FRAGMENT

fragment이란 무엇인가

여러개의 fragment를 결합해서 하나의 액티비티에 보이게 할 수도 있으며 하나의 fragment를 여러 액티비티에서 재사용할 수 있다.

fragment란, 어플리케이션에서 화면에 직접 보이는 공간인 Activity내에서 분할시키고 다른 화면으로 전환할 수 있는 화면 공간의 단위입니다.

Fragment를 이용하려면, 그 상위에 있는 Activity에서 출력할 layout을 제어해줘야 합니다.

fragment와 연결

▹xml파일과 fragment 클래스 연결
import androidx.fragment.app.Fragment
Fragment1 extends Fragment
onCreateView
return inflater.inflate(R.layout.fragment, container, false)

▹fragment.xml의 widget과 fragment.java과 연결
ViewGroup rootView = (ViewGroup) inflater.inflate(R.layout.fragment_main, container, false);
Button button = rootView.findViewById(R.id.button);

▹activity와 fragment연결
You can add your fragment to the activity's view hierarchy either by defining the fragment in your activity's layout file or by defining a fragment container in your activity's layout file and then programmatically adding the fragment from within your activity.
In either case, you need to add a FragmentContainerView that defines the location where the fragment should be placed within the activity's view hierarchy. It is strongly recommended to always use a FragmentContainerView as the container for fragments, as FragmentContainerView includes fixes specific to fragments that other view groups such as FrameLayout do not provide.

두가지 방법이 있다. activity의 xml파일에 fragmentview를 할당해주거나 fragment를 담을 container을 담을 공간을 만들어 준 다음 activity 클래스에서 연결해줄 수 있다.
두 상황 모두에서 fragmentcontainerview를 사용하자

fragmentcontainerview 초기화 방법

1. fragmentcontainerview에는 android:name="com.example.ExampleFragmen"를 넣어서 초기화 할 수 있다. The android:name attribute specifies the class name of the Fragment to instantiate. When the activity's layout is inflated, the specified fragment is instantiated,
2. FragmentTransaction을 이용 할 수도 있다.
   fragmentManager.beginTransaction().replace(R.id.container, mainfragment).commit();

fragment의 전환

▹activity 밖에서 fragment를 바꾸고 싶을 때
fragment의 전환은 activity 그리고 그 안에 있는 fragmentmanager가 관장한다. 따라서 activity에 전환 메소드를 만들고 fragment의 버튼에서는 activity.메소드로 그 메소드를 불러와야 한다.

1. activity에 화면 전환 메서드 만들기 fragment manager는 getSupportFragmentManager를 이용해서 초기화해야 한다.

```
public void onFragmentChanged(int Index) {
    switch (Index) {
      case 0:
        //main
        fragmentManager.beginTransaction()
            .replace(R.id.container_frame,mainFragment).commit();
        break;
      case 1:
        //preview
        fragmentManager.beginTransaction()
            .replace(R.id.container_frame, previewFragment).commit();
        break;
      case 2:
        //time set
        fragmentManager.beginTransaction()
            .replace(R.id.container_frame, timeLockFragment).commit();
        break;
      case 3:
        // app setting
        fragmentManager.beginTransaction()
            .replace(R.id.container_frame, allowAppSettingFragment).commit();
        break;
    }
  }
```

2. activity 객체를 만들어서 getActivity로 container을 포함하고 있는 activity를 가져오고 activity안에 있는 화면 전화 메서드를 불러와서 전환한다.

```
MainActivity activity = (MainActivity) getActivity();
activity.onFragmentChanged(0);
```

activity에서 fragment 안에 있는 view, widget 제어하기

https://kitesoft.tistory.com/82

activity에서 fragment view를 바꾸려면 두 가지 방법이 있다

1. fragment안의 view를 직접 가져오는 방법이 있고
2. fragment ㅋ클래스 내에 view를 바꾸는 메서드를 만들고 그 메서드를 가져오는 것이다.

두번째 방법이 옳다. fragment의 동작은 fragment내에 만들어 줘야 깔끔하다. 두번째 방법만 정리하겠다.

1. fragment안에 view를 바꾸는 메서드를 만든다

```
public void setTextView(String str){
        positive_text.setText(str);
   }
```

2. fragment 객체를 가져와서 fragment 내의 메서드를 가져온다

```
mainFragment = new MainFragment();
mainFragment.setTextView();
```

Interface를 이용해 fragment의 정보 교환

▹정보를 주는 fragment에서 listener interface를 만든다

```
public static interface DialogListener {
    public void textChange(String str);
}
```

▹activity가 fragment A 안의 interface를 implement하게 하고 interface의 메서드를 activity에서 구현한다. 그 메서드 안에 activity에서 할 동작을 적는다

```
Mainactivity extends AppCompatActivity implements
CustomDialog.DialogListener
```

```
@Override
    public void textChange(String str) {
        //이 안에 activity에서 작동하는 코드를 입력한다
    }
```

▹activity와 fragment가 연결되는 순간 fragment의 onAttach에서 activity context안에 담긴 listener을 가져온다. 그 안에 activity에서 만든 코드가 포함되어 들어온다
try catch문을 이용하면 좋다. 이러면 implement되지 않았으면 context안에 있는 callback을 가져오지 못하고 넘어간다
//val
DialogListener listener;

```
@Override
    public void onAttach(@NonNull Context context) {
        super.onAttach(context);
       // ctrl+alt+t
        try {
            listener = (DialogListener) context;
        } catch (ClassCastException e) {
            throw new ClassCastException(context.toString() +
                    "must implement DialogListener")
        }
    }
```

▹onAttach를 통해 가져온 listener 객체로 activity의 메서드를 참조하고 그 메서드에 값을 넘겨줌으로서 정보를 전단한다

```
@Override
public void onClick(DialogInterface dialogInterface, int i) {
    str_positive = positiveText.getText().toString();
    listener.textChange(str_positive);
}
```

?VIEWMODEL

https://developer.android.com/guide/fragments/communicate

▹To properly react to user events, or to share state information, you often need to have channels of communication between an activity and its fragments or between two or more fragments. To keep fragments self-contained, you should not have fragments communicate directly with other fragments or with its host activity.

쉽게 설명을 하면 두 fragment는 직접적으로 통신을 하면 안된다. 그래서 보통은 communicate 전용 interface를 만든 다음 이것을 activity에 implement하고 그 interface를 통해 소통한다.

▹fragment 사이나 fragment와 activity 사이 user event, state information 공유 방법은 두가지가 있다.

