모바일프로그래밍기초
- 안드로이드(Android) -

- 안드로이드 기능/실습 6
  - 프레퍼런스

- 안드로이드 기능/실습 7
  - 내부 파일 저장

❍ 목적

- ❑ 기본 데이터, 입력, 설정 정보 저장 혹은 전달한 데이터를 저장, 관리하기 위한 데이터를 다루기 위한 기본을 습득

❍ 기능/실습 내용

- ❑ 내용
  - ❖ 프레퍼런스
  - ❖ 파일
  - ❖ 데이터베이스

- ❑ 프레퍼런스와 파일 부분에 대해서 학습, 데이터베이스 및 외부 저장소는 별도 진행

❑ 안드로이드 기능 / 실습 6
프레퍼런스

목적

- 프레퍼런스 기초 사용법 습득

내용

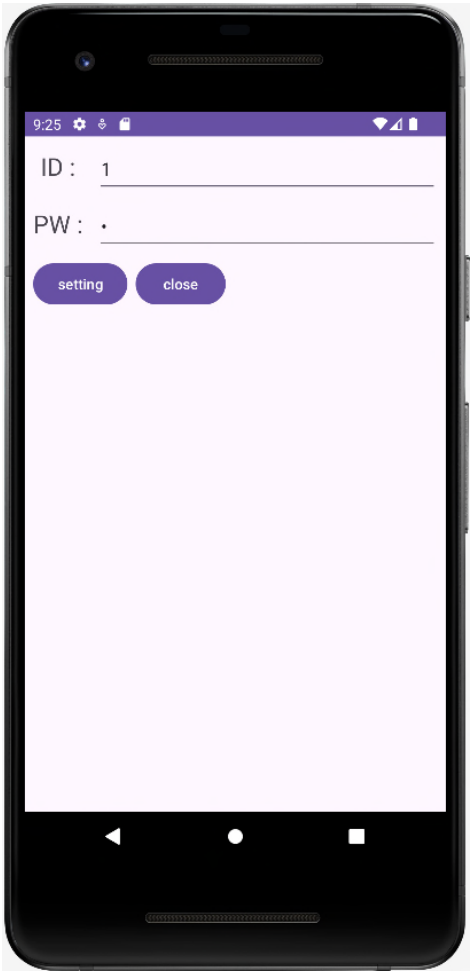
- 사용자가 입력한 정보, 환경 설정 정보 등을 간단히 저장 관리 할 수 있는 프레퍼런스의 사용법을 간단한 액티비티 구현을 통해서 확인
- 세부적인 기능은 개별 확인

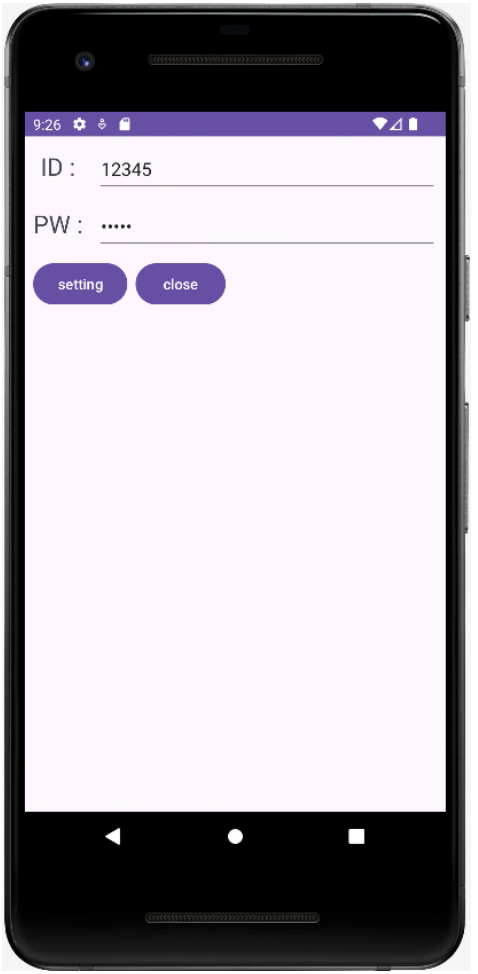
❑ 프레퍼런스(Preference)

- 제일 단순한 저장 형태
- 각 애플리케이션에 고유한 설정값을 지정
- <키, 값>의 조합으로 데이터 저장
  - 값에 이름을 부여하여 저장
  - 환경설정에 유용
- 주요 메소드
  - SharedPreferences 인터페이스
    - ❖ getSharedPreferences()에 의해 반환된 프레퍼런스 객체를 접근/수정 제공
    - ❖ SharedPreferences.Editor editor = pref.edit()
      - ▪ 프레퍼런스 객체를 수정 후, commit() 또는 apply() 연산으로 배치 처리

❑ 실행 모습

- 처음 실행 시 기본 설정된 값으로 동작
- 사용자 정보를 입력 후에 setting 버튼을 누르면 해당 정보를 저장, 다시 실행 시 해당 정보를 불러와 사용



