**Security Gallery**

**ANDROID STUDIO 에서**

**1대1 단일 치환 기법을 적용**

**동국대학교 경주캠퍼스**

**컴퓨터공학과**

**한지용**

gkswldyd555@naver.com

목차

[1. 서론 3](#_Toc39258954)

[2. 기본 연구 4](#_Toc39258955)

[**2.1 – Android Studio** 4](#_Toc39258956)

[**2.2 – 1대1 단일 치환 암호** 5](#_Toc39258957)

[3. 시스템 설계 5](#_Toc39258958)

[**3.1 – 전체적 구성** 5](#_Toc39258959)

[**3.2 – Manifests** 6](#_Toc39258960)

[**3.3 – Adapter** 8](#_Toc39258961)

[**3.4 – Fragment** 9](#_Toc39258962)

[**3.5 - Util** 12](#_Toc39258963)

[**3.5.1 – DB 생성** 12](#_Toc39258964)

[**3.5.2 – Utils** 13](#_Toc39258965)

[**3.6 - 4개의 구성요소(Activity)** 14](#_Toc39258966)

[**3.6.1 – ImageViewActivity** 14](#_Toc39258967)

[**3.6.2 – IntroActivity** 16](#_Toc39258968)

[**3.6.3 - LockImageListActivity** 19](#_Toc39258969)

[**3.6.4 – MainActivity** 20](#_Toc39258970)

[**3.7 – UI** 21](#_Toc39258971)

[**3.7.1 – Font** 21](#_Toc39258972)

[**3.7.2 – Layout** 22](#_Toc39258973)

[4. 실험 23](#_Toc39258974)

[**4.1 – 실험 환경** 23](#_Toc39258975)

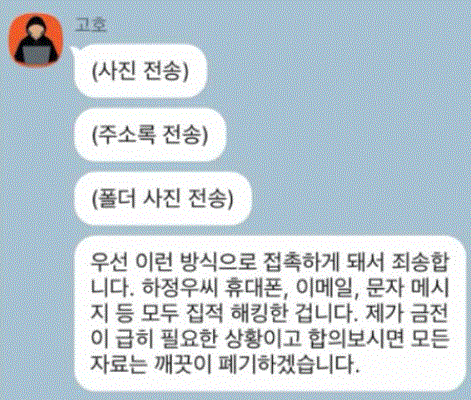
[**4.2 – 실험 진행** 23](#_Toc39258976)

[5. 결론 및 개선 24](#_Toc39258977)

# **1. 서론**

이동 단말기를 통한 전화, 사진, 인터넷 등의 기술이 발전함에 따라 스마트폰이 널리 퍼졌으며, 우리의 일상 속에서 스마트폰은 빠질 수 없는 존재가 되었다. 분명 여러 좋은 장점들이 있는 것은 사실이다. 하지만 본인은 단점들 중에서 스마트폰 보안에 관련되어 생각해 보았다. 스마트폰 보안과 관련되어 배우 ‘하정우’의 휴대폰 해킹 사건이 있었다. 휴대폰이 해킹 당함으로써, 많은 개인정보들이 해커에게 유출되었고, 그것을 약점으로 삼아 정보 유출을 시킨다는 명목 하에 거액의 금전을 요구하는 사건이다.

이 사건을 보아 우리는 스마트폰의 보안과 관련되어 심각히 위험성을 인지해야 함을 느끼는 바이다.



[그림1] 배우 하정우 휴대폰해킹 협박문자

따라서, 본 연구에서는 1:1 단일치환 암호방식을 이용하여 Android Studio 를 통하여

사진을 암호화시켜 보안하는 기능을 구현한다. 구현한 기술을 어플리케이션을 통하여

접목시켜 보면 사진을 보안 할 수 있다. 스마트폰 보안 방법으로 미약하지만 도움이 될 수 있을 것이다.

본 논문 2장에서는 Android Studio와 단일 치환암호에 대해 알아보고, 3장에서는 구현한 프로그램의 설명, 4장에서는 실험한 결과, 5장에서는 결론과 문제점 및 개선 해야 할

부분을 쓰며 마친다.