1. ViewModel: 더 광범위한 상황에서 지속해야하는 data를 주고 받을 때
2. Fragment Result API: 일회성으로 넘겨주고 쓷나는 data이고 Bundle에 들어갈 크기면

▸ViewModel

▹viewmodel을 사용해야 할 때

1. activity 안에 두개의 fragment가 서로 소통을 해야 할 때
2. fragment 안에 또 다른 fragment가 있고 서로 소통을 할 때: fragment에서 dialog나 picker을 띄워도 이들은 activity 위에서 생성되는 것이다.
3. navigation view와 fragment와 소통을 할 때

☆ViewModel 이용하기

1. viewmodel에 필요한 dependencies을 추가하고
2. viewmodel java class를 만들고 viewmodel을 상속한다
3. private MutableLiveData<원하는 type> text = new MutableLiveData<>(); mutable을 사용해야만 setText를 사용해서 추가할 수 있다.
4. data를 더라하고 빼는 메서드를 추가한다.
5. fragmentA에 들어가서 onActivityCreated override해서 activity와 observer설정한다

◇ Fragment에서 Toast 사용하기

https://horae.tistory.com/entry/%EC%95%88%EB%93%9C%EB%A1%9C%EC%9D%B4%EB%93%9C-Fragment-%EC%97%90%EC%84%9C-Toast-%EC%82%AC%EC%9A%A9%ED%95%98%EA%B8%B0

toast를 사용하려면 toast가 표시되는 context가 필요하다.

1. Context 객체 만들기
2. fragment를 감싸고 있는 context 가져오기
3. makeText(context, “string”, length)

◇ FragmentManager

https://developer.android.com/guide/fragments/fragmentmanager

▹ FragmentManager is the class responsible for performing actions on your app's fragments, such as adding, removing, or replacing them, and adding them to the back stack.

▹ Every FragmentActivity and subclasses thereof, such as AppCompatActivity, have access to the FragmentManager through the getSupportFragmentManager() method.

▹Fragment는 다른 Fragment를 안에 품을 수 있으며 getChildFragmentManager(), getParentFragmentManager() 메서드를 통해 각각의 fragmentmanager을 참조할 수 있다.

◇ Fragment Transactions

https://developer.android.com/guide/fragments/transactions

fragmentmanager는 fragment의 추가, 제서, 교체를 관장하는데 fragmenttransaction class를 통해 fragment 변경이 가능하다. 이때 manager가 관리하는 back stack에 transaction 과정을 저장해서 뒤로 돌아갈 수 있다.

FragmentManager fragmentManager = getFragmentManager();
FragmentTransaction fragmentTransaction = fragmentManager.beginTransaction();

▹ .beginTransaction()
▹.setReorderingAllowed()
▹.add()
▹.remove()
▹.replace()
▹.commit()

▹ .show() .hide()
▹.detach() .attach()

◇ Fragment LifeCycle

▹By building Fragment on top of Lifecycle, you can use the techniques and classes available for Handling Lifecycles with Lifecycle-Aware Components.

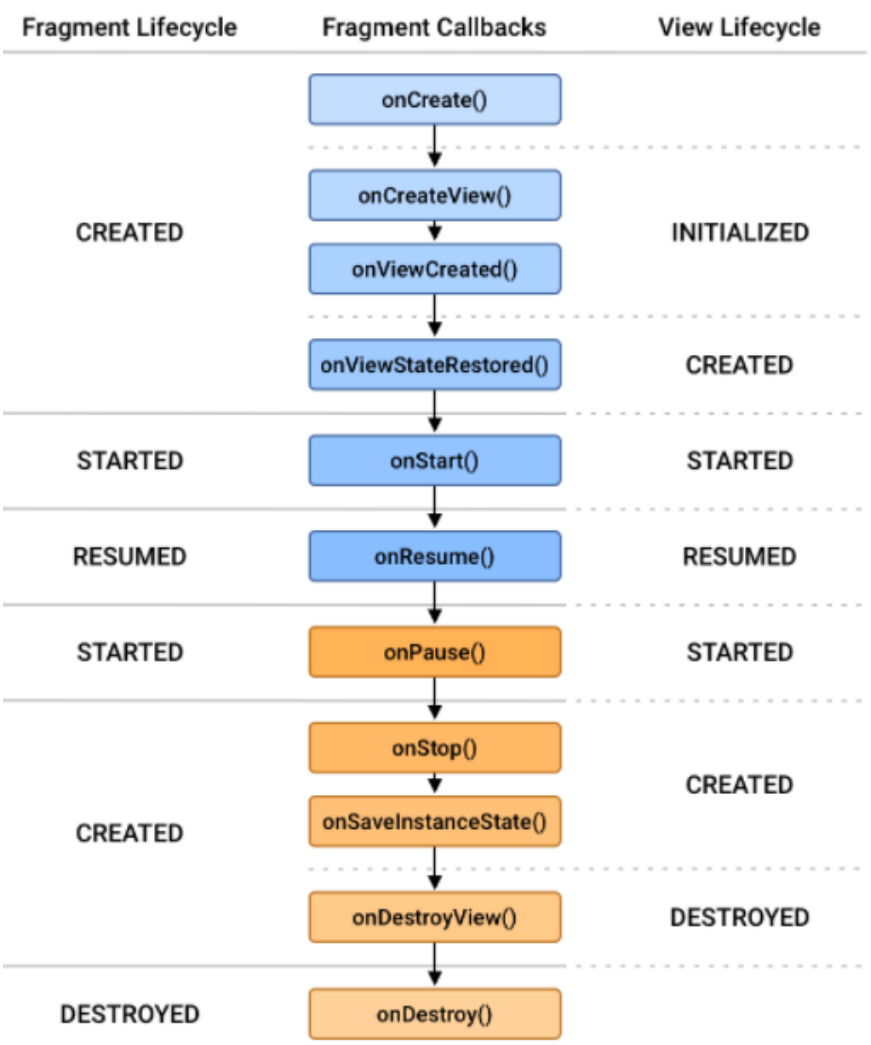
This component could automatically start listening when the fragment becomes active and stop when the fragment moves to an inactive state.

▹ fragment의 lifecycle 확인 방법

1. LifeCycleOwner을 implement해서 getlifecycle() method를 사용한다. getlifecycle() method는 Lifecycle객체를 return하는데 이를 통해 현재 어떤 state에 있는지 확인 가능하다
2. Fragment class에는 callback method가 있는데 callback method는 fragment의 lifecycle와 연동되어 해당 state에 도달하면 자동으로 시작된다.

▹ fragment View의 lifecycle

fragment의 view는 fragment와는 독립적인 lifecycle을 가진다.



▹ INITIALIZED
When a fragment is instantiated, it begins in the INITIALIZED state. For a fragment to transition through the rest of its lifecycle, it must be added to a FragmentManager. The FragmentManager is responsible for determining what state its fragment should be in and then moving them into that state.

- onAttach()
  The onAttach() callback is invoked when the fragment has been added to a FragmentManager and is attached to its host activity. At this point, the fragment is active, and the FragmentManager is managing its lifecycle state. At this point, FragmentManager methods such as findFragmentById() return this fragment.

▹CREATED
When your fragment reaches the CREATED state, it has been added to a FragmentManager and the onAttach() method has already been called.

- onCreate()
  created transition invokes the onCreate() callback. The callback also receives a savedInstanceState Bundle argument containing any state previously saved by onSaveInstanceState()

- onCreateView()
  The fragment's view Lifecycle is created only when your Fragment provides a valid View instance. In most cases, you can use the fragment constructors that take a @LayoutId, which automatically inflates the view at the appropriate time. You can also override onCreateView() to programmatically inflate or create your fragment's view.