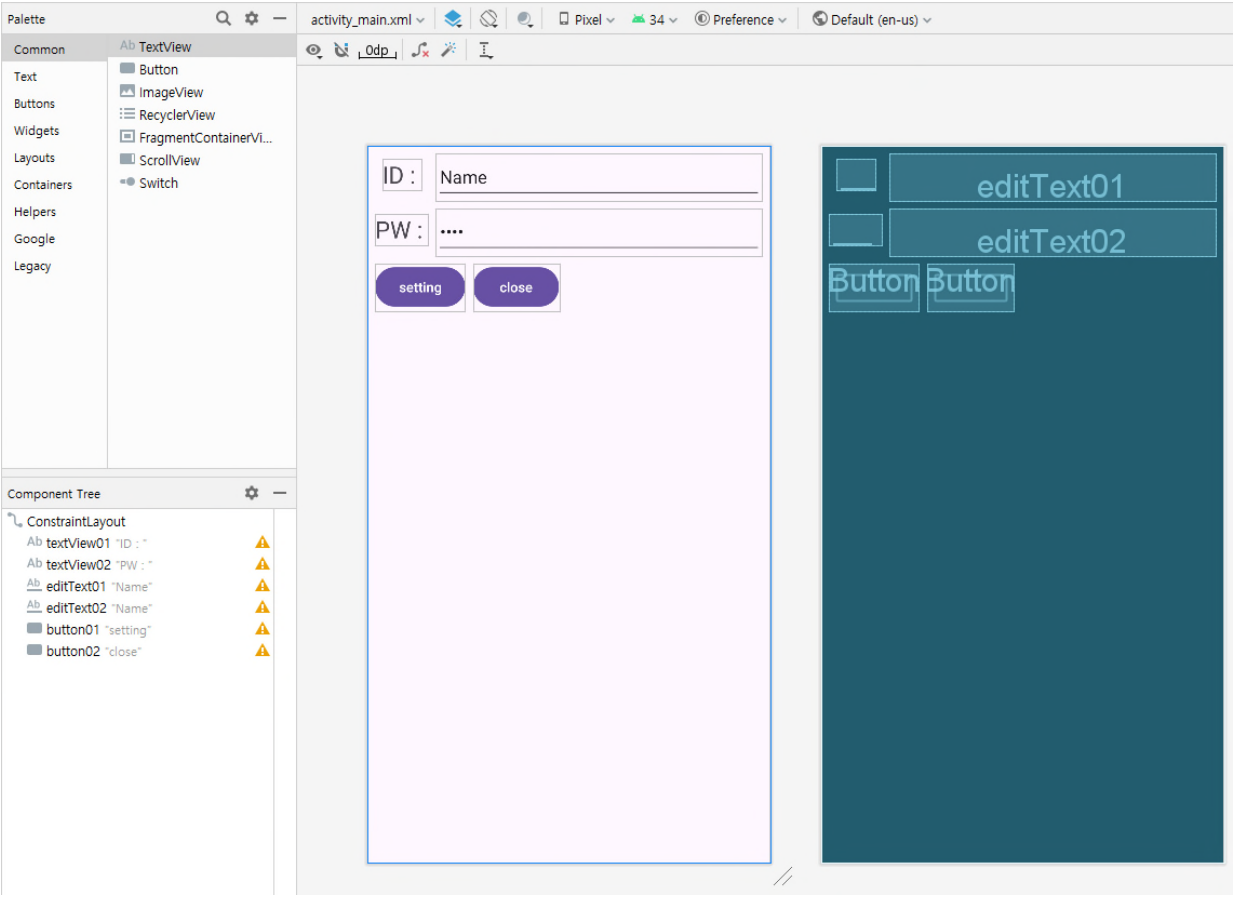


❑ 레이아웃 구성

- 프레퍼런스 기능을 확인하기 위한 부분이라 간단한 사용자 입력창 만으로 구성
  - 여기서는 setting 버튼을 눌렀을 때만 사용자가 입력한 정보를 프레퍼런스를 이용하여 저장하도록 구성

1개는 일반 EditText
1개는 Password
형태로 입력

사용자가 입력한
정보를 프레퍼런스에
저장하기 위한
버튼 1개와
종료버튼 1개로 구성



❑ 레이아웃 작성

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/textView01"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="ID : "
        android:textSize="24sp"
        app:layout_constraintBaseline_toBaselineOf="@+id/editText01"
        app:layout_constraintEnd_toEndOf="@+id/textView02"
        app:layout_constraintStart_toStartOf="@+id/textView02" />

    <TextView
        android:id="@+id/textView02"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:text="PW : "
        android:textSize="24sp"
        app:layout_constraintBaseline_toBaselineOf="@+id/editText02"
        app:layout_constraintStart_toStartOf="parent" />
```

❑ 레이아웃 구성

```
<EditText
    android:id="@+id/editText01"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginTop="8dp"
    android:layout_marginEnd="8dp"
    android:ems="10"
    android:inputType="text"
    android:text="Name"
    android:minHeight="48dp"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="@+id/editText02"
    app:layout_constraintTop_toTopOf="parent" />
```

```
<EditText
    android:id="@+id/editText02"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginStart="8dp"
    android:layout_marginTop="8dp"
    android:layout_marginEnd="8dp"
    android:ems="10"
    android:inputType="textPassword"
    android:text="Name"
    android:minHeight="48dp"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toEndOf="@+id/textView02"
    app:layout_constraintTop_toBottomOf="@+id/editText01" />
```

❑ 레이아웃 구성

```
    <Button
        android:id="@+id/button01"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:text="setting"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/editText02" />

    <Button
        android:id="@+id/button02"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:text="close"
        app:layout_constraintBaseline_toBaselineOf="@+id/button01"
        app:layout_constraintStart_toEndOf="@+id/button01" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