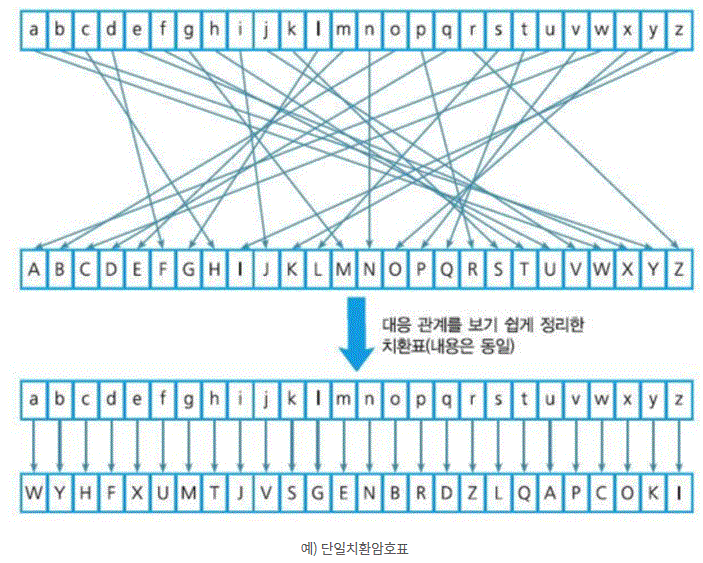
# **2. 기본 연구**

## **2.1 – Android Studio**

안드로이드 스튜디오는 안드로이드 및 안드로이드 전용 어플리케이션 제작을

위한 공식 통합 개발 환경(IDE)이다. 안드로이드 스튜디오는 gradle빌드 시스템을 사용하고 있고, 프로그래밍 언어로 자바와 코틀린을 지원한다.

## **2.2 – 1대1 단일 치환 암호**



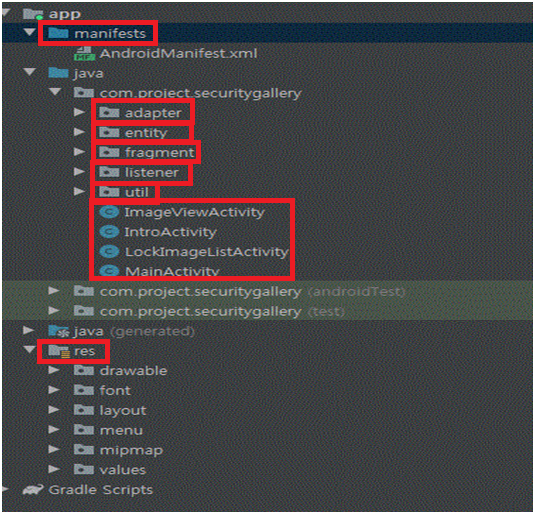
[그림2] 1대1 단일 치환 암호 예시

1:1 단일 치환 암호는 ‘시저 암호’ 또는 ‘카이사르 암호’로 불리며, 알파벳을 평행이동 시킴으로써 암호문을 생성하는 암호방식이다. 기존 알파벳 26문자를 무작위로 나열한 알파벳 집합과 서로 1대1 대응 시킴으로써 암호문을 생성한다.

1:1 단일 치환 기법의 복호화는 암호화 과정의 정반대이다.

# **3. 시스템 설계**

## **3.1 – 전체적 구성**



[그림3] 전체적 설계 구성

위 [그림3]과 같이 전체적 설계는 크게 Manifests, Java, Res(리소스) 부분으로

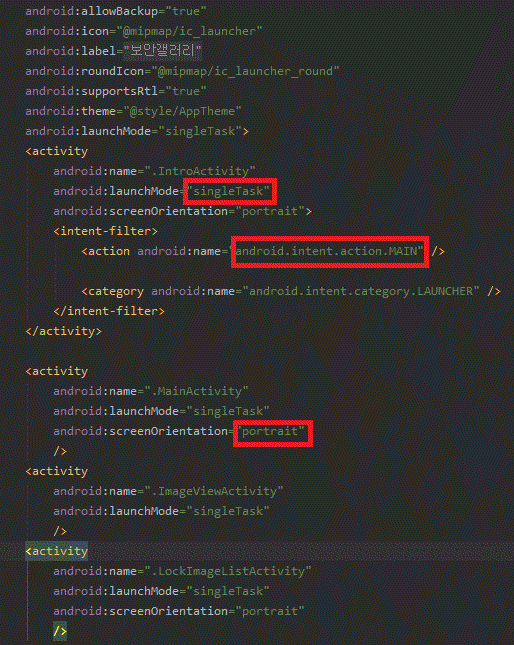
나뉜다. Manifests는 안드로이드 시스템이 코드 실행 전 가지고 있어야 하는 필수 정보를 시스템에 제공하는 부분이다. 또한 구성요소들의 클래스 이름을 지정하고 상호작용하기 위한 권한 생성 및 구성요소가 실행 될 수 있는 조건을 기술한다. 구성요소는 Java안에 4개의 Activity 부분이 있고, 이것이 구성요소가 된다.