- onViewCreate()
  If and only if your fragment's view is instantiated with a non-null View, that View is set on the fragment and can be retrieved using getView(). The getViewLifecycleOwnerLiveData() is then updated with the newly INITIALIZED LifecycleOwner corresponding with the fragment's view. The onViewCreated() lifecycle callback is also called at this time.

▷STARTED
▷RESUMED
the fragment is ready for user interaction and onResumed() callback is invoked.
Fragments that are not RESUMED should not manually set focus on their views or attempt to handle input method visibility.

▷DOWNWARD STARTED
- onPause()

JAVA 문법

Super와 Super()

http://www.tcpschool.com/java/java_inheritance_super

1. super 키워드
   parent.class를 child.class가 상속받았을 때 child안에서 super.을 이용해 parent안에 있는 메소드나 변수를 참조해서 조작할 수 있다.
2. super() 메소드
   parent.class를 child.class가 상속받았을 때 child안에서 super()을 이용해 parent가 필요로 하는 생성자와 함께 parent.class를 불러와 부모의 생성자를 호출한다.

```
// ex.3)
class A{
    public void doA() {
        System.out.println("A 클래스의 doA() 입니다.");
    }
}
class B extends A{
    B(){
        // A 클래스의 doA() 함수를 호출합니다.
        super.doA();
    }
}

public class Ex01 {
    public static void main(String[] args) {
        new B();
        // 결과 : A 클래스의 doA() 입니다.
    }
}
```

```java
// ex.4)
class AAA{
    AAA(){
        System.out.println("AAA 생성자");
    }
    AAA(int a){
        System.out.println(a);
    }
}
class BBB extends AAA{
    BBB(){
        super();
        System.out.println("BBB 생성자");
    }
    BBB(int a){
        super(a);
        System.out.println("a = " + a);
    }
}
public class Ex01 {
    public static void main(String[] args) {
        // BBB 클래스의 BBB(int a) 를 호출하면
        // 부모 클래스 AAA 클래스의 AAA(int a) 를 호출합니다.

        /* 결과:
         * 20
         * a = 20
         *
         */
        new BBB(20);
    }
}
```

상속과 생성자 그리고 Interface

1. 상속은 부모 객체가 가지고 있는 특성을 모두 가지는 새로운 객체를 만드는 것이다.

- 부모 객체가 가지고 있던 변수, 메서드를 자식은 바로 사용할 수 있다.
  String name;
  void sleep( ) { System.out.println(this.name + " zzz"};
- overriding을 이용해서 부모 객체 안의 메서드를 고쳐서 사용할 수 있다.
  void sleep( ) { System.out.println(this.name + " abc"};
- overloading을 이용해서 부모 객체 안의 메서드를 추가해서 사용할 수 있다.
  overloading은 같은 이름의 메서드가 여러 형태의 자료형을 받고 그에 따라 다른 행동을 함을 의미한다.
  void sleep(int num) { System.out.println(this.name + num +" zzz"};

2. 생성자는 상속을 할 때 ~~반드시 추가해야 하는 생성에 필요한 argument~~를 의미한다.(x)
   생성자는 상속을 할 때 반드시 실시되는 명령을 의미한다. 이때 argument는 필요할 수도 있고 필요하지 않을 수도 있다. 생성자는 argument를 나타내는 말이 아니고 그 안에 있는 명령을 의미한다.

   ```
   class HouseDog extends Dog {
       HouseDog(String name) { this.setName(name);}}
   ```

   여기에서 this.setName(name)을 의미한다.

3. 인터페이스는 새로운 분류기준을 만들어서 객체에 추가함으로써 부모 클래스와는 다른 기준으로 객체를 묶고 새로운 변수, 베서드를 추가함을 의미한다.

   ```
   interface Predator { String getFood}

   class Tigar extends Animal implements Predator {
       public String getFood() { return "apple"}}
   ```

java의 List 자료형

list는 array와 비슷하나 크기가 정해져 있지 않고 저장되는 정보의 양에 따라 동적으로 변한다는 특징이 있다.

☆ Generics
<>안에는 어떤 객체를 list안에 저장할 것인지 명시하는 것이 generics이다. 이때 기본 자료형 뿐만 아니라 내가 만든 class 예를 들면 <Dog> <Animal>을 넣어서 그 객체를 저장할 수 있다.

ArrayList<String> pitches = new ArrayList<String>();
보통 뒷 부분의 자료형은 생략 가능하다.
ArrayList<String> pitches = new ArrayList<>();

▹ list 자료형은 interface여서 이 interface를 구현한 하위 자료형이 있고 이들을 쓴다.

