 디렉토리 등이<br>모여 있는 곳                    |
| Local Repository           | 내 PC에 파일이 저장되는 개인<br>저장소                         |
| Remote Repository          | 원격 저장소   |
| 저장소 (Repository)           | 프로젝트의 저장 공간, repo라고도<br>함                        |

## ● 깃허브 명령어

| 명령어          | 기능   |
|--------------|--|
| git init     | 깃 저장소 초기화                                      |
| git help     | 깃 명령어 확인                                       |
| git status   | 저장소 체크   |
| git commit   | 변경사항 만든 후 스냅샷 남기기                              |
| git branch   | 나만의 변경사항 추가 시 커밋 타임라인 제작                       |
| git checkout | 현재 위치하지 않은 저장소 체크아웃                            |
| git merge    | 브랜치에서 작업 완료 후 Master 브랜치로 병합                   |
| git push     | 로컬 컴퓨터에서 작업 후 커밋을 깃허브에서도 온라인으로도 확인할 수 있도록 push |
| git pull     | 작업하고 있는 저장소의 최신 버전을 다운                         |



- 
- git reset, revert 로 되돌리기

- **reset**

```
git commit -m "1"
```

```
git commit -m "2"
```

```
git commit -m "3"
```

- ▲ commit

```
git reset HEAD^
```

- ▲ commit 을 직전 상황으로 되돌림

```
git reset HEAD~2
```

- ▲ 여러 개의 commit 이전으로 되돌리는 경우

- 1 커밋으로 되돌아감
- HEAD~2 부분에 커밋 해쉬를 사용해도 되돌아감

---

```
git commit -m "1"
```

```
git commit -m "2"
```

```
git commit -m "3"
```

```
git reset --hard [1 번 commit hash]
```

```
git push
```

▲ 위처럼 실행할 경우 2, 3 번 커밋 반영 내용 모두 삭제

```
git commit -m "1"
```

```
git commit -m "2"
```

```
git commit -m "3"
```

```
git reset --mixed [1 번 commit hash]
```

```
git add .
```

```
git commit -m "~"
```

```
git push
```

---

▲ 이력은 삭제되지만 코드는 남아있는 상태

Add 명령어를 통해 stage 에 반영하고 commit 가능

```
git commit -m "1"
```

```
git commit -m "2"
```

```
git commit -m "3"
```

```
git reset --soft [1 번 commit hash]
```

```
git commit -m "~"
```

```
git push
```

▲ 이력은 삭제되지만 변경 내용은 남아 stage 되어 있는 상태.

Add 명령어 없이 바로 commit 가능

## ● revert

```
git commit -m "1 번 커밋"
```

```
git commit -m "2 번 커밋"
```

```
git commit -m "3 번 커밋"
```

```
git revert [1 번 commit hash]
```

▲ 1 번 커밋에 해당하는 내용만 삭제

Revert "1 번 커밋"에는 1 번 커밋이 삭제된 이력 기입

```
Revert "1 번 커밋"
```

```
3 번 커밋
```

```
2 번 커밋
```

```
1 번 커밋
```

▲ git log 로 확인한 모습

---

```
git revert [커밋해쉬]..[커밋해쉬]
```

### ▲ 여러 개의 커밋 되돌리기

```
git revert [1 번커밋해쉬]..[2 번커밋해쉬]
```

```
git log
```

```
Revert "2 번커밋해쉬"
```

```
Revert "1 번커밋해쉬"
```

### ▲ git log 로 확인한 모습

## ● 졸업작품 팀 프로젝트 수행 후기



본 교과목을 통해 졸업작품 팀 프로젝트를 수행하면서, 하나의 프로젝트를 진행하는 데에 있어서 혼자서 수행하는 것보다 팀원들과 역할분담을 통해 서로 협력하며 수행하는 것이 더 효율적이라는 점을 깨달았습니다.

프로젝트 내 기능을 구현하는 데에 있어서 미흡한 점으로 인해 프로젝트 중 여러 차례 막힌 부분이 있었고, 이러한 시행착오를 팀원 간의 협력을 통해 해결해 나가면서 결국 최종발표까지 이르게 되었습니다. 물론 초기에 구상했던 기능 중에 구현에 어려움을 겪었던 여러 기능들이 다른 기능으로 대체되거나 누락되었지만 이러한 팀 프로젝트 경험으로 자신을 되돌아보고 다시 재정비할 수 있는 기회가 된 것 같아 좋은 경험이 되었다고 생각합니다.

또한, 진행해온 과정들을 포트폴리오를 통해 다시 한 번 정리하면서 복습할 수 있게 되었고, 포트폴리오와 같은 이러한 자료들은 추후 사회에 진출하게 되었을 때 분명 도움이 될 것이라 생각합니다.

프로젝트를 진행하면서 여러 차례 의사소통의 기회를 가지면서 서로 간의 의견이나 아이디어를 주고 받으며 미흡했던 점을 보완해 나갈 수 있었기 때문에 함께 팀 프로젝트를 수행한 팀원들에게 감사하다는 이야기를 전하고 싶습니다.