Java는 설계 시, 필요한 코드들을 모아놓은 부분이다.

Res(리소스)는 어플리케이션을 만들 때 꼭 필요한 레이아웃(화면구성), 메뉴바,

폰트 등 디자인쪽을 담당하는 부분이다.

## **3.2 – Manifests**



[그림4] Manifests 구현

Manifests는 3-1 에서 말했듯이, 구성요소들의 권한 생성 및 구성요소가 실행 될

수 있는 조건을 기술한다.

구현한 어플리케이션에는 4개의 Activity 구성요소들을 가지고 있고 런치모드를 싱글태스크로 설정하였다. 런치모드를 싱글태스크로 지정하면, 시스템이 새 작업의 루트에 액티비티를 생성하고 인텐트를 해당 액티비티로 라우팅한다. 그러나 액티비티가 이미 존재하는 경우 새 인스턴스를 생성하는 대신 onNewIntent() 메소드를 호출하여 인텐트를 기존

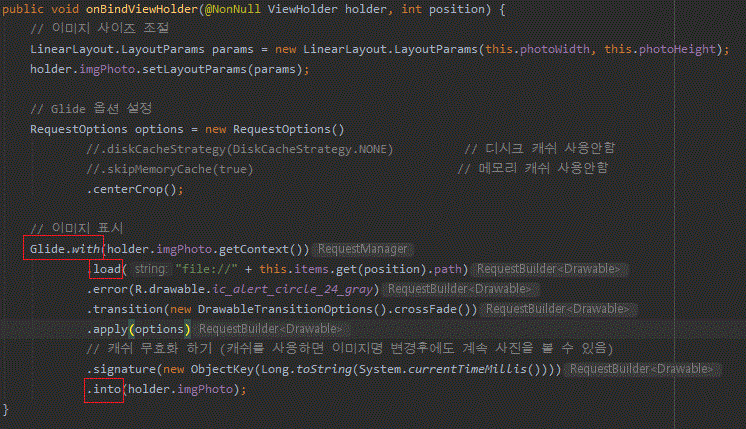
인스턴스로 라우팅한다.

IntroActivity부터 싱글태스크를 했으므로 인스턴스 Activity가 되고

인트로를 호출하고 필터안에 인텐트를 Main으로 줬기 때문에 시간이 지나 MainActivity로 넘어간다. 그 후로 3번째, 4번째 Activity가 실행된다.

Portait은 Activity 화면을 Layout(화면구성) 에서 세로로 고정하는 역할을 한다.

## **3.3 – Adapter**



[그림5] Adapter 구현

Adapter 부분은 사용자 데이터 리스트로부터 아이템 뷰를 만든 클래스이다.

쉽게 설명을 하면 초기 어플리케이션을 실행 시, 앨범의 리스트 항목들을 구현한

부분이다. Glide는 구글에서 공개한 이미지 라이브러리 이며, 웹 상에서 이미지를

로드하여 보여주기 위해 고려해야 할 사항들을 미리 구현하여, 사용자가 이용하기 쉽게 만든 라이브러리 이다.

Glide 클래스는 빌더 패턴으로 구현되어 있고, 3개의 필수 파라미터를 요구하는데,

With, Load, Into 가 있다. With는 안드로이드의 많은 API를 이용하기 위해 필요하며,

Load는 이미지 경로를 나타내고, into는 다운로드 받은 이미지를 보여줄 이미지 뷰이다.

Glide 클래스는 기본적으로 메모리 & 디스크에 이미지를 캐싱한다.