1. arrayList 자료형
2. Vector 자료형
3. LinkedList 자료형

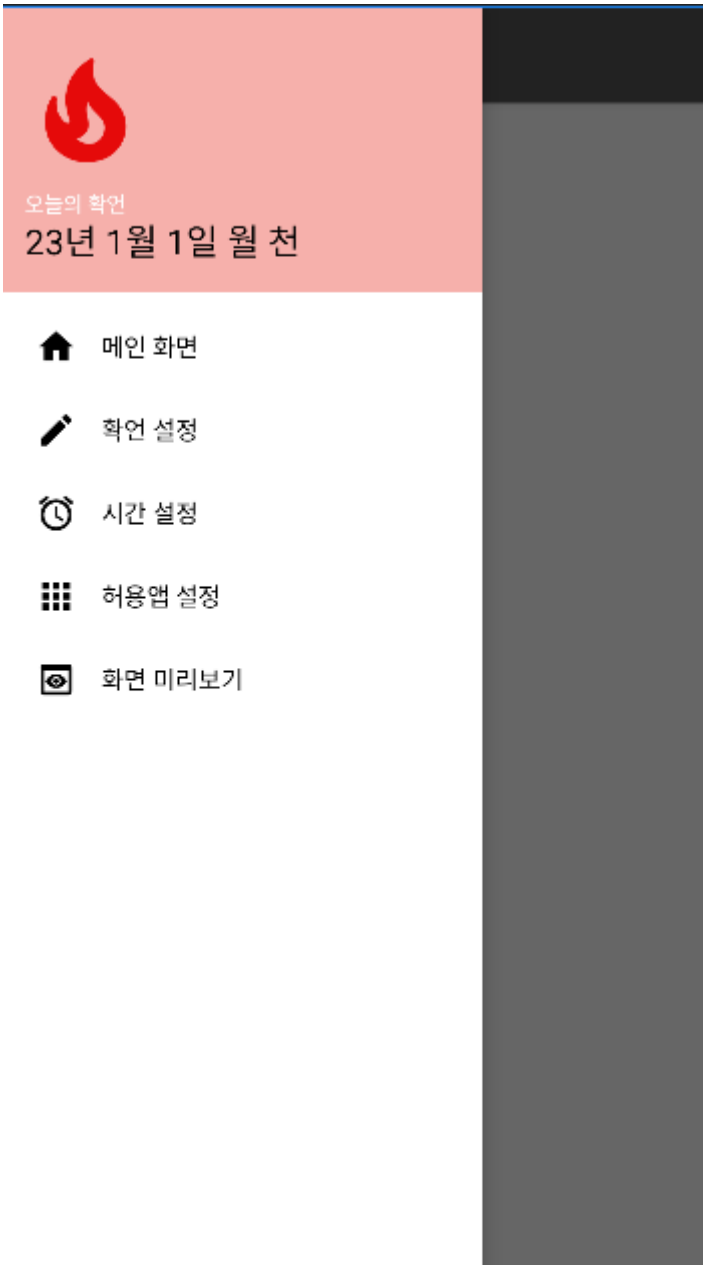
For Each문

array와 list에서만 사용할 수 있는 for문으로 자료형 안에 있는 모든 객체에 대해 for문 안에 있는 반복문을 반복한다.

```
String[] numbers = {"one", "two", "three"};
for(int i=0; i<numbers.length; i++) {
    System.out.println(numbers[i]);
}
```

```
String[] numbers = {"one", "two", "three"};
for(String number: numbers) {
    System.out.println(number);
}
```

NAVIGATION VIEW



```
<?xml version="1.0" encoding="utf-8"?>
<androidx.drawerlayout.widget.DrawerLayout 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:id="@+id/drawer_layout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity"
    android:fitsSystemWindows="true"
    tools:openDrawer="start">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical">

        <androidx.appcompat.widget.Toolbar
            android:id="@+id/toolbar"
            android:layout_width="match_parent"
            android:layout_height="?attr/actionBarSize"
            android:background="#555555"
            app:title=" "
            android:elevation="4dp"
            android:theme="@style/ThemeOverlay.AppCompat.Dark.ActionBar"
            app:popupTheme="@style/ThemeOverlay.AppCompat.Light" />

        <androidx.fragment.app.FragmentContainerView
            android:id="@+id/container_frame"
            android:name="org.texchtown.fullconcentration3.MainFragment"
            android:layout_width="match_parent"
            android:layout_height="match_parent" />

    </LinearLayout>


    <com.google.android.material.navigation.NavigationView
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:layout_gravity="start"
        android:id="@+id/nav_View"
        app:headerLayout="@layout/nav_header"
        app:menu="@menu/nav_menu">

    </com.google.android.material.navigation.NavigationView>

</androidx.drawerlayout.widget.DrawerLayout>
```

APPBAR

▹ 앱바 툴바 액션바
https://developer.android.com/training/appbar/setting-up

appbar를 만드는 방법은 actionbar와 toolbar가 있는데 activity를 만들면 기본으로 actionbar가 붙는다.

하지만 appbar를 만들 때 꼭 toolbar을 이용해라. 여러 device에서 문제 없이 구현되고 더 많은 기능을 포함한다.

▹ Toolbar로 appbar을 구현하기 위한 준비과정

1. v7 appcompat support library를 project에 추가한다.
2. activity가 appcompatactivity를 상속해야 한다.
3. app manifest에서 <application> 부분을 NoActionbar theme으로 바꿔야 기본 actionbar를 사용한 appbar가 없어진다.

▹ Toolbar 위젯을 activity에 추가하고 android:elevation 값을 주어 화면 위에 떠 보이게 한다.

```
<androidx.appcompat.widget.Toolbar
android:id="@+id/toolbar"
android:layout_width="match_parent"
android:layout_height="?attr/actionBarSize"
android:background="#555555"
app:title=" "
android:elevation="4dp"
android:theme="@style/ThemeOverlay.AppCompat.Dark.ActionBar"
app:popupTheme="@style/ThemeOverlay.AppCompat.Light" />
```

▹ Toolbar를 앱의 appbar로 사용되게 만들기: setSupportActionBar()메소드의 인자로 toolbar 객체를 넣는다. 반드시 이과정을 거쳐야 toolbar가 actionbar로 구동을 한다.

```
//set toolbar as actionbar
toolbar = findViewById(R.id.toolbar);
setSupportActionBar(toolbar);
actionBar = getSupportActionBar();
```

▹액션바에 menu 추가하기 (버튼 추가하기)
menu > toolbar_menu 추가하고 그 안에 들어갈 item 추가한다

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:app="http://schemas.android.com/apk/res-auto">
   <item
      android:id="@+id/menu_start_fire"
      android:icon="@drawable/ic_baseline_local_fire_department_24"
      android:title="@string/menu_btn_fire"
      app:showAsAction="always"/>
</menu>
```

activity에서 onCreateOptionMenu를 오버라이드해서 toolbar의 menu를 활성화한다

```
@Override
  public boolean onCreateOptionsMenu(Menu menu) {
     MenuInflater menuInflater = getMenuInflater();
     menuInflater.inflate(R.menu.toolbar_menu, menu);
     return true;
  }
```

☆메뉴를 선택했을 때 처리하는 방법
사용자가 하나의 menu item을 선택했을 때 자동 호출되는 onOptionsItemSelected을 override해서 item의 id값을 입력하고 원하는 작업을 추가하면 된다.

```
@Override
public boolean onOptionsItemSelected(@NonNull MenuItem item) {

  int curId = item.getItemId();
  switch (curId) {
     case R.id.menu_refresh:
        Toast.makeText(this, "새로고침 메뉴가 선택되었습니다.",
Toast.LENGTH_LONG).show();
        break;
,,,
     default:
        break;   }
 return super.onOptionsItemSelected(item); }
```

Navigation

https://developer.android.com/guide/navigation

navigation이란 앱 내의 다양한 content 사이를 돌아다니고, 들어갔다가 나갔다 하는 모든 상호과정을 의미한다.
구현 방법은 두가지이다

1. drawerlayout과 navigationview를 이용해 만든다
2. Android JetPack 안의 Navigation component를 이용해 만든다

▹Navigation state is represented as a stack of destinations
앱이 시작했을 때부터 여러 activity와 fragment들을 backstack에 쌓이고 빠지는 과정을 거친다. navigation component의 필요성은 이런 모든 activity와 fragment가 쌓여 있는 backstack을 미리 잘 정리해서 사용자에게 보여줌으로써 편리성을 증가시키는 것이다.
그냥 버튼을 사용해서 fragment들을 교체해주는 것과 navigation component들을 사용하는 것의 차이는 navigation은 복잡한 화면 교체를 하나로 묶어 stack에 넣어서 관리함으로써 back버튼이나 다른 navigation item을 눌렀을 때 많은 변화가 한번에 일어나는 것이다.

▹ Deep Link
deep link는 앱에 있는 특정 부분으로 바로 가는 link를 제공한다. 이때 그 fragment이나 activity로만 가는 것이 아니고 그 목적지에 도달하기 위해 거쳐야 하는 activity와 fragment들을 backstack에 순서에 맞게 저장해서 그 backstack 자체를 가져오는 것이다.

NAVIGATION with navigationView

https://www.youtube.com/watch?v=zYVEMCiDcmY

▹ xml만들기

1. menu에 xml을 만들어 navigationDrawer에 들어갈 목록을 만든다.

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    tools:showIn="navigation_view">

    <group>
        <item
            android:id="@+id/menu_main"
            android:icon="@drawable/ic_baseline_home_24"
            android:title="@string/menu_home"/>
        <item
```

```
            android:id="@+id/menu_textedit"
            android:icon="@drawable/ic_baseline_edit_24"
            android:title="@string/menu_textedit"/>
        <item
            android:id="@+id/menu_alarm"
            android:icon="@drawable/ic_baseline_alarm_24"
            android:title="@string/menu_alarm"/>
        <item
            android:id="@+id/menu_app"
            android:icon="@drawable/ic_baseline_apps_24"
            android:title="@string/menu_app_allowed"/>
        <item
            android:id="@+id/menu_preview"
            android:icon="@drawable/ic_baseline_preview_24"
            android:title="@string/menu_preview"/>
    </group>

</menu>
```