❑ 코드 분석 / 설명

```
package com.practice.ex.preference;

import androidx.appcompat.app.AppCompatActivity;

import android.content.SharedPreferences;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;

public class MainActivity extends AppCompatActivity implements View.OnClickListener {

    3 usages
    EditText edit01, edit02;
    2 usages
    Button btn01, btn02;

    4 usages
    SharedPreferences spref;
    4 usages
    SharedPreferences.Editor editor;
```

getSharedPreference()에 의해 반환된 프레퍼런스 객체를 접근/수정 제공

객체의 값을 수정하는데 사용되는 인터페이스
SharedPreference.Editor editor = spref.edit()
프레퍼런스 객체를 수정 후, commit() 또는 apply() 연산으로 처리

❑ 코드 분석 / 설명

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    edit01 = (EditText) findViewById(R.id.editText01);
    edit02 = (EditText) findViewById(R.id.editText02);

    btn01 = (Button) findViewById(R.id.button01);
    btn02 = (Button) findViewById(R.id.button02);

    btn01.setOnClickListener(this);
    btn02.setOnClickListener(this);

    spref = getSharedPreferences( name: "gpref", MODE_PRIVATE);
    editor = spref.edit();

    String temp1 = spref.getString( key: "editText01",  defValue: "1");
    String temp2 = spref.getString( key: "editText02",  defValue: "2");

    edit01.setText(temp1);
    edit02.setText(temp2);
}
```

프레퍼런스 설명 참조

디폴트값

❑ 코드 분석 / 설명

```
    @Override
    public void onClick(View v) {
        String txt01, txt02;
        txt01 = edit01.getText().toString();
        txt02 = edit02.getText().toString();