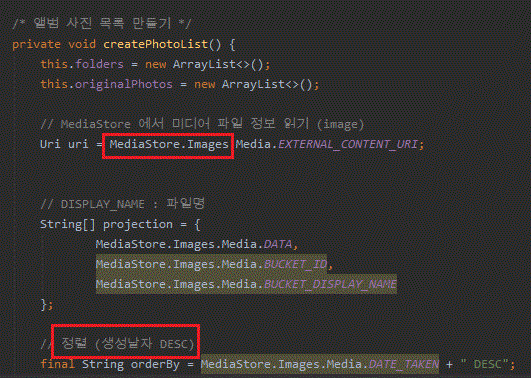
따라서 불필요한 네트워크 사용 연결을 줄이기 위하여

skipMemoryCache(true) = 메모리 캐싱 off

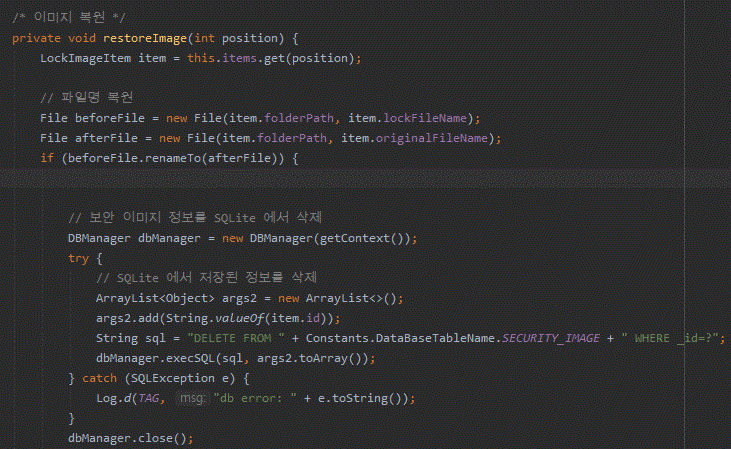
diskCacheStrategy(DiskCacheStrategy.NONE) = 디스크 캐싱 off

라는 명령어를 호출해야 한다.

## **3.4 – Fragment**



[그림6] Fragment 구현1



[그림7] Fragment 구현2

Fragment 부분은 3-1 전체적 구성에서 설명한 4개의 Activity 구성요소에 얹을 것들을

모아놓은 부분이다.

[그림6] Fragment 구현1 에서 보듯이 앨범 사진 목록 만들기라는 함수가 나중에

4개의 구성요소중 하나인 MainActivity에 얹혀 실행될 때 Fragment와 연동되어 실행

될 것이다. MediaStore라는 API를 사용하여 미디어 파일에 액세스 할 수 있는 인터페이스를 활용 하였고, 어플리케이션을 실행 시 앨범 파일의 날짜를 DESC(내림차순)

형식으로 정렬하게 하였다.

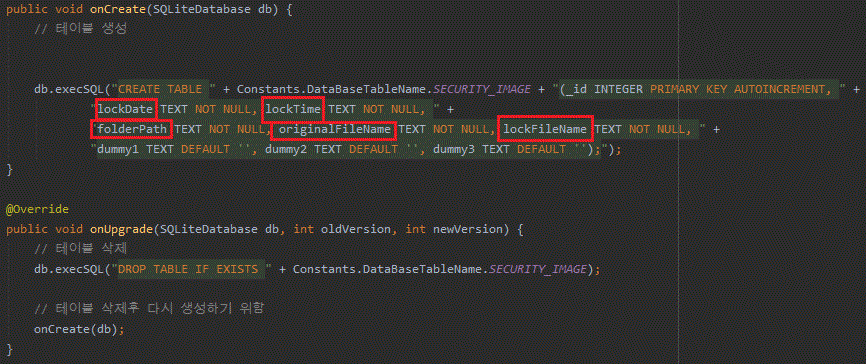
[그림7] Fragment 구현2 에서 사진 파일을 보안하고 복원 할 때의 함수를 구현 하였는데 기본적으로 안드로이드 스튜디오는 내장DB를 지원한다.

따라서 SQLite를 활용하여 보안된 파일 리스트에서 파일을 제거하고 기존폴더로

다시 복구하는 작업을 구현하였다.