2. activity.xml 파일에 root layout을 drawerlayout으로 설정하고 toolbar와 naviationView를 추가한다

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.drawerlayout.widget.DrawerLayout
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:id="@+id/drawer_layout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity"
    android:fitsSystemWindows="true"
    tools:openDrawer="start">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical">

        <androidx.appcompat.widget.Toolbar
            android:id="@+id/toolbar"
            android:layout_width="match_parent"
            android:layout_height="?attr/actionBarSize"
            android:background="#555555"
            app:title=" "
            android:elevation="4dp"
            android:theme="@style/ThemeOverlay.AppCompat.Dark.ActionBar"
            app:popupTheme="@style/ThemeOverlay.AppCompat.Light" />
```

```
    <androidx.fragment.app.FragmentContainerView
        android:id="@+id/container_frame"
        android:name="org.texchtown.fullconcentration3.MainFragment"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />

</LinearLayout>


<com.google.android.material.navigation.NavigationView
    android:layout_width="wrap_content"
    android:layout_height="match_parent"
    android:layout_gravity="start"
    android:id="@+id/nav_View"
    app:headerLayout="@layout/nav_header"
    app:menu="@menu/nav_menu">

</com.google.android.material.navigation.NavigationView>

</androidx.drawerlayout.widget.DrawerLayout>
```

▷ activity.java

1. NavigationView.OnNavigationItemSelectedListener 인터페이스를 implement하고 onNavigationItemSelected 메서드를 오버라이드해서 목록을 선택할 때 어떤 작업을 할지를 그 안에 구현한다

```
public class MainActivity extends AppCompatActivity implements
NavigationView.OnNavigationItemSelectedListener

@Override
    public boolean onNavigationItemSelected(@NonNull MenuItem item) {

        switch (item.getItemId()) {
            case R.id.menu_main:
                //main
                onFragmentChanged(0);
                break;
            case R.id.menu_textedit:
                //main
                onFragmentChanged(0);
                //show dialog to edit and pass text to main fragment
                openDialog();
                break;
            case R.id.menu_alarm:
                //time picker
                onFragmentChanged(2);
```

```
            break;
        case R.id.menu_app:
            // ?? new fragment for setting apps
            onFragmentChanged(3);
            break;
        case R.id.menu_preview:
            //preview fragment
            onFragmentChanged(1);
            break;
    }
    drawerLayout.closeDrawer(GravityCompat.START);
    return true;
}
```

2. onBackPress메서드를 오버라이드해서 뒤로가기 누르면 navigationDrawer가 들어가게 만든다

```
@Override
    public void onBackPressed() {
        if(drawerLayout.isDrawerOpen(GravityCompat.START)) {
            drawerLayout.closeDrawer(GravityCompat.START);
        } else {
            super.onBackPressed();
        }
    }
```

3. drawerLayout 객체를 만들어 drawer_layout view를 연결해서 toolbar에 navigation icon이 뜨게 만든다

```
drawerLayout = findViewById(R.id.drawer_layout);
        ActionBarDrawerToggle toggle = new ActionBarDrawerToggle(this,
drawerLayout, toolbar,
                R.string.navigation_drawer_open, R.string.navigation_drawer_close);
        drawerLayout.addDrawerListener(toggle);
        toggle.syncState();
```

4. ActionBarDrawerToggle을 사용하면 actionBar에 햄버거 아이콘이 생겨서 navigation drawer을 열었다가 닫았다 할 수 있다.
   ActionBarDrawerToggle 객체를 만들 때 시각장애인을 위한 string 값을 두개 인수로 전달해야 한다.

   res>string.xml에 string 두개를 추가한다.

```
<string name="navigation_drawer_open">Open navigation drawer</string>
<string name="navigation_drawer_close">Close navigation drawer</string>
```

5. navigationview안에 있는 item이 클릭되었을 때 작동할 listener을 연결하고 override한다.

```
NavigationView navigationView = findViewById(R.id.nav_view);
```

navigationView.setNavigationItemSelectedListener(this);

6. navigationView.setNavigationItemSelectedListener(this) 은 callback 어쩌구 때문에 this 대신 new OnNavigationItemSelectedListener을 써도 되지만
   코드 좋아보이게 this 쓰고 activiity에 implements NavigationView.OnNavigationItemSelectedListener하고 onNavigationitemSelected을 override한다.

7. 처음 앱이 켜질 때 activity에 보이는 fragment와 menu가 같은지 확인해준다

```
if(savedInstanceState == null) {
//        //initialize mainFragment to container
//        fragmentManager.beginTransaction()
//                .replace(R.id.container_frame, mainFragment).commit();
          //initialize home item in menu
          navigationView.setCheckedItem(R.id.menu_main);
      }
```

DIALOG / PICKER

Dialog

https://developer.android.com/guide/topics/ui/dialogs

▹Dialog
dialog는 화면 전체를 차지 하지 않는 view로 modal event를 위해 쓰인다
modal event는 앱의 진행이 멈추고 특정 input을 받은 후 앱의 진행이 이어지는 것이다

▹Dialog class 사용법
dialog의 기본적인 성질은 dialog class에 다 정의되어 있지만 subclass인 Alertdialog와 PickerDialog를 사용해야 한다.
progressdialog는 deprecate되었다.

▹Dialog와 DialogFragment
dialog class를 이용해 dialog view를 만드는 방법은 두 가지가 있다. 하나는 dialog class와 builder를 이용하는 방법이고 다른 방법은 dialog fragment를 상속하는 클래스를 만들고 그 class를 show()방식으로 띄우는 것이다.

▹dialog의 활용
https://material.io/components/dialogs#usage

☆ dialog 만들기
dialog는 굳이 바로 fragment로 만들어 쓸 필요 없이 띄우고 싶은 곳에서 builder를 이용해서 만들면 된다.
하지만 반복적으로 같은 dialog를 띄우거나 코드를 분리하고 싶으면 fragment를 사용하면 된다.

Dialog without fragment

▹ dialog builder 객체를 만들고 new를 이용해 초기화한다.