        if(v.getId() == R.id.button01) {
            editor.putString("editText01", txt01);
            editor.putString("editText02", txt02);
            editor.apply();
        }
        if(v.getId() == R.id.button02) {
            finish();
        }
    }
}
```

프레퍼런스 설명 참고

❑ 안드로이드 기능 / 실습 7
파일 저장 – 파일 다루기

❍ 목적

- ❑ 액티비티 간의 실행, 데이터 전달, 수정 등의 작업이 수행될 때 필수적인 데이터 저장 관리를 위함
  - ❖ 데이터베이스(SQLite) 부분은 추후 진행

❍ 내용

- ❑ 메모장과 같이 작성한 내용을 저장할 수 있도록 함
- ❑ 어플리케이션의 재 시작시 이전 실행시 최종 작성한 결과를 보여주도록 함
  - ❖ 액티비티 생명주기 이해
- ❑ 작성한 내용을 저장하는 파일이 없거나 처음 실행하는 경우에는 기본적으로 지정해서 가지고 있는 파일을 보여줌

❑ 파일

- 디바이스나 저장 매체에 직접적으로 파일을 저장
- 데이터 읽기
  - Context.openFileInput() 메소드 호출, FileInputStream 오브젝트를 얻어서 처리
- 데이터 쓰기
  - Context.openFileOutput() 메소드 호출, FileOutputStream 오브젝트를 얻어서 처리

- 작성된 파일을 패키지에 포함시키는 방법
  - 파일 위치 : res/raw
  - 특징 : 읽을 수만 있음