## **3.5 - Util**

### **3.5.1 – DB 생성**

****

[그림8] DB 생성

기본적으로 안드로이드 스튜디오에서는 내장 DB를 지원한다. 따라서 본인은 DB에

보안 정보들을 담고 [그림8]과 같이 onCreate 라는 명령어를 사용하여 DB 테이블을

생성하고 DB 테이블이 생성되었으면, 그 후 다시 실행하였을 때는 DB 테이블이 있으므로 onCreate는 실행이 되지 않는다 . 따라서 버전을 더 높은값(업그레이드)을 준다면

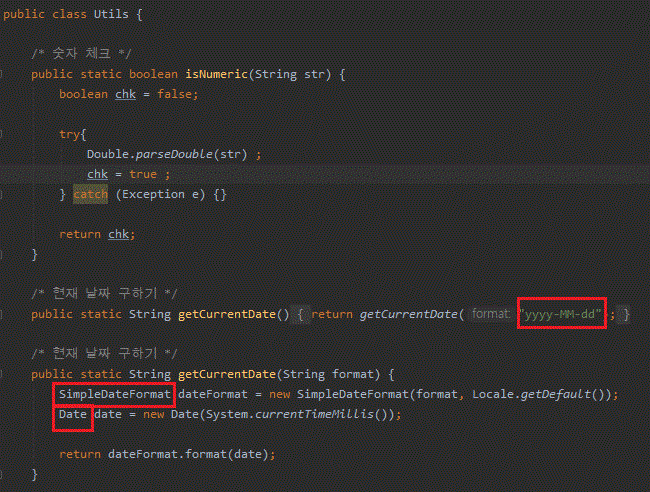
다시 onUpgrade가 실행되어 기존의 테이블을 지운 후 onCreate를 실행하여 새로운

테이블을 생성하도록 구현하였다.

lockDate : 잠금일 / lockTime : 잠금시간 / folderPath : 폴더경로

originalFileName : 원본파일이름 / lockFileName : 보안파일이름 을 의미한다.

### **3.5.2 – Utils**

****

[그림9] Utils 구현

Utils 부분은 1대1 단일 치환 암호 기법을 사진파일에 적용 했을 시, 새로운 파일을

생성한다. 하지만 파일을 생성하기 위해서는 어플리케이션 작동 당시의 현재 시간과

현재의 날짜를 알아야 생성을 할 수 있다.

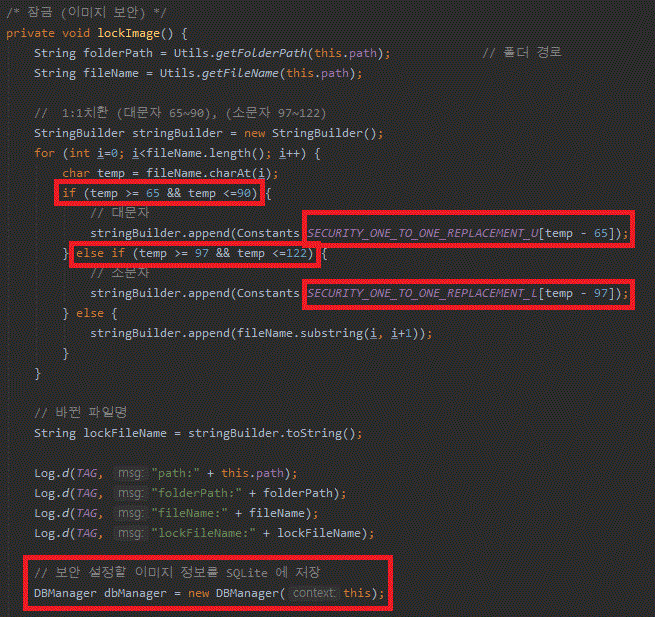
따라서 위 [그림9] 와 같이 현재의 날짜를 구하기 위하여 Date 라이브러리를 import

하여 사용 하였으며, SmpleDateFormat 라이브러리를 import하여 yyyy-MM-dd

형식으로 출력을 하였다.

## **3.6 - 4개의 구성요소(Activity)**

### **3.6.1 – ImageViewActivity**



[그림10] ImageViewActivity 구현

ImageViewActivity 구성요소 부분은 이미지 파일의 정보를 어떻게 보여줄 것 인가에

대한 부분들을 구현한 클래스이다. 가장 중요하게 보아야 할 부분은 1대1 단일 치환

기법을 사용하여 보안한 이미지 파일을 안드로이드 스튜디오 내장DB에 저장을 하는 것이다.

본인은 이라는 것을 따로 함수로 만들어 놓고, 그 함수에는 대문자 ‘A’ 는 ‘Z’로 바뀌고 ‘B’ 는 ‘Y’ 로 소문자 ‘a’ 는 ‘z’ ‘b’는 ‘y’

처럼 1대1 매칭을 역순으로 배열 형식으로 지정해 놓았다.