```
android.app.AlertDialog.Builder builder = new android.app.AlertDialog.Builder(this);
```

▹ dialog를 위해 만든 custom view xml파일을 inflate하고 setView를 이용해 builder와 연결한다

```
LayoutInflater inflater = (LayoutInflater)
this.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
View view = inflater.inflate(R.layout.custom_dialog, null);
builder.setView(view);
```

▹custom view를 사용하지 않을 경우 임의로 title subtitle을 추가해야 한다

```
//title, subtitle
        builder.setMessage(R.string.dialog_message)
             .setTitle(R.string.dialog_title);
//edittext
        EditText editTextConfidence = new EditText(MainActivity.this);
        editTextConfidence.setInputType(InputType.TYPE_CLASS_TEXT);
        myDialog.setView(editTextConfidence);

//list
        builder.setItems(R.array.colors_array, new DialogInterface.OnClickListener() {
                    public void onClick(DialogInterface dialog, int which) {
                     // The 'which' argument contains the index position
                     // of the selected item
//arraylist 등 여러가지를 추가할 수 있다.
```

▹ negative positive button을 만들고 acrtion을 지정한다

```
        builder.setPositiveButton("저장", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialogInterface, int i) {
        //positive action          }
      });
      builder.setNegativeButton("취소", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialogInterface, int i) {
        //negative action          }
      });
```

▹ alertdialog 객체를 만들고 builder.create()을 이용해 초기화해준다 그리고 .show()를 이용해 화면에 pop시킨다

```
        android.app.AlertDialog alertDialog = builder.create();
        alertDialog.show();
```

Dialog using fragment

https://developer.android.com/guide/fragments/dialogs

▹dialogfragment를 상속하는 class를 만들고 onCreateDialog()를 오버라이드한다

```
public class CustomDialog extends AppCompatDialogFragment
```

▹onCreateDialog 내에서 Builder객체를 만들어 주고 getActivity()로 context를 넘겨준다

```
@Override
  public Dialog onCreateDialog(@Nullable Bundle savedInstanceState) {
    AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());
```

▹사용하고 싶은 layout을 담은 xml파일을 준비하고 inflater를 이용해 view객체에 담는다.

```
LayoutInflater inflater = getActivity().getLayoutInflater();
View view = inflater.inflate(R.layout.custom_dialog, null);
```

▹ builder.setView()를 이용해 사용하고 싶은 layout을 추가하고, positivebutton과 negativebutton을 추가한다.

```
builder.setView(view);
//actions
builder.setPositiveButton("저장", new DialogInterface.OnClickListener() {
@Override
public void onClick(DialogInterface dialogInterface, int i) {          }
});
builder.setNegativeButton("취소", new DialogInterface.OnClickListener() {
@Override
public void onClick(DialogInterface dialogInterface, int i) {          }
});
```

▹ builder.create()를 반환해서 dialog를 만든다

```
return builder.create();
```

▹dialogfragment를 activity에서 열기 위해서는 dialogfragment 객체를 만들고 .show() 메서드를 이용하면된다.

```
public void openDialog() {
      CustomDialog customDialog = new CustomDialog();
      customDialog.show(getSupportFragmentManager(), "positive saying dialog");
  }
```

PICKER

안드로이드에서 제공하는 dialog의 한 형태로 사용자가 입력하는 시간이나 날짜를 입력 받기 쉬운 모양으로 구성된다.
dialog와 동일하게 fragment를 이용해서 구성해도 되고 builder를 이용해서 activity에서 직접 구현해도 된다.

?Calendar

https://developer.android.com/reference/java/util/Calendar

Calendar는 시간에 대한 정보를 처리하기 위해 사용되는 추상 클래스로 특정 시간(현재)을 가져오거나 calendar field를 설정하거나 조작하는데 사용된다.
Calendar 객체는 가장 많이 date-time을 format하고 저장하는데 사용된다.

Picker Dialog without fragment

▹ TimerPIckerDialog.onTimesetLIstener 인터페이스 객체를 만든다. onTimeSet 메서드를 override하고 그 안에 선택한 값으로 바꿀 것을 적는다

```
TimePickerDialog.OnTimeSetListener onTimeSetListener = new
TimePickerDialog.OnTimeSetListener() {
  @Override
  public void onTimeSet(TimePicker timePicker, int i, int i1) {
        start_hour = i;
        start_minute = i1;
        startTime.setText(String.format(Locale.getDefault(), "%02d:%02d", start_hour,
         start_minute));  }
};
```

▹PickerDialog를 생성하기 위해 TimePickerDialog 객체 만들고 context, theme, listener, 초기 시간, 초기 분, 24시간 표기 여부를 인자로 전달한다. 그리고 setTitle()로 제목을 정해주고 show()를 이용해서 화면에 띄운다

```
TimePickerDialog timePickerDialog = new TimePickerDialog(getActivity(),
AlertDialog.THEME_HOLO_DARK
            , onTimeSetListener, 0,0,true);
 timePickerDialog.setTitle("시작 시간");
 timePickerDialog.show();
```

▷ Createing a Picker

1. DialogFragment를 상속해야 하고,
2. DatePickerDialog.OndateSetListener를 implement해야 한다.
3. 추상 메서드 onDateSet을 추가해서 고른 날짜로 무엇을 할 지 적어준다
4. onCreateDialog를 통해 .xml파일 없이 제공되는 달력 dialog를 쓸 수 있다. 이때 DatePickerDialog를 return해줘야 한다

▷ Showing a Picker

dialogFragment를 켜주고 싶은 버튼 onclicklistener 내에 FragmentDateDialog 객체를 만들어주고 .show()메서드를 사용한다.

Picker Dialog using fragment

https://developer.android.com/guide/fragments/dialogs

▷ DialogFragment를 상속하는 class를 만들고 onCreateDialog를 override해서 그 안에 timepickerdialog를 만든다. timepickerdialog 같은 경우는 custom xml파일을 만들 수 없다.

```
public class CustomPicker extends AppCompatDialogFragment implements
TimePickerDialog.OnTimeSetListener {

 @Override
   public Dialog onCreateDialog(@Nullable Bundle savedInstanceState) {
      //default value
      Calendar c = Calendar.getInstance();
      int hour = c.get(Calendar.HOUR_OF_DAY);
      int minute = c.get(Calendar.MINUTE);
      ///create picker
      return new TimePickerDialog(getActivity(), this, hour, minute
            , DateFormat.is24HourFormat(getActivity()));
   }
}
```

▷onTimeSet 메서드를 오버라이드해서 그 안에 입력한 시간이나 날짜를 가지고 어떤 액션을 취할지 코딩한다.

```
@Override
   public void onTimeSet(TimePicker timePicker, int i, int i1) {
   }
```

▷dialogfragment를 activity에서 열기 위해서는 dialogfragment 객체를 만들고 .show() 메서드를 이용하면된다.