  - Resource 클래스의 openRawResource() 메소드를 호출, 매개변수로 파일명 대신 리소스 ID를 지정하여 이용

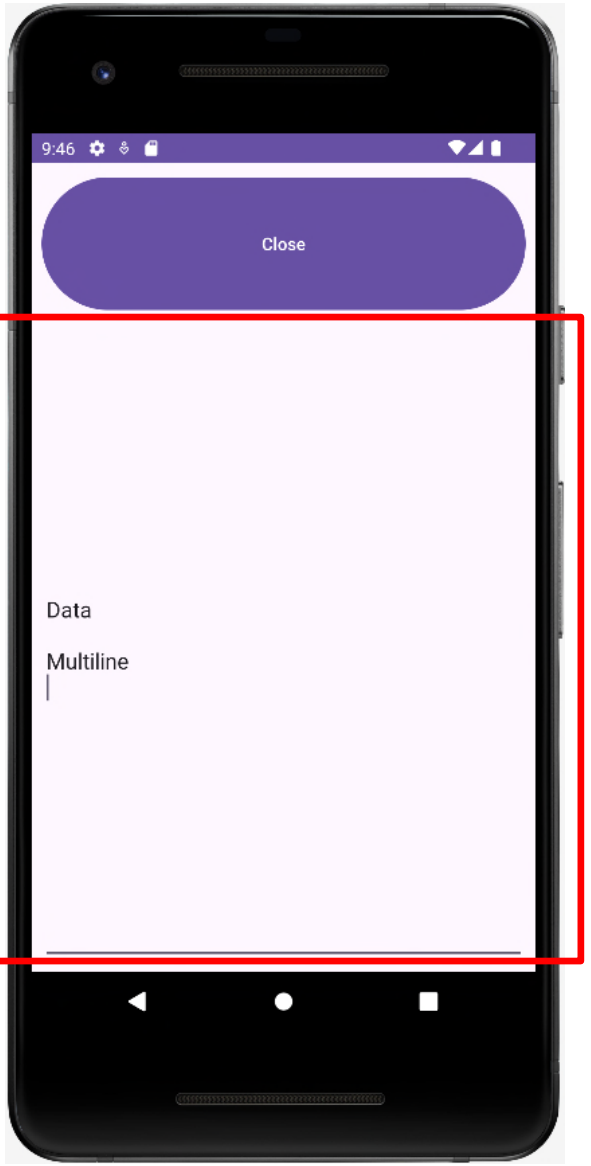
```
InputStream in = getResources().openRawResource(R.raw.data_name);
```

❑ 실행 모습

작성한 내용이 없는 경우, 가지고 있는 원본 파일 (raw/raw_data.txt)을 보여줌.

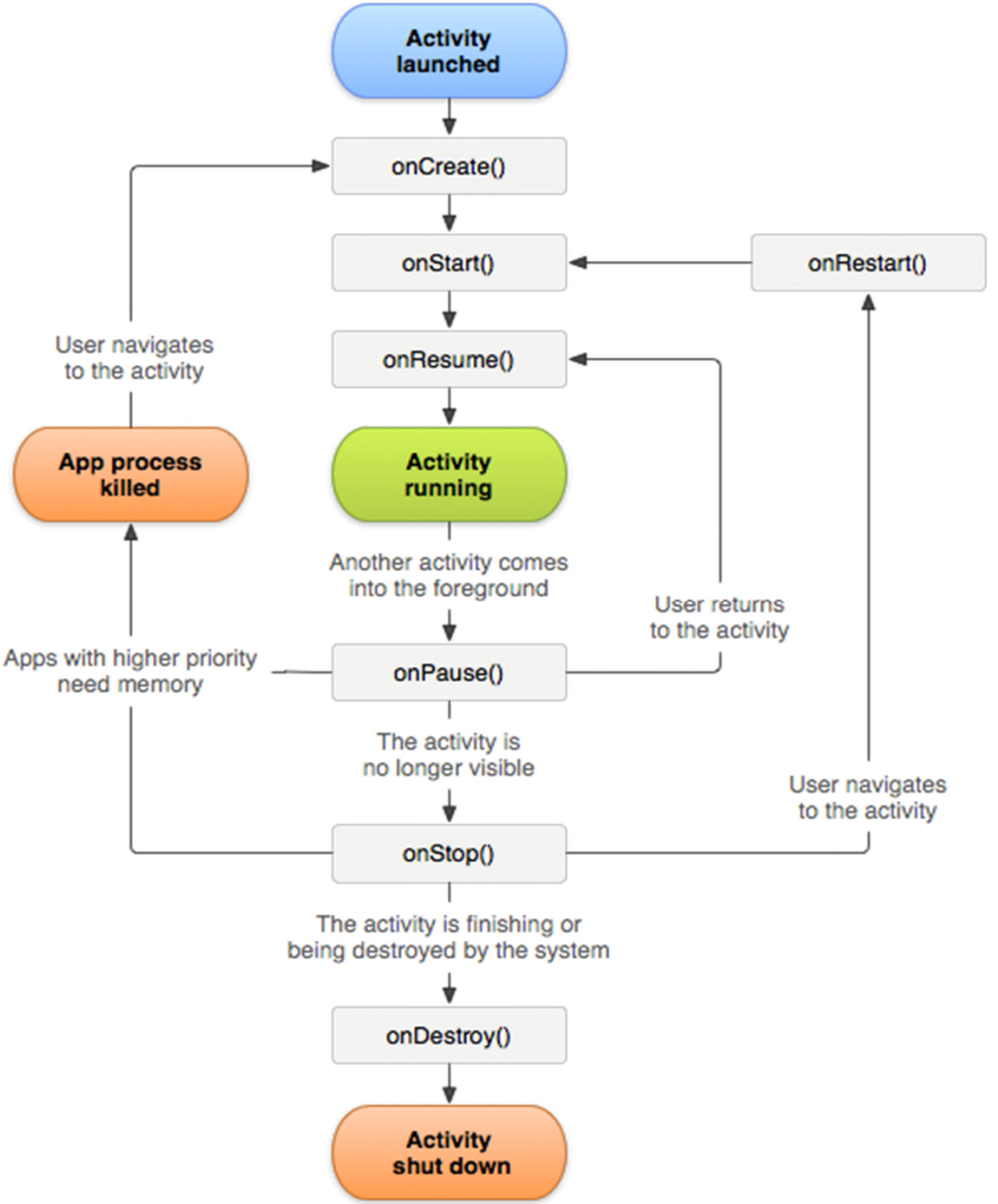
기존에 사용자가 작성한 내용이 있는 경우에는 저장된 작성 내용을 보여줌.

❑ 실행 모습 (사용자 저장 내용이 있는 경우)



작성된 내용이 있을 경우,
실행시에 가지고 있는
원본 파일 (raw/raw_data.txt)
이 아닌 작성되어 저장된 내
용이 보임

❑ 액티비티 생명주기


Activity launched
onCreate()
onStart()
onRestart()
onResume()
User navigates to the activity
App process killed
Activity running
Another activity comes into the foreground
User returns to the activity
Apps with higher priority need memory
onPause()
The activity is no longer visible
User navigates to the activity
onStop()
The activity is finishing or being destroyed by the system
onDestroy()