ASCII 코드에서 보면 대문자는 65번 ~ 90번까지, 소문자는 97번 ~ 112번까지 이다.

만약 ‘a’ 를 읽는다면, temp에 97이 들어가며 [temp – 97] 는 ‘0’ 이 되므로

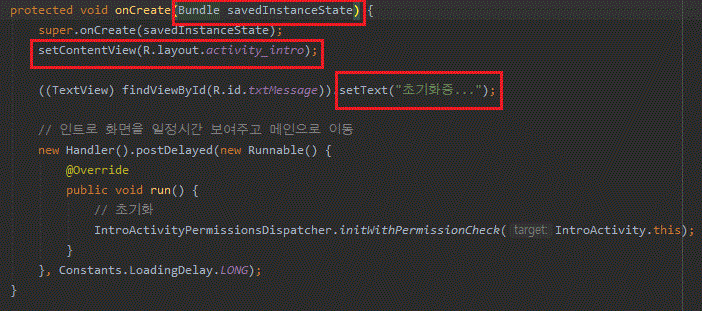
SECURITY\_ONE\_TO\_ONE\_REPLACEMENT\_L 함수에서 첫번째 배열의 값 z를 불러온다.

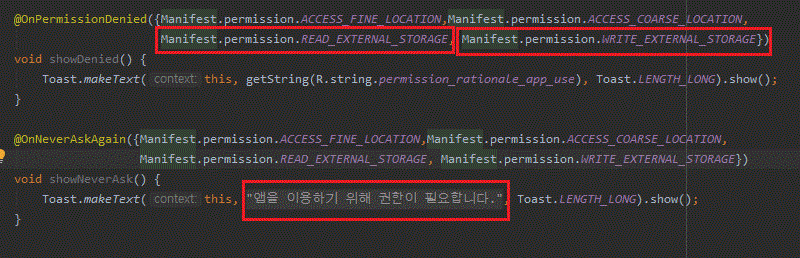
따라서, 위 [그림10] 과 같이 1대1 단일 치환 기법을 적용하였으며, 적용된 기법으로

새로 생성된 파일을 안드로이드 스튜디오 내장DB에 저장하도록 구현하였다

.

### **3.6.2 – IntroActivity**

[그림11] IntroActivity 구현



[그림12] Permission 권한 구현

IntroActivity 부분은 어플리케이션 실행 시, 초창기 화면을 어떻게 구성할 지에 대하여

구현 하였으며, 해당 어플리케이션을 스마트폰에서 설치 할 시, 앱을 사용하기 위한

퍼미션(권한) 을 생성하는 부분을 구현한 구성요소 부분이다.

[그림11]을 보면, Bundle saveInstanceState라는 것을 사용하였는데, Bundle은 사전적 의미

로는 꾸러미, 묶음이란 뜻이다. 안드로이드 스튜디에서의 Bundle은 Bundle 객체가

상태 정보를 저장하고 있다란 뜻이다. 즉 새로운 활동 이전의 상태 정보를 저장 하고 있다는 뜻이며, 어플리케이션이 실행 혹은 종료했다가 다시 불러 올 때 Bundle의 상태

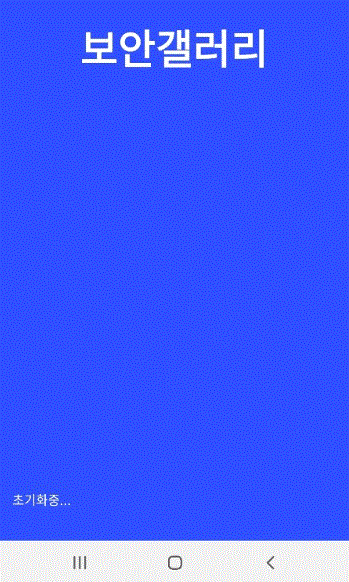
정보를 가져오며, saveInstanceState는 어플리케이션이 셧다운 된 후, 다시 실행될 경우

Bundle은 가장 최근에 공급된 상태 정보를 포함한 데이터를 가져온다는 뜻으로 쓰인다.

어플리케이션을 실행 시, Res(리소스)의 Layout 클래스의 Activity\_intro를 불러오며

초기화중… 이라는 메시지를 띄워 Intro 초창기 화면을 구현하였다.

아래의 사진은 스마트폰에서 [그림11]을 작동시킨 결과이다.