```
        public void openDialog() {
            CustomDialog customDialog = new CustomDialog();
            customDialog.show(getSupportFragmentManager(), "positive saying dialog");
        }
```

SQLite Database

◇ SQLite syntex

https://intellipaat.com/blog/tutorial/sql-tutorial/

▹Table
여러 개의 column이 모여서 table이 되고 여러 table이 모여서 database를 이룬다
Sql은 표의 형식으로 data를 저장한다.
그리고 표 안에 몇개의 column을 넣을 것인지 명시해야 한 그룹의 data에 어떤 정보가 들어가는지 알 수 있다.
표 안의 row 개수는 결국 우리가 data 그룹를 저장한 만큼 생성된다.

▹Create
create는 table을 새로 생성할 때 필수적으로 필요하다
table의 이름을 설정하고 그 table에 들어갈 column의 개수와 각각 이름을 정해준다. 그리고 각 column에 들어갈 data type를 정한다

▹Drop
Sql에서 drop a table은 그 table과 그 안에 있는 data를 삭제한다는 의미이다.

▹Insert into
Iinsert 구문은 하나의 record(row)를 table에 추가할 때 사용한다. 이때 추가하는 data는 반드시 table과 같은 개수의 column을 가지고 각 column안에 저장된 자료형도 같아야 한다.

INSERT INTO tablename VALUES (value1, value2,….valueN);

▹Select
select구문은 table에서 정해진 규칙에 따라 data를 가져오거나 table 전체를 가져오는 것을 도와준다
select구문은 column을 선택할 수는 있지만 column에서 어떤 row를 고를지는 선택할 수 없다.

SELECT column1, column2, columnN
FROM tablename;

▹Where
where문을 이용하면 select구문에서 조건을 추가할 수 있다.

SELECT column1, column2, columnN
FROM tablename
WHERE [condition];

▹연산자
where문에서 condition을 추가할 때는 and, or, not 조건문을 사용할 수 있다.

```
SELECT column1, column2, …, columnN
FROM tablename
WHERE [condition1], … AND [conditionN];

SELECT column1, column2, … columnN
FROM tablename
WHERE [condition1], …, OR [conditionN];

SELECT column1, column2, … columnN
FROM tablename
WHERE NOT [condition1];
```

▹IN
연산자 in은 where과 함께 쓰여 특정 colomn안에서 row를 고를 수 있게 도와준다.

```
SELECT column1, column2….
FROM table_name
WHERE column1 IN (value1, value2….);
```

▹BETWEEN
연산자 between을 이용하면 select문을 이용해서 column을 고른 다음 where과 between을 이용해서 특정 구간의 row를 가져올 수 있다.

```
SELECT column1, column2….
FROM table_name
WHERE column1 BETWEEN Value1 AND Value2;
```

▹DELETE & TRUNCATE in SQL
DELETE는 특정 row나 row전부를 지우는 명령어이다. 조건문을 만족하는 row는 모두 삭제된다.

```
DELETE FROM table_name
[WHERE condition];
```

TRUNCATE는 table의 구조 빼고 안에 저장되어 있는 모든 row를 삭제한다. 구조란 처음에 설정한 column의 개수와 저장할 자료형이다.

```
TRUNCATE TABLE table_name;
```

▹UPDATE in SQL
update란 data가 일단 한번 저장된 다음 이미 저장되어 있는 data를 고칠 때를 의미한다.
추가를 할 때는 insert into 문을 쓰면 된다.
한 row안에 있는 여러 columnn을 고칠 수도 있고 한 column안에 여러 row를 한번에 고칠 수 있다.

```
UPDATE table_name
```

```
SET col1=val1, col2=val2…
[Where condition];
```

◇ DATA Types in SQLite

https://www.sqlite.org/datatype3.html

▹Storage Class and Data type
Storage Class는 SQLite database에 저장될 수 있는 자료형의 묶음이다. Storage Class의 종류는 null, integer, real, text, blob가 있다. 각 class는 data의 길이에 따라 더 세분화 되지만 세분화되어 저장될 뿐 입력할 때나 출력할 때는 다시 5가지 storage class의 type으로 변환된다.

▹Boolean type
Boolean type는 따로 존재하지 않는다. 대신 integer자료형으로 저장하고 sql은 true와 false를 1과 0으로 변역해서 integer에 저장한다.

▹Date and Time type

Date and Time type는 따로 존재하지 않는다. 대신 Date Time Funtion을 이용해 text, real, integer로 변환해서 저장하면 된다.

◇ Save data using SQLite

▹sqliteopenhelper클래스를 상속하는 클래스를 만들고 constructor를 만들어 줘야 한다. constructor는 context인자만 놔두고 나머지는 삭제한다.

```
public class TitleDatabaseHelper extends SQLiteOpenHelper {
 public TitleDatabaseHelper(@Nullable Context context) {
       super(context, DATABASE_NAME, null, DATABASE_VERSION);
   }
}
```

▹그리고 상수 설정을 이용해 database이름, database 버전, table이름, 사용할 column을 설정해 줘야한다.

```
   //TAG
   private static final String TAG = "DBHelper";

   //database create var
   private static final String DATABASE_NAME = "positive.db";
   private static final int DATABASE_VERSION = 1;

   //table & column name
   private static final String TABLE_NAME = "tb_positive";
   private static final String COLUMN_ID = "_id";
   private static final String COLUMN_SAYING = "saying";
```

▹그리고 사용할 query, 즉 sql명령어 문장들을 선언해 준다. 필수적이지는 않으나 유지보수에 용의해진다.

```
   //query
   private static final String CREATE_QUERY = "create table tb_positive (_id integer primary
   key autoincrement, saying text);";
   private static final String UPDATE_QUERY = "drop table tb_positive";
   private static final String RAW_READ_QUERY = "select saying from tb_positive";
```

▹ create database using SQL helper
SQLiteOpenHelper를 상속하는 class를 만들어 그 안에 override된 onCreate메서드에서 database를 만들어야 한다.

```
 @Override
   public void onCreate(SQLiteDatabase sqLiteDatabase) {
      sqLiteDatabase.execSQL(CREATE_QUERY);
   }
```

▹oncreate 메서드를 오버라이드해서 db.execSQL 메서드와 database create query를 이용해 database을 만들어 준다. 인자는 id, 그리고 사용할 column을 전달하면 된다

```
@Override
   public void onUpgrade(SQLiteDatabase sqLiteDatabase, int i, int i1) {
      sqLiteDatabase.execSQL(UPDATE_QUERY);
      onCreate(sqLiteDatabase);
   }
```

▹database에 정보 넣기

```
        // Gets the data repository in write mode
        SQLiteDatabase db = dbHelper.getWritableDatabase();

        // Create a new map of values, where column names are the keys
        ContentValues values = new ContentValues();
        values.put(FeedEntry.COLUMN_NAME_TITLE, title);
        values.put(FeedEntry.COLUMN_NAME_SUBTITLE, subtitle);

        // Insert the new row, returning the primary key value of the new row
        long newRowId = db.insert(FeedEntry.TABLE_NAME, null, values);
```

▹ Read information from a database

일단 getReadableDatabase() 메서드를 이용해 read모드로 들어간다.