Activity shut down

❑ 진행 단계

- 해당 작업을 수행하기 위해서 필요한 내용, 작업 순서를 생각해 보세요.

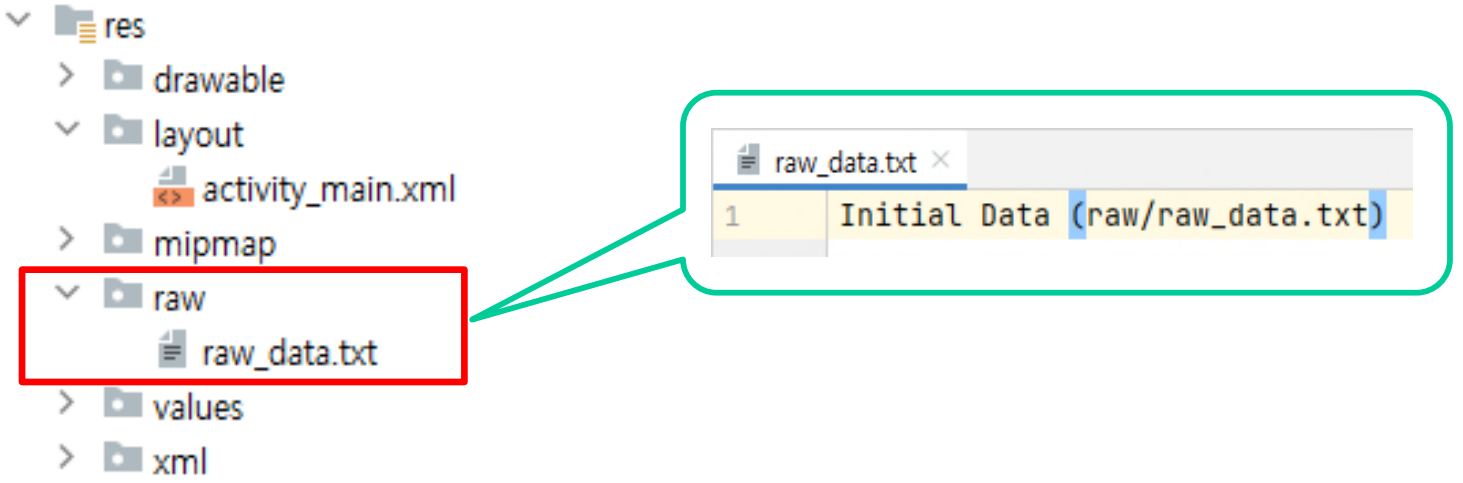
- 필요한 구성 요소는 무엇인가?

- 진행 순서는 어떻게 되는가?

❑ 원본 파일 추가

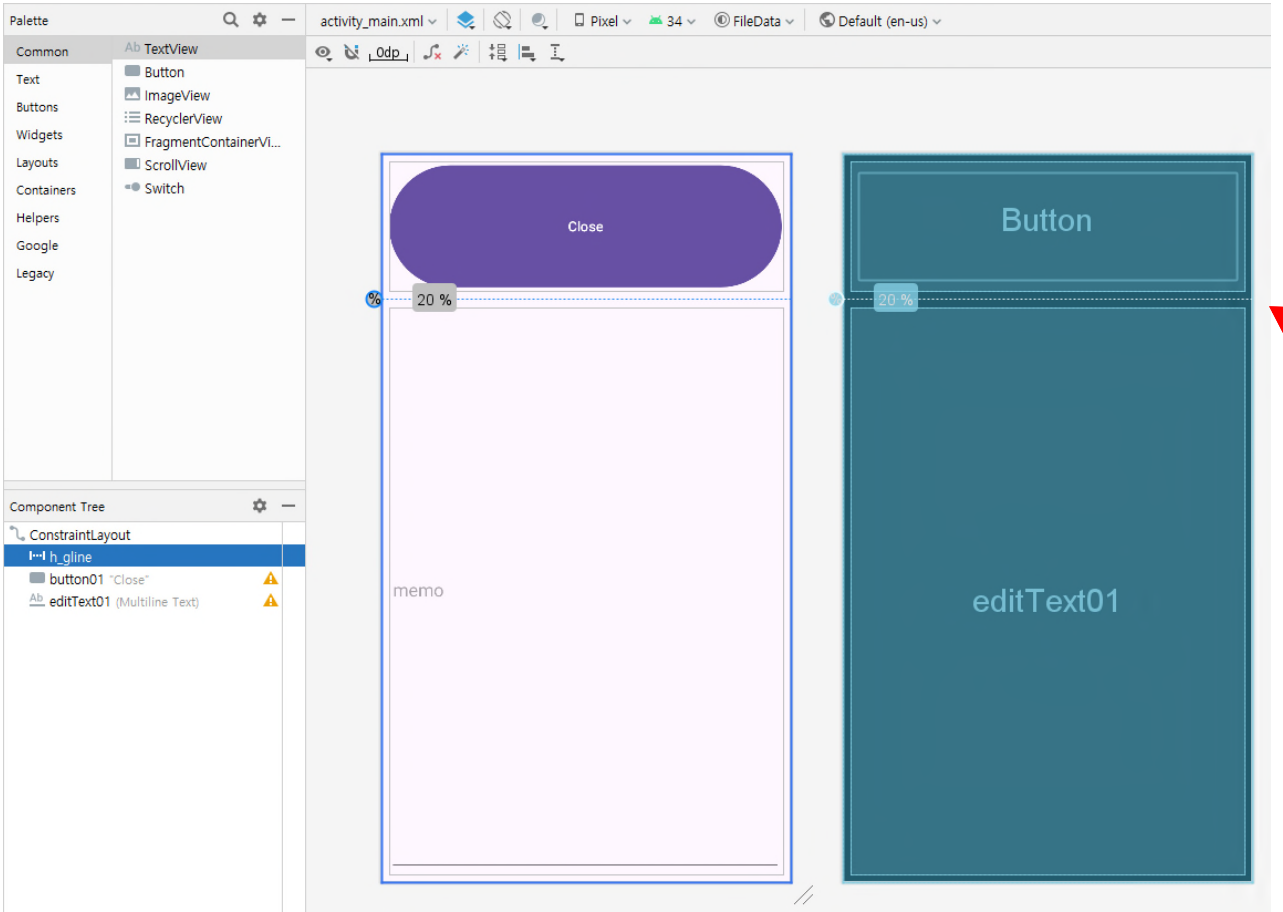
- 처음실행 시, 또는 저장된 내용이 없으면 보여주어야 할 원본 파일을 추가

- 패키지에 포함된 res/raw/파일명
  - 수정 불가
  - 읽기 전용



❑ 레이아웃 작성

- 메모 기능을 할 수 있도록 레이아웃 설정
- 자동 저장되는 기능을 확인하기 위함이라 편집할 수 있는 텍스트 (Plain Text – EditText 객체)와 종료시 사용할 버튼으로 구성
- ConstraintLayout 구성 과정에서 필요한 비율 지정 방법 확인

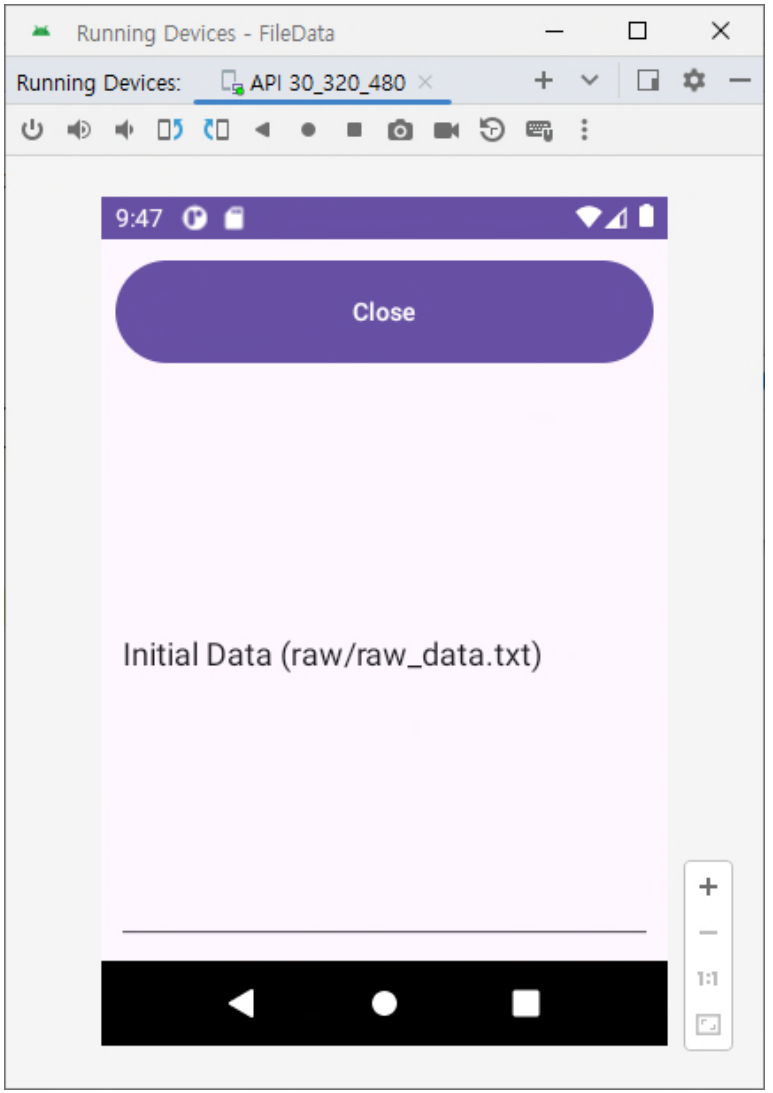


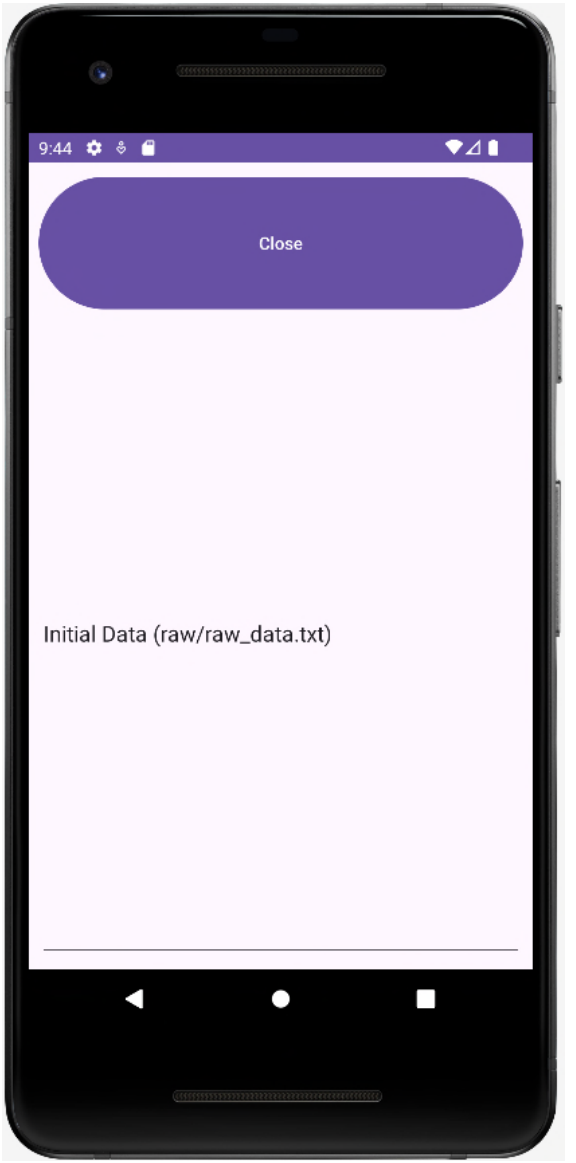
Button

Guideline (20%)

EditText

❍Guideline 사용시





❑ 레이아웃 작성

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <androidx.constraintlayout.widget.Guideline
        android:id="@+id/h_gline"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        app:layout_constraintGuide_percent="0.20" />
```