[그림13] 그림11을 스마트폰에서 작동한 결과

[그림12]는 어플리케이션 설치 시, 스마트폰의 저장소(Storage)에 읽기, 쓰기 권한을 부여

하기 위하여 구현한 부분이다.

3-2 Manifests 에서 4개의 Activity에 실행 조건을 기술한다고 하였다.

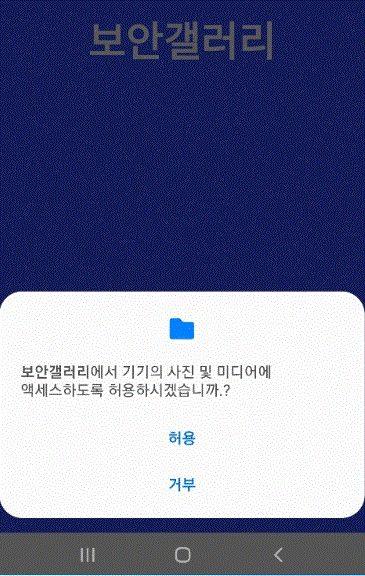
Manifests 부분에서 스마트폰의 저장소에 읽기, 쓰기 권한을 주게 하였으며

Manifest.permission.READ\_EXTERNAL\_STORAGE Manifest.permission.WRITE\_EXTERNAL\_STORAGE

위와 같은 코드로 IntroActivity 클래스에서 Manifests 클래스를 호출하여 그곳에 쓰여진 퍼미션을 사용 할 수 있게끔 구현하였다. READ는 저장소를 읽는 권한 WRITE는 저장소에

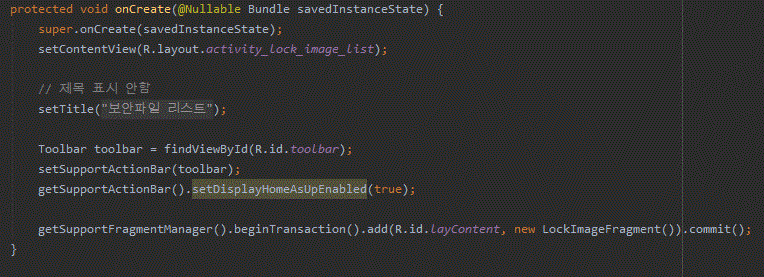
쓰기 권한을 요청하는 부분이다.

아래의 사진은 스마트폰에서 [그림12]을 작동시킨 결과이다



[그림14] 그림12를 스마트폰에서 작동한 결과

### **3.6.3 - LockImageListActivity**

****

[그림15] LockImageListActivity 구현

LockImageListActivity 부분은 1대1 단일 치환 기법을 적용한 이미지 파일을 모아둔

리스트를 구현한 부분이다.

Res(리소스)의 Layout 클래스의 activity\_lock\_image\_list를 호출하여 화면을 설정하고

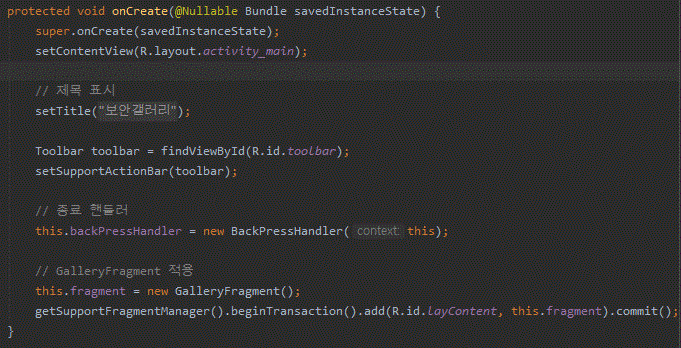
위 구성요소에서도 Bundle saveInstanceState를 사용하여 앱을 작동 시, 이전의 상태

정보들을 저장하고 있게끔 하였다. 3.4 Fragment에 보안리스트들에 대한 정보들을

구현한 부분이 있고, 그 부분을 LockImageListActivity에 얹어서 사용한다.

구현한 화면은 4장에서 확인 할 수 있을 것이다.

### **3.6.4 – MainActivity**



[그림16] MainActivity 구현

MainActivity 부분은 앱의 메인화면을 담당하는 구성요소 부분이다.

LockImageListActivity이 보안한 이미지 파일을 모아둔 리스트를 구현하였다면,

MainActivity는 스마트폰의 Storage에 있는 이미지 파일을 모두 보여준다.