```
        SQLiteDatabase db = dbHelper.getReadableDatabase();
```

db.query()메서드나 db.rawquery()메서드를 이용해서 읽고 싶은 datagroup을 만들고 그 group을 가리키는 cursor 객체를 return 받는다.

```
        String query = "select positive_saying from tb_positive order by _id desc limit 1";
        Cursor cursor = DB.rawQuery(query, null);
```

반드시 while cursor.movetonext()를 이용해 처음에 -1 위치에 있던 cursor를 오른쪽으로 움직이면서 data를 읽어야 한다.

```
        List itemIds = new ArrayList<>();
        while(cursor.moveToNext()) {
          long itemId = cursor.getLong(
              cursor.getColumnIndexOrThrow(FeedEntry._ID));
          itemIds.add(itemId);
        }
        cursor.close();
```

▹ Delete information from a database

```
        // Define 'where' part of query.
        String selection = FeedEntry.COLUMN_NAME_TITLE + " LIKE ?";
        // Specify arguments in placeholder order.
        String[] selectionArgs = { "MyTitle" };
        // Issue SQL statement.
        int deletedRows = db.delete(FeedEntry.TABLE_NAME, selection, selectionArgs);
```

▹Update a database

▹Persisting database connection

getWritableDatabase()와 getReadableDatabase()는 database가 close()되어 있을 때는 접속하기 힘드니 database를 최대한 오랫동안 열어 놓아야 한다. 따아서 보통 Activity의 onDestroy()메서드에서 db를 close()한다.

```
@Override
protected void onDestroy() {
    dbHelper.close();
    super.onDestroy();
}
```

Recyclerview

▹layout xml에 recycler view를 추가한다

```
<androidx.recyclerview.widget.RecyclerView
    android:id="@+id/app_list"
    android:layout_width="380dp"
    android:layout_height="300dp"
    android:background="@color/white"
    android:layout_marginTop="20dp"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/search_current_app" />
```

▹root layout이 cardview인 새로운 xml를 만든다. 여기에는 한 item을 표시하는 모양을 만든다. cardview의 layout에 marginTop을 설정해야 각 cardview사이 간격을 만들 수 있다.

```
<androidx.cardview.widget.CardView
xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
android:layout_marginTop="10dp"
   android:layout_marginRight="10dp"
   android:layout_marginLeft="10dp"

   xmlns:app="http://schemas.android.com/apk/res-auto">


   <androidx.constraintlayout.widget.ConstraintLayout
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:orientation="horizontal">
      <ImageView
         android:id="@+id/app_logo"
         android:src="@mipmap/ic_launcher"
         android:layout_width="100dp"
         android:layout_height="100dp"
         android:adjustViewBounds="true"
         android:padding="10dp"
         app:layout_constraintLeft_toLeftOf="parent"
         app:layout_constraintRight_toRightOf="parent"
         app:layout_constraintTop_toTopOf="parent"/>

      <TextView
         android:id="@+id/app_name"
         android:layout_width="80dp"
         android:layout_height="30dp"
         android:background="#B6B6B6"
```

```
        android:gravity="center"
        android:text="app_name"
        android:textColor="#000000"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toBottomOf="@id/app_logo" />

    <CheckBox
        android:id="@+id/app_status"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:gravity="right"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"/>


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

▹하나의 cardview에 들어갈 정보를 관리하는 class를 만든다. 보여주고 싶은 자료형을 정의하고 getting setting method, equal method도 추가한다.;

```
public class AppInfo {

  //data type to save
  String packageName;
  String appName;
  Drawable appLogo;
  boolean appStatus;

  @Override
  public boolean equals(Object o) {
    if (this == o) return true;
    if (!(o instanceof AppInfo)) return false;
    AppInfo appInfo = (AppInfo) o;
    return Objects.equals(getPackageName(), appInfo.getPackageName());
  }

  @Override
  public int hashCode() {
    return 0;
  }

  //getting setting method
  public String getPackageName() {
    return packageName;
```

```
    }

    public void setPackageName(String packageName) {
        this.packageName = packageName;
    }

    public String getAppName() {
        return appName;
    }

    public void setAppName(String appName) {
        this.appName = appName;
    }

    public Drawable getAppLogo() {
        return appLogo;
    }

    public void setAppLogo(Drawable appLogo) {
        this.appLogo = appLogo;
    }

    public boolean isAppStatus() {
        return appStatus;
    }

    public void setAppStatus(boolean appStatus) {
        this.appStatus = appStatus;
    }
}
```

▹Adapter class를 만들어서 방금 만들었던 view에 들어갈 자료형을 각 cardview에 할당하고 전체적인 recyclerview를 만든다.
adapter class가 RcyclerView.Adapter<Adapter.ViewHolder>를 상속하게 한다.

```
public class AppListAdapter extends RecyclerView.Adapter<AppListAdapter.ViewHolder> {

    List<AppInfo> appList;

    Context context;

    //constructor for AppListAdapter
    //adding appInfos input into appList
    AppListAdapter(List<AppInfo> appInfos) {
        this.appList=appInfos;
    }

    @NonNull
```

```
    @Override
    public ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
        if(context == null)
            context = parent.getContext();
        View view = LayoutInflater.from(context).inflate(R.layout.app_item, parent, false);
        ViewHolder viewHolder = new ViewHolder(view);

        return viewHolder;
    }

    @Override
    public void onBindViewHolder(@NonNull ViewHolder holder, int position) {
        holder.appLogo.setImageDrawable(appList.get(position).appLogo);
        holder.appName.setText(appList.get(position).appName);
        if(appList.get(position).appStatus)
            holder.appStatus.setChecked(true);
        else
            holder.appStatus.setChecked(false);
    }

    @Override
    public int getItemCount() {
        //return size to notify how many card needed to be made
        return appList.size();
    }

    public class ViewHolder extends RecyclerView.ViewHolder {
        //object data for the card view
        ImageView appLogo;
        TextView appName;
        CheckBox appStatus;
        public ViewHolder(@NonNull View itemView) {
            //using super connect objects with input View
            super(itemView);
            appLogo = itemView.findViewById(R.id.app_logo);
            appName = itemView.findViewById(R.id.app_name);
            appStatus = itemView.findViewById(R.id.app_status);
        }
    }

}
```

▹mainActivity에서 recycler 객체와 layoutmanager, adapter연결하기

```
//recycler 객체와 view 연결하기
RecyclerView appList;
appList = rootView.findViewById(R.id.app_list);

//layoutmanager 설정하고 .setlayoutmanager로 recycler객체에 연결하기
GridLayoutManager gridManager = new GridLayoutManager(getActivity(), 4);
appList.setLayoutManager(gridManager);

// setAdapter를 이용해서 adapter을 연결한다.
AppListAdapter adapter = new AppListAdapter(apps);
appList.setAdapter(adapter);
```