기준 위치 설정, 비율 지정 등이 필요할 때 Guideline 사용

필수 속성은 좌측 코드 참조

android:orientation =
“horizontal”
→ 가로 기준선을 생성
“vertical”
→ 세로 기준선을 생성

비율로 지정하는 경우 : layout_constraintGuide_percent
시작점을 기준으로 거리를 지정하는 경우 : layout_constraintGuide_begin
끝점을 기준으로 거리를 지정하는 경우 : layout_constraintGuide_end

❑ 레이아웃 작성

```
<Button
    android:id="@+id/button01"
    android:layout_width="0dp"
    android:layout_height="0dp"
    android:layout_marginStart="8dp"
    android:layout_marginTop="8dp"
    android:layout_marginEnd="8dp"
    android:layout_marginBottom="8dp"
    android:text="Close"
    app:layout_constraintBottom_toTopOf="@+id/h_gline"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
```

❑ 레이아웃 작성

```
    <EditText
        android:id="@+id/editText01"
        android:layout_width="0dp"
        android:layout_height="0dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:layout_marginEnd="8dp"
        android:layout_marginBottom="8dp"
        android:ems="10"
        android:hint="memo"
        android:inputType="textMultiLine"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="@+id/h_gline" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

❑ 코드 분석 / 설명

```
package com.practice.ex.filedata;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;

import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.io.OutputStreamWriter;
import java.io.PrintWriter;

public class MainActivity extends AppCompatActivity implements View.OnClickListener {

    3 usages
    private static final String LOCAL_FILE = "memo_data.txt";
    4 usages
    EditText edit01;
    2 usages
    Button btn01;
```

❑ 코드 분석 / 설명

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    edit01 = (EditText) findViewById(R.id.editText01);

    btn01 = (Button) findViewById(R.id.button01);
    btn01.setOnClickListener(this);

    InputStream in;
    BufferedReader reader;
```

- InputStream
  - 입력 스트림 클래스들의 최상위 클래스
- BufferedReader
  - Reader 스트림에 버퍼링 기능을 추가한 입력 스트림 클래스

❑ 코드 분석 / 설명

```
        try {
            in = openFileInput(LOCAL_FILE);
        } catch (FileNotFoundException e) {
            in = getResources().openRawResource(R.raw.raw_data);
        }

        try {
            reader = new BufferedReader(new InputStreamReader(in,  charsetName: "UTF-8"));
            String s;
            while ( (s = reader.readLine()) != null) {
                edit01.append(s);
                edit01.append("\n");
            }
        } catch (IOException e) {
            Log.e( tag: "Local File", e.getMessage());
        }
    }
```

readLine() 메소드를
이용하여
한줄씩 읽어서 처리

- OutputStream
  - 데이터스트림의 기본 출력 담당
  - 모든 출력 스트림 클래스들의 최상위 클래스
- openFileInput, openFileOutput
  - InputStream과 OutputStream의 인스턴스를 받아오는 부분

❑ 코드 분석 / 설명

```
@Override
protected void onPause() {
    super.onPause();

    String s = edit01.getText().toString();
    if(s.length() == 0) {
        deleteFile(LOCAL_FILE);
        return;
    }
    try {
        OutputStream out = openFileOutput(LOCAL_FILE, MODE_PRIVATE);
        PrintWriter writer = new PrintWriter(new OutputStreamWriter(out, charsetName: "UTF-8"));
        writer.append(s);
        writer.close();
    } catch (IOException e) {
        Log.e( tag: "Local File", e.getMessage());
    }
}

@Override
public void onClick(View v) {
    finish();
}
}
```

❍ onPause()

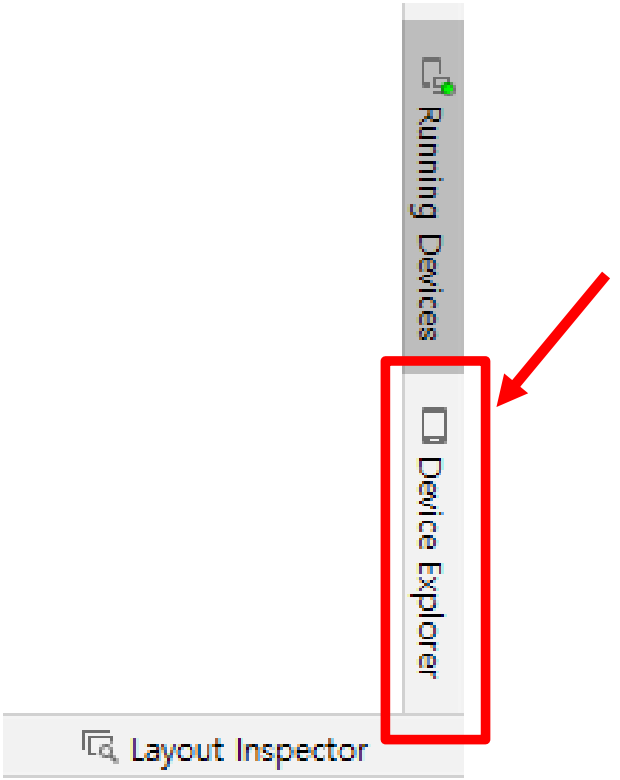
- ❑ 절전 상태 혹은새로운 액티비티가 시작될 경우 (액티비티 생명주기 참조)
- ❑ 초점 상실
- ❑ 재개(resume)되기 전 데이터 저장, 애니메이션 중지, 프로세서를 소비하는 작업 중단