Res(리소스)의 Layout 클래스의 activity\_main을 호출하여 화면을 설정하고

위 구성요소에서도 Bundle saveInstanceState를 사용하여 앱을 작동 시, 이전의 상태

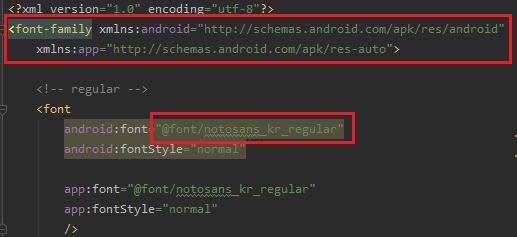
정보들을 저장하고 있게끔 하였다. [그림6] , [그림7]을 보면 MainActivity에서 사용 할

부분을 구현하였다.

## **3.7 – UI**

UI 부분은 프로그램 설계 시 Res(리소스) 부분에서 설계를 했다.

### **3.7.1 – Font**



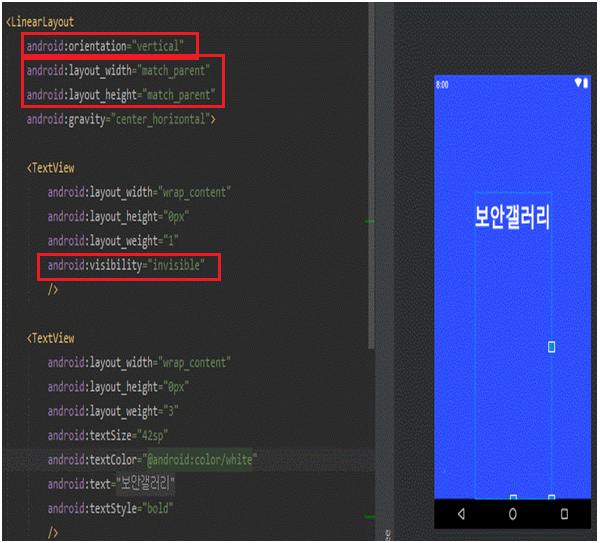
[그림17] 적용한 폰트

Font 부분은 상업적으로 무료로 사용 가능한 본고딕(Noto Sans)체를 지정하여

앱의 전체적인 부분의 폰트를 담당한다. 다른 폰트를 사용 하고 싶을 때에는,

oft파일(폰트파일)을 다운로드 받아 Font 클래스에 추가하고 xml 경로를 바꿔주면 된다.

### **3.7.2 – Layout**



[그림18] Layout 구현

Layout 클래스는 View 위젯들을 화면에 배치하는 과정에서 위치를 정렬

하거나, 그룹화하는 역할을 수행한다.

위 [그림18]은 초기 어플리케이션 실행 시 나타나는 인트로 화면이다.

Layout에는 orientation 이라는 속성을 가지고 있고 vertical은 세로를 나타낸다.

Vertical에 값을 넣으면 세로형태로 쌓이며, layout의 width & height를

match\_parent 값을 줘 꽉차게 파란배경으로 구성했다.

‘보안갤러리’ 라는 글씨가 들어갈 layout을 구현하였고, invisible값을 줘 layout을 감췄으며 그안에 텍스트를 적는 부분을 구현하였다.

본인은 표시형태가 직관적으로 나타날 수 있고 View를 수직 또는 수평 방향으로

배치 할 수 있는 LinearLayout 형태를 사용했다.

# **4. 실험**

## **4.1 – 실험 환경**

|  |  |
| --- | --- |
| Android Studio version | Android Studio 3.6.1 |
| Android Studio Amulator | Nexus 5X API 28 |
| 실험한 스마트폰 기종 | Samsung Galaxy S8+ |
| 스마트폰 Android version | Android 9 (Pie) |

[표1] 실험 환경

## **4.2 – 실험 진행**

# **5. 결론 및 개선**

**참고문헌**

**[1] 위키백과, 우리 모두의 백과사전**

<https://ko.wikipedia.org/wiki/%EC%95%88%EB%93%9C%EB%A1%9C%EC%9D%B4%EB%93%9C_%EC%8A%A4%ED%8A%9C%EB%94%94%EC%98%A4>

**[2] 안드로이드 개발자**

<https://developer.android.com/?hl=ko>

**[3] Leaf Pic(앨범 어플)**

<https://github.com/HoraApps/LeafPic>