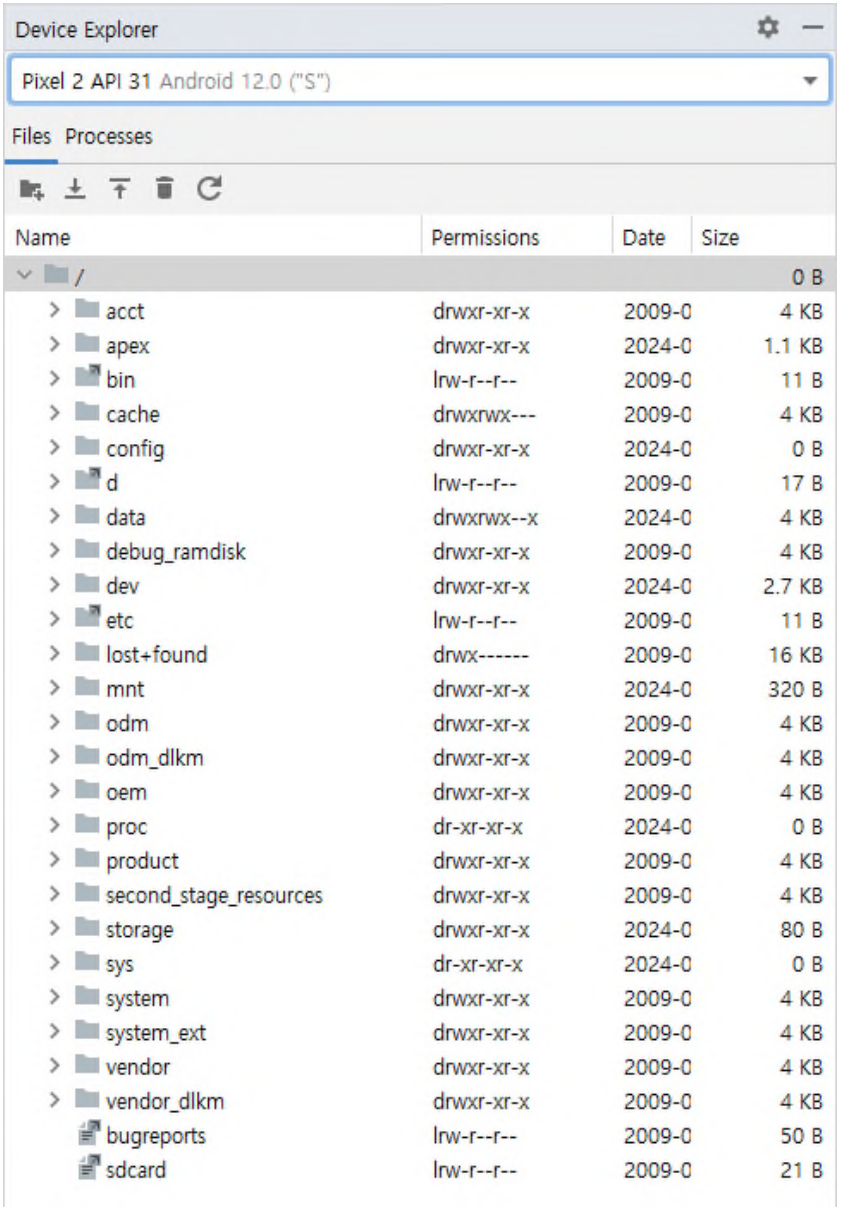
문자단위로 특정한 스트림에 작성

MODE_PRIVATE 다른어플에서 접근 불가

❑ 저장된 파일 확인

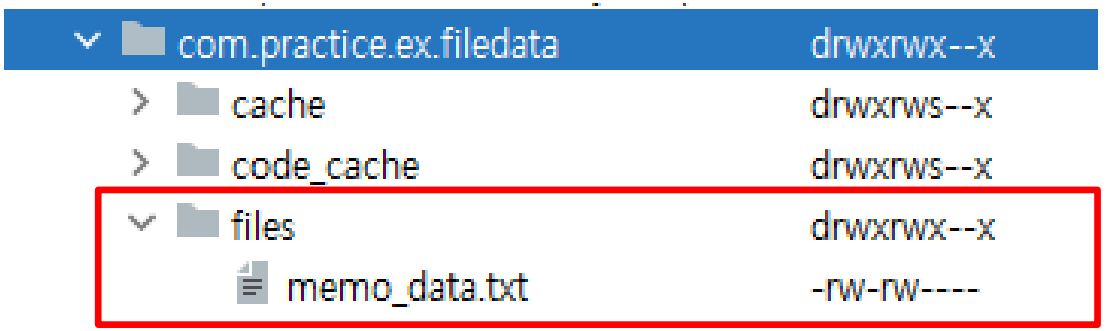
❍ 에뮬레이터 실행 후에 Studio 하단 우측 Device File Explorer 선택 사용





❑ 저장된 파일 확인

- 해당 프로젝트가 수행된 에뮬레이터가 동작 중이어야 함
  - Device File Explorer → data → data → 해당 패키지 경로 → files → 저장파일명 확인
  - 저장된 파일 위치 확인 (해당 프로젝트 패키지 경로)



- 저장된 파일 확인
  - 우클릭 후 해당 메뉴를 통해 업로드, 다운로드, 새로 생성 등이